Índice

1 World Wide Web e a linguagem PHP....1
1.1 Introdução....1
1.2 O que é o PHP?....1
1.3 Origem da linguagem PHP....1
1.4 Características do PHP....2
1.4.1 Código de fonte−aberta....2
1.4.2 Interpretador....2
1.4.3 Suporte a inúmeros bancos de dados....2
1.4.4 Independente de plataforma....2
1.4.5 Simples....2
1.5 Sites úteis....3

2 Básico PHP....4
2.1 Introdução....4
2.2 Básico....4
2.3 Variáveis....5
2.4 Tipos de dados, concatenação e constantes....6
2.4.1 Concatenação....6
2.4.2 Constantes....7
2.5 Operadores Aritméticos....8
2.5.1 Combinação de operadores....8
2.5.2 Incremento e decremento....8
2.6 Operadores de comparação....9
2.7 Operadores Lógicos....9
2.8 Scap....9

3 Instruções de Condição e Loops....10
3.1 Introdução....10
3.2 Instruções de Condição....10
3.2.1 Instrução if....10
3.2.2 Instrução if then else....10
3.2.3 Instrução Else if....11
3.2.4 Instrução Switch....11
3.3 Loops....12
3.3.1 While....12
3.3.2 Do While....13
3.3.3 For....13

4 Arrays....14
4.1 Introdução....14
4.2 Criação e inicialização de Arrays....14
4.3 Os tipos de índices....14
4.4 Funções para a manipulação de arrays....15

5 Funções em PHP....17
5.1 Introdução....17
5.2 Definição....17
5.3 Procedimentos sem passagem de parâmetros....17
5.4 Procedimentos com passagem de parâmetros....17
5.5 Funções sem passagem de parâmetros....18
5.6 Funções com passagem de parâmetros....18
5.7 Passagem de parâmetros por referência....18
5.8 Arquivos de Inclusão....19

Índice

6 Manipulação de Arquivos 20
6.1 Introdução 20
6.2 Abrindo e fechando arquivos 20
6.2.1 Fopen 20
6.2.2 Fclose 21
6.3 Gravação em um arquivo 21
6.4 Leitura de um arquivo 21
6.5 Bloquear arquivos 22
6.6 Utilizando loops para acessar arquivos 23
6.7 Funções diversas para a manipulação de arquivos 23
6.7.1 rewind() 23
6.7.2 fseek() 23
6.7.3 copy() 23
6.7.4 unlink() 24
6.7.5 rename() 24
6.7.6 file_exists() 24
6.7.7 filesize() 24
6.7.8 filetype() 24
6.7.9 is_dir() 24
6.7.10 is_executable() 24
6.7.11 is_readable() 24
6.7.12 is_writeable() 24
6.8 Diretórios 24
6.8.1 chdir() 25
6.8.2 opendir() 25
6.8.3 readdir() 25
6.8.4 closedir() 25
6.8.5 mkdir() 25
6.8.6 rmdir() 25
6.9 Exemplo Prático − Contador de Acessos 25

7 Manipulação de Strings e Expressões Regulares 27
7.1 Introdução 27
7.2 Manipulação de Strings 27
7.2.1 substr() 27
7.2.2 strlen() 28
7.2.3 trim() 29
7.2.4 ucfirst() 29
7.2.5 ucwords() 29
7.2.6 strpos() 30
7.2.7 strrpos() 30
7.2.8 str_replace() 30
7.2.9 chr() 30
7.2.10 strcmp() 31
7.2.11 strcasecmp() 31
7.2.12 ereg() 31
7.2.13 ereg_replace() 32
7.2.14 split() 32
7.3 Expressões Regulares 33
7.3.1 Metacaracteres tipo Representante . [ ] [^] 33
7.3.2 Metacaracteres tipo Quantificadores ? * + { } 34
7.3.3 Metacaracteres tipo Âncoras ^$ \b 35
7.3.4 Outros Metacaracteres \c | ( ) 35

Índice

8 Formulários HTML 37
8.1 Introdução 37
8.2 Entendendo o funcionamento dos formulários 37
8.3 Conhecendo os objetos dos formulários 37
8.4 Criando um formulário 39
8.5 Utilizando os campos do formulário 39
8.6 Inserindo caixas de seleção e botões de rádio 41
8.7 Inserindo listas e menus 42
8.8 Adicionando botões de formulário 44
8.9 Inserindo um objeto image 45
8.10 Adicionando o objeto file 45

9 Processamento de Formulários 47
9.1 Introdução 47
9.2 Primeiro Processamento 47
9.3 Validação de Formulários 48
9.4 Cabeçalhos HTTP 50
9.5 Redirecionamento de páginas 50
9.6 Trabalhando com datas 51
9.6.1 date() 51
9.6.2 checkdate() 52

10 Envio de Informações 53
10.1 Introdução 53
10.2 Correio Eletrônico 53
10.3 Feedback 53
10.4 Upload de Arquivos 54

11 Sessões e Cookies em PHP 57
11.1 Introdução 57
11.2 Como utilizar sessões? 57
11.3 Esquema de Sessão 57
11.4 Funções Definidas 57
11.5 Passando a ID de Sessão 58
11.6 Criar uma sessão e mostrar o ID de sessão no navegador 58
11.7 Criar variáveis dentro de uma sessão 58
11.8 Sessão usando cookies 59
11.9 Carregando valores através de formulários usando uma sessão 61

12 Introdução ao MySQL 64
12.1 Introdução 64
12.2 Por que o MySQL? 64
12.3 O que é MySQL? 64
12.4 Básico 64
12.5 Conectando e desconectando do servidor 64
12.6 Executando consultas 65
12.7 Criando e usando um banco de dados 66
12.7.1 Criando e selecionando um banco de dados 67
12.7.2 Criando uma tabela 67
12.7.3 Carregando dados para uma tabela 70
12.7.4 Recuperando informação de uma tabela 70
12.7.4.1 Selecionando todos dados 71
12.7.4.2 Selecionando linhas particulares 71
12.7.4.3 Selecionando colunas particulares 72

Índice

12 Introdução ao MySQL
12.7.4.4 Ordenando linhas....72
12.7.4.5 Casamento de padrões....73
12.7.4.6 Contando linhas....73
12.7.4.7 Restrições....74
12.7.4.8 Cláusula auto_increment....74
12.7.5 Usando múltiplas tabelas....75
12.8 Alterando linhas....76
12.9 Deletando linhas....76

13 Conectando PHP com MySQL....78
13.1 Introdução....78
13.2 Conexões MySQL....78
13.2.1 mysql_connect()....78
13.2.2 mysql_pconnect()....78
13.2.3 mysql_close()....78
13.2.4 mysql_create_db()....78
13.2.5 mysql_drop_db()....79
13.2.6 mysql_select_db()....79
13.2.7 mysql_query()....79
13.2.8 mysql_db_query()....79
13.2.9 mysql_list_dbs()....79
13.2.10 mysql_list_tables()....79
13.2.11 mysql_list_fields()....80
13.2.12 mysql_result()....80
13.2.13 mysql_num_fields()....80
13.2.14 mysql_num_rows()....80
13.2.15 mysql_affected_rows()....80
13.2.16 mysql_fetch_array()....81
13.2.17 mysql_insert_ID()....81
13.3 Primeira Aplicação....81
13.4 Preenchendo Menus....82
13.5 Inserindo um registro....85
13.6 Atualização de Registro....88
13.7 Excluindo um Registro....92
13.8 Registros Duplicados....94

14 Anexo A − Conceitos Utilizados....95
14.1 Softwares Livres....95
14.2 Servidor Web....96
14.3 Server Side....96
14.4 Client Side....97
14.5 Interpretador CGI....97
14.6 Módulo Apache....97
14.7 Configuração para utilizar a função mail():....97

15 Anexo B − Instalação e Configuração do PHP....99
15.1 Windows....99
15.1.1 Download dos itens necessários....99
15.1.2 Instalação....99
15.1.3 Configuração do PHP....99
15.1.4 Configuração do Apache....99
15.1.5 Configuração do MySQL....100
15.1.6 Finalização....100

Índice

15 Anexo B − Instalação e Configuração do PHP

15.2 Linux.....100

1 World Wide Web e a linguagem PHP

1.1 Introdução

Com o passar do tempo, as mudanças na World Wide Web são cada vez maiores, já se foi a época em que a Internet era tratada como uma grande enciclopédia usada apenas para a obtenção de informações, onde a linguagem HTML (Hyper Text Markup Language) era o suficiente para prover as estruturas dos sites.

Hoje em dia, as *home pages* têm visuais e conteúdos muito mais dinâmicos, totalmente voltados para a intereção com o usuário da rede. Queremos sites com interfaces amigáveis, conteúdos interessantes e significativos, fácil uso, funcionalidades corretas e que possam, de certa forma, realizar nossos propósitos.

O desenvolvedor Web tem inúmeras opções para a realização de seus projetos, mas deve−se ter a consciência de que com o passar dos tempos, tanto hardwares quanto softwares passarão por modificações, portanto, tais projetos devem de certa forma manterem−se constantemente funcionais. É ae que entram os softwares de fonte aberta, que de certa forma são a melhor garantia para um desenvolvedor que deseje que seus projetos funcionem hoje e amanhã em diversas máquinas diferentes.

1.2 O que é o PHP?

PHP, sinônimo de *PHP (Personal Home Pages) Hypertext Preprocessor*, é uma linguagem de elaboração de scripts usada unicamente para o processamento de páginas da web. Podemos dizer, que assim você poderá transformar seu site em um aplicativo Web com conteúdo e informações dinâmicas, contrastando com as inumeras coleções de páginas estáticas com informações limitadas que não podem ser atualizadas frequentemente. Sites que têm como objetivo trabalharem frequentemente com o processamento de dados e informações, devem ser desenvolvidos como aplicativos Web.

PHP é uma linguagem embutida no HTML e que reside no servidor em que o servidor web está rodando, isto significa que, o usuário através do browser faz a chamada de uma página PHP, o servidor web a processa e, depois, envia as informações de volta para o navegador do usuário. O PHP roda no ***Server Side*** (ler Anexo A − Conceitos Utilizados).

O código PHP pode rodar junto com páginas HTML de uma forma em pode se ter o melhor dos dois mundos: pode−se ter uma página HTML que em certas partes do seu código utiliza o PHP para processar e manipular certas informações, ou pode−se ainda, ter uma página PHP produzindo um página HTML pura com seu processamento normal, exemplos de ambas as formas serão apresentads no decorrer do livro.

Vamos entender um pouco melhor como PHP funciona: quando o navegador tenta carregar uma página com código PHP, podendo ser um script PHP ou um documento HTML com PHP embutido, o servidor web irá processar tal código (leia Anexo A − Conceitos Utilizados para maiores informações) e executá−lo. Assim, o código executado substitui o código fonte original da página e o servidor web, envia a página de volta para o navegador.

1.3 Origem da linguagem PHP

Em 1994 surgiu PHP (na época, apenas Personal Home Page), um pocote de ferramentas feito a partir da linguagem Perl por Rasmus Lerdorf. Em vista do interesse que seu pacote ferramentas estava tendo em meados de 1995, Lerdorf desenvolveu um sistema de processamento de scripts com uma ferramenta para analisar formulários HTML: FI, Form Interpreter, surgiu ae o PHP/FI ou PHP2.

Com a maior utlização destas ferramentas para a realização de tarefas mais complicadas, mais pessoas se uniram a Lerdorf para o desenvolvimento do projeto, assim surgiu o PHP3, que contava com um sistema de

processamento de scripts mais aprimorado, maior funcionalidade para o código e uma sintaxe também aprimorada. A sintaxe da linguagem PHP é de certa forma semelhante às sintaxes de linguagens como C, C++ e Java, significando que desenvolvedores com conhecimentos destas linguagens terão ainda mais facilidade para aprender as estruturas básicas da linguagem PHP.

Em 2000, surgiu o PHP4 que se baseia no sistema de processamento de scripts Zend, que foi desenvolvido desde o zero para poder ser facilmente incorporado em aplicativos diferentes. O PHP4 já utiliza conceitos de linguagem orientada à objetos, o que foi mais desenvolvido em 2003, com o lançamento do PHP5.

1.4 Características do PHP

- Código de fonte−aberta
- Interpretador
- Suporte à inúmeros bancos de dados
- Independente de plataforma
- Simples

1.4.1 Código de fonte−aberta

Em posse do código−fonte, qualquer modificação no código pode ser feita (desde que não violem os direitos autorais da linguagem, é claro), se o desenvolvedor quer que algum novo recurso seja adiconado ou um certo bug seja corrigido, ele mesmo pode fazer tais modificações. Podendo ainda certa modificação ser bastante atrativa aos olhos de outros desenvolvedores, o que pode ajudar ainda mais no desenvolvimento da linguagem.

Com o código−fonte aberto, o PHP também pode ser adquirido de inúmeras formas diversas, podendo ser gratuito ou com um certo valor comercial, dependendo apenas da onde o usuário irá adquiri−lo. Um grande exemplo da liberdade que os Softwares Livres nos propoem.

1.4.2 Interpretador

PHP não é compilador, e sim um interpretador, ou seja, ele interpreta e executa o código da maneira exata como ele foi desenvolvido. Tal interpretação e execução é realizada no próprio servidor web.

1.4.3 Suporte a inúmeros bancos de dados

Uma forma de dinamizar uma Home Page é a utilização de um banco de dados para guardar valores dinâmicos, ficando com a linguagem PHP a função de manipulá−los. Como PHP tem código−fonte aberto, inúmeros bancos de dados são suportados, com nenhuma ou pouquíssimas modificações no código. Para fins didáticos vamos utilizar o banco de dados MySQL.

1.4.4 Independente de plataforma

PHP é independente de plataforma podendo rodar em sistemas operacionais como Linux e Windows sem problemas, graças ao seu código−fonte aberto.

1.4.5 Simples

Uma das garndes características desta linguagem, a simplicidade para se usar banco de dados e ter independência de plataforma. Sendo esta uma característica que pode exemplificar o porquê da utilização da linguagem PHP ao invés de utilizar outras ferramentas como: ASP, Cold Fusion, Perl, Java Server Pages, etc.

Tendo sido criada para a programação web, PHP se distingue nesta área, principalmente, devido à sua simplicidade de como realizar certas ações, como por exemplo, acessar e consultar um banco de dados, o que pode ser realizado com 2 ou 3 linhas de código apenas.

1.5 Sites úteis

- http://www.php.net/
- http://www.phpbuilder.com/
- http://www.apache.org/
- http://www.mysql.com/

2 Básico PHP

2.1 Introdução

Neste capítulo iniciaremos os aspectos práticos da linguagem PHP, tendo em consideração as informações essenciais sobre sintaxe e funções, focando principalmente os tópicos: variáveis, constantes, operadores aritméticos e operadores comparativos.

2.2 Básico

Como já foi comentado, você pode ter um script PHP que gera HTML, ou um script HTML que executa PHP dentro daquele documento, não interessando qual tipo de script seja, o código PHP deve se delimitado do resto código, podendo estas delimitações serem as seguintes:

<?php ou <?	Marcas inicias do código PHP
?>	Marca final do código PHP

Lembre−se:

- Os caracteres de escape ASP: <% (início) e %> (fim) também podem ser utilizados desde que a linguagem seja configurada para tal propósito.

A linguagem PHP também possibilita que o desenvolvedor utilize comentários em seu código para que seu script possa ter um entendimento um pouco mais fácil. Veja a tabela abaixo:

#	Iniciar uma única linha de comentário
//	Iniciar uma única linha de comentário
/*	Iniciar comentário que terá mais de uma linha
*/	Terminar comentários iniciados com /*

Vamos agora escrever nosso primeiro script PHP para nos adaptarmos cada vez mais com a estrutura da linguagem, lembre−se que o código pode ser escrito em qualquer editor de texto.

script1.php

```
<?php
# Este é um comentário, o browser não irá mostrá−lo
echo "Meu primeiro script PHP! <br>";
echo "<hr>";
# A função echo imprime informações na tela
?>
```

Agora vamos entender o código acima:

- Note que o código é cercado por delimitações já apresentadas, isto significa que quando o servidor HTTP encontra a primeira delimitação (<?php), ele começa a processar o código ali contido como um script PHP até que ele encontre o seu término (?>)

- Duas linhas de comentários foram incluídas, o browser não irá mostrá−las
- Uma nova instrução é apresentada: echo − ela produz uma saída para o navegador. No primeiro caso imprime uma string na tela e há uma quebra de linha devido a tag HTML
 e, no segundo caso, imprime uma linha na tela através de outra tag HTML <hr>
- Note que todas as instruções terminam em ';'

Lembre−se:

- Todas as instruções em PHP devem terminar com ';', caso contrário ocorrerá um erro na execução do código.
- Alguns programadores usam a função sinônima print ao invés da função echo, que pode ser usada de duas maneiras:

```
echo "Sistemas Abertos!";
echo ("Sistemas Abertos!");
```

2.3 Variáveis

Como em todas as linguagens de programação, em PHP podemos armazenar dados em variáveis, e então podemos acessá−los utilizando os nomes de suas respectivas variáveis. Você pode atribuir valores às variáveis antes de precisar delas em seu script. Veja exemplos desta atribuição:

$nome = "Rafael";
$empresa = "Sistemas Abertos";

Note que o sinal de igualdade '=' é o responsável pela atribuição dos textos (sempre entre aspas) nas variáveis em questão. Você pode exibir apenas os valores das varíaveis na tela, ou também exibir seus valores junto à outras frases:

```
<?php
echo $nome;
//Exibe apenas o conteúdo da variável $nome, no caso: "Rafael"
echo "Meu nome é $nome !";
//Imprime na tela a frase "Meu nome é Rafael !"
echo "Meu nome é $nome e trabalho na $empresa !";
#Imprime na tela "Meu nome é Rafael e trabalho na Sistemas Abertos !"
?>
```

Todos os nomes de variáveis devem ser precedidos por um cifrão '$' e seguirem uma série de simples regras: qualquer palavra pode ser o nome de uma variável, desde que não começe com um número, um sublinhado '_' ou qualquer outro caractere especial como < $ & ^ ". Além destas palavras não poderem conter espaços e nem devem ter mais de 32 caracteres de comprimento. Veja alguns exemplos válidos e outros inválidos:

$3nome Não é válido
_empresa Não é válido
meu nome Não é válido
meu_nome É válido

Lembre−se:

- As variáveis não precisam ser declaradas em PHP.
- Os nomes das variáveis fazem diferenciação entre letras maiúsculas e minúsculas, portanto: $nome é diferente de $Nome, $Empresa é diferente de $EmPrEsA, etc.

- Procure sempre usar nomes de variáveis que tenham significados próximos aos valores nelas armazenados, isto ajuda no entendimento do seu script.

A atribuição de valores às variáveis também podem ser feitas no mesmo momento em que imprimir a variável na tela. Veja:

```
<?php
$nome = "Rafael";
echo $nome = "Akira";
/* A variável $nome terá seu valor alterado
assim que o resultado é 'ecoado' na tela */
?>
```

Agora imaginem a seguinte situação, queremos imprimir na tela a seguinte frase: "*Como você está, "Amigo\a"?*" com todas as aspas e a barra invertida. Para imprimirmos tais caracteres na tela devemos fazer o que chamamos de 'scap', utiliza−se uma barra invertida antes do caractere que queremos imprimir, veja:

```
<?php
echo "Como você está, \"Amigo\\a\" ?";
/*Outra maneira seria utilizar aspas simples
para não confundir o interpretador*/
echo "Como você está, 'Amigo\\a' ?";
?>
```

2.4 Tipos de dados, concatenação e constantes

Em PHP existem os diferentes tipos de dados:

- Inteiros − números inteiros
- Pontos flutuantes − números decimais
- Literais − caracteres ou uma cadeia de caracteres (srting)
- Arrays − falaremos sobre este tipo mais tarde
- Objetos − são utilizados na programação orientada a objetos, o que é um tópico do curso de PHP avançado
- Booleanos − são os valores "true" (verdadeiro) e "false" (falso)

Para retornar o tipo do dado, usamos a função *gettype* , observe:

```
<?php
$nome = "Rafael";
echo (gettype($nome));
#Imprimirá na tela string
?>
```

2.4.1 Concatenação

Concatenação é a união de informações, no nosso caso, usaremos para a união de variáveis. Para isto devemos usar o *operador de concatenação* representado pelo '.', vejamos o exemplo abaixo:

Tem−se duas variáveis:

$primeironome = "Rafael";
$sobrenome = "Berquó";

Agora as uniremos:

$nome = $primeironome.$sobrenome;

A expressão abaixo imprime **RafaelBerquó** :

```
echo $nome;
```

Vamos agora escrever e entender um outro script para fixarmos as características da concatenação em PHP:

script2.php

```
<html>
<body>
<center>
<h1>Bem-vindos!</h1>
<hr>

<?php
$primeironome = "Rafael";
$sobrenome = "Berquó";
$nome = $primeironome ." " .$sobrenome;
# une as variáveis com um espaço entre elas
$empresa = "Sistemas Abertos";
echo "Olá! Meu nome é $nome!";
echo "Trabalho na empresa" ." " .$empresa ."<br>";
/* une a cadeia de caracteres a um espaço, à variável
e há uma quebra de linha */
?>

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

Lembre-se:

- Este script foi iniciado com marcas HTML, mas isso não quer dizer que ele seja um HTML puro. Nomeando-o como .php, o servidor web saberá como executar o código PHP, deixando as marcas HTML para o navegador interpretar.

2.4.2 Constantes

Em PHP também é possível utilizar constantes, ou seja, valores que permanecerão sem alterações durante todo o código. Para atribuir uma constante deve-se utilizar a palavra *define* para que o PHP saiba que o valor da constante está para ser armazenado. Observe o formato da definição:

```
define (nome da constante, "valor da constante");
```

Exemplos:

```
define (PI, "3.14");
echo " O valor de PI é: " .PI;
```

Lembre-se:

- Não se esqueça de utilizar a ',' na hora de definir as constantes
- Nunca utilize o $ antes de constantes, elas não são variáveis
- Constantes podem ser definidas em qualquer parte do código, sendo que devem ser definidas antes de utilizadas, lógico

- Utilize o operador de concatenação na hora de utilizar a constante, pois ao utilizar o nome da constante dentro de um comando echo, o interpretador PHP pode interpreta−lo com um pedaço de texto

---

2.5 Operadores Aritméticos

Já vimos que para atribuir valores à variáveis devemos usar o operador '=', agora iremos conhecer os operadores para a realização de operações aritméticas:

Operador	Significado
+	Adição
−	Subtração
*	Multiplicação
/	Divisão
%	Módulo

Exemplo de como utilizar os operadores:

script3.php

```
<?php
echo "Quanto é : 2 + 2?";
echo (2 + 2);
/*imprime na tela o resultado da operação, os
parênteses não são necessários */
$preço = "15";
$frete = "2.5";
echo "Quanto fica o preço total já com frete?";
$valortotal = $preço + $frete;
echo "O valor é: $valortotal";
?>
```

2.5.1 Combinação de operadores

São operadores utilizados para agilizar a escrita do código:

Operador	Exemplo	Significado
+ =	$a + = 5	$a = $a + 5
− =	$a − = 5	$a = $a − 5
/ =	$a / = 5	$a = $a / 5

2.5.2 Incremento e decremento

Em PHP também é possível a utilização de operadores para incrementação e decrementação, sua estrutura é muito parecida com a utilizada em linguagens como C/C++ e Java.

Exemplo	O que faz	Significado
++$a	pré−incremento	$a = $a + 1
$a++	pós−incremento	$a = $a + 1
−−$a	pré−decremento	$a = $a − 1
$a−−	pós−decremento	$a = $a − 1

2.6 Operadores de comparação

Os operadores de comparação são mais utilizados junto à instruções de condição que será visto mais adiante na obra. Em PHP os operadores de comparação verificam se:

Operador	Exemplo	Significado
= = =	($a = = = $b)	Se $a for igual a $b e forem do mesmo tipo
= =	($a = = $b)	Se $a for igual a $b
<	($a < $b)	Se $a é menor que $b
>	($a > $b)	Se $a é maior que $b
< =	($a < = $b)	Se $a é menor ou igual a $b
>	($a > = $b)	Se $a é maior ou igual a $b
! =	($a ! = $b)	Se $a é diferente de $b
< >	($a < > $b)	Se $a é diferente de $b

2.7 Operadores Lógicos

Os operadores de comparação apenas nos dão o resultado de uma simples comparação, e se quiséssemos fazer comparações compostas como por exemplo: um número deve ser menor do que três e maior do que 1. Para isso usamos os operadores lógicos:

Operador	Exemplo	Significado
&&, e	($a = = 15 && $b = = 1)	$a é igual a 15 e $b é igual a 1
||, ou	($a = = 15 || $b = = 1)	$a é igual a 15 ou $b é igual a 1
!	!($a = = $b)	$a NÃO é igual a $b

Lembre−se:

- Use sempre parênteses para determinar qual será a precendência dos operadores usados

2.8 Scap

A linguagem PHP permite usar seqüências de caracteres de controle para caracteres especiais, para isso usa−se o caractere de sacap "\", veja a tabela:

Seqüência de controle	Nome
\b	backspace (volta um espaço)
\t	tab (tabulação)
\n	quebra de linha
\r	carriage return (retorno de carro)
\”	aspas (duplas)
\'	apóstrofe (aspas simples)
\\	barra invertida (barra)

3 Instruções de Condição e Loops

3.1 Introdução

O uso de instruções de condição e Loops é obrigatório para que um script seja capaz de realizar escolhas, ferramentas muito importante para uma página web dinâmica. Aprenderemos neste capítulo as instruções de condição da linguagem PHP e suas estruturas de Loop.

3.2 Instruções de Condição

Em PHP pode−se utilizar as intruções: if, if then else, else if e switch. Vejamos cada uma delas.

3.2.1 Instrução if

A estrutura if é utilizada para o controle de fluxo, desviando ou executando ações de acordo com um condição. Veja sua sintaxe:

```
if (condição)
{ ação a ser realizada; }
```

Vejamos um exemplo:

instrucaoif.php

```
<?php
$cor = "azul";
if ($cor == "azul")
{ echo "A cor escolhida é azul <br>";
}
?>
```

Lembre−se:

- O uso das chaves é obrigatório quando realiza−se mais de uma ação, podendo as mesmas não serem utilizadas quando for realizada apenas uma ação.

3.2.2 Instrução if then else

Esta estrutura complementa o comando if, pois adiciona uma outra alternativa caso a condição não seja atendida. Sua sintaxe:

```
if (condição)
{ ação a ser realizada caso condição seja atendida; }
else
{ ação a ser realizada caso condição não seja atendida;}
```

Vejamos um exemplo:

instrucaoifthenelse.php

```
<?php
$nome = "Rafael";
if ($nome == "Rafael")
{ echo "Seu nome é Rafael";
```

```
}
else
{ echo "Você não se chama Rafael!";
}
?>
```

3.2.3 Instrução Else if

Esta instrução é um simples aninhamento de instruções if após um else, fazendo assim com que obtenhamos inúmeros resultados acresentando novas condições após um primeiro teste de condição. Veja sua sintaxe:

```
if (condição1)
{ ação a ser realizada caso condição1 seja atendida; }
elseif (condição2)
{ ação a ser realizada caso condição1 não seja atendida
  e a condição2 seja atendida;}
else
{ ação a ser realizada caso condição1 e condição2 não
  sejam atendidas;}
```

Vejamos um exemplo:

instrucaoelseif.php

```
<?php
$nome = "Rafael";
if ($nome == "Rafael")
{ echo "Seu nome é Rafael";
}
elseif ($nome =="Akira"
{ echo "Seu nome é Akira";
}
else
{ echo "Você não se chama Rafael e nem Akira!";
}
?>
```

3.2.4 Instrução Switch

Esta instrução testa múltiplas condições, evitando uso de repetitivos comandos ifthenelse. Sua sintaxe:

```
switch (condição)
{
 case resultado1:
  ação(ões) a ser(em) realizada(s);
  break;

 case resultado2:
  ação(ões) a ser(em) realizada(s);
  break;
   .
   .
   .

 default:
  ação(ões) a ser(em) realizada(s);
 }
```

Vejamos um exemplo:

estruturaswitch.php

```
<?php
$nome = "Rafael";
switch ($nome)
{
        case "Rafael":
        echo "Seu nome é Rafael";
        break;
        case "Akira":
        echo "Seu nome é Akira";
        break;
        case "João":
        echo "Seu nome é João";
        break;
        default:
        echo "Seu nome não está cadastrado!";
}
?>
```

Lembre−se:

- Nunca se esqueça de iniciar e terminar a instrução switch com chaves.
- É muito importante não se esquecer de colocar o ":" no final de cada resultado ou após o default.
- O comando "break;" evita que o processo continue através do resto da instrução switch.
- O resultado "default" apenas será acessado se nenhum outro resultado for atendido. Ele não necessita do comando "break".

3.3 Loops

Os loops, mais conhecidos como laços, são um forma convencional de programação que evitam que o programador fique repetindo inúmeras vezes em seu código um mesmo comando para realizar uma única ou inúmeras tarefas. O loop não para até que uma certa condição seja alcançada.

3.3.1 While

Este loop pode ser usado quando queremos executar algum bloco de comandos baseado na avaliação de uma determinada expressão lógica, ou seja, enquanto a expressão continuar sendo verdadeira, o laço continua a ser executado. Sua sintaxe:

```
while (condição)
{
  comandos;
}
```

Dentro do bloco de comandos, deve−se implementar algum comando que possa modificar, ou não, a condição imposta no comando while com o objetivo que o loop em algum momento chegue ao fim. Vejamos um exemplo:

loopwhile.php

```
<?php
$contador = 0; // Inicalização do contador
while($contador <=10)
{
        echo "O valor atual da variável 'contador' é " .$contador;
        $contador++; // Incremento do contador
        //Quando $contador chegar a 11, o laço será encerrado
}
?>
```

3.3.2 Do While

Do While é praticamente igual ao While, sendo que o bloco de comandos é executado ao menos uma vez antes da condição ser testada. Sua sintaxe:

```
do{
comandos;
}while (condição);
```

Vejamos um exemplo:

loopdowhile.php

```
<?php
$contador = 0;
do {
        echo "O valor atual da variável 'contador' é " .$contador;
        $contador++;
        //Este comando será executado uma vez mesmo que a
        //condição em while seja falsa

} while ($contador <=10)
?>
```

3.3.3 For

O for tem a mesma função do while, mas sua sintaxe é mais direta, o que significa: obter o mesmo resultado utilizando um código menor, o que é sinal de eficiência. Sua sintaxe:

```
for(inicialização; condição; incremento ou decremento)
   {
    comandos;
   }
```

Vejamos o mesmo exemplo feito quando falamos do while, mas agora utilizaremos o for:

loopfor.php

```
<?php
for ($contador = 0; $contador <=10; $contador++;)
{
        echo "O valor atual da variável 'contador' é: " .$contador;
}
?>
```

Lembre−se:

- A não especificação dos campos do for acarretar em um loop infinito, o que pode não ser aconselhável em seu projeto. Exemplo: for (;;)
- O comando "break;" pode ser utilizado dentro de um laço para que o mesmo seja interrompido e encerrado, ou seja, no momento em que ele for interpretado, o laço é abortado imediatamente.
- Há também o comando "continue;" que tem a mesma função do "break;" com apenas um porém, após a interrupção do laço, o mesmo não é abortado, e sim, testado novamente.

---

4 Arrays

4.1 Introdução

Podemos pensar em arrays como uma variável que guarda inúmeras outras variáveis (elementos), sendo todas do mesmo tipo de dados e cada uma, identificada por um índice que serve para localizar cada elemento dentro do array. Lembre−se que tais índices podem ser vetoriais ou associativos, mas falaremos disto mais adiante.

4.2 Criação e inicialização de Arrays

Para a criação de um array, utilizamos a função *array* e podemos inicializá−lo imediatamente ou deixar esta tarefa para mais tarde. Vejamos o exemplo a seguir:

firstarray.php

```
<?php
$cores = array();
//inicializa um array vazio
$nomes = array("Rafael", "Akira");
//inicializa um array com dois elementos
echo $nomes[0];
// exibe na tela o elemento de índice 0 do array "nomes", no caso: "Rafael"
$cores[] = "Vermelho";
//adiciona um elemento no array "cores"
?>
```

Lembre−se:

- Os indices de arrays em PHP iniciam−se em 0, ou seja, se um array tem 10 elementos, o índice do primeiro será 0 e do último será 9

Como um array é normalmente utilizado para guardar inúmeros elementos, devemos utilizar métodos mais práticos para percorrermos seu conteúdo, como a utilização de loops. Em PHP há um tipo de loop especial para percorrer arrays, o *foreach*. Veja:

looparray.php

```
<?php
$linguagens = array("C++", "Java", "Perl", "Delphi", "PHP");
foreach ($linguagens as $valor) {
        echo "Linguagem: $valor <br>";
}
?>
```

Além deste novo loop pode−se utilizar qualquer outro tipo de laço, dependendo da preferência do programador, assim, há duas funções que vale a pena serem lembradas para realizar tais ações: *count($nome_do_array)* e *sizeof($nome_do_array)*, ambas retornam o número de elementos do array.

4.3 Os tipos de índices

Como já dito, um índice é o identificador da varíavel dentro de um array, até agora apenas utilizamos índices vetorias, ou seja, o índice é baseado em números (int) que começam do 0. Agora conheceremos os índices associativos, tais índices não são números, mas sim strings. Vejamos exemplos:

Agora vamos criar algumas associações:

indiceassociativo.php

```
<?php
$lista = array("nome1" => "Rafael", "nome2" => "Akira", "nome3" => "Renata", "nome4" => "Sílvia")
echo $lista["nome1"];
// imprime "Rafael"
echo $lista["nome4"];
// imprime "Sílvia"
echo $lista[0];
// imprime "Rafael"
echo $lista[3];
// imprime "Sílvia"
?>
```

Utilizando este tipo de índice temos a criação de dois conceitos: "chave"(key) e "valor"(value). Dizemos que a "chave" é o índice associativo e "valor" é a variável para a qual tal índice é associado. Em nosso exemplo anterior, cada "chave" (por exemplo: "nome1") é associada ao seu respectivo "valor" (por exemplo: "Rafael") através do operador "=>".

Pode−se realizar loops para percorrer arrays com índices associativos normalmente, uma maneira nova e prática que atribue cada elemento do array a uma variável $value. Este loop while é ligeiramente diferente do antes apresentado, mas também de fácil entendimento. Veja o exemplo abaixo:

looparray2.php

```
<?php
$lista = array("nome1" => "Rafael", "nome2" => "Akira");

while (list($key,$value) = each ($lista) )
{
        echo "$key: $value <br>";
}

?>
```

Lembre−se:

- Esta nova forma de usar o loop while também pode ser utilizada em arrays de índices vetoriais, mas com uma pequena alteração: não utiliza−se o "$key", veja:

```
while (list(,$value) = each ($nome_do_array) )
{
        echo "$key: $value <br>";
}
```

4.4 Funções para a manipulação de arrays

A linguagem PHP traz uma gama de funções para a manipulação de arrays, todas podem ser encontradas no site http://www.php.net, com suas respectivas explicações e exemplos. Agora iremos apresentar apenas algumas destas funções.

- *is_array($nome_do_array)* − retorna "true" ou "false" dependendo se a variável é ou não do tipo array, respectivamente.
- *in_array("valor", $nome_do_array)* − retorna "true" ou "false" dependendo se o "valor" está ou não dentro do array em questão, respectivamente.

- *asort($nome_do_array)* − ordena o array de acordo com seus valores mantendo sua associação com as chaves.
- *ksort($nome_do_array)* − igual asort(), mas ordena o array de acordo com suas chaves.
- *sort($nome_do_array)* − ordena o array de acordo com seus valores, mas não mantém suas relações com suas respectivas chaves.
- *array_merge($array1, $array2,...)* − une os arrays que se encontrarem entre parênteses.

Existem inúmeras funções para a manipulação de arrays e todas estão no site http://www.php.net. Para conhecimentos adicionais, procure informações sobre as funções: explode(), implode(), split(), array_pop(), array_push(), array_rand(), array_reverse(), array_shift(), dentre outras.

5 Funções em PHP

5.1 Introdução

Função nada mais é do que um bloco de código que somente é executado quando for solicitado. As funções funcionam como uma caixa preta onde inúmeras linhas de código podem ser guardadas. Ela representa uma forma simplificada de executarmos um mesmo trecho de código em diversos pontos do mesmo script.

5.2 Definição

Observe a sintaxe básica de uma função:

```
function nome_da_função(parâmetro1, parâmetro2,...)
     {
       Bloco de Comandos;
       return valor;
     }
```

Qualquer código PHP válido pode estar contido no interior de uma função. Em PHP não é necessário declarar qual será o valor de retorno da função, mas deve−se ficar atento para que os parâmetros utilizados sejam dos tipos necessários para que não ocorra nenhum erro dentro do bloco de comandos.

As funções podem ou não terem parâmetros sendo passados para as mesmas, assim como também podem ou não retornar um valor. Funções que não retornam valores são normalmente chamadas de procedimentos.

Lembre−se:

- Uma função ou procedimento pode ser chamada antes de ser declarada, mas isto não é aconselhável, pois pode atrapalhar o entendimento do código

---

5.3 Procedimentos sem passagem de parâmetros

São funções que não retornam valores e não recebem parâmetros, observe:

procedimento1.php

```
<?php
//declaração do procedimento EscreveMsg
function EscreveMsg()
{
        echo "Você aprendendo a utilizar funções em PHP!"
}
//agora iremos chamar o procedimento
EscreveMsg();
//imprime na tela "Você aprendendo a utilizar funções em PHP!"
?>
```

5.4 Procedimentos com passagem de parâmetros

São funções que não retornam valores e recebem parâmetros para sua execução, observe:

procedimento2.php

```
<?php
//declaração do procedimento EscreveMsg2
function EscreveMsg2($msg)
{
        echo $msg;
}
//agora iremos chamar o procedimento
EscreveMsg2("Curso PHP Total!");
//imprime na tela "Curso PHP Total!"
?>
```

5.5 Funções sem passagem de parâmetros

São funções que retornam valores e não recebem parâmetros, observe:

funcao1.php

```
<?php
//declaração da função DoisAoQuadrado()
function DoisAoQuadrado()
{
        return (2 * 2);
}
//agora iremos chamar a função
$resultado = DoisAoQuadrado();
echo "Dois ao quadrado é: $resultado";
//imprime na tela "Dois ao quadrado é: 4"
?>
```

5.6 Funções com passagem de parâmetros

São funções que retornam valores e recebem parâmetros para sua execução, observe:

funcao2.php

```
<?php
//declaração da função Quadrado
function Quadrado($numero)
{
        return ($numero * $numero);
}
//agora iremos chamar a função recebendo o valor 2
$Resultado = Quadrado(2);
echo "O quadrado de $numero é: $Resultado";
//imprime na tela "O quadrado de 2 é: 4"
?>
```

5.7 Passagem de parâmetros por referência

A passagem de parâmetros mostradas até agora são chamadas de "passagem por valor", o que significa que se o valor da variável for alterada dentro do bloco de comandos da função, fora dele, a variável não terá seu valor alterado. Por exemplo:

```
<?php
function Adiciona2($numero)
{
        $numero += 2;
}

$a = 3;
```

```
echo Adiciona2($a);
//imprime 5 na tela, mas $a continua valendo 3
?>
```

As funções trabalham com cópias das variáveis que lhe são passadas como parâmetros para preservar seus valores iniciais. Se a passagem de valor fosse feita por referência, a variável $a teria 5 como valor. Para realizar uma passagem por referência pode−se utilizar duas maneiras, em ambas utiliza−se o caractere "&", veja:

```
<?php
function Adiciona2(&$numero1, $numero2) {
        $numero1 += 5;
        $numero2 += 5;
}
$a = $b = $c = $d = 1;
echo " <br> Valor de 'a' e 'b' antes da chamada da funcao: <br> a = $a <br> b = $b";
Adiciona2($a, $b);
echo " <br> Valor de 'a' e 'b' depois da chamada da funcao: <br> a = $a <br> b = $b";
//só $numero1 terá seu valor alterado
echo " <br> Valor de 'c' e 'd' antes da chamada da funcao: <br> c = $c <br> d = $d";
Adiciona2($c, &$d);
echo " <br> Valor de 'c' e 'd' depois da chamada da funcao: <br> c = $c <br> d = $d";
//Ambas as variáveis terão seus valores alterados
?>
```

5.8 Arquivos de Inclusão

Com o tempo, você poderá ter criado inúmeras funções que podem ser reutilizadas em novos scripts, para que você não seja obrigado a declarar todas estas funções novamente em seu novo script, pode−se escrever no topo do mesmo a frase: *require (funcoes.php)*, sendo que, no script funcoes.php devem estar declaradas todas as funções que você utilizará.

Lembre−se:

- O script funcoes.php utilizado junto ao require é um nome fictício, ficando a critério do programador nomear seu script de funcoes
- Se o script de funções não se encontrar no mesmo diretório que nossos scripts normais, deve−se especificar a localização completa do script. Por exemplo: require ("/home/httpd/html/php/funcoes.php");

---

6 Manipulação de Arquivos

6.1 Introdução

A manipulação de arquivos é uma ferramenta muito importante da linguagem PHP, pois às vezes não temos em nossa disposição um banco de dados, ou então, queremos apenas guardar informações em arquivos temporários e logo após, sermos capazes de lê−los e editá−los através do nosso próprio script PHP.

6.2 Abrindo e fechando arquivos

Sempre que trabalhamos com arquivos devemos realizar no mínimo duas operações: sua abertura e fechamento. Para abrir um arquivo, utilizamos a função fopen() e para fechar fclose(), vejamos cade um deles:

6.2.1 Fopen

Esta é a função utilizada para abrir um arquivo ou para criá−lo caso ele já não tenha sido, vejamos sua sintaxe:

fopen(filename, mode);

- filename: pode ser simplesmente o nome do arquivo ou o caminho completo para o mesmo.
- mode: especifica o modo de abertura, ou seja, se o arquivo deve ser aberto para leitura, escrita, etc. Vejamos tais modos:

Modo	Descrição
a	abre o arquivo para anexar dados, posiciona o ponteiro no final do arquivo − se o arquivo não existir, será criado um novo
a+	abre o arquivo para anexo/leitura, posiciona o ponteiro no final do arquivo − se o arquivo não existir, será criado um novo
r	abre o arquivo para leitura e posiciona o ponteiro no início do arquivo − o arquivo deve existir
r+	abre o arquivo para leitura/escrita, posiciona o ponteiro no início do arquivo − se o arquivo não existir, um novo será criado
w	abre o arquivo no modo somente escrita − se o arquivo já existir, será sobrescrito, senão, será criado um novo
w+	abre o arquivo para escrita/leitura − se o arquivo já existir, será sobrescrito, senão, será criado um novo
b	abre o arquivo como um binário

Ao abrir um arquivo, deve−se atribuir o resultado desta operação à uma variável, pois tal variável pode funcionar como um "ponteiro" na hora de fechar o arquivo e também pode ser útil para a verificação de abertura do arquivo.

```
$arq_aberto = fopen("/tmp/arq.txt","w");
/* $arq_aberto será 0 se o arquivo não abrir ou será um número
inteiro positivo caso o arquivo seja aberto com sucesso*/
```

Veja um exemplo para realizar a verificação de abertura de arquivo:

verifabertura.php

```
<?php
if (!$arq_aberto = fopen("/tmp/arq.txt","w"))
{
        echo "Arquivo não foi aberto!";
```

```
        exit;
}
?>
```

Lembre−se:

- O operador "!" no exemplo acima, é usado para testar se $arq_aberto contém um valor positivo (true) ou o valor 0 (false)
- Caso o arquivo não seja aberto, junto com a impressão de erro, também será impresso o erro interno da linguagem PHP. Para contornar isto utiliza−se o símbolo "@" antes do início da instrução condicional ou antes da chamada de fopen:

```
if (@!$arq_aberto = fopen("/tmp/arq.txt","w"))
 ou
if (!$arq_aberto = @fopen("/tmp/arq.txt","w"))
```

6.2.2 Fclose

Fechar um arquivo é muito simples, deve−se apenas utilizar o "ponteiro" concebido no momento de abertura do arquivo, junto com a função fclose:

fclose($arq_aberto);

6.3 Gravação em um arquivo

Para a gravação de dados em um arquivo pode−se utilizar as funções: *fwrite()* ou *fputs()*, ambas funcionam da mesma maneira, veja o exemplo:

gravacao.php

```
<?php
$linha = "Linha adicional!";
//verifica se o arquivo foi aberto corretamente
if (!$arq_aberto = fopen("/tmp/arq.txt","w"))
{
        echo "Arquivo não foi aberto!";
        exit;
}
//vamos agora inserir linhas
fputs($arq_aberto, "Primeira linha do meu arquivo\n");
fwrite($arq_aberto, "Segunda linha do meu arquivo\n");
fwrite($arq_aberto, $linha);
//não vamos nos esquecer de fechar o arquivo
fclose($arq_aberto);
?>
```

6.4 Leitura de um arquivo

Para ler as informações de um arquivo pode−se utilizar inúmeras funções, elas são:

- fread() e fgets(): Estas funções permitem ler strings gravadas em um arquivo com uma pequena diferença entre elas. Veja o exemplo:

leitura1.php

```
<?php
```

```
$arq_aberto = fopen("/tmp/arq.txt","w");
$text = fread($arq_aberto, 255);
//lê 255 bytes do arquivo
$text2 = fgets($arq_aberto, 255);
//lê 255 bytes do arquivo ou até o final da linha
echo $text;
echo $text2;
fclose($arq_aberto);
?>
```

- fgetc(): Permite ler caractere por caractere de um arquivo. Veja o exemplo:

leitura2.php

```
<?php
$arq_aberto = fopen("/tmp/arq.txt","w");
do {
        $char = fgetc($arq_open);
}
while($char)
/* coloca cada caractere do arquivo em $char,
até o final do conteúdo do arquivo */
fclose($arq_aberto);
?>
```

- file(): Esta função lê um arquivo completo, e armazena cada linha do arquivo como um elemento de um array. Depois de ler todo o conteúdo do arquivo, file() o fecha automaticamente, não sendo necessária uma chamada a fclose(), veja o exemplo:

leitura3.php

```
<?php
$linhas = file("/tmp/arq.txt");
//$linhas receberá cada linha do arquivo como um elemento do array
echo "Primeira linha: " .$linhas[0];
echo "Segunda linha: " .$linhas[1];
?>
```

6.5 Bloquear arquivos

As vezes é necessário limitar o acesso a um arquivo, para que o mesmo possa não se corromper se for acessado por muitos usuários ao mesmo tempo, para isto, podemos bloquear o acesso à arquivos utilizando a função *flock()*. Seu fomato básico é:

flock("ponteiro", operação);

São aceitas 3 operações:

Operação	Descrição
1 − Compartilhado	usuários podem ler, mas ninguém pode gravar
2 − Exclusivo	nenhum outro usuário pode ler ou gravar até o arquivo ser liberado
3 − Liberação	libera qualquer operação antes definida

Vejamos o exemplo:

bloqueio.php

```
<?php
$arq_aberto = fopen("/tmp/arq2.txt","w");
```

```
flock($arq_aberto,2);
//bloqueia o arquivo logo após a sua abertura
fputs($arq_aberto, "Estou escrevendo no arquivo!");
//o arquivo está bloqueado para outros usuários
flock($arq_aberto,3);
//libera o acesso ao arquivo
fclose($arq_aberto);
?>
```

6.6 Utilizando loops para acessar arquivos

Novamente utilizamos os loops para nos poupar de algum trabalho manual, agora queremos evitar de acessar linha por linha de um arquivo, para isto, utilizamos uma nova função que verifica se chegamos ao final de um arquivo, a função *feof()*. Vejamos um exemplo de seu uso:

looparquivo.php

```
<?php
if (!$arq_aberto = fopen("/tmp/arq3.txt","r"))
{
        echo "Arquivo não foi aberto!";
        exit;
}
else
{
        while (!feof ($arq_aberto))
        {
                $linha = fgets($arq_aberto, 255);
                echo "$linha <br>";
        }
        fclose($arq_aberto);
}
?>
```

6.7 Funções diversas para a manipulação de arquivos

Apresentaremos agora algumas funções para um maior entendimento do assunto e também para apresentar sugestões de como utilizar os arquivos em PHP. Volto a lembrar que todas estas funções podem ser encontradas no manual da linguagem PHP em http://www.php.net.

6.7.1 rewind()

Esta função redefine a posição corrente para o início do arquivo, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: rewind($arq_aberto);

6.7.2 fseek()

Esta função move o cursor para um posição específica no arquivo, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: fseek($arq_aberto,*offset*); sendo *offset* o nº de bits dese o início do arquivo até a posição desejada.

6.7.3 copy()

É a função utilizada para copiar arquivos de um local (origem) para outro (destino), retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: copy("/root/file.txt", "/temp/file.txt"); − copia o arquivo file.txt de "/root/" para "/temp".

6.7.4 unlink()

Esta é a função utilizada para excluir um arquivo criado, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: unlink ("/temp/file.txt");

6.7.5 rename()

Altera o nome de um arquivo, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: rename("oldfile.txt", "newfile.txt");

6.7.6 file_exists()

Como diz o próprio nome, esta função verifica se o arquivo realmente existe, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: file_exists("newfile.txt");

6.7.7 filesize()

Esta função retorna em bytes o tamanho do arquivo especificado, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: filesize("newfile.txt");

6.7.8 filetype()

Esta função retorna uma string com o tipo do arquivo especificado, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: filetype("newfile.txt");

6.7.9 is_dir()

Verifica se o arquivo em questão é um diretório, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: is_dir("file.txt");

6.7.10 is_executable()

Verifica se o arquivo em questão é um executável, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: is_executable("file.txt");

6.7.11 is_readable()

Verifica se no arquivo é permitido a leitura, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: is_readable("file.txt");

6.7.12 is_writeable()

Verifica se no arquivo é permitido a escrita, retornando TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: is_writeable("file.txt");

6.8 Diretórios

Apesar de não ser muito comum, PHP também nos oferece inúmeras funções para a manipulação de diretórios, vejamos algumas:

6.8.1 chdir()

Esta função transforma o diretório especificado no diretório corrente, ou seja, aquele em que a página propriamente dita reside. Exemplo: chdir("/temp");

6.8.2 opendir()

Função usada para abrir o diretório, retornando um número inteiro que podemos chamar de "manipulador de diretório". Exemplo: $dirmanip = opendir("/temp");

6.8.3 readdir()

É um função utilizada para ler o conteúdo do diretório, ela lista o primeiro registro dentro do mesmo. Esta função usa como parâmetro o "manipulador de diretório" apresentado anteriormente e retorna TRUE caso tenha sucesso e FALSE no caso de falha. Exemplo: $file = readdir($dirmanip);

6.8.4 closedir()

Função usada para fechar o diretório utilizando como parâmetro o "manipulador de diretório". Exemplo: closedir("/temp");

6.8.5 mkdir()

Função usada para criar um novo diretório Exemplo: mkdir("/temp/new", 0700); − note que o segundo parâmetro especifica as permissões de acesso para um diretório UNIX, sendo ignorado no Windows.

6.8.6 rmdir()

Simplesmente remove o diretório. Exemplo: rmdir("/temp/new");

6.9 Exemplo Prático − Contador de Acessos

Para uma melhor utilização e entendimento das funções manipuladoras de arquivos, tentem estruturar um script para controlar os acessos à uma página de intenet utilizando arquivos. Abaixo mostraremos um modelo que pode ser tomado como exemplo:

contador.php

```
<?php
//vamos criar uma função que conte os acessos
function contador()
{
        #Esta variavel conterá o caminho completo na árvore de diretórios e o nome do auquivo que
        $arq_cont = "/var/www/contador.txt";
        #verifica se o arquivo que irá conter o nº de acessos existe
        if (file_exists($arq_cont))
        {
                $arq_aberto = fopen($arq_cont, "r+");
                #apenas um usuário poderá acessar o arquivo por vez
                flock($arq_aberto, 2);
                #coloca na variável os 4 primeiros dígitos do arquivo
                $contador = fgets($arq_aberto, 4);
                $contador++;
                rewind($arq_aberto);
                #atualiza o valor do contador no próprio arquivo
                fputs($arq_aberto, $contador);
```

```
                #libera o acesso ao arquivo
                flock($arq_aberto, 3);
                fclose($arq_aberto);
                echo $contador;
                #este "else" só ocorrerá caso o arquivo não tenha sido criado
        } else {
                $arq_aberto = fopen($arq_cont, "w");
                #inicia o contador com o valor 1 para que depois ele seja incrementado
                $contador = "1";
                flock($arq_aberto, 2);
                fputs($arq_aberto, $contador);
                flock($arq_aberto, 3);
                fclose($arq_aberto);
                echo $contador;
        }
}
contador();
?>
```

Agora seria necessário apenas utilizar a função contador onde você bem quiser na sua página web lembrando de utilizar o comando: require("contador.php");

7 Manipulação de Strings e Expressões Regulares

7.1 Introdução

Com certeza você irá precisar tratar as strings de seus scripts em algum ponto, para isso, a linguagem PHP também fornece inúmeras funções. Neste capítulo iremos aprender algumas destas funções e iremos explorar expressões regulares.

7.2 Manipulação de Strings

Ao programar em PHP é indispensável a manipulação de string, pois através dela fazemos as devidas validações de dados inseridos pelos usuários, evitando assim a entrada de dados que não são válidas para o sistema. O PHP oferece diversas funções que podem nos ajudar a processar uma cadeia de caracteres ou retornar mensagens de confirmação ou erro ao usuário.

Utilizando estas funções para tratar as cadeias de caracteres podemos realizar as tarefas de manipulação de string de maneira mais fácil e eficiente. Algumas das funções mais utilizadas são as seguintes:

7.2.1 substr()

Retorna uma parte de uma cadeia de caracter, utilizamos esta função fornecendo dois ou três argumentos: o primeiro argumento *string* é a cadeia original a ser analisada, o segundo argumento *início* é a posição de início do retorno da substring, e o terceiro argumento *tamanho* (opcional) determina o tamanho da cadeia que será retornada, caso não seja informado será retornado a substring até o final da cadeia original.

substr (string, início);

substr (string, início, tamanho);

exemplo 1:
Temos uma string que possui a seguinte cadeia de caracteres “Sistemas Abertos”, e queremos retornar apenas “Abertos”

```
<?php
     $nome=”SistemasAbertos”;
     echo substr($nome, 8);
?>
```

O resultado deste código no navegador é:

`Abertos`

exemplo 2:
Temos uma string que possui a seguinte cadeia de caracteres “Sistemas Abertos”, e queremos apenas a cadeia “Sistemas”

```
<?php
     $nome=”SistemasAbertos”;
     echo substr($nome, 0, 8);
?>
```

O resultado deste código no navegador é:

`Sistemas`

Se utilizarmos o segundo ou terceiro argumento como número negativo, a contagem será feita do final da cadeia para o início da string original.

exemplo 3:
O código abaixo retorna uma string de 6 caracteres começando com o sétimo caracter de trás para frente da string original.

```
<?php
     $nome=”SistemasAbertos”;
     echo substr($nome, −7, 6);
?>
```

Resultado no navegador:

Aberto

exemplo 4:
O código abaixo retorna uma string com início na sétima posição de trás para frente até a penúltima posição.

```
<?php
     $nome=”SistemasAbertos”;
     echo substr($nome, −7, −1 );
?>
```

Resultado no navegador:

Aberto

Lembre−se:

- Ao utilizar a função substr() e não informar o terceiro argumento, o retorno da cadeia será feito a partir da posição de início até o final da string.

7.2.2 strlen()

Retorna um número inteiro que informa o comprimento da cadeia, ou seja a quantidades de caracteres da string. Esta função possui apenas um argumento, a própria cadeia.

strlen (string);

exemplo 5

```
<?php
     $curso=”Expressões Regulares”;
     echo “A cadeia possui “.strlen($curso).” caracteres<br>”;
     echo “Sistemas Abertos possui “.strlen(“Sistemas Abertos”).” caracteres”;
?>
```

O resultado do código acima é:

```
A cadeia possui 20 caracteres
Sistemas Abertos possui 16 caracteres
```

7.2.3 trim()

Retira os espaços em branco no início e final de uma cadeia de caracteres, esta função não exclui os espaços em branco no meio da cadeia. O argumento que esta função recebe é a própria cadeia.

trim(string);

exemplo 6

```
<?php
     $linguagem="   Hypertext Preprocessor   ";
     $linguagem_limpa=trim($linguagem);
     echo "início:".$linguagem."fim.<br>";
     echo "início".$linguagem_limpa."fim.";
?>
```

O resultado do código acima é:

início: Hypertext Preprocessor fim.
inícioHypertext Preprocessorfim.

Para limpar os caracteres brancos apenas do início da string , usamos o ltrim().
E para limpar apenas os caracteres brancos do final da string usamos o chop().

7.2.4 ucfirst()

Retorna uma string com o primeiro caracter em maiúsculo. O argumento da função é a cadeia a ser alterada.

ucfirst(string);

exemplo 7

```
<?php
     $name = "daniella balbino";
     $name = ucfirst($name);
     echo "O nome é ".$name;
?>
```

O resultado deste código é:

O nome é Daniella balbino

7.2.5 ucwords()

Utilizada em strings que possuem mais de uma palavra, retorna os primeiros caracteres de cada palavra em maiúsculo.

ucwords(string);

exemplo 8

```
<?php
     $name = "daniella balbino";
     $name = ucwords($name);
     echo "A função ucwords retorna ".$name;
?>
```

O resultado do código é: **A função ucwords retorna Daniella Balbino**

7.2.6 strpos()

Utilizada para fazer pesquisa de uma palavra dentro de uma cadeia. A função retorna a posição da primeira ocorrência da combinação de caracteres referentes a palavra recebida como argumento. O primeiro argumento é a cadeia de caracter onde será efetuada a pesquisa, e o segundo argumento é a palavra (sequência de caracteres) que será pesquisada na cadeia.

7.2.7 strrpos()

Retorna a posição da última ocorrência da palavra pesquisada.

strpos(string, substring) strrpos(string,substring)

exemplo 9

```
<?php
     $mensagem=”Daniella não conseguiu passar no teste,
     ela não atingiu a nota mínima”;
     echo strpos($mensagem, “não”); //retorna a posição do primeiro “não”
     echo strrpos($mensagem, “não”); //retorna a posição do último “não”
?>
```

7.2.8 str_replace()

Utilizada para fazer a substituição de uma sequência de caracteres(palavra). Esta função recebe três argumentos: palavra procurada, palavra que será escrita na cadeia principal e a cadeia principal.

str_replace(palavra procurada, nova palavra, cadeia principal);

Para transformar a cadeia de caracteres “Daniella foi reprovada no teste” para “Daniella foi aprovada no teste”, utilizamos a função str_replace().

exemplo 10

```
<?php
     $resultado=”Daniella foi reprovada no teste”;
     $resultado=str_replace(“reprovada”,”aprovada”,$resultado);
     echo $resultado;
?>
```

O resultado do código acima no navegador é:

Daniella foi aprovada no teste

7.2.9 chr()

Recebe como argumento um número inteiro, e retorna o caracter correspondente a este número da tabela ASCII.

chr(número que representa um código ASCII);

exemplo 11

```
<?php
```

```
     echo chr(34);
     echo chr(64);
     echo chr(34);
?>
```

O resultado será: “@”

7.2.10 strcmp()

Compara duas cadeias de caracteres, retorna 0 se as duas strings são exatamente iguais (a comparação é feita em case sensitive), retorna < 0 se a primeira string é maior que a segunda e retorna > se a segunda string é maior que a primeira.

strcmp(string1, string2);

exemplo 12

```
<?php
     $senha=”123456”;
     $acesso=”123456”;
     $resultado=(strcmp ($senha, $acesso));
     if ($resultado==0)
          echo “Senha correta”;
     else
          echo “Senha inválida”;
?>
```

O resultado deste código no navegador é:

Senha correta

7.2.11 strcasecmp()

Compara duas cadeias de caracteres da mesma forma que a função strcmp(), exceto pelo fato de desconsiderar o case sensitive(diferenças entre caracteres maiúsculos e minúsculos).

strcasecmp(string1,string2);

exemplo 13

```
<?php
     $nome1=”Carlos José”;
     $nome2=”carlos José”;
     $resultado= strcasecmp ($nome1, $nome2);
     if ($resultado==0)
          echo “Os nomes são iguais”;
     else
          echo “Os nomes não são iguais”;
?>
```

O resultado no navegador é:

Os nomes são iguais

7.2.12 ereg()

Utilizada para procurar em uma cadeia de caracter uma certa combinação de padrão(expressão regular) Verifica se a $variavel é igual a expressão regular definida em $expressao em um modo sensível a distinção

de caracteres (case sensitive). Esta função retorna verdadeiro se a variável estiver nos padrões da expressão e falso se a variável não estiver nos padrões.

ereg(“expressão”, $cadeia,);

exemplo 14

```
<?php
     $cadeia=”Para manipular strings em cadeia de caracteres é preciso estudar muito”;
     $expressao=”estudar”;
     if(ereg($expressao,$cadeia))
          echo”Foi encontrado ocorrência da expressao na cadeia”;
     else
          echo”Não foi encontrado ocorrência da expressao na cadeia”;
?>
```

O resultado no navegador é´:

Foi encontrado ocorrência da expressão na cadeia

7.2.13 ereg_replace()

Essa função busca na terceira string recebida como argumento resultados para a expressão determinada no primeiro argumento. Substituindo a sequência de caracteres que estiver nos padrões da expressão, pela string recebida no segundo argumento. Se não for encontrado uma sequência de caracteres que esteja de acordo com os padrões da expressão, o resultado será a $variável inalterada.

ereg_replace ($expressao, $substituicao, $cadeia)

exemplo 15

```
<?php
     $cadeia=”Para manipular string em cadeia de caracteres é preciso estudar muito”;
     $expressão=”muito”;
     $nova_expressão=”bastante”;
     echo ereg_replace($expressao, $nova_expressão, $cadeia);
?>
```

O resultado desta execução de código no navegador é:

Para manipular strings em cadeia de caracteres é preciso estudar bastante

7.2.14 split()

Utilizada para transformar uma cadeia de caracteres em um array, para isto é necessário determinar qual caracter será utilizado para dividir os campos do array.

split($limite, $cadeia);

exemplo 16

```
<?php
     $cadeia=”Para manipular strings em cadeia de caracteres é preciso estudar muito”;
     $limite=” “;
     $vetor=split($limite, $cadeia);
     foreach ($vetor as $valor)
          echo $valor.”<br>”;
?>
```

O resultado deste código no navegador é:

Para manipular strings em cadeia de caracteres é preciso estudar muito

7.3 Expressões Regulares

Agora que sabemos as principais funções de manipulação de strings, vamos aprender as expressões regulares que serão muito úteis para padronizar o conteúdo de variável literal. Este padrão determina os caracteres utilizados em sua formação a quantidade de ocorrência destes caracteres.

A expressão regular é um método formal de se especificar um padrão de texto, ou seja, é uma composição de símbolos, caracteres com funções especiais que, agrupados entre si e com caracteres literais, formam uma sequência (expressão). Essa expressão é interpretada com uma regra, que indicará sucesso se uma entrada de dados qualquer combinar com essa regra, ou seja, obedecer exatamente a todas as suas condições.

As expressões são formadas por símbolos e caracteres literais chamados metacaracteres, que possuem funções especiais.

7.3.1 Metacaracteres tipo Representante . [ ] [^]

Utilizado para representar um ou mais caracteres.

. ponto – representa qualquer caracter, pode ser um número, uma letra, caracteres especiais (@, %, &, *, e outros). Podemos utilizar o ponto em um texto para procurar palavras, da seguinte forma:

```
express.o        expressão, expressao, expressAo, ...
.epresentante    Representante, representante, ...
pont.            ponto, ponta, ...
16.30            16:30, 16 30, 16.30, ...
<..>             <br>, <tr>, <td>, <th>, ...
```

[ ] lista utilizada para limitar os metacaracteres que serão permitidos na expressão, ou seja, só é possível utilizar os metacaracteres especificados na lista, veja os seguintes exemplos:

```
ling[uü]iça      linguiça, lingüiça
[Bb]rasil        Brasil, brasil
16[:. ]30        16:30, 16.30, 16 30
```

Dentro da lista todos os caracteres são normais, exceto o traço (−). Portanto o ponto é apenas um ponto e não um metacaracter.

A lista contém todos os caracteres permitidos em uma posição. Uma lista que determina que em uma posição exista apenas números é a seguinte: [0123456789], agora imagine como seria uma lista para representar uma letra do alfabeto.

Usamos o conceito de intervalo para determinar os intervalos existentes, desta forma podemos determinar os intervalos da seguinte forma:

```
[0−9]   números de 0 a 9
[a−z]    letras minúsculas de a a z
[A−Z]   letras maiúsculas de A a Z
[a−zA−z]   letras maiúsculas e minúsculas de A a Z
```

Para representar o traço(−) literal em uma lista, é necessário colocá−lo no final da lista. Temos também o colchete literal que são os delimitadores da lista, os colchetes que abre ([ ), podemos colocar em qualquer lugar da lista, pois ela já está aberta e não podemos ter uma lista dentro de outra lista. Agora o colchete que

fecha ( ]) deve ser colocado no começo da lista, ou seja, ser o primeiro item da lista, para não ser confundido com o colchete que termina a lista.

```
[][-]    representa os caracteres literais dos seguintes símbolos: ], [, -
```

[^] lista negada utilizada para representar os caracteres literais que não fazem parte da posição, as regras que são aplicadas na lista normal são válidas à lista negada. A diferença é que na lista negada é especificado os elementos que não podem ser representados e o primeiro caracter da lista é um circunflexo, ele indica que esta é uma lista negada.

A lista negada é útil quando sabemos o que não pode existir em determinada posição. Por exemplo, de acordo com a regra de boa escrita após caracter de pontuação, devemos colocar um espaço em branco separando do resto do texto. Então para procurar por qualquer coisa que não o espaço após a pontuação, utilizamos a seguinte expressão: [:;,.!?][^]

7.3.2 Metacaracteres tipo Quantificadores ? * + { }

São utilizados para indicar o número de repetições permitidas para entidades imediatamente anterior, ou seja, a quantidade de repetições que a entidade anterior pode aparecer. As entidades podem ser um caracter ou um metacaracter.

? opcional é um quantificador utilizado para ter ou não a ocorrência da entidade anterior, o opcional(?) repete o caracter ou metacaracter anterior 0 (zero) ou 1 (uma) vez.

```
Programas?      Programa, Programas
aula [1-5]?      aula, aula 1, aula 2, aula 3, aula 4, aula 5
brincadeira[s!]?   brincadeira, brincadeiras, brincadeira!
```

**** asterisco*** – usado para ter, não ter, ou ter vários, infinitos caracteres da entidade anterior, em outras palavras o (*) indica que a entidade anterior pode aparecer em qualquer quantidade.

```
[ar]*a   a, ara, arara, ararara, ra, raaa, rrrra, ...
Humm*!   Hum!, Humm!, Hummm!, Hummmmm!
```

Verificamos até agora dois metacaracteres extremamentes abrangentes, como o ponto (qualquer caracter) e o asterisco (em qualquer quantidade). Se juntarmos os dois .* teremos qualquer caracter em qualquer quantidade, é o que chamamos de curinga pois representa tudo ou nada. O nada por que “qualquer quantidade” também é igual a “nenhuma quantidade”, e o tudo pois “qualquer quantidade” também é igual a “tudo o que vier”.

+ *mais* é um quantificador utilizado para exigir a ocorrrência da entidade anterior pelo menos uma vez, podendo repetir a quantidade de vezes que quiser. É muito parecido com o asterisco (*), a única diferença é que o mais (+) não é opcional, exigindo o mínimo uma repetição, ele se comporta como um asterisco mais exigente.

```
[ar]+a   aa, ra, rraa, rra, ...
Humm+!   Humm!, Hummm!, ...
```

{n,m} chaves as chaves é a solução para uma quantificação mais controlada, onde podemos especificar exatamente quantas repetições se quer da entidade anterior, {n,m} significa de n até m vezes, temos também a sintaxe {n} que podemos omitir a quantidade final, ou ainda, especificar exatamente o número de repetições.

```
{1,5}   de 1 a 5
{5,}   pelo menos 5 (5 ou mais)
{0,5}   nenhuma ou até 5
{5}   exatamente 5
{0,1}   zero ou 1 (igual ao opcional)
```

```
{0,}   zero ou mais (igual ao asterisco)
{1,}   um ou mais (igual ao mais)
```

7.3.3 Metacaracteres tipo Âncoras ^$ \b

Utilizados para marcar uma posição específica na linha. Eles não podem ser quantificados, então os metacaracteres tipo quantificadores (+ * { }) não exercem influência sobre eles.

^ ***circunflexo*** este metacaracter marca o começo de uma linha, ele só tem esta função se estiver no início da expressão regular.

Lembre−se:

- O circunflexo como primeiro elemento de uma lista é o marcador da lista negada, fora dela ele é âncora que determina como a linha deve ser iniciada.

```
^[*#]
```

Isto quer dizer: a partir do começo da linha, combine (* ou #), procuramos linhas que começam com * ou #. Ele é muito utilizado para procurar palavras no começo da linha.

$ cifrão o cifrão marca o fim de uma linha e só é válido no final de uma expressão regular.

b borda – utilizada para marcar uma borda de palavra, ou seja, ela marca os limites de uma palavra (onde ela começa e/ou termina). Veja como se comportam as expressões regulares nas palavras dia, diafragma, radial, melodia, bom−dia!:

```
dia       dia, diafragma, radial, melodia, bom−dia!
\bdia     dia, diafragma, bom−dia!
dia\b     dia, melodia, bom−dia!
\bdia\b   dia, bom−dia!
```

Portanto a borda força um começo ou terminação de palavra. Palavra aqui é um conceito que engloba apenas[A−Za−z0−9_], ou seja, letras, números e sublinhado. Por isso \bdia\b também combina com bom−dia! pois o traço e a exclamação não são parte de uma palavra.

7.3.4 Outros Metacaracteres \c | ( )

Chamamos de outros os metacaracteres representados aqui, por que eles não possuem uma função específica e não relacionam entre si. Mas isto não quer dizer que eles são inferiores, pois são muito utilizados aumentando as possibilidades de uso das expressões regulares.

\ ***escape*** utilizado para transformar um metacaracter em caracter literal, ou seja, fazer com que os metacaracteres não se comporte de acordo com suas funções, ou melhor sejam representados apenas pelo seu significado literal. Seria a mesma coisa que colocar este metacaracter em uma lista, pois de acordo com o conceito de lista, todos os caracteres possuem valor literal(exceto o traço).

Um exemplo de expressão para representar o número de RG:

```
   [0−9]\.[0−9]{3}\.[0−9]{3}−[0−9]

\*   caracter literal * (mesmo que [*] )
\^   caracter literal ^ (mesmo que [^] )
\?   caracter literal ? (mesmo que [?] )
```

*| ou * representado pela barra vertical |, serve para validar mais de uma alternativa. Quando em uma posição específica da expressão podemos combinar mais de uma alternativa, utilizamos o ou para determinar as combinações possíveis. Por exemplo:

```
boa−tarde | boa−noite
```

Lemos essa expressão regular da seguinte forma: “boa−tarde ou boa−noite”. A lista se comporta de maneira semelhante ao ou, mas apenas para uma letra, então:

```
[bcdgp]ata      bata, cata, data, gata, pata
```

(...) grupo responsável por juntar vários sujeitos em um mesmo local, dentro de um grupo podemos ter um ou mais caracteres, metacaracteres e inclusive outros grupos. Assim como em uma expressão matemática, os parênteses definem um grupo, e seu conteúdo pode ser visto como um bloco na expressão.

Os metacaracteres quantificadores tem seu poder ampliado pelo grupo, pois ele lhes dá mais abrangência. Veja alguns exemplos:

```
(ai)+   ai, aiai, aiaiai, aiaiaiai, ...
(10)+   10, 1010, 101010, 10101010, ...
```

Nestes exemplos, combinamos várias repetições da palavra (ou sequência) especificada dentro do grupo.

Podemos também criar subgrupos, por exemplo vamos procurar o nome de um supermercado em uma listagem e não sabemos se este é um mercado, supermercado ou um hipermercado, para isto utilizaremos a seguinte expressão:

```
((su|hi)per)?mercado
```

***1 ... *9 retrovisor** chamamos de retrovisor por que é possível buscar um trecho já casado, muito útil para casar trechos repetidos em uma mesma linha. Ele é representado pelo escape \ seguido dos números de 1 a 9 (podemos ter no máximo 9 retrovisores na expressão) , os números indicam o grupo que está sendo referenciado. Por exemplo vamos procurar literalmente por :

```
(bom)\1         bombom
(quero)−\1      quero−quero
```

Então o retrovisor \1 é uma referência a combinação especificada no primeiro grupo, nesses casos foram repetidos as palavras encontradas no primeiro grupo (bom e quero).

8 Formulários HTML

8.1 Introdução

Para aqueles que desejam prosseguir na programação web, aprendendo posteriormente linguagens de script cliente e servidor, devem entender os formulários e suas funcionalidades.

Os formulários são essenciais para que haja uma interação entre o usuário que acessa nossa página. Isso quer dizer que os elementos declarados dentro da tag <form>, são a principal fonte de feedback ou entrada de dados da máquina cliente, que serão armazenados no servidor e interpretados pela linguagem PHP.

Lembre−se que neste capítulo apenas falaremos sobre os formulários PHP em si, deixando para o próximo capítulo o processamento dos mesmos.

8.2 Entendendo o funcionamento dos formulários

Como adiantamos na introdução deste capítulo, a tag <form> é a responsável pela declaração de uso de um formulário dentro de nossa página. Dentro desta tag serão inseridos todos os objetos que farão a interação com o usuário.

A tag <form> possui algumas propriedades importantes que devemos entender:

- ***action*** esta propriedade define qual programa ou script deverá analisar e processar os dados que serão enviados para o servidor, é aqui que deve−se especificar o script PHP a interpretar as variáveis provindas do formulário.
- ***method*** os métodos de envio podem ser dois:
    - ***get*** a diferença básica entre esses dois métodos é o fato de que, ao usarmos o método get, todos os dados a serem enviados serão passados diretamente na URL. Embora ligeiramente codificados, esses dados são passíveis de entendimento pelo usuário. Caso os dados possuam certa discrição, torna−se incompatível o seu uso.
    - ***post*** já o método post, submete os dados de forma oculta, não alterando a URL. Deve ser utilizado em casos de autenticação de usuários, consultas de bancos de dados e onde haja tráfego de informações que não devem se tornar públicas.
- ***enctype*** com esta propriedade informamos qual é o tipo de mídia ou dado que está transitando pelos nossos formulários. O tipo mais comum e default é o application/x−www−form−urlencoded, responsável pela transmissão de dados típicos de formulários, ou melhor, pelos valores que esses elementos estão assumindo. Há uma exceção no caso de transmissão de arquivos, como em um upload. Neste caso, o enctype a ser usado é o multipart/form−data. E se os dados forem enviados diretamente por e−mail, o valor a ser utilizado é o text/plain.
- ***name*** é a propriedade que define o nome de acesso ao nosso objeto em questão, o formulário.

Nos tópicos a seguir, veremos a declaração e uma implementação progressiva do uso de formulários.

8.3 Conhecendo os objetos dos formulários

Os objetos dos formulários são vários e iremos estudá−los um de cada vez. Eles possuirão características comuns e talvez funcionalidades comuns, dependendo da forma que serão utilizados. Mas todos eles devem estar declarados dentro das tags <form></form>. Caso algum deles esteja fora dessas tags, eles não serão enviados ao servidor.

Vejamos agora uma lista de objetos que podemos usar:

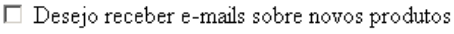
- ***text*** o objeto text é uma caixa de texto, limitando o tamanho do texto a 255 caracteres ou de acordo com a propriedade maxlength.

- ***password*** campo texto destinado à inserção de valores de senha. É igual ao objeto text, porém o texto digitado pelo usuário aparece em forma de asteriscos ou marcadores para ocultar a senha.

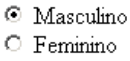
- ***textarea*** também permite a entrada de texto, porém de forma livre, sem um limite de caracteres pré−estabelecido.

- ***hidden*** o objeto hidden armazena um determinado valor, pré−estabelecido ou não, de forma oculta ao usuário.

- ***checkbox*** a checkbox permite que o usuário decida se quer ou não aceitar uma determinada cláusula estabelecida na página, como receber e−mails de propaganda ou newsletters.



- ***radio*** os botões de rádio permitem que o usuário escolha apenas uma entre várias alternativas exibidas na página, como por exemplo o campo sexo em um cadastro qualquer. Sabemos que é uma escolha de uma única resposta.



- ***select*** o objeto select se parece muito com as combo box de outras linguagens de programação. É dado ao usuário uma lista com várias opções e ele pode escolher apenas uma.

- ***select multiple*** a sintaxe é igual ao objeto select, exceto pela cláusula multiple, que permitirá ao usuário selecionar uma ou mais opções, utilizando as teclas shift ou ctrl para a seleção múltipla.

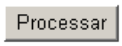
- ***image*** muitas páginas utilizam imagens como botões. Isso poderia ser feito com o auxílio de uma linguagem de script cliente, como o JavaScript. Porém o HTML fornece o objeto image, que tem a mesma funcionalidade.

- ***file*** o objeto file somente será aproveitável com o auxílio de uma outra linguagem que poderá tratar o dado que foi submetido, no caso um arquivo. Este objeto é composto de uma caixa de texto e um botão, que ao ser clicado abre a árvore de diretórios da máquina do usuário.

- ***submit*** este é um botão que já vem com uma tarefa específica: enviar todos os dados para o programa ou script que irá tratar esses dados, que foi definido na propriedade action da tag <form>.

- ***reset*** este também é um botão com tarefa pré−definida: limpar os valores de todos os objetos incluídos no formulário.

- ***button*** este último botão não tem atividade pré−programada. A programação dele depende de outra linguagem de script, como o JavaScript. Essa programação dependerá do objetivo desejado, como por exemplo processar uma determinada informação.



8.4 Criando um formulário

Para se criar um formulário, basta inserir a tag form com suas respectivas propriedades. Vamos iniciar aqui, uma declaração que será aproveitada até o final do capítulo em todos os exemplos. Vamos trabalhar também com tabelas, para organizarmos nossos objetos e também treinarmos um pouco mais.

```
<HTML>
        <HEAD>
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON−Line</center><br><br>
        <form action="curriculo.php" method="get">
        </form>
</BODY>
</HTML>
```

Note que utilizamos o script exemplo *curriculo.php* após o action, pois será este script o responsável pela manipulação dos dados passados através do formulário. Após a visualização deste exemplo, você ainda não poderá ver nada do formulário, porque ainda não declaramos nenhum objeto. Faremos isso nos tópicos seguintes.

8.5 Utilizando os campos do formulário

Conforme o texto inicial da página, iremos montar, ao final de todo o capítulo, um formulário para cadastro de currículos, porém sem nenhuma funcionalidade. O HTML produz apenas páginas estáticas, que não podem interagir com o usuário.

Vamos então inserir algumas caixas de texto para obter os dados do usuário, como nome e endereço.

```
<HTML>
        <HEAD>
```

```
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON-Line</center><br><br>
        <form action="curriculo.php" method="get">
        <table width="50%" align="center" border="0">
        <tr>
        <td align="right">Nome:</td>
        <td>
        <input type="text" name="nome" size="40"
        maxlength="50">
        </td>
        </tr>
        <tr>
        <td align="right">Endereço:</td>
        <td>
        <input type="text" name="ende" size="40"
        maxlength="100">
        </td>
        </tr>
        </table>
        </form>
</BODY>
</HTML>
```

Vamos entender agora a declaração e as propriedades das caixas de texto. As caixas de texto são tags <input> com tipo text:

<input type="text name= nome size=ì0 maxlength=í0 >

- ***type*** define o tipo do objeto input, neste caso, uma caixa de texto(text).
- ***name*** é o nome pelo qual outras linguagens acessarão o valor deste objeto.
- ***size*** define um tamanho para o objeto, baseado em números de caracteres desejados.
- ***maxlength*** limita o tamanho do texto, conforme o valor estabelecido.

O exemplo será alterado agora, para que o usuário possa inserir uma senha de acesso ao seu currículo. Estamos falando do objeto password. O password também é um objeto input, só que com o tipo password. Veja a declaração:

<input type= password name= senha size=é5 maxlength=î >

Agora observe como ficará o código:

```
<HTML>
        <HEAD>
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON-Line</center><br><br>
        <form action="curriculo.php" method="get">
        <table width="50%" align="center" border="0">
        <tr>
        <td align="right">Nome:</td>
        <td>
        <input type="text" name="nome" size="40" maxlength="50">
        </td>
        </tr>
        <tr>
        <td align="right">Endereço:</td>
        <td>
        <input type="text" name="ende" size="40" maxlength="100">
        </td>
        </tr>
```

```
        <tr>
        <td align="right">Senha:</td>
        <td>
        <input type="password" name="senha" size="15" maxlength="6">
        </td>
        </tr>
        </table>
        </form>
</BODY>
</HTML>
```

As propriedades de password são as mesmas do objeto text. Todos os objetos tem ainda uma propriedade que contém o valor do objeto. Nas caixas de texto não é comum usá−la, pois espera−se que o usuário insira algum texto. Somente em casos particulares seu uso pode ser útil, como por exemplo induzir a entrada com um determinado formato.

8.6 Inserindo caixas de seleção e botões de rádio

A funcionalidade que vamos declarar agora é a caixa de seleção ou checkbox. O checkbox é um objeto input do tipo checkbox.

Vamos perguntar ao usuário se ele quer que seus dados sejam públicos.

<input type= checkbox name= publicar value= sim checked>

Procurem o final da última linha e acrescentem o novo código:

```
<HTML>
        <HEAD>
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON−Line</center><br><br>
        <form action="curriculo.php" method="get">
        <table width="50%" align="center" border="0">
        .
        .
        .
        <tr>
        <td align="right"> </td>
        <td>
        <input type="checkbox" name="publicar" value="sim" checked>
        Quero que meu currículo seja público.
        </td>
        </tr>
        </table>
        </form>
</BODY>
</HTML>
```

Vamos entender as propriedades do objeto checkbox:

- ***type*** o tipo do objeto será checkbox.
- ***name*** é o nome pelo qual outras linguagens poderão acessar o checkbox.
- ***value*** define o valor que será enviado caso o usuário marque a checkbox. Caso não seja declarada, o valor default será on se estiver marcado.
- ***checked*** informa que o checkbox inicia já marcado. Caso queira dar essa opção ao usuário, é só omitir esta cláusula.

Perguntaremos o sexo do usuário utilizando o objeto de escolha radio. Sua sintaxe de declaração é a seguinte:

<input type= radio name= sexo value= Masculino >

Vejam o código:

```
<HTML>
        <HEAD>
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON-Line</center><br><br>
        <form action="curriculo.php" method="get">
        <table width="50%" align="center" border="0">
        .
        .
        .
        <tr>
        <td align="right">Sexo:</td>
        <td>
        <input type="radio" name="sexo" value="Masculino">Masculino<br>
        <input type="radio" name="sexo" value="Feminino">Feminino
        </td>
        </tr>
        </table>
        </form>
</BODY>
</HTML>
```

Vamos entender as propriedades deste objeto:

- ***type*** o tipo deste input é radio.
- ***name*** é nome pelo qual este objeto será acessado por outras linguagens de programação. No caso de objetos radio, que podem ser vários, todos aqueles que pertencerem a um mesmo grupo de informação deverão ter o mesmo nome. Do contrário, mais de uma opção poderá ser selecionada. Não é o que queremos. Sexo só poderá ser Masculino ou Feminino, nunca os dois.
- ***value*** valor que será enviado de acordo com a seleção do usuário.
- ***checked*** permite definir se uma determinada opção já iniciará com valor checkado ou não. Se esta cláusula for declarada, deverá ser feita diretamente na opção preterida. Se não existir em nenhuma das opções, a opção ficará em branco para o usuário escolher.

8.7 Inserindo listas e menus

Agora vamos perguntar ao usuário qual é o seu nível de escolaridade. Esta pergunta também só poderá ter uma única resposta. Um objeto que pode nos auxiliar com esta questão é o select. Vamos ver como ficará o código e logo ao final estaremos discutindo sua declaração e propriedades:

```
<HTML>
        <HEAD>
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON-Line</center><br><br>
        <form action="curriculo.php" method="get">
        <table width="50%" align="center" border="0">
        .
        .
        .
        <tr>
        <td align="right">Escolaridade:</td>
        <td>
        <select name="escolaridade">
        <option value="sup">Superior</option>
```

```
        <option value="2">2º Grau</option>
        <option value="1">1º Grau</option>
        </select>
        </td>
        </tr>
        </table>
        </form>
</BODY>
</HTML>
```

Vamos entender a declaração deste objeto. Cada opção que o usuário terá deverá estar dentro das tags <select></select>. Estamos falando das tags

- ***name*** é o nome pelo qual este objeto será acessado por outras linguagens de programação.
- ***value*** como termos várias opções para o usuário, é conveniente que cada uma delas tenha um valor diferente. Após a submissão deste formulário teremos um objeto select com apenas um valor, aquele que foi selecionado pelo usuário.

Queremos que o usuário também informe quais as tecnologias que ele domina. Como podem ser várias, precisamos de um objeto que possa permitir a seleção de uma ou mais opções. Este objeto é o mesmo select visto no exemplo anterior, mas agora, com uma cláusula multiple, que habilitará este objeto para que seja possível um seleção múltipla com auxílio das teclas shift e ctrl.

Vejam o código e no final a explicação da declaração e propriedades deste objeto:

```
<HTML>
        <HEAD>
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON-Line</center><br><br>
        <form action="curriculo.php" method="get">
        <table width="50%" align="center" border="0">
        .
        .
        .
        <tr>
        <td align="right">Tecnologia:</td>
        <td>
        <select name="tecnologias" multiple size="3">
        <option value="php">PHP</option>
        <option value="html">HTML</option>
        <option value="js">Javascript</option>
        <option value="java">Java</option>
        </select>
        </td>
        </tr>
        </table>
        </form>
</BODY>
</HTML>
```

Como vimos, a declaração é exatamente igual ao objeto select, exceto pela cláusula multiple. Vejamos que propriedades este objeto possui:

- ***name*** nome pelo qual outros linguagens de programação poderão acessar este objeto.
- ***size*** define quantas opções serão mostradas ao usuário sem criar barras de rolagem no objeto. Caso o número de opções seja superior ao informado, uma barra de rolagem é automaticamente inserida.
- ***value*** são os valores que serão transmitidos após a submissão. Como podem ser vários, este objeto será interpretado como um array no script que irá processar este formulário. Isso será melhor entendido em um curso de JavaScript ou PHP.

- ***multiple*** permite que seja selecionados várias opções ao mesmo tempo.

8.8 Adicionando botões de formulário

O formulário terá que ter botões para que ele possa ser enviado e dar a opção ao usuário para limpar os campos caso deseje reiniciar o preenchimento.

Vamos começar com o botão de submissão. O input de tipo submit, é o responsável pelo envio dos dados para o script ou programa de processamento. Observe a sintaxe:

<input type= submit name= enviar value= Enviar >

Veja como ficará no exemplo:

```
<HTML>
        <HEAD>
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON-Line</center><br><br>
        <form action="curriculo.php" method="get">
        <table width="50%" align="center" border="0">
        .
        .
        .
        <tr>
        <td align="center" colspan="2">
        <input type="submit" name="enviar" value="Enviar">
        </td>
        </tr>
        </table>
        </form>
</BODY>
</HTML>
```

Vamos entender as propriedades do botão submit:

- ***type*** é o tipo deste input, que é submit, ou seja, um botão que tem a função de submeter o formulário.
- ***name*** nome pelo qual este objeto será acessado por outras linguagens de programação.
- ***value*** é o valor que será exibido no botão para informar ao usuário, qual a função do mesmo.

Aproveitaremos o código acima para adicionar um botão que tem a tarefa de limpar todos os campos que estão declarados dentro das tags form></form>.

Escrevam na mesma célula do botão submit.

```
<HTML>
        <HEAD>
                <TITLE>Trabalhando com formulários!!!</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center>Currículo ON-Line</center><br><br>
        <form action="curriculo.php" method="get">
        <table width="50%" align="center" border="0">
        .
        .
        .
        <tr>
        <td align="center" colspan="2">
```

```
        <input type="submit" name="enviar" value="Enviar">
         
        <input type="reset" name="limpar" value="Limpar">
        </td>
        </tr>
        </table>
        </form>
</BODY>
</HTML>
```

As propriedades para o botão reset são as mesmas do botão submit. Apenas o tipo de input é diferente: como a função deste botão é limpar o conteúdo dos campos do formulário, então seu tipo é reset.

Não vamos inserir um input do tipo button, porque não teríamos funcionalidade para ele, uma vez que não possuímos conhecimento para acrescentar−lhe uma funcionalidade. Suas propriedades também são idênticas aos botões submit e reset, apenas seu tipo será button. Vejamos apenas sua sintaxe de declaração:

<input type= button name= processar value= Processar >

8.9 Inserindo um objeto image

O objeto image pode ser visto como uma interface mais amigável de um botão. Este objeto tem a função default igual ao botão submit, ou seja, ao ser clicado, enviará o formulário conforme a propriedade action do formulário.

Como já utilizamos botões no exemplo acima, vamos entender apenas a sintaxe de declaração e as propriedades deste objeto:

<input type= image src= enviar.gif name= enviar border=è >

Observe agora uma explicação sobre essas propriedades:

- ***type*** este é um input do tipo image.
- ***src*** define o nome e o caminho da imagem a ser utilizada como botão.
- ***name*** nome de acesso para outras linguagens.
- ***border*** define se o botão terá ou não borda.
- ***width*** como o objeto em questão é uma imagem, podemos manipular seu comprimento através desta propriedade.
- ***height*** o valor aqui estabelecido definirá a altura do objeto.

8.10 Adicionando o objeto file

Com este objeto podemos dar a opção do usuário selecionar um arquivo de seu computador e enviá−lo para o servidor, ou seja, realizar um upload, desde que haja um programa ou script de processamento para este formulário. Veremos este tópico com mais detalhes nos próximos capítulos.

Como ainda não aprendemos a processar os valores recebidos do usuário, trabalharemos apenas com a declaração e propriedades deste formulário:

<input type= file name= arquivo >

Este é um input de tipo file. A propriedade name permitirá que o programa ou script que irá receber este arquivo saiba qual objeto está acessando. Entretanto, a propriedade mais importante, o value, só será definida quando o usuário selecionar um arquivo para enviá−lo.

Como vimos em todo o capítulo, os formulários são a principal fonte de interação entre nossas páginas e o usuário. Embora este formulário de currículo não seja processado, a sua codificação foi muito importante para entendermos cada objeto de formulário e sabermos onde usar cada um deles.

O entendimento deste capítulo será de fundamental importância quando vocês estiverem aprendendo as linguagens de script que tanto falamos.

9 Processamento de Formulários

9.1 Introdução

No capítulo anterior mostramos como criar formulários HTML, agora iremos fazer o processamento dos mesmos e realizar algumas tarefas e aplicações que também podem ser executadas, como por exemplo a validação de formulários.

9.2 Primeiro Processamento

Veremos agora um exemplo básico de processamento de formulários, faremos um simples formulário com algumas informações e um script php para processá−las, vejamos o código HTML e o código do script PHP:

form.html

```
<HTML>
        <HEAD>
                <TITLE>Processamento</TITLE>
        </HEAD>
        <BODY bgcolor="#FFFFCC">
        <center><b>Entre com suas informações</b></center><br><br>
        <form action="proc.php" method="get">
        Nome Completo:<br><input type = "text" name = "nomecompleto"> <br>
        Endereço Completo:<br><textarea name = "endereco" rows = "3" cols = "40">
        </textarea> <br>
        Área de Atuação:<br><select name = "atuacao">
        <option>Desenvolvimento Geral
        <option>Desenvolvimento Web
        <option>Redes Windows
        <option>Redes Linux
        </select> <br>
        Horário de Trabalho:<br>
        Matuntino <input type = "radio" name = "horario" value = "Matutino">
        Vespertino <input type = "radio" name = "horario" value = "Vespertino">
        Noturno <input type = "radio" name = "horario" value = "Noturno">
        <br><br>
        <input type = "submit" value  = "Enviar Dados!">  
        <input type = "reset" value  = "Limpar"
        </form>
</BODY>
</HTML>
```

Agora veremos o script *proc.php* que é o responsável pelo processamento das informações obtidas com o formulário anterior, note que as strings colocadas nos campos "name" do formulário acima será o respectivo nome de cada variável manipulada, veja o script:

proc.php

```
<HTML>
<BODY>
<?php
echo "Seja bem−vindo $nomecompleto!<br>";
echo "Seu endereço completo é $endereco!<br>";
echo "Você atua na área de $atuacao! <br>";
echo "Você trabalha no período $horario! <br>";
?>
<br>
<a href = "form.html"> Voltar para o formulário!</a>
</body>
</html>
```

O processamento de qualquer formulário não é muito parecido com o realizado no exemplo acima, veremos mais alguns exemplos que podem ilustrar situações diferentes.

Observe o campo "área de atuação" do formulário acima, imagine que o profissional atue em mais de uma área, assim, o formulário deve aceitar seleções múltiplas. Tal implementação pode ser feita com uma simples mudança no *form.html*. Veja:

```
Área de Atuação:<br><select name = "atuacao[]" Multiple>
```

Esta modificação implica que a variável "atuacao" é um array, que poderá ser preenchido quando o usuário selecionar as opções desejadas, não esquecendo de deixar a tecla "Ctrl" pressionada durante tal ação.

Agora a exibição do campo é muito simples, apenas imprima na tela os valores do array como já foi ensinado anteriormente nesta mesma obra. Veja a modificação necessário no script *proc.php*:

```
if (sizeof($atuacao) > "1")
   {
     echo "Você atua na(s) área(s) de: <br>";
     foreach($atuacao as $valor)
     {
      echo "$valor <br>";
     }
   }
 else
   {
     echo "Nenhuma área de atuação foi selecionada!"
   }
```

Note que no código acima a instrução *if* verifica se a variável $atuacao foi incializada com algum valor, ou seja, se o usuário selecionou algum campo de atuação. Isto é uma técnica de validação de formulários que veremos a seguir.

9.3 Validação de Formulários

Existem inúmeros critérios para a validação de formulários, totalmente dependentes da finalidade para qual o mesmo foi implementado. Poderíamos validar se um um certo campo foi preenchido, se um certo nome tem tamanho muito pequeno, se o valor para um campo numérico é válido, enfim, as opções são inúmeras, lembrando que quanto mais completo o processamento do seu formulário, mais eficiente ele será.

No exemplo anterior, chegamos a fazer uma pequena validação para vermos se o campo atuação continha algum item selecionado ou não, veremos agora mais alguns exemplos de validação.

valid_form.php

```
<HTML>
<BODY>
<?php
//variável para indicar se houve algum campo não preenchido
$erro = false;
//verifica se o campo nome completo foi preenchido
if ($nomecompleto == "")
{
        $erro = true;
        echo "Você deve entrar com o seu nome completo!<br>";
} else {
        echo "Seja bem-vindo $nomecompleto!<br>";
}
//verifica se o campo endereço foi preenchido
$tamanhoendereco = strlen($endereco);
```

```
if ($tamanhoendereco == 0)
{
        echo "Você não entrou com seu endereço!<br>";
} else {
        echo "Seu endereço completo é $endereco!<br>";
}
//verifica se a área de atuação foi escolhida
if (sizeof($atuacao) >= "1")
{
        echo "Você atua na(s) área(s) de: <br>";
        foreach($atuacao as $valor)
        {
                echo "$valor<br>";
        }
}
else
{ $erro = true;
        echo "Nenhuma área de atuação foi selecionada!<br>"
}
//verifica se o horário de trabalho foi selecionado
if (empty($horario))

{
        $erro = true;
        echo "Você não selecionou seu horário de trabalho!<br>";
} else {
        echo "Você trabalha no período $horario!<br>";
}
/* caso tenha ocorrido algum erro de preenchimento, uma mensagem será escrita na tela */
if ($erro)
{
        echo "Alguns campos não foram preenchidos, preencha novamente o <a href=\"form.html\">for
}

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

Agora, teste inúmeras possibilidades de erro para o fomulário antes criado e agora processado pelo script do exemplo anterior.

Lembre−se:

- Os métodos utilizados para verificar certas informações provindas de um formulário não precisam ser os mesmos dos utilizados nesta obra, ficando a critério do programador como implementá−las.
- Se certas validações forem muito parecidas procures utilizar operadores lógicos como || e &&, isto pode diminuir em muito o seu trabalho.

Muito cuidado ao implementar validações para verificar se a informação passada é um número ou não, pois quando um formulário é enviado para processamento, a cadeia de consulta (POST ou GET) é transformada em uma cadeia de texto, portando quando você tenta comparar números, o que você está fazendo é comparar números que são cadeias.

Para evitar problemas, use a função *is_numeric*, pois ela apenas verifica se o conteúdo da variável nela indicada é um número, independente de seu tipo, veja o exemplo:

valid_num.php

```
<?php
if (is_numeric($idade))
{
        echo "$idade é um número!";
```

```
} else {
        echo "$idade não é um número!";
}
?>
```

Abordaremos até o final do capítulo algumas aplicações que poderão tornar seus formulários um pouco mais eficientes.

9.4 Cabeçalhos HTTP

Quando um documento HTML é carregado, o servidor envia algumas informações sobre o documento carregado, informações estas que são chamadas "linhas de cabeçalho HTTP" ou "conteúdo de cabeçalhos HTTP". Normalmente enviamos cabeçalhos para realizar redirecionamentos de páginas e outras aplicações.

Devemos lembrar que nenhuma outra informação deve ser passada para o navegador antes de usar um cabeçalho, ou seja, o cabeçalho sempre deve ir primeiro do que qualquer output para o navegador. Caso isto não seja feito, uma mensagem de erro será gerada.

9.5 Redirecionamento de páginas

Esta é uma prática que com certeza todos desenvolvedores web devem estar acustumados a realizar. Falaremos agora do redirecionamento forçado, ou seja, o usuário não poderá interagir com o redirecionamento, ele será automaticamente redirecionado quando o cabeçalho HTTP for enviado.

O formato para utilizar o cabeçalho HTTP é o que se segue:

```
header ("Location: página destino");
```

Lembrando que a "página destino" deve conter um texto URL completo, como por exemplo: http://www.sistemasabertos.com.br

Podemos utilizar a estrutura mostrada para realizar um redirecionamento mais amigável, para isto, veja o exemplo a seguir:

redir.html

```
<HTML>
        <HEAD>
                <TITLE>Redirecionamento</TITLE>
        </HEAD>
        <BODY>
                <form action="redirec.php" method="post">
                Selecione a url desejada:<br>
                <select name = "link">
                <option selected value = "http://www.yahoo.com.br">Yahoo
                <option value = "http://www.google.com.br">Google
                <option value = "http://www.cade.com.br">Cadê
                <option value = "http://www.sistemasabertos.com.br">Sistemas Abertos
                </select>
                <br><br>
                <input type = "submit" value  = "Ir agora!">
                </form>
        </BODY>
</HTML>
```

redirec.php

```
<?php
```

```
header("Location: $link");
?>
```

Lembre−se:

- Até agora, apenas trabalhamos com um script para o formulário e outro script sendo chamado para processar o primeiro. Em PHP podemos chamar o próprio script que contém o formulário para processá−lo. Para isso usamos a variável *$PHP_SELF* entre aspas, no campo "action" do formulário:

```
action = "$PHP_SELF"
```

9.6 Trabalhando com datas

Saber trabalhar com datas é muito importante em qualquer linguagem de programação. O PHP fornece inúmeros tipos de formatação de datas, veremos nesta sessão as funções *date* e *checkdate*. Para intuito de estudo, procure no manual do PHP(http://www.php.net), as funções *time* e *mktime*.

9.6.1 date()

Sintaxe:

```
date(string formato, int[data e hora em timestamp]);
```

Vejamos as formatações possíveis do primeiro parâmetro:

Formato	Definição	Exemplo
A	prefixo "AM" ou "PM"	"PM"
j	dia no formato numérico	"1" a "31"
d	dia no formato numérico	"01" a "31"
S	número do dia com *th* ou *nd*	"22nd"
D	3 letras do dia textual	"fri"
l	dia em formato texo	Friday
F	mês textual	"january"
M	mês textual em 3 dígitos	"Jan"
m	mês no formato "01" a "12"	"01"
n	mês no formato "1" a "12"	"4"
Y	ano com 4 dígitos	2005
y	ano com 2 dígitos	05
t	número de dias do mês (28−31)	"29"
w	número do dia da semana (0 = domingo a 6 = sábado)	"3"
g	hora no formato "1" a "12"	"9"
G	hora no formato de "0" a "23"	"9"
h	hora no formato "00" a "12"	"09"
H	hora no formato de "00" a "23"	"19"
i	minutos de "0" a "59"	"53"
s	segundos de "00" a "59"	"17"

Com estes dados, você poderá facilmente mostrar a data e hora atual da seguinte forma:

```
date("d−m−Y h:i")
```

O exemplo acima mostra a data e a hora atual. Lembre−se que utilizando a função date, você pode tanto mostrar a data e a hora juntos, quanto apenas a data, ou a hora.

9.6.2 checkdate()

Sintaxe:

```
checkdate(int mês, int dia, int ano);
```

Como o próprio nome já diz, esta função serve para verificar se uma data específica é válida ou não, retornando true caso obtenha sucesso ou, caso contrário, retorne false. Vejamos o exemplo:

```
<?php
if(checkdate(12,11,2005))
{
        echo "Data válida!";
} else {
        echo "Data Inválida!";
}
//irá retornar "Data válida!"
$dia = 34;
$mes = 12;
$ano = 1900;
if(checkdate($dia,$mes,$ano))
{
        echo "Data válida!";
} else {
        echo "Data Inválida!";
}
//irá retornar data inválida pois não existe o dia 34
?>
```

10 Envio de Informações

10.1 Introdução

Com nosso estudo sobre formulários, vimos que podemos passar através deles uma pequena quantidade de informações, o que deveríamos fazer se agora, esta quantidade de informação fosse muito grande? É exatamente sobre isso que iremos falar neste capítulo, como enviar uma grande quantidade de informações provindas de um formulário e também carregar arquivos para o servidor de web.

10.2 Correio Eletrônico

Aprenderemos agora a enviar uma correspondência através de um próprio script PHP, utilizando a função *mail()*. Lembrando que para que tal função funcione corretamente, você deverá configurar o PHP para aceitar o envio de e−mails, leia o "Anexo A − Conceitos Utilizados" para obter mais informações.

A estrutura geral da função mail() é:

```
mail(para, assunto, conteúdo);
```

Lembrando que "para" deve ser o e−mail destino da mensagem, "assunto" é o título do e−mail e "conteúdo" é a mensagem em sí. Vejamos um simples exemplo para um maior entendimento:

```
<?php
mail(alguem@dominio.com.br, "Primeiro E−mail", "Este é o primeiro e−mail que envio através de um
?>
```

Dependendo do tamanho do e−mail, utilizar o método apresentado no e−mail anterior não seria muito prático, por isso apresentaremos uma outra maneira que também pode ser utilizada:

```
<?php
$para = "alguem@dominio.com.br";
$assunto = "Novo e−mail!"
$conteudo = "Estou testanto novamente esta função que aprendi!";
mail($para, $assunto, $conteudo);
?>
```

Lembre−se:

- Também é possível enviar o e−mail para mais de um destinatário, para isto, separe os e−mail utilizando a vírgula. Exemplo:

```
$para = "eu@domain.com, voce@domain.com";
```

10.3 Feedback

É muito comum hoje em dia, encontrarmos em inúmeros sites um campo "Contato", onde podemos enviar mensagens para o próprio administrador do site. Estes são os chamados formulários de feedback, faremos agora um exemplo desta tão usada ferramenta.

Faremos primeiramente um simples formulário para que o usuário deixe seu nome, e−mail e mensagem, veja:

contato.html

```
<HTML>
        <BODY>
                <center>
                        <h1>Entre em contato conosco!</h1>
                        <h2><form action="mail.php" method="post">
                        nome: <input name = "nome" type="text"> <br>
                        e-mail: <input name = "email" type="text"> <br>
                        assunto: <input name = "assunto" type="text"> <br>
                        mensagem:
                        <br>
                        <textarea name = "msg" cols = 25 rows = 6></textarea>
                        <br>
                        <input type = "submit" value = "Enviar Comentário!">  
                        <input type = "reset" value="Limpar Tudo!">  
                        </form>
                        </h2>
                </center>
        </BODY>
</HTML>
```

Vamos agora implementar o script de processamanto do formulário:

mail.php

```
<?php
//e-mail do administrador do site
$para = "admim@dominio.com.br"
//info do usuário
$remetente = "from: ".$nome ."<" .$email .">";
//note o 4º parâmetro da função mail()
if (mail($para, $assunto, $msg, $remetente))
{
        echo "E-mail enviado com sucesso!";
} else {
        echo "E-mail não enviado!";
}
?>
```

O 4º parâmetro da função mail() é adicional e normalmente utilizado para guardar mais informações sobre usuário que enviou a mensagem, podendo ser muito útil caso queiramos futuramente guardar tais informações em um banco de dados.

Note também que a função mail() retorna um booleano, podendo ser utilizado em uma instrução de condição para validar o sucesso ou a falha do envio.

10.4 Upload de Arquivos

No capítulo sobre Formulários HTML falamos sobre este assunto, mas só agora podermos implementá-lo. Se estivéssemos em um site interno de uma empresa, a habilidade de carregar arquivos para o servidor de web facilitaria em muito a nossa vida.

Para ser capaz de fazer o upload de um arquivo através de um formulário, deve-se usar o método POST junto ao elemento "multipart/form-data", veja o exemplo:

<method = post enctype = "multipart/form-data">

Lembremos agora como criar um botão no formulário para realizar o upload:

<input type = file name = "fileupload">

Utilizando esta tag, será necessário para o usuário escrever o endereço completo do arquivo na caixa de texto que aparecerá, ou então clicar no botão que será criado no navegador que mostrará o visualizador de sistemas de arquivo padrão do sistema. Com o arquivo selecionado, deve−se criar um script para processá−lo.

Vejamos um formulário para o upload de arquivos:

upload.html

```
<HTML>
        <BODY>
                <center>
                        <h1>Fazendo o upload de arquivos!</h1>
                        <h2><form action="uploadproc.php" method="post" enctype = "multipart/form
                        Entre com o nome do arquivo: <br>
                        <input type = file  name = "fileupload">
                        <br><br>
                        <input type = "submit" value = "Mandar arquivo!">  
                        <input type = "reset" value="Limpar!">  
                        </form>
                        </h2>
                </center>
        </BODY>
</HTML>
```

Após um arquivo ser carregado, ele pode ser acessado anteriormente através da variável $fileupload como definimos no formulário anterior. A linguagem PHP tmbém cria algumas variáveis ao fazer um carregamento de arquivo, variáveis estas que podem ser úteis de diversas maneiras, como por exemplo, no controle do tamanho máximo exigido para que um arquivo possa ser carregado. Veja o quadro:

Variável	Definição
$fileupload_name	Nome e caminho do arquivo carregado
$fileupload_size	Tamanho do arquivo em bytes
$fileupload_type	Tipo do arquivo
$fileupload	Guarda o nome do arquivo temporariamente

Lembre−se:

- Note no quadro acima que utilizamos a variável "fileupload" pois foi assim que a chamamos no formulário.

Veremos agora o script de processamento do formulário:

uploadproc.php

```
<HTML>
<BODY>
<?php
echo "nome do arquivo temporário para o processo de carregamento: $fileupload";
echo "nome do arquivo remoto original: $fileupload_name <br>";
echo "tamanho do arquivo em bytes: $fileupload_size <br>";
echo "o tipo do arquivo: $fileupload_type <br> <hr>";
//se o arquivo do cliente não for passado para o servidor, a variável $fileupload receberá "none"
if($fileupload == "none")
{
        echo "O arquivo não foi enviado!<br>";
        echo "Voltar para o formulário de <a href = 'upload.html'>upload</a>";
        exit;
}
//verifica se o arquivo é menos do que 3 Kb
```

```
if ($fileupload_size < 3072)
{ // copia o arquivo para o endereço específico
        if(copy($fileupload, "/home/httpd/ldocs/$fileupload_name")
        {
                echo "Upload feito com sucesso!";
                //exclui o nome do arquivo temporário
                unlink($fileupload);
        } else {
                echo "Upload não realizado!";
        }
} else {
        echo "Os arquivos devem ser menores do que 3Kb para realizar o upload!<br>";
}
?>
Voltar para o formulário de <a href = "upload.html">upload</a>!
</BODY>
</HTML>
```

11 Sessões e Cookies em PHP

11.1 Introdução

As sessões tem uma grande importância em um website, elas permitem criar uma área reservada para cada usuário em um site. Ao iniciar uma sessão, é designado para cada usuário um (ID) identificador único de sessão (SID − Session ID). Este identificador é guardado em um cookie do lado do usuário ou está na URL e será mantido até o momento de finalizar a sessão.

O suporte a sessões permite registrar um número qualquer de variáveis que deverão ser preservadas entre as requisições, ou seja, quando uma sessão é criada, podemos registrar quantas variáveis desejarmos. Enquanto a sessão existir, essas variáveis estarão disponíveis em qualquer página com o mesmo domínio que quiser acessá−las.

11.2 Como utilizar sessões?

Os passos de utilização de sessão são muito simples: primeiramente cria−se uma sessão, em seguida é enviado o ID para o usuário, esse ID será utilizado para manter a sessão aberta em todas as páginas do site até o usuário efetuar o “logout” (saída) da sessão, onde esta será destruída.

11.3 Esquema de Sessão

- Primeira página: Início da Sessão, envio do ID para o usuário. Quando um visitante acessar o site ele recebe um ID (identificador) único, que é a chave para o controle de sua sessão. Esse ID é armazenado em um cookie ou enviado via URL.
- Páginas visitadas no site: Envio do ID para o servidor, para retornar a sessão aberta;
- Úlmima página: Fim da Sessão, a sessão é destruída e todas as suas informações (variáveis de sessão) são apagadas.

11.4 Funções Definidas

session_start() utilizado para iniciar dados de sessão. Esta função cria uma sessão ou mantém uma sessão aberta, para isto é necessário utilizá−la em todos os scripts do site. A inicialização de uma sessão com esta função deve ser feita antes de qualquer saída HTML.

Quando esta função é usada para iniciar uma sessão, ou seja, quando o navegador acessa o site, o ambiente PHP irá verificar se um ID de sessão já foi criado para aquele acesso. Se não existir um ID de sessão, será criado um que será enviado para o usuário identificando este acesso.

session_register() Utilizada para armazenar informações em uma sessão. Para armazenar informação em uma sessão temos que criar as variáveis de sessão que serão utilizadas. É possível armazenar em uma sessão qualquer quantidade de variáveis de qualquer tipo de variáveis, inclusive arrays.

session_destroy() Destrói todos os dados registrados para uma sessão, ou seja, destrói todos os dados associados com a sessão corrente. Ela não elimina nenhuma das variáveis globais associadas com a sessão, e nem o cookie de sessão.

session_id() Obtém e/ou define a id da sessão atual, ou seja, retorna a id de sessão para a sessão atual.

session_is_registered() Usada para verificar se uma variável global está registrada em uma sessão. Esta função retorna true se existe uma variável global (com o nome do parâmetro desta função) registrada na sessão atual.

session_unset() Libera todas as variáveis de sessão atualmente registradas.

Utilizando estas funções torna−se muito simples fazer um sistema de autenticação de usuário utilizando sessões.

- Cria−se uma sessão quando o usuário entra no site;
- O usuário autentica−se com o seu login e password.
- Esta informação é verificada, e se for um login correto, registra−se uma variável de sessão com o ID do usuário autenticado.
- Os scripts que necessitam que o usuário esteja autenticado, apenas tem de verificar se existe um ID de usuário na variável de sessão. Se sim, o usuário está autenticado; se não ainda não está autenticado.
- Para efetuar o logout da autenticação, utiliza−se simplesmente uma chamada ao “session_destroy”, que destrói a sessão, e toda a informação nela contida (deixa de existir um ID de usuário na sessão).
- Da próxima vez que o usuário carregar uma nova página do site, é atribuído uma nova sessão, e caso necessite, o usuário pode voltar a autenticar−se normalmente.

11.5 Passando a ID de Sessão

Há dois métodos para programar uma id de sessão:

- Cookies;
- Parâmetro URL.

Os Módulos de sessão suportam ambos os métodos. Cookies são mais eficientes, mas se eles não estiverem autorizados (os usuários não são obrigados a aceitá−los), não é possível utilizá−los. O segundo método inclui a ID de sessão diretamente na URL, ou seja, o PHP permite anexar o ID a URL manualmente, ou então modificar as configurações do PHP, para que ele faça isso automaticamente.

11.6 Criar uma sessão e mostrar o ID de sessão no navegador.

id_sessao.php

```
<?php
session_start();
echo"O seu ID de sessão é: ". session_id();
?>
```

Executando este script no navegador teríamos algo parecido:

O seu ID de sessão é: e7b440cc28c0ba27fed874d505781489

No exemplo id_sessao.php inicializamos uma sessão e em seguida mostramos o ID de sessão no navegador.

11.7 Criar variáveis dentro de uma sessão.

criar_variaveis_sessao.php

```
<?php
session_start();
```

```
session_register("nome","sobrenome");
$nome="Maria";
$sobrenome="Silva";
echo "Os dados serão gravados na sessão de id".session_id();
echo "<br><br><a href=\"visualizar_variaveis_sessao.php\">Visualizar variáveis de sessão</a>";
echo "<br><br><a href=\"verificar_variaveis_sessao.php\">Verificar variaveis de Sessão</a>";
?>
```

visualizar_variaveis_sessao.php

```
<?php
session_start();
session_register(nome,sobrenome);
echo "O conteúdo das variáveis de sessão: nome e sobrenome são respectivamente: ".$nome." e ".$so
echo "<br><br>ID = ".$PHPSESSID;
echo "<br><br><a href=\"verificar_variaveis_sessao.php\">Verificar variaveis de Sessao</a>";
echo "<br><br><a href=\"criar_variaveis_sessao.php\">Início</a>";
?>
```

verificar_variaveis_sessao.php

```
<?php
session_start();
echo "O conteúdo das variáveis de sessão: nome e sobrenome são respectivamente: ".$nome." e ".$so
echo "<br><br>ID = ".session_id();
echo "<br><br><a href=\"criar_variaveis_sessao.php\">Inicio</a>";
echo "<br><br><a href=\"visualizar_variaveis_sessao.php\">Visualizar variáveis de sessão</a>";
?>
```

A primeira vez que é definida a sessão não é possível verificar as variáveis de sessão, elas só estarão disponíveis após fazer uma atualização ou recarregar a página.

Para criar variáveis dentro de uma sessão, utilizamos a função *session_register* , estas informações são gravadas em um arquivo geralmente em /tmp . Ao olhar o diretório /tmp, você poderá verificar que um arquivo foi criado com um nome de arquivo que é o mesmo que o id de sessão.

O script *criar_variaveis_sessao.php* cria uma sessão com duas variáveis ($nome, $sobrenome), com os valores "Maria" e "Silva" respectivamente.

O script *visualizar_variaveis_sessao.php* imprime no navegador o conteúdo das variáveis $nome e $sobrenome, além do id da sessao armazenado na variável $PHPSESID.

O script *verificar_variaveis_sessao.php* tem a mesma função do script *visualizar_variaveis_sessao.php*, ou seja, mostrar o id da sessão e o valor das variáveis. Neste exemplo o id de sessão foi obtido pela função *session_id.*

11.8 Sessão usando cookies

Utilizamos cookies para salvar de forma mais simples os estado de sessão na internet. Os cookies são pequenos arquivos de texto, que contém informações que podem ser passados de um formulário para outro. Eles são gravados no HD do cliente a não ser que este bloqueia este tipo de gravação.

Um arquivo de cookie não pode ter mais que 4K, geralmente os cookies utilizam apenas 1K. Quando um cookie é emitido, não é possível ler aquele cookie até que o cliente faça uma atualização da página ou carregue outra página. Só é possível enviar um cookie se não tiver emitido qualquer output PHP ou HTML anterior.

Quando um servidor web envia um cookie para um cliente, apenas este servidor poderá ler o cookie gravado no cliente, isto oferece uma segurança maior para as informações que são enviadas e recebidas entre cliente e servidor.

Para criar um cookie que termine quando a conexão atual do navegador terminar, basta deixar o tempo de expiração em branco.

Para definir um cookie no lado do cliente utilizamos a função *setcookie* .

setcookie(nome do cookie, valor, tempo de expiração, caminho)

Componentes do cookie:

- Nome do cookie − o nome atribuído ao cookie criado;
- Valor − O conteúdo do cookie;
- Tempo de expiração − O período de tempo que o cookie é válido;
- Caminho − Para que diretório o cookie é válido no servidor;

entrar.php

```
<html>
<body>
<center><b>SISTEMAS ABERTOS</br>Login para entrar no Site</b><br><br></center>
<center>Digite a Senha</center>
<form method="post" action="acessar.php">
<table border=2 align=center>
<tr><td>Senha:</td></tr>
<tr><td><input type="Password" name="password"</td></tr>
</table>
<center><input type="submit" value="Login"></center>
</form>
</body>
</html>
```

acessar.php

```
<?php
if ($password == "teste"){
        setcookie("cookie_access", $password);
        echo "Acesso permitido para visitar o <a href=\"pag.php\">Site</a>";
}else{
        echo "Senha Errada <a href=\"entrar.php\">Tentar Novamente</a>";
}
?>
```

logout.php

```
<?php
setcookie("cookie_access");
?>
<br><br><center>
Logout realizado com sucesso. Clique <a href="pag.php">AQUI</a> para voltar a página principal.</
```

pag.php

```
<?php
if(empty($cookie_access)){
        echo "Para visualizar esta página é necessário fazer o <a href=\"entrar.php\" >Login</a>"
        exit();
}else{
```

```
        if($cookie_access != "teste"){
                echo "Erro de login";
                exit;
        }else {
                echo "<h3><center>Página a ser visualizada</center><h3><br><br>";
                echo "<a href=\"pag2.php\">Próxima página<br><br></a>";
                echo "<a href=\"logout.php\">Logout</a>";
        }
}
?>
```

pag2.php

```
<?php
if(empty($cookie_access)){
        echo "Para visualizar esta página é necessário fazer o <a href=\"entrar.php\">Login</a>";
        exit();
}else{
        if($cookie_access != "teste"){
                echo "Erro de login";
                exit();
        }else {
                echo "<h3><center>Segunda Página</center><h3><br><br>";
                echo "<a href=\"pag.php\">Voltar</a><br><br>";
                echo "<a href=\"logout.php\">Logout</a>";
        }
}
?>
```

No script *entrar.php* criamos um campo input do tipo PASSWORD para receber a senha de acesso, neste caso utilizamos a senha "teste". Este formulário está utilizando o método POST, pois queremos esconder da URL o que o usuário está enviando quando o formulário é submetido. Assim que o usuário clicar em "Login" o formulário é postado com a variável $password contendo a senha para o script *acessar.php* .

O script *acessar.php* verifica se a senha é igual a "teste", se ela for igual será definido um cookie que utilizará a senha como o valor para a variável $password. Ao chamar a função *setcookie("cookie_access", $password);* não fornecemos data de expiração, desta forma este cookie terminará quando o navegador fechar. Se a senha digitada no script anterior não for igual a "teste" então um hiperlink será apresentado para o usuário voltar a página de login (entrar.php).

Os scripts *pag.php* e *pag2.php* estão apenas simulando o conteúdo das páginas visitadas no site. O código que aparece no início dos scripts são necessários para proteger os sites visitados, ou seja, o usuário precisa estar logado com a senha correta e o cookie deve existir. Caso ocorra do usuário tentar acessar estas páginas diretamente, será feita uma checagem, se existe o cookie e a senha correta, o usuário terá permissão para continuar visitando as páginas do site. Caso contrário será apresentado ao usuário um hiperlink redirecionando para a página de login (entrar.php).

Para finalizarmos a sessão temos o script *logout.php* , o processo é extremamente simples e envolve uma nova chamada à instrução setcookie apenas com o nome do cookie que queremos eliminar. O link para esta página, que chamaremos de logout.php, deverá estar presente em pontos−chaves do site. Após eliminar os cookies criados, redirecionamos o browser para outra página do site, neste caso *pag.php.*

11.9 Carregando valores através de formulários usando uma sessão

Utilizando *session_start* para iniciar uma sessão o PHP cria um cookie automaticamente, ele será único e as informações serão armazenadas no cliente e no servidor. A sessão terminará quando o navegador fechar ou utilizarmos a funçao session_destroy. O arquivo no servidor é preenchido quando são registradas as variáveis

de sessão, ao mesmo tempo o PHP cria diversas variáveis da web. Ao criar scripts que necessitam de dados e estados da sessão é necessário chamar a função session_start primeiro para recuperar estes dados. A função session_start deve ser chamada no início do script antes que ocorra qualquer output HTML ou PHP.

sess1.php

```
<?php
session_start();
$employee_num = session_id();
session_register( "employee_num" );
echo "<center><H2>Delivery etc − New Employee</h2></center>
<form method=get action=\"sess2.php\"><b>Employee:</b><BR>
<input type=\"text\" name=\"name\"><BR><b>Department:</b><BR>
<select name=\"dept\">
<option>Warehouse
<option>Accounts
<option>Administration
</select>
<BR><b>Pay Grade</b><BR>
<select name=\"pay_grade\">
<option selected>Top Job
<option>Normal
<option>Overtime
</select>
<BR><b>Country Location</b><BR>
<select name=\"location\">
<option selected>USA
<option>England
<option>Germany
</select>
<BR><BR>
<input type=\"submit\" value=\"Next Page\">
</form>";
echo $form;
?>
```

Este script inicializa uma sessão com a função session_start(), em seguida cria uma variável que será armazenada o id da sessão aberta. Para criar variáveis de sessão basta referenciá−las como parâmetros na função session_register. Neste exemplo foram criadas como variáveis de sessão: name, dept, pay_grade, location e employee_num. Podemos criar a qualquer momento em qualquer quantidade variáveis de sessão, para isto devemos referenciá−las como argumento da função session_register(). Este formulário será postado pelo método get para o script sess2.php.

sess2.php

```
<?php
session_start();
session_register( "name", "dept", "pay_grade", "location" );
$form = "<center><H2>Delivery etc − New Employee</H2></center>
<form method=get action=sess3.php>
<b>Employee Addres</b><BR>
<input type=\"text\" name=\"address\"><BR><BR>
<b>Telephone num</b><BR>
<input type=\"text\" name=\"telephone\"><BR><hr>
<input type=\"submit\" value=\"Next Page\">
</form>";
echo "Employee name: ".$name."<BR>
Department: ".$dept."<BR>
Pay Grade: ".$pay_grade."<BR>
Location: ".$location."<BR>
Employee ID: ".$employee_num."<BR>";
echo $form;
?>
```

Neste script utilizamos a função *session_start()* para recuperar os dados da sessão aberta em sess1.php. Se for preciso utilizar alguma variável de sessão que não foi referenciada na função *session_register()*, podemos chamar novamente esta função passando as variáveis necessárias no script como parâmetro da função. A qualquer momento podemos adicionar uma nova variável de sessão.

sess3.php

```
<?php
session_start();
session_register( "address", "telephone" );
?>

<center>Here are the new employee details</center>

<?php
$display="<pre>
Employee name: $name<BR>
Department: $dept<BR>
Pay Grade: $pay_grade<BR>
Location: $location<BR>
Home Address: $address<BR>
Telephone: $telephone<BR>
Employee ID: $employee_num<BR></pre>";

echo $display;
session_unset();
session_destroy();
?>
```

O script sess3.php mostra no navegador o valor de todas as variáveis de sessão que foram referenciadas, e destroy a sessão ao finalizar o script. Para destruir a sessão que está aberta, utilizamos a função *session_destroy* que irá destruir todas as variáveis associadas com a sessão atual. Podemos também acionar session unset para remover todas as variáveis que foram registradas em uma sessão. Para ter certeza de destruir as variáveis da sessão atual e da propriamente dita, chamamos as duas funções juntas no script.

12 Introdução ao MySQL

12.1 Introdução

Neste capítulo é dado uma introdução ao banco de dados MySQL, dando mais ênfase ao aspecto da linguagem SQL. Para quem deseja aprofundar mais sobre SQL, este capítulo não será suficiente. Para aqueles que já sabem SQL, este capítulo será uma revisão e uma introdução às particularidades do banco de dados MySQL. Grande parte deste capítulo faz parte da tradução do tutorial do MySQL encontrado na documentação em sua documentação oficial.

12.2 Por que o MySQL?

- O MySQL é um banco de dados cliente−servidor gratuito
- É simples, tem alto desempenho, é disponível para várias plataformas e é robusto
- Possui bom suporte à java (possui driver jdbc)

12.3 O que é MySQL?

MySQL é um servidor de banco de dados SQL multi−usuário, com suporte à múltiplas linhas de execução [multi−threaded]. SQL é a linguagem de banco de dados mais popular no mundo. MySQL é uma implementação cliente/servidor que consiste de servidor [server daemon] mysqld e vários programas clientes e bibliotecas.

SQL é a linguagem padronizada que torna fácil armazenar, atualizar e acessar informação. Por exemplo, você pode usar SQL para recuperar informação de produtos e armazenar informação de clientes para um site Internet. MySQL é também suficientemente veloz e flexível para armazenar dados históricos e figuras.

Os principais objetivos do MySQL são: velocidade, robustez e facilidade de uso.

12.4 Básico

'mysql' (algumas vezes é referenciado como "monitor") é um programa interativo que permite conectar a um servidor MySQL, executar consultas e ver os resultados. 'mysql' também pode ser usado em modo de execução em lote [batch mode]: você armazena suas consultas em um arquivo, então orienta ao 'mysql' executar os conteúdos do arquivo. Ambas maneiras de usar o 'mysql' são cobertos aqui.

Para ver uma lista de opções fornecidas pelo 'mysql', invoque−o com a opção de comando '−−help'

```
> mysql −−help
```

12.5 Conectando e desconectando do servidor

Para conectar ao servidor, você irá necessitar de fornecer um nome de usuário do próprio Linux, quando o 'mysql' é invocado e uma senha. Se o servidor está funcionando em uma máquina diferente da que você está efetuando logon, também será necessário especificar o nome de máquina. Por exemplo:

```
> mysql −u root −p
Enter password: ********
```

Os caracteres `********' representam sua senha; entre quando 'mysql' mostra o prompt 'Enter password'.

Se der tudo certo, você deverá ver informações introdutórias seguidas pelo prompt 'mysql>'.

```
shell> mysql −u root
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a−log

Type 'help' for help.

mysql>
```

O prompt significa que o 'mysql' está pronto para você entrar comandos.

Depois de ter conectado com sucesso, você pode desconectar qualquer momento digitando 'QUIT' no prompt 'mysql>':

```
mysql> QUIT
Bye
```

Você também pode desconectar acionando as teclas CTRL+D.

12.6 Executando consultas

Tenha a certeza que você está conectado ao servidor, como foi discutido na seção anterior.

Aqui é mostrado um comando simples que solicita ao servidor sua versão e a data atual.

```
mysql> SELECT VERSION(), CURRENT_DATE;
+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−+
| version()    | CURRENT_DATE |
+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−+
| 3.22.20a−log | 1999−03−19   |
+−−−−−−−−−−−−−−+−−−−−−−−−−−−−−+
1 row in set (0.01 sec)
mysql>
```

Esta consulta ilustra vários pontos sobre o 'mysql':

- Um comando normalmente consiste de uma setença SQL seguida por um ponto−e−vírgula.
- Quando você executa um comando, 'mysql' envia ao servidor para execução e mostra os resultados, então imprime outro 'mysql>' para indicar que está pronto para outro comando.
- 'mysql' mostra a saída de consultas como tabelas (linhas e colunas). A primeira linha contém os rótulos para as colunas. As linhas seguintes são os resultados da consulta.

As palavras−chave podem ser entradas sem diferenciação de maiúsculas/minúsculas. As seguintes consultas são equivalentes:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

É permitido entrar com múltiplas sentenças em uma única linha. Simplesmente finalize cada sentença com um ponto−e−vírgula:

```
mysql> SELECT VERSION(); SELECT NOW();
+−−−−−−−−−−−−−−+
| version()    |
+−−−−−−−−−−−−−−+
| 3.22.20a−log |
```

```
+--------------+

+---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

Existe uma forma bastante livre para digitar sentenças. É permitido pular linhas entre sentenças:

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;

+--------------------+--------------+
| USER()             | CURRENT_DATE |
+--------------------+--------------+
| joesmith@localhost | 1999-03-18   |
+--------------------+--------------+
```

Se você decidir que você não quer executar um comando que você está em processo de digitação, cancele-o digitando '\c':

```
mysql> SELECT
    -> USER()
    -> \c
mysql>
```

Lembre-se que para finalizar um comando, finalize-o com ponto-e-vírgula.

```
mysql> SELECT USER()
    -> ;
+--------------------+
| USER()             |
+--------------------+
| joesmith@localhost |
+--------------------+
```

12.7 Criando e usando um banco de dados

Agora que você já sabe entrar comandos, estamos no momento certo de acessar um banco de dados.

Suponha que você deseje criar um banco de dados de contatos de clientes. Esta seção mostra como recuperar dados de tabelas e também:

- Como criar um banco de dados
- Como criar uma tabela
- Como carregar dados para a tabela
- Como recuperar dados da tabela de várias formas
- Como utilizar tabelas múltiplas

O banco de dados de clientes será simples (deliberadamente), mas não será difícil imaginar situações de mundo real no qual um tipo de banco de dados similar poderá ser usado.

Use a sentença 'SHOW' para checar quais bancos de dados existem atualmente no servidor:

```
mysql> SHOW DATABASES;
+----------+
| Database |
```

```
+----------+
| mysql    |
| test     |
+----------+
```

O banco de dados 'mysql' é necessário porque descreve os privilégios de acesso dos usuários. O banco de dados 'test' é geralmente fornecido como um espaço de trabalho para usuários para testes.

Tente acessar o banco de dados teste:

```
mysql> USE test
Database changed
```

O banco de dados test permite que todos que tenham acesso a ele criar e remover as suas tabelas. Portanto, é recomendável utilizar um banco de dados separado para cada aplicação. Desta maneira, você (ou o administrador do banco de dados) poderá configurar o acesso de uma maneira mais apropriada.

12.7.1 Criando e selecionando um banco de dados

Para criar um banco de dados:

```
mysql> CREATE DATABASE contatos;
```

Sob sistemas Unix, nomes de banco de dados são sensíveis a caixa (diferente das palavras−chave SQL). Portanto, você deve sempre referenciar seu banco de dados como 'contatos', não como 'Contatos', 'CONTATOS' ou qualquer outra variante. Isto também é aplicável para nomes de tabelas.

Para conectar no banco de dados:

mysql> USE contatos Database changed

12.7.2 Criando uma tabela

Neste ponto o banco de dados está vazio, como é mostrado no comando 'SHOW TABLES':

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

A parte complexa é decidir como será a estrutura de seu banco de dado: quais tabelas serão necessárias, e quais colunas terão cada uma. Este trabalho é um trabalho de análise de banco de dados, que foge do escopo deste curso. Suponhamos que este trabalho já foi realizado, e tenhamos a seguinte estrutura:

Tabelas do sistema de contatos de clientes

Pessoa

Nome da coluna	Tipo	Nulo?	Descrição
cpf	char(11)	não	Cpf da pessoa
Nome	char(40)	não	Nome completo da pessoa
telefone1	char(15)	sim	Telefone residencial
telefone2	char(15)	sim	Telefone comercial
email	char(30)	sim	Email da pessoa
data_nascimento	Date	sim	Data de nascimento
Endereco	char(50)	sim	Endereço residencial

bairro	int unsigned1	sim	Código do Bairro residencial
cidade	int unsigned	sim	Código da Cidade residencial
uf	char(2)	sim	Unidade de Federação da residência
cep	char(8)	sim	CEP da residência

Bairro

Nome da coluna	Tipo	Nulo?	Descrição
cod_bairro	int unsigned	não	Identificador do bairro
descricao	char(40)	não	Descrição do bairro

Cidade

Nome da coluna	Tipo	Nulo?	Descrição
cod_cidade	int unsigned	não	Identificador da cidade
descricao	char(40)	não	Descrição da cidade

Empresa

Nome da coluna	Tipo	Nulo?	Descrição
cgc	char(20)	não	Cgc/cnpj
nome_fantasia	char(40)	não	Nome fantasia
razao_social	char(40)	sim	Razão social
responsavel	char(20)	sim	Cpf do responsável pela empresa
site_internet	char(50)	sim	Site Internet
email	char(30)	sim	Email do contato com a empresa
telefone	char(15)	sim	Telefone de contato
fax	char(15)	sim	Fax
endereco	char(50)	sim	Endereço
bairro	int unsigned	sim	Código do Bairro
cidade	int unsigned	sim	Código da Cidade
uf	char(2)	sim	Unidade de Federação
cep	char(8)	sim	CEP

Contato

Nome da coluna	Tipo	Nulo?	Descrição
num_contato	int unsigned	não	Identifica o contato (valor sequencial)
tipo_cliente	char(1)	não	p=pessoa; e=empresa
cpf	char(20)	sim	Cpf (caso seja pessoa)
cgc	char(20)	sim	Cgc (caso seja empresa)
data	date	não	Data de contato com o cliente
observacao	text	sim	Observações, informações sobre o contato com o cliente

Através desta base de dados, além de você ter o cadastro de clientes, empresas ou pessoas, que poderá utilizá−la em uma mala direta. Note que com o uso de tabelas para cidade e bairro, você poderá utilizar informações para classificar área de maior concentração geográfica de seus clientes. Através da tabela de contatos você poderá cadastrar os contatos realizados com cada cliente, possuindo um histórico dos contatos realizados.

Você pode forçar que determinada coluna não aceita nulos, através da cláusula NOT NULL.

Use um comando 'CREATE TABLE' para especificar o layout de cada tabela:

```
mysql> CREATE TABLE PESSOA (
    ->   cpf char(11) not null,
    ->   nome char(40),
    ->   telefone1 char(15),
    ->   telefone2 char(15),
    ->   email char(30),
    ->   data_nascimento date,
    ->   endereco char(50),
    ->   bairro int unsigned,
    ->   cidade int unsigned,
    ->   uf char(2),
    ->   cep char(8));
```

Agora que voce criou uma tabela, 'SHOW TABLES' deve produzir a saída:

```
mysql> SHOW TABLES;
+--------------------+
| Tables in contatos |
+--------------------+
| PESSOA             |
+--------------------+
```

Para verificar que sua tabela foi criada da maneira que você espera, use o comando 'DESCRIBE':

```
mysql> DESCRIBE PESSOA;
+-----------------+----------+------+-----+---------+-------+
| Field           | Type     | Null | Key | Default | Extra |
+-----------------+----------+------+-----+---------+-------+
| cpf             | char(20) |      |     |         |       |
| nome            | char(40) | YES  |     | NULL    |       |
| telefone1       | char(15) | YES  |     | NULL    |       |
| telefone2       | char(15) | YES  |     | NULL    |       |
| email           | char(30) | YES  |     | NULL    |       |
| data_nascimento | date     | YES  |     | NULL    |       |
| endereco        | char(50) | YES  |     | NULL    |       |
| bairro          | int(11)  | YES  |     | NULL    |       |
| uf              | char(2)  | YES  |     | NULL    |       |
| cep             | char(8)  | YES  |     | NULL    |       |
+-----------------+----------+------+-----+---------+-------+
```

Por enquanto, iremos apenas criar a tabela EMPRESA:

```
mysql> CREATE TABLE EMPRESA (
    -> cgc char(20) not null,
    -> nome_fantasia char(40) not null,
    -> razao_social char(40),
    -> responsavel char(11),
    -> site_internet char(50),
    -> email char(30),
    -> telefone char(15),
    -> fax char(15),
    -> endereco char(50),
    -> bairro int unsigned,
    -> cidade int unsigned,
    -> uf char(2),
    -> cep char(8));
```

12.7.3 Carregando dados para uma tabela

Depois de criar suas tabelas, você precisa populá−las. Os comandos 'LOAD DATA' e o 'INSERT' são úteis para isto.

Suponha que seus registros podem ser descritos como mostrado abaixo. (Observe que o ***MySQL*** aceita datas no formato 'YYYY−MM−DD'; isto pode ser diferente do que você está acostumado.

Tabela Pessoa

cpf	nome	telefone1	telefone2	email	data_nascimento	endereco	bairro	cidade	uf	cep
1111	João da Silva	111−1111	111−1112	joao@uol.com.br	1984−11−11	Rua 11, no. 11	1	1	AM	11111111
2222	Maria Nunes	222−2222	222−2223	maria@bol.com.br	1971−11−22	Rua 22, no. 22	2	2	GO	22222222
3333	José Pereira	333−3333	null	jose@sol.com.br	1961−03−03	Rua 33, no. 33	2	2	BA	22222222

Desde que você está iniciando com uma tabela vazia, uma forma fácil de populá−la é criar um arquivo texto contendo uma linha para cada pessoa, então carregar o conteúdo do arquivo para a tabela com um único comando.

Você pode criar um arquivo texto 'pessoa.txt' contendo um registro por linha, com valores separados por tabs, e fornecendo dados na ordem na qual as colunas foram listadas no comando 'CREATE TABLE'. Para valores não aplicáveis ou não existentes (tal como segundo telefone), você pode usar valores 'NULL'. Para representar estes valores em seu arquivo texto, use '\N'. Por exemplo, o registro para José Pereira seria (onde há espaço em branco entre valores existe um caractere tab):

cpf	nome	telefone1	telefone2	email	data_nascimento	endereco	bairro	cidade	uf	cep
3333	José Pereira	333−3333	\N	jose@sol.com.br	1968−03−03	Rua 33, no. 33	2	2	BA	22222222

Para carregar o arquivo texto 'pessoa.txt' na tabela 'PESSOA', use este comando:

```
mysql> LOAD DATA LOCAL INFILE "pessoa.txt" INTO TABLE PESSOA;
```

Quando você desejar adicionar novos registro um por vez, o comando 'INSERT' é útil. Nesta forma mais simples, você fornece valores para cada coluna, na ordem na qual as colunas foram listadas no comando 'CREATE TABLE':

```
mysql> INSERT INTO PESSOA
    -> VALUES ('4444', 'Ana Souza', '444−4444', NULL, 'ana@aol.com.br',
    -> '1955−04−04', 'Rua 44, no. 44', 1, 1, 'GO', '44444444');
```

Note que os valores de string e datas são especificados como strings entre aspas. Também, com 'INSERT', você pode inserir 'NULL' diretamente para representar um valor ausente. Você não usa '\N' como é utilizado no comando 'LOAD DATA'.

12.7.4 Recuperando informação de uma tabela

O comando 'SELECT' é usado para obter informação de uma tabela. A forma geral do comando é:

```
SELECT o_que_selecionar
FROM qual_tabela
WHERE condições_de_consulta
```

'o_que_selecionar' indica o que você deseja ver. Isto pode ser uma lista de colunas, ou '*' para indicar "todas colunas". 'qual_tabela' indica a tabela do qual os dados serão recuperados. A cláusula 'WHERE' é opcional. Se está presente, 'condições_de_consulta' especifica condições com as quais as linhas devem satisfazer para qualificar a recuperação.

12.7.4.1 Selecionando todos dados

A forma mais simples de 'SELECT' recupera tudo de uma tabela:

```
mysql> SELECT * FROM PESSOA;
+-----------+---------------+-----------+-----------+------------------+-----------------+-------
| cpf       | nome          | telefone1 | telefone2 | email            | data_nascimento | endere
+-----------+---------------+-----------+-----------+------------------+-----------------+-------
| 1111      | João da Silva | 111-1111  | 111-1112  | joao@uol.com.br  | 1984-11-11      | Rua 11
| 2222      | Maria Nunes   | 222-2222  | 222-2223  | maria@bol.com.br | 1971-11-22      | Rua 22
| 3333      | José Pereira  | 333-3333  | NULL      | jose@sol.com.br  | 1961-03-03      | Rua 33
| 4444      | Ana Souza     | 444-4444  | NULL      | ana@aol.com.br   | 1955-04-04      | Rua 44
+-----------+---------------+-----------+-----------+------------------+-----------------+-------
```

12.7.4.2 Selecionando linhas particulares

Você pode selecionar somente linhas particulares de sua tabela. Por exemplo, se você quiser selecionar qual é a pessoa de cpf de no. 3333, você executaria uma consulta como esta:

```
     mysql> SELECT * FROM PESSOA WHERE CPF = "3333";
+-----------+--------------+-----------+-----------+-----------------+-----------------+---------
| cpf       | nome         | telefone1 | telefone2 | email           | data_nascimento | endereco
+-----------+--------------+-----------+-----------+-----------------+-----------------+---------
| 3333      | José Pereira | 333-3333  | NULL      | jose@sol.com.br | 1961-03-03      | Rua 33,
+-----------+--------------+-----------+-----------+-----------------+-----------------+---------
```

Você pode especificar condições em qualquer coluna, não somente 'cpf'. Por exemplo, se você desejar quais pessoas nasceram depois de 1970, utilize a coluna data_nascimento:

```
     mysql> SELECT * FROM PESSOA WHERE DATA_NASCIMENTO >= "1970-1-1";
+-----------+---------------+-----------+-----------+------------------+-----------------+-------
| cpf       | nome          | telefone1 | telefone2 | email            | data_nascimento | endere
+-----------+---------------+-----------+-----------+------------------+-----------------+-------
| 1111      | João da Silva | 111-1111  | 111-1112  | joao@uol.com.br  | 1984-11-11      | Rua 11
| 2222      | Maria Nunes   | 222-2222  | 222-2223  | maria@bol.com.br | 1971-11-22      | Rua 22
+-----------+---------------+-----------+-----------+------------------+-----------------+-------
```

Você pode combinar condições, por exemplo, para localizar pessoas que moram em Goiás e que nasceram depois de 1970:

```
     mysql> SELECT * FROM PESSOA WHERE DATA_NASCIMENTO >= "1970-1-1" AND UF = "GO";
+-----------+-------------+-----------+-----------+------------------+-----------------+---------
| cpf       | nome        | telefone1 | telefone2 | email            | data_nascimento | endereco
+-----------+-------------+-----------+-----------+------------------+-----------------+---------
| 2222      | Maria Nunes | 222-2222  | 222-2223  | maria@bol.com.br | 1971-11-22      | Rua 22,
+-----------+-------------+-----------+-----------+------------------+-----------------+---------
```

A consulta anterior usou o operador lógico 'AND'. Há também o operador 'OR':

```
     mysql> SELECT * FROM PESSOA WHERE DATA_NASCIMENTO >= "1970-1-1" OR UF = "GO";

+-----------+---------------+-----------+-----------+------------------+-----------------+-------
| cpf       | nome          | telefone1 | telefone2 | email            | data_nascimento | endere
+-----------+---------------+-----------+-----------+------------------+-----------------+-------
| 1111      | João da Silva | 111-1111  | 111-1112  | joao@uol.com.br  | 1984-11-11      | Rua 11
| 2222      | Maria Nunes   | 222-2222  | 222-2223  | maria@bol.com.br | 1971-11-22      | Rua 22
```

```
| 4444      | Ana Souza     | 444-4444  | NULL      | ana@aol.com.br   | 1955-04-04      | Rua 44
+-----------+---------------+-----------+-----------+------------------+-----------------+-------
```

12.7.4.3 Selecionando colunas particulares

Se você não desejar ver todas colunas de sua tabela, simplesmente forneça os nomes das colunas que você tiver interesse, separados por vírgulas. Por exemplo, se você desejar obter uma lista de nomes com seus respectivos e-mails, selecione as colunas 'nome' e 'email':

```
     mysql> SELECT nome, email FROM PESSOA;
+---------------+------------------+
| nome          | email            |
+---------------+------------------+
| João da Silva | joao@uol.com.br  |
| Maria Nunes   | maria@bol.com.br |
| José Pereira  | jose@sol.com.br  |
| Ana Souza     | ana@aol.com.br   |
+---------------+------------------+
```

Para selecionar quais estados (unidades de federação) as pessoas se encontram, use esta consulta:

```
     mysql> SELECT UF FROM PESSOA;
+------+
| UF   |
+------+
| AM   |
| GO   |
| BA   |
| GO   |
+------+
```

No entanto, note que a consulta simplesmente recupera o campo 'uf' de cada registro, e alguns deles aparecem mais do que uma vez. Para minimizar a saída, recupere cada saída de registro unicamente adicionando a palavra-chave 'DISTINCT':

```
     mysql> SELECT DISTINCT UF FROM PESSOA;
+------+
| UF   |
+------+
| AM   |
| BA   |
| GO   |
+------+
```

12.7.4.4 Ordenando linhas

Você deve ter notado nos exemplos anteriores que as linhas resultantes são mostradas em uma ordem particular. No entanto, é mais fácil examinar a saída de uma consulta quando as linhas são ordenadas em uma forma significativa. Para ordenar um resultado, use uma cláusula 'ORDER BY'.

Aqui estão os aniversários das pessoas, ordenadas pela data:

```
     mysql> SELECT nome, data_nascimento FROM PESSOA ORDER BY data_nascimento;
+---------------+-----------------+
| nome          | data_nascimento |
+---------------+-----------------+
| Ana Souza     | 1955-04-04      |
| José Pereira  | 1961-03-03      |
| Maria Nunes   | 1971-11-22      |
| João da Silva | 1984-11-11      |
+---------------+-----------------+
```

Para ordenar em ordem decrescente, adicione a palavra−chave 'DESC' (descendente) para o nome da coluna que você está ordenando:

```
     mysql> SELECT nome, data_nascimento FROM PESSOA ORDER BY data_nascimento DESC;
+--------------+-----------------+
| nome         | data_nascimento |
+--------------+-----------------+
| João da Silva | 1984−11−11      |
| Maria Nunes   | 1971−11−22      |
| José Pereira  | 1961−03−03      |
| Ana Souza     | 1955−04−04      |
+--------------+-----------------+
```

12.7.4.5 Casamento de padrões

Para encontrar nomes começando com 'J':

```
     mysql> SELECT cpf, nome FROM PESSOA WHERE nome LIKE "J%";
+-----------+---------------+
| cpf       | nome          |
+-----------+---------------+
| 1111      | João da Silva |
| 3333      | José Pereira  |
+-----------+---------------+
```

Para encontrar nomes que terminam com 'a':

```
     mysql> SELECT cpf, nome FROM PESSOA WHERE nome LIKE "%a";
+-----------+---------------+
| cpf       | nome          |
+-----------+---------------+
| 1111      | João da Silva |
| 3333      | José Pereira  |
| 4444      | Ana Souza     |
+-----------+---------------+
```

12.7.4.6 Contando linhas

Banco de dados são geralmente usados para responder a questão, "quanto um determinado tipo de dados ocorrem em uma tabela?". Por exemplo, você pode querer saber quantas pessoas você possui, ou quantas pessoas existem por um determinado tipo de estado ou bairro.

Para contar o número total de pessoas:

```
     mysql> SELECT COUNT(*) FROM PESSOA;
+----------+
| COUNT(*) |
+----------+
|        4 |
+----------+
```

Para selecionar o número de pessoas, agrupado por estado:

```
     mysql> SELECT UF, COUNT(*) FROM PESSOA GROUP BY UF;
+------+----------+
| UF   | COUNT(*) |
+------+----------+
| AM   |        1 |
| BA   |        1 |
| GO   |        2 |
+------+----------+
```

Note o uso de 'GROUP BY' para agrupar todos registros para cada estado.

12.7.4.7 Restrições

Suponha que você deseje evitar que uma pessoa seja cadastrada duas vezes. Isto pode ser evitado elegendo uma coluna como chave. Em termos técnicos, devemos eleger uma chave primária. A chave primária por si deve refletir algum dado que não se repete. Por exemplo, se elegermos a coluna nome como chave primária podemos estar cometendo um erro, pois podem ter pessoas homônimas (mesmo nome). Por outro lado, se escolhermos a coluna cpf será uma boa escolha, pois sabemos que nunca teremos um cpf repetido.

Por definição, uma chave primária nunca deve admitir nulos.

Para definir uma coluna como chave primária, você pode utilizar a cláusula PRIMARY KEY na criação da tabela:

```
CREATE TABLE PESSOA (
CPF CHAR(11) NOT NULL PRIMARY KEY,
...
```

Para definir uma coluna como chave primária, para uma tabela já existente, execute o comando ALTER TABLE com a cláusula PRIMARY KEY.

```
mysql> ALTER TABLE PESSOA ADD PRIMARY KEY (CPF);

mysql> ALTER TABLE EMPRESA ADD PRIMARY KEY (CGC);
```

12.7.4.8 Cláusula auto_increment

Na tabela PESSOA, você deve ter notado que as colunas CIDADE e BAIRRO contém valores inteiros. Na verdade, estas colunas fazem referência a outros valores de outras tabelas; respectivamente às tabelas CIDADE e BAIRRO.

Execute os próximos comandos. Nestes comandos é introduzido a cláusula auto_increment, que indica que o valor será auto incrementado quando for criado uma nova linha:

```
mysql> CREATE TABLE BAIRRO (
    -> cod_bairro int unsigned not null auto_increment primary key,
    -> descricao char(40) not null);

mysql> CREATE TABLE CIDADE (
    -> cod_cidade int unsigned not null auto_increment primary key,
    -> descricao char(40) not null);

mysql> CREATE TABLE CONTATO (
    -> num_contato int unsigned not null auto_increment primary key,
    -> tipo_cliente char(1),
    -> cpf_ou_rg char(20),
    -> cgc char(20),
    -> data date,
    -> observacao text);
```

Para popular as tabelas CIDADE, BAIRRO, EMPRESA e CONTATO, execute os comandos seguintes, que irão carregar valores para o banco de dados a partir dos arquivos cidade.txt, bairro.txt, empresa.txt e contato.txt respectivamente . Caso não tenha sido criado todos os arquivos , crie−os , conforme já foi explicado , e faça o carregamento dos mesmos no banco da seguinte maneira.

```
mysql> LOAD DATA LOCAL INFILE "bairro.txt" INTO TABLE BAIRRO;
mysql> LOAD DATA LOCAL INFILE "cidade.txt" INTO TABLE CIDADE;
mysql> LOAD DATA LOCAL INFILE "empresa.txt" INTO TABLE EMPRESA;
```

```
mysql> LOAD DATA LOCAL INFILE "contato.txt" INTO TABLE CONTATO;
```

Para experimentar a propriedade de auto incremento, execute:

```
mysql> select num_contato, tipo_cliente, cpf, cgc, data FROM CONTATO;
+-------------+--------------+-----------+--------+------------+
| num_contato | tipo_cliente | cpf       | cgc    | data       |
+-------------+--------------+-----------+--------+------------+
|           1 | P            | 1111      | NULL   | 2000-04-21 |
|           2 | P            | 2222      | NULL   | 2000-04-21 |
|           3 | E            | NULL      | 121212 | 2000-04-21 |
|           4 | P            | 1111      | NULL   | 2000-04-22 |
+-------------+--------------+-----------+--------+------------+

mysql> INSERT INTO CONTATO ('P', '6666', NULL, '2000-04-23', 'Precisa do programa do curso d

mysql> select num_contato, tipo_cliente, cpf_ou_rg, cgc, data FROM CONTATO;
+-------------+--------------+------------+--------+------------+
| num_contato | tipo_cliente | cpf_ou_rg  | cgc    | data       |
+-------------+--------------+------------+--------+------------+
|           1 | P            | 1111       | NULL   | 2000-04-21 |
|           2 | P            | 2222       | NULL   | 2000-04-21 |
|           3 | E            | NULL       | 121212 | 2000-04-21 |
|           4 | P            | 1111       | NULL   | 2000-04-22 |
|           5 | P            | 6666       | NULL   | 2000-04-23 |
+-------------+--------------+------------+--------+------------+
```

Note neste exemplo, que no comando INSERT, não foi necessário fornecer o valor da coluna num_contato. Automaticamente o banco de dados verifica o maior valor existente e insere o valor seguinte para a coluna com propriedade de auto−incremento.

12.7.5 Usando múltiplas tabelas

É muito comum necessitarmos de informações que somente são obtidas cruzando informações entre tabelas. Por exemplo: Qual é o email pessoal dos responsáveis por cada empresa ? Qual é o dia de aniversário dos responsáveis pela empresa ? Quantos contatos existem por estado (unidade de federação) ? Na verdade, podemos criar inúmeras consultas, cruzando informações entre tabelas através de uma operação denominada junção [join].

Como exemplo, vamos obter o email dos responsáveis por empresas:

```
mysql> SELECT B.NOME_FANTASIA, A.NOME, A.EMAIL
    -> FROM PESSOA A, EMPRESA B
    -> WHERE A.CPF = B.RESPONSAVEL;

+---------------+---------------+------------------+
| NOME_FANTASIA | NOME          | EMAIL            |
+---------------+---------------+------------------+
| Mabel         | João da Silva | joao@uol.com.br  |
| Arisco        | Maria Nunes   | maria@bol.com.br |
+---------------+---------------+------------------+
```

Note:

- É utilizado as letras 'A' e 'B' como apelidos [alias] para as tabelas PESSOA e EMPRESA, respectivamente.
- Você pode referenciar colunas de tabelas simplesmente indicando o alias da tabela, 'ponto' e o nome da coluna. Por exemplo, B.NOME_FANTASIA é a coluna NOME_FANTASIA da tabela EMPRESA.
- Todo join tem uma condição de junção, especificada na parte de 'WHERE' da cláusula SELECT.

Neste caso, a condição de junção é que o responsável da empresa tenha o mesmo número de cpf da tabela de pessoas.

12.8 Alterando linhas

Para se alterar alguma linha, é utilizado o comando update, com a seguinte sintaxe:

```
UPDATE
SET =
WHERE =
```

Basicamente, é necessário apenas especificar a tabela em que ser quer fazer a atualização; o campo e seu novo valor; e finalmente, a condição de filtragem. Por exemplo:

```
mysql> select cpf, nome
    -> from pessoa;
+------+----------------+
| cpf  | nome           |
+------+----------------+
| 1111 | Joao da Silva  |
| 2222 | Maria Nunes    |
| 3333 | Jose da Silva  |
| 4444 | Ana Souza      |
+------+----------------+

mysql> update pessoa
    -> set nome="Joao da Silva Pereira"
    -> where cpf="1111";

mysql> select cpf, nome
    -> from pessoa;
+------+-----------------------+
| cpf  | nome                  |
+------+-----------------------+
| 1111 | Joao da Silva Pereira |
| 2222 | Maria Nunes           |
| 3333 | Jose Pereira          |
| 4444 | Ana Souza             |
+------+-----------------------+
```

12.9 Deletando linhas

Para se alterar alguma linha, é utilizado o comando update, com a seguinte sintaxe:

```
DELETE FROM
WHERE =
```

Basicamente, é necessário apenas especificar a tabela em que ser quer fazer a deleção; e a condição de filtragem. Deve se ter cuidado com este comando, pois o simples comando DELETE sem a cláusula WHERE, significa a deleção de todas as linhas da tabela, Por exemplo:

```
mysql> select cpf, nome
    -> from pessoa;
+------+----------------+
| cpf  | nome           |
+------+----------------+
| 1111 | Joao da Silva  |
| 2222 | Maria Nunes    |
| 3333 | Jose da Silva  |
| 4444 | Ana Souza      |
+------+----------------+
```

```
mysql> delete from pessoa
    -> where cpf="1111";

mysql> select cpf, nome
    -> from pessoa;
+------+----------------------+
| cpf  | nome                 |
+------+----------------------+
| 2222 | Maria Nunes          |
| 3333 | Jose Pereira         |
| 4444 | Ana Souza            |
+------+----------------------+

mysql> delete from pessoa;

mysql> select cpf, nome
    -> from pessoa;
Empty set (0.00 sec)
```

13 Conectando PHP com MySQL

13.1 Introdução

Agora que você já tem uma idéia básica de comandos MySQL, poderemos ver como a linguagem PHP pode interagir com este banco de dados através de inúmeras funções. Funções de conexão de MySQL são chamadas de APIs (Application Programming Interfaces), mas iremos tratá−las apenas como funções para um melhor entendimento.

13.2 Conexões MySQL

Veremos agoras as funções mais comuns para realizar a conexão entre PHP e MySQL, a maioria destas funções retornam TRUE em caso de sucesso e FALSE em caso de falha, quando o valor retornado for de outro tipo, diremos na descrição da respectiva função. Vejamos:

13.2.1 mysql_connect()

Realiza uma conexão inicial para MySQL, utilizando como parâmetros: "hostname" (nome do servidor), "user" (usuário MySQL que esta sendo conectado) e "password" (senha do usuário que esta se conectando ao banco de dados).

```
mysql_connect("localhost", "administrador", "12345");
```

Lembrando que: "localhost" é o "hostname", "administrador" é o "user" e "12345" é o "password". É possível também realizar uma conexão anônima apenas deixando os compos "user" e "password" em branco. Exemplo:

```
mysql_connect("localhost", "", "");
```

Usa−se um "Link de Conexão" (no exemplo a seguir: $conexao) para confirmarmos de que a conexão foi ou não realizada:

```
$conexao = mysql_connect("localhost", "administrador", "12345");
```

13.2.2 mysql_pconnect()

Chamada de conexão persistente, age da mesma forma que a função anterior, sendo que neste caso, quando o MySQL é deixado, a conexão é mantida aberta, e quando o usuário retornar, a mesma conexão será utilizada.

```
mysql_pconnect("localhost", "administrador", "12345");
```

13.2.3 mysql_close()

Simplesmente fecha uma conexão atual, note que esta função utiliza um "link de conexão".

```
mysql_close($conexao);
```

13.2.4 mysql_create_db()

Cria um novo banco de dados (database) para ser utilizado futuramente:

```
mysql_create_db($nomedatabase, $conexao);
```

$conexao usa a conexão atual − é opcional o seu uso.

13.2.5 mysql_drop_db()

Exclui um database anteriormente criado:

```
mysql_drop_db($nomedatabase, $conexao);
```

$conexao usa a conexão atual − é opcional o seu uso.

13.2.6 mysql_select_db()

Seleciona um database para ser trabalhado:

```
mysql_select_db($nomedatabase, $conexao);
```

$conexao usa a conexão atual − é opcional o seu uso.

13.2.7 mysql_query()

Envia a declaração SQL para o servidor MySQL:

```
mysql_query($declaracaoSQL, $conexao);
```

$declaracaoSQL é uma string contendo a declaração SQL como: CREATE, ALTER, DROP, DELETE, INSERT e UPDATE.
$conexao usa a conexão atual − é opcional o seu uso.

13.2.8 mysql_db_query()

Semelhante à função anterior, mas agora há um parâmetro adicional que especifica o nome do database ativo:

```
mysql_db_query($nomedatabase, $declaracaoSQL, $conexao);
```

$nomedatabase é uma string com o nome do banco de dados.
$declaracaoSQL é uma string contendo a declaração SQL como: CREATE, ALTER, DROP, DELETE, INSERT e UPDATE.
$conexao usa a conexão atual − é opcional o seu uso.

13.2.9 mysql_list_dbs()

Lista todos os database disponíveis no servidor.

```
mysql_list_dbs($conexao);
```

$conexao usa a conexão atual − é opcional o seu uso.

13.2.10 mysql_list_tables()

Lista todas as tabelas disponíveis dentro do database especificado.

```
mysql_list_tables($nomedatabase, $conexao);
```

$nomedatabase contém o nome do database atual.
$conexao usa a conexão atual − é opcional o seu uso.

13.2.11 mysql_list_fields()

Lista todos os campos de uma tabaela especificada.

```
mysql_list_fields($nomedatabase, $nometabela, $conexao);
```

$nomedatabase contém o nome do database onde a tabela se encontra.
$nometabela − nome da tabela.
$conexao usa a conexão atual − é opcional o seu uso.

13.2.12 mysql_result()

Obtém os dados retornados através das funções *mysql_query, mysql_db_query, mysql_list_tables ou mysql_list_dbs.*

```
mysql_result($resultado);
```

$resultado é o *identificador resultado* que deve ser obtido da seguinte maneira, lembrando que a função poder ser *mysql_query, mysql_db_query, mysql_list_tables ou mysql_list_dbs*:

```
$resultado = mysql_query($declaracaoSQL, $conexao);
```

É possível escolher a linha e o campo pelos quais o resultado será obtidos, veja:

```
mysql_result($resultado, $linha, $campo);
```

A função retornará o conteúdo da linha *$linha* e a coluna *$campo* de $resultado.

13.2.13 mysql_num_fields()

Retorna o número de campos existentes dentro de $resultado. Veja:

```
mysql_num_fields($resultado);
```

$resultado novamente é o *identificador resultado* descrito na função anterior.

13.2.14 mysql_num_rows()

Retorna o número de linhas existentes dentro de $resultado provindo de uma declaração SELECT. Veja:

```
mysql_num_rows($resultado);
```

$resultado novamente é o *identificador resultado*, mas agora provindo apenas de uma declaração SELECT.

13.2.15 mysql_affected_rows()

Retorna o número de linhas afetadas na consulta SQL anterior, podendo estas declarações serem apenas: INSERT, DELETE, REPLACE ou UPDATE.

```
mysql_affected_rows($conexao);
```

$conexao usa a conexão atual − é opcional o seu uso.

13.2.16 mysql_fetch_array()

Obtém o resultado de uma declaração SQL, e coloca cada linha obtida em um campo de um array específico, veja:

```
$linhas = mysql_fetch_array($resultado, $conexao);
```

O array $linhas comportará todas as linhas provindas de $resultado, lembrando que:

$resultado novamente é o *identificador resultado* retornado por *mysql_query, mysql_db_query, mysql_list_tables ou mysql_list_dbs*.
$conexao usa a conexão atual.

13.2.17 mysql_insert_ID()

Retorna o ID de incrementação automática gerado pelo último comando SQL INSERT executado em uma tabela que continha uma coluna AUTO_INCREMENT.

```
mysql_insert_ID($conexao);
```

$conexao usa a conexão atual − é opcional o seu uso.

13.3 Primeira Aplicação

Iremos agora desenvolver um script PHP para facilitar a compreensão das funções acima. O script irá imprimir na tela todos os campos ('titulo', 'conteudo' e 'link') da tabela 'noticias' que se encontra dentro do database 'portal_de_noticias'. Vejamos o código:

Este arquivo irá criar o database portal_de_noticias, a tabela noticias com suas respectivas colunas e inserir dados em seus campos.

portal.sql

```
CREATE DATABASE portal_de_noticias;
use portal_de_noticias;
CREATE TABLE noticias (titulo char(50) not null,conteudo char(200),link char(50));
INSERT INTO noticias VALUES ('Sistemas Abertos','informatica ltda','www.sistemasabertos.com.br');

Obs: para carrega−lo conecte−se so banco de dados e digite o seguinte comando:
mysql> source /caminho_na_árvore_de_diretórios/portal.sql
```

firstconection.php

```
<?php

$conexao = mysql_connect("localhost", "", "");
//realiza a conexão com o servidor MySQL
if (!$conexao)
{
        echo "Conexão ao servidor MySQL não realizada!";
        exit;
}
//verifica se a conexao foi realizada com sucesso

$db = mysql_select_db("portal_de_noticias", $conexao);
//seleciona o database a ser trabalhado
if (!$db)
{
```

```
        echo "Mudança de database não realizada!";
        exit;
}
//verifica se a seleção do database foi realizada com sucesso

$consultasql = "SELECT * FROM noticias";

$query = mysql_query($consultasql, $conexao);
//realiza a consulta MySQL

$num_linhas = mysql_num_rows($query);
//retorna o nº de linhas da consulta realizada

if ($num_linhas == 0)
{
        echo "Não há nenhum dado!";
} else {
        //coloca o resultado da consulta em um array e imprime cada elemento
        while ($linha = mysql_fetch_array($query))
        {
                $titulo = $linha["titulo"];
                $conteudo = $linha["conteudo"];
                $link = $linha["link"];

                echo "$titulo: $conteudo <br> $link";
        }
}
mysql_close($conexao);
//fecha a conexão com o servidor MySQL
?>
```

Lembre−se:

- As vezes mensagens de erro podem ser geradas pelas funções de conexão, para desabilitar tal ação, utilize um'@' no início de sua função, veja:

```
$conexao = @mysql_connect("localhost", "", "");
```

13.4 Preenchendo Menus

Este tópico irá focalizar a interação entre o usuário e o formulário. Faremos um script onde estará impresso na tela todos os títulos das notícias disponíveis para que o usuário possa fazer sua escolha, e em seguida, será impresso o título, conteúdo e link da notícia selecionada.

Para evitarmos de ficar repetindo o mesmo código para realizar a conexão com o servidor MySQL, criaremos um script apenas com esta função, e depois iremos incluí−lo em nosso projeto. Veja o script:

conection.php

```
<?php
$conexao = mysql_connect("localhost", "root", "");
//realiza a conexão com o servidor MySQL
if (!$conexao)
{
        echo "Conexão ao servidor MySQL não realizada!";
        exit;
}
?>
```

selecao.php

```
<HTML>

<BODY><CENTER><b>Selecione a notícia desejada:</b></CENTER>

<?php

include ("conection.php");

$db = mysql_select_db("portal_de_noticias", $conexao);

if (!$db)
{
        echo "Mudança de database não realizada!";
        exit;
}

$consultasql = "SELECT titulo FROM noticias";
//consulta o campo 'titulo' da tabela 'noticias'

$query = mysql_query($consultasql, $conexao);

$num_linhas = mysql_num_rows($query);

if ($num_linhas == 0)
{
        echo "Não há nenhum dado!";
        //caso a tabela esteja vazia
} else {

        echo "<form method = get action = 'ver_dados.php'>";
        echo "<select name = 'form'>";
        while ($linha = mysql_fetch_array($query))
        {
                $titulo = $linha["titulo"];
                //coloca os valores do campo 'titulo' no array 'linha'
                echo "<option value = '$titulo'>$titulo";
                //imprime cada valor do campo 'titulo' para a seleção
        }
        echo "</select>";
}
echo "<br><br>";
echo "<input type = 'submit' value = 'Mandar dados!'>";
echo "<input type = 'reset' value = 'Limpar!'>";
mysql_close($conexao);

?>
</BODY>
</HTML>
```

Muito cuidado na hora de implementar o código de seleção, em nosso exemplo, nossa tabela é simples e não contém valores duplicados, ou seja, com o mesmo título, mas caso tivesse acontecido, seria necessário usar um campo específico para cada notícia. Para isto usaríamos um ID.

Cada notícia teria seu próprio ID, não interessando seu título, conteúdo ou link, resolvendo este pequeno impasse. Vejamos agora como ficaria o código anterior com a inclusão do campo ID:

selecaoid.php

```
<HTML>

<BODY><CENTER><b>Selecione a notícia desejada:</b></CENTER>

<?php

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

```
$db = mysql_select_db("portal_de_noticias", $conexao);

if (!$db)
{
        echo "Mudança de database não realizada!";
        exit;
}

$consultasql = "SELECT titulo, id FROM noticias";
//agora também selecionamos o campo id

$query = mysql_query($consultasql, $conexao);

$num_linhas = mysql_num_rows($query);
if ($num_linhas == 0)
{
        echo "Não há nenhum dado!";
} else {

        echo "<form method = get action = 'ver_dados.php'>";
        echo "<select name = 'form'>";
        while ($linha = mysql_fetch_array($query))
        {
                $titulo = $linha["titulo"];
                $id = $linha["id"];
                //note que o campo 'id' também sendo passado para o array com seu respectivo 'tit

                echo "<option value = '$id'>$titulo";
                //note que o valor de 'form' será $id e não $titulo como anteriormente
        }
        echo "</select>";
}
echo "<br><br>";
echo "<input type = 'submit' value = 'Mandar dados!'>";
echo "<input type = 'reset' value = 'Limpar!'>";
mysql_close($conexao);

?>
</BODY>
</HTML>
```

Agora criaremos o script que imprimirá na tela os dados da notícia selecionada através do script *selecao.php*.

ver_dados.php

```
<HTML>

<BODY>

<?php

include ("conection.php");

$db = mysql_select_db("portal_de_noticias", $conexao);

if (!$db)
{
        echo "Mudança de database não realizada!";
        exit;
}

$consultasql = "SELECT * FROM noticias WHERE titulo = '$form'";
//selecionamos todos os dados sobre o título selecionado

$query = mysql_query($consultasql, $conexao);
```

```
$num_linhas = mysql_num_rows($query);
if ($num_linhas == 0)
{
        echo "Não há nenhum dado!";
} else {
        //faremos uma tabela para melhor visualizar os valores obtidos
        echo "<table align = 'center' border = '3'>";
        echo "<tr><th>Titulo</th><th>Conteúdo</th><th>Link</th></tr>";

        while ($linha = mysql_fetch_array($query))
        {
                $titulo = $linha["titulo"];
                $conteudo = $linha["conteudo"];
                $link = $linha["link"];
                echo "<tr><td>$titulo</td><td>$conteudo</td><td>Link</td></tr>";
        }
}

mysql_close($conexao);

?>
</table>
//note que a tabela html é fechada fora do código PHP
<br><a href = "selecao.php">Voltar!</a>
</BODY>
</HTML>
```

Caso estivéssemos utilizando o scrip *selecaoid.php* faríamos apenas 2 modificações:

- Mudança de $consultasql:

```
$consultasql = "SELECT * FROM noticias WHERE ID = '$form'";
```

- Alteração do link de retorno:

```
<a href = "selecaoid.php">Voltar!</a>
```

13.5 Inserindo um registro

Um site que tem a capacidade de alterar informações é sinônimo de dinamicidade. Criaremos agora um script para o usuário inserir um registro e logo depois verificarmos as mudanças.

Primeiro criaremos um formulário para que o usuário coloque todos os seus dados pessoais. Usaremos o database 'dados_pessoais' que contém uma tabela MySQL 'paises' com os campos 'pais'(para que o usuário selecione o país desejado) e 'id'(cada uma de um país) e a tabela 'dados' para armazenar os dados do usuário. Veja:

forminserir.php

```
<HTML>

<BODY>
<CENTER><B>Insira seus dados em sua ficha pessoal</B></CENTER>
<?php

include ("conection.php");

$db = mysql_select_db("dados_pessoais", $conexao);

if (!$db)
{
```

```
        echo "Seleção de database não realizada!";
        exit;
}
//função para verificar caracteres especiais nos campos do formulário
function decodifica($str){
        $str = urldecode($str);
        $str = stripslashes($str);
        $str = htmlspecialchars($str);
        return ($str);
}


$consultasql = "SELECT pais FROM paises";

$query = mysql_query($consultasql, $conexao);

$num_linhas = mysql_num_rows($query);
if ($num_linhas == 0)
{
        echo "Não há nenhum dado!";
} else {

        echo "<form method = get action = 'inserir.php'>";
        //as próximas 3 linhas serão utilizadas se o script se repetir
        $nome = decodifica($nome);
        $endereco = decodifica($endereco);
        $cidade = decodifica($cidade);
        //preenchimento dos campos
        echo "Nome completo: <br> <input type = 'text' name = 'nome' value = '$nome'><br>";
        echo "Endereço: <br> <input type = 'text' name = 'endereco' value = '$endereco'><br>";
        echo "Cidade: <br> <input type = 'text' name = 'cidade' value = '$cidade'><br>";
        echo "Selecione seu país:<br>";
        echo "<select name = 'pais'>";
        //exibição dos países da tabela 'paises'
        while ($linha = mysql_fetch_array($query))
        {
                $pais = $linha["pais"];

                echo "<option>$pais";
        }
        echo "</select>;
}

        echo "<br><br>";
        echo "<input type = 'submit' value = 'Enviar dados!'>;
        echo "</form>";

        echo "<form method = get action = '$PHP_SELF'>";
        echo "<input type = 'submit' value = 'Limpar dados!'>";

        mysql_close($conexao);
        ?>
        </form>

        </BODY>

        </HTML>
```

Agora vamos explicar mais detalhadamente o script acima. Se o formulário for enviado de volta devido à uma validação falha (como campos faltando), precisaremos decodificar a informação da cadeia de consulta(para campos que foram acessados). Isto evita que usuários tenham que redigitar informações. Assim decodificamos a cadeia de consulta(urldecode), habilitamos caracteres especiais como ", % e !(stripslashes) e protegemos caracteres como & e >(htmlspecialchars). Todas estas verificações foram feitas no código abaixo:

```
function decodifica($str){
```

```
        $str = urldecode($str);
        $str = stripslashes($str);
        $str = htmlspecialchars($str);
        return ($str);
}
```

Se o usuário clicar em *Enviar Dados!*, e alguns campos não estiverem preenchidos, o formulário será enviado de volta com as informações nos campos que ele entrou, assim como dito anteriormente. No entanto, isto representaria um problema: se o usuário decidir zerar os campos (usando a reiniciação), os campos que continham dados ainda preencherão os campos que o usuário acabou de entrar.

Isto ocorre porque a cadeia de consulta ainda conterá informações do *Enviar Dados!* anterior, ou seja, as variáveis do formulário que haviam sido preenchidas, ainda estarão preenchidas com os valores anteriores.

Foi por isso que utilizamos o seguinte código:

```
echo "<form method = get action = '$PHP_SELF'>";
echo "<input type = 'submit' value = 'Limpar dados!'>";
```

Assim, criamos o nosso próprio botão especial de reinicialização que chama a si mesmo, forçando uma limpeza completa em qualquer cadeia de consulta.

Vejamos agora o script que irá manipular os dados de nosso formulário:

inserir.php

```
<HTML>

<BODY>

<?php

include ("conection.php");

$db = mysql_select_db("dados_pessoais", $conexao);

if (!$db)
{
        echo "Seleção de database não realizada!";
        exit;
}
//se o formulário voltar, protegeremos os dados já preenchidos
function protecao($str){
        $str = urlencode($str);
        $str = stripslashes($str);
        return ($str);
}

//verificação se os campos estão vazios
$erro = false;
if ($nome == " ")
{
        $erro = true;
        echo "Campo nome não preenchido!<br>";
}

if ($endereco == " ")
{
        $erro = true;
        echo "Campo endereço não preenchido!<br>";
}

if ($cidade == " ")
```

```
{
        $erro = true;
        echo "Campo cidade não preenchido!<br>";
}

//se os campos estiverem vazios, fazemos a proteção de caracteres para o retorno ao formulário
if ($erro)
{
        $nome = protecao($nome);
        $endereco = protecao($endereco);
        $cidade = protecao($cidade);
        //o usuário e os dados já preenchidos são direcionados novamente ao formulário original
        echo "<br><a href = 'forminserir.php?nome=$nome&endereco=$endereco&cidade=$cidade'> Volta
        exit;
}

//inserção dos novos dados
$consultasql = "INSERT INTO dados (nome, endereco, cidade, pais) VALUES ('$nome, '$endereco', '$c

if (!$query = mysql_query($consultasql, $conexao))
{
        echo "Erro, dados não gravados, aperte o botão de retorno e tente novamente!<br>";
        exit;
} else {
        echo "Tabela alterada [" .mysql_affected_rows() ."] dado(s) gravado(s)!<br>";
}

echo "<br><a href = 'forminserir.php'>Voltar!</a>";

?>
</BODY>

</HTML>
```

Para realizar a verificação se um campo foi preenchido ou não, atribuimos 'true' ou 'false' à variável $erro. Inicialmente, inicializamos a variável como 'false', e se algum campo estiver vazio, alteramos seu valor para 'true'. A partir da linha 24 é feita esta verificação em todos os campos.

Primeiramente, se qualquer um dos campos não estiver preenchido, a consulta será codificada antes de ser enviada de volta para o formulário, veja o código:

```
if ($erro)
{
        $nome = protecao($nome);
        $endereco = protecao($endereco);
        $cidade = protecao($cidade);
```

Primeiro a cadeia é codificada, depois qualquer caractere especial é protegido e finalmente removemos qualquer proteção de caracteres como ^, & e (,). Deixamos o resto da codificação para o script *forminserir.php*.

Logo após a proteção dos dados, o usuário é direcionado de volta ao formulário original junto com as informações já preenchidas através de um hiperlink com valores anexados à cadeia URL. Veja:

```
echo "<br><a href = 'forminserir.php?nome=$nome&endereco=$endereco&cidade=$cidade'> Voltar!</a>";
exit;
}
```

13.6 Atualização de Registro

No tópico anterior aprendemos a inserir um registro, e se agora quiséssemos alterar apenas o endereço ou

adicionar um sobrenome? Para isto utilizaremos o campo ID de cada registro, pois ele é a melhor maneira de identificar seus registros de forma única.

Uma maneira possível de se realizar tal atualização seria criar uma lista de nomes a serem atualizados, para que assim o usuário selecione a opção desejada. Observe o código abaixo:

form_selecao.php

```
<HTML>

<BODY><CENTER><b>Selecione a notícia desejada:</b></CENTER>

<?php

include ("conection.php");

$db = mysql_select_db("dados_pessoais", $conexao);

if (!$db)
{
        echo "Seleção de database não realizada!";
        exit;
}

$consultasql = "SELECT nome, endereco, cidade, pais, ID FROM dados ORDER BY pais";

$query = mysql_query($consultasql, $conexao);
$num_linhas = mysql_num_rows($query);

if ($num_linhas == 0)
{
        echo "Não há nenhum dado!";
} else {

        echo "<table align = 'center' border = '8'>";
        echo "<tr><th>Nomes</th><th>Endereços</th><th>Países</th></tr>";
        while ($linha = mysql_fetch_array($query))
        {
                $nome = $linha["nome"];
                $endereco = $linha["endereco"];
                $cidade = $linha["cidade"];
                $pais = $linha["pais"];
                $id = $linha["id"];

                echo "<tr><td><a href = 'modif.php?linha_id=$id'>$nome</a></td><td>$endereco</td>
        }
}

mysql_close($conexao);

?>

</table>

</BODY>

</HTML>
```

Este script imprime na tela uma tabela para que o usuário clique no nome desejado, tendo acesso a um hiperlink para realizar suas modificações. O único trecho de código que pode parecer novo a seus olhos é a seguinte linha de comando:

```
echo "<tr><td><a href = 'modif.php?linha_id=$id'>$nome</a></td><td>$endereco</td><td>$pais</td></
```

Este trecho é a implementação do hiperlink que levará ao script 'modif.php', onde as modificações serão realizadas. Note que 'linha_id' será a variável responsável para a identificação de qual registro deve ser modificado. Vejamos o código do próximo script:

modif.php

```
<HTML>

<BODY><CENTER><b>Realize as atualizações desejadas:</b></CENTER>

<?php

include ("conection.php");

$db = mysql_select_db("dados_pessoais", $conexao);

if (!$db)
{
        echo "Seleção de database não realizada!";
        exit;
}

$sql1 = "SELECT * FROM dados WHERE ID = '$linha_id'";
$sql2 = "SELECT pais FROM paises";

$query = mysql_query($sql1, $conexao);
$num_linhas = mysql_num_rows($query);

if ($num_linhas == 0)
{
        echo "Não há nenhum dado!";
} else {

        while ($linha = mysql_fetch_array($query))
        {
                $nome = $linha["nome"];
                $endereco = $linha["endereco"];
                $cidade = $linha["cidade"];
                $pais = $linha["pais"];
                $id = $linha["id"];
        }
        echo "<form method =get action = 'modif_agora.php'>";
        echo "Nome<br><input type = 'text' name = nome value = '$nome'><br>";
        echo "Endereço<br><input type = 'text' name = endereco value = '$endereco'><br>";
        echo "Cidade<br><input type = 'text' name = cidade value = '$cidade'><br>";
        echo "País<br>";

        echo "<select name = 'pais'>";
        $query2 = mysql_query($sql2, $conexao);
        echo "<option selected>$pais";
        while ($linha = mysql_fetch_array($query2))
        {
                $pais = linha["pais"];
                echo "<option>$pais";
        }

        echo "</select>";
        echo "<br><br>";

        echo "<input type = 'HIDDEN' name= 'ID' value = '$id'>";
        echo "<input type = submit value = 'Atualizar registro!'>";
} //fim do else

mysql_close($conexao);

?>
```

```
</form>

</BODY>

</HTML>
```

Após ter realizado, ou não, as devidas modificações, o usuário clica no botão para que as atualizações sejam feitas. Para passarmos o valor de $id para o 3º script, usamos campos HIDDEN, pois estes campos não eram parte do input do formulário, assim, precisamos de uma outra forma para passá−los.

modif_agora.php

```
<HTML>

<BODY>

<?php

include ("conection.php");

$db = mysql_select_db("dados_pessoais", $conexao);

if (!$db)
{
        echo "Seleção de database não realizada!";
        exit;
}

$nome = (urldecode($nome));

$erro = false;
if ($nome == " ")
{
        $erro = true;
        echo "Campo nome não preenchido!<br>";
}

if ($endereco == " ")
{
        $erro = true;
        echo "Campo endereço não preenchido!<br>";
}

if ($cidade == " ")
{
        $erro = true;
        echo "Campo cidade não preenchido!<br>";
}

if ($erro)
{
        echo "Todos os campos devem estar preenchidos! Preencha o formulário corretamente!";
        echo "<br><a href = 'form_selecao.php'> Voltar!</a>";
        exit;
}

$sql = "UPDATE dados SET nome = '$nome', endereco = '$endereco', cidade = '$cidade', pais = '$pai

if (!$query = mysql_query($consultasql, $conexao))
{
        echo "Erro, atualização não realizada!<br>";
        echo "<br><a href = 'form_selecao.php'> Voltar!</a>";
        exit;
} else {
```

```
        echo "Tabela atualizada [" .mysql_affected_rows() ."] dado(s) atualizados(s)!<br>";
}

echo "<br><a href = 'form_selecao.php'> Voltar!</a>";
mysql_close($conexao);
?>
</BODY>

</HTML>
```

Novamente realizamos a decodificação da variável $nome e depois verificamos se os outros campos estão devidamente preenchidos utilizando a variável $erro.

Note que todos os campos são atualizados e não apenas os selecionados.

13.7 Excluindo um Registro

Agora que já trabalhamos com tantos scripts, a exclusão de um registro será muito fácil de ser implementada. Para a seleção do registro a ser excluído, utilizaremos o script *form_selecao.php* novamente, logicamente, modificando o link para direcionar para nosso novo script que confirmará a deleção.

Realize a seguinte modificação em *form_selecao.php*:

```
echo "<tr><td><a href = 'apagar.php?linha_id=$id'>$nome</a></td><td>$endereco</td><td>$pais</td><
```

Vamos ao script:

apagar.php

```
<HTML>

<BODY>
<?php

include ("conection.php");

$db = mysql_select_db("dados_pessoais", $conexao);

if (!$db)
{
        echo "Seleção de database não realizada!";
        exit;
}

$sql = "SELECT * FROM dados WHERE ID = '$linha_id'";

$query = mysql_query($sql, $conexao);
$num_linhas = mysql_num_rows($query);

if ($num_linhas == 0)
{
        echo "Não há nenhum dado!";
} else {
        echo "<form method = get action = 'apagar_agora.php'>";
        while ($linha = mysql_fetch_array($query))
        {
                $nome = $linha["nome"];
                $endereco = $linha["endereco"];
                $cidade = $linha["cidade"];
                $pais = $linha["pais"];

                echo "Nome<br><b>$nome</b><br><br>";
```

```
                echo "Endereço<br><b>$endereco</b><br><br>";
                echo "Cidade<br><b>$cidade</b><br><br>";
                echo "País<br><b>$pais</b><br><br>";
                echo "<br><br>";
        }

        echo "<input type = 'HIDDEN' name= 'ID' value = '$ID'>";
        echo "<input type = submit value = 'Atualizar registro!'>";
        echo "</form>";

        echo "<form method = 'post' action = 'form_selecao.php'>";
        echo "<input type = 'submit' value = 'Cancelar exclusão!'>";
        echo "</form>";
} //fim do else

mysql_close($conexao);

?>

</form>

</BODY>

</HTML>
```

Neste script, novamente o ID do registro a ser deletado será passado para o próximo script através de um campo HIDDEN, para identificarmos de modo único o registro a ser esxcluído. Vejamos agora o script responsável pela deleção do registro no banco de dados:

apagar_agora.php

```
<HTML>

<BODY>

<?php

include ("conection.php");

$db = mysql_select_db("dados_pessoais", $conexao);

if (!$db)
{
        echo "Seleção de database não realizada!";
        exit;
}

$nome = (urldecode($nome));

$sql = "DELETE FROM dados WHERE ID = '$ID'";

if (!$query = mysql_query($sql, $conexao))
{
        echo "Erro, exclusão não realizada!<br>";
        echo "<br><a href = 'form_selecao.php'> Voltar!</a>";
        exit;
} else {
        echo "Tabela atualizada [" .mysql_affected_rows() ."] dado(s) excluído(s)!<br>";
}

echo "<br><a href = 'form_selecao.php'> Voltar!</a>";
mysql_close($conexao);
?>
</BODY>
```

```
</HTML>
```

13.8 Registros Duplicados

Ao inserir ou atualizar novos registros, pode ocorrer de já haver um registro com tais dados. Para fazer tal pesquisa, deve−se apenas ter consciência da declaração sql seguinte:

```
SELECT * FROM dados WHERE nome = '$nome' AND endereco = '$endereco'
AND cidade = '$cidade' AND pais = '$pais'
```

Agora utilize a função *mysql_query()* e logo após *mysql_num_rows()* com tal declaração, se o valor retornado for diferente de zero, um registro com os mesmos dados já existe.

14 Anexo A − Conceitos Utilizados

14.1 Softwares Livres

Vejamos o conceito encontrado na página oficial do projeto GNU (http://www.gnu.org):

Nós mantemos esta definição do Software Livre para mostrar claramente o que deve ser verdadeiro à respeito de um dado programa de software para que ele seja considerado software livre.

"Software Livre" é uma questão de liberdade, não de preço. Para entender o conceito, você deve pensar em "liberdade de expressão", não em "cerveja grátis".

"Software livre" se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro tipos de liberdade, para os usuários do software:

* A liberdade de executar o programa, para qualquer propósito (liberdade no. 0) * A liberdade de estudar como o programa funciona, e adaptá−lo para as suas necessidades (liberdade no. 1). Aceso ao código−fonte é um pré−requisito para esta liberdade. * A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo (liberdade no. 2). * A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie (liberdade no. 3). Acesso ao código−fonte é um pré−requisito para esta liberdade.

Um programa é software livre se os usuários tem todas essas liberdades. Portanto, você deve ser livre para redistribuir cópias, seja com ou sem modificações, seja de graça ou cobrando uma taxa pela distribuição, para qualquer um em qualquer lugar. Ser livre para fazer essas coisas significa (entre outras coisas) que você não tem que pedir ou pagar pela permissão.

Você deve também ter a liberdade de fazer modifcações e usá−las privativamente no seu trabalho ou lazer, sem nem mesmo mencionar que elas existem. Se você publicar as modificações, você não deve ser obrigado a avisar a ninguém em particular, ou de nenhum modo em especial.

A liberdade de utilizar um programa significa a liberdade para qualquer tipo de pessoa física ou jurídica utilizar o software em qualquer tipo de sistema computacional, para qualquer tipo de trabalho ou atividade, sem que seja necessário comunicar ao desenvolvedor ou a qualquer outra entidade em especial.

A liberdade de redistribuir cópias deve incluir formas binárias ou executáveis do programa, assim como o código−fonte, tanto para as versões originais quanto para as modificadas. Está ok se não for possível produzir uma forma binária ou executável (pois algumas linguagens de programação não suportam este recurso), mas deve ser concedida a liberdade de redistribuir essas formas caso seja desenvolvido um meio de cria−las.

De modo que a liberdade de fazer modificações, e de publicar versões aperfeiçoadas, tenha algum significado, deve−se ter acesso ao código−fonte do programa. Portanto, acesso ao código−fonte é uma condição necessária ao software livre.

Para que essas liberdades sejam reais, elas tem que ser irrevogáveis desde que você não faça nada errado; caso o desenvolvedor do software tenha o poder de revogar a licença, mesmo que você não tenha dado motivo, o software não é livre.

Entretanto, certos tipos de regras sobre a maneira de distribuir software livre são aceitáveis, quando elas não entram em conflito com as liberdades principais. Por exemplo, copyleft (apresentado de forma bem simples) é a regra de que, quando redistribuindo um programa, você não pode adicionar restrições para negar para outras pessoas as liberdades principais. Esta regra não entra em conflito com as liberdades; na verdade, ela as

protege.

Portanto, você pode ter pago para receber cópias do software GNU, ou você pode ter obtido cópias sem nenhum custo. Mas independente de como você obteve a sua cópia, você sempre tem a liberdade de copiar e modificar o software, ou mesmo de vender cópias.

"Software Livre" não significa "não−comercial". Um programa livre deve estar disponível para uso comercial, desenvolvimento comercial, e distribuição comercial. O desenvolvimento comercial de software livre não é incomum; tais softwares livres comerciais são muito importantes.

Regras sobre como empacotar uma versão modificada são aceitáveis, se elas não acabam bloqueando a sua liberdade de liberar versões modificadas. Regras como "se você tornou o programa disponível deste modo, você também tem que torná−lo disponível deste outro modo" também podem ser aceitas, da mesma forma. (Note que tal regra ainda deixa para você a escolha de tornar o programa disponível ou não.) Também é aceitável uma licença que exija que, caso você tenha distribuído uma versão modificada e um desenvolvedor anterior peça por uma cópia dele, você deva enviar uma.

No projeto GNU, nós usamos "copyleft" para proteger estas liberdades legalmente para todos. Mas também existe software livre que não é copyleft. Nós acreditamos que hajam razões importantes pelas quais é melhor usar o copyleft, mas se o seu programa é free−software mas não é copyleft, nós ainda podemos utilizá−lo.

14.2 Servidor Web

Um servidor web é um aplicativo responsável por fornecer ao computador do cliente (usuários de sites e páginas eletrônicas), em tempo real, os dados solicitados.

O processo se inicia com a conexão entre o computador onde está instalado o servidor e o computador do cliente; como na web não é possível prever a que hora se dará essa conexão, os servidores precisam estar disponíveis dia e noite. A partir daí é processado o pedido do cliente, e conforme as restrições de segurança e a existência da informação solicitada, o servidor devolve os dados.

Quando falamos em servidor web, estamos na verdade falando de servidores capazes de lidar com o protocolo HTTP, que é o padrão para transmissão de hipertexto. Muitos servidores trabalham, paralelamente, com outros protocolos, como HTTPS (que é o HTTP "seguro"), FTP, RPC, etc.

Além de transmitir páginas HTML, imagens e aplicativos Java, os servidores também podem executar programas e scripts, interagindo mais com o usuário, como por exemplo, interpretar scripts da linguagem PHP.

Deve−se lembrar que tais aplicativos se ligam ao servidor web através de portas de serviço, como por exemplo a porta 6667 muito utilizada por aplicativos IRC.

14.3 Server Side

Aplicativos SERVER SIDE, ou melhor, aplicativos do 'lado do servidor', são aplicativos que são executados pelo próprio servidor web, que após realizar o processamento, devolve os resultados para a máquina do cliente, mais precisamente, para o browser do usuário que está acessando o web site.

Um exemplo prático é a linguagem PHP, que apenas é executada no servidor web que devolve o resultado do processamento através de código HTML para a máquina do cliente.

14.4 Client Side

Ao contrário de SERVER SIDE, aplicativos CLIENT SIDE são executados do lado do cliente, mais precisamente, no próprio browser do usuário.

Um exemplo é o HTML e o Javascript que são totalmente processados pelo navegador web do usuário, e não pelo servidor web.

14.5 Interpretador CGI

CGI significa Common Gateway Interface e age da seguinte maneira: com o sistema PHP configurado para ser um interpretador CGI, sempre que um script PHP vai ser interpretado, o servidor web gera uma instância do interpretador PHP, o qual interpreta o script, ou seja, o PHP é executado em um processamento separado ao processamento do servidor web, o que pode causar uma degradação de desempenho.

14.6 Módulo Apache

Já quando o sistema PHP é compilado como módulo Apache, ele executa no mesmo espaço de endereços que o próprio processamento do servidor web e propicia desse modo uma melhora significativa de desempenho com relação aos interpretadores CGI tradicionais que são processados sepárados.

14.7 Configuração para utilizar a função mail():

Para que possamos iniciar a configuração do PHP para aceitar o envio de e−mails precisamos primeiramente localizar o arquivo php.ini em nosso servidor. Independente da plataforma que está sendo utilizada, o arquivo de configuração sempre será o php.ini.

Na plataforma Windows você especificará um servidor de SMTP para que você possa enviar e−mails através de seus scripts. O servidor de SMTP não precisa estar instalado na mesma máquina que o Web server.

Então vamos a parte prática. Abra o arquivo php.ini que deve estar em C:\WINDOWS\ (ou C:\WINT). Depois localize dentro do arquivo o bloco idêntico ao abaixo:

```
[mail function]
; For Win32 only.
;SMTP =
;sendmail_from =

; For Unix only.  You may supply arguments as well (default: 'sendmail −t −i').
;sendmail_path =
```

Observe que todas as linhas estão comentadas, ou seja, estão com um ; (ponto e vírgula) no início das linhas. Para fazer a configuração para Windows você precisará deixar o seu bloco similar ao abaixo:

```
[mail function]
; For Win32 only.
SMTP = smtp.brturbo.com
sendmail_from =thiago_algo@brturbo.com

; For Unix only.  You may supply arguments as well (default: 'sendmail −t −i').
;sendmail_path =
```

Note que as duas linhas iniciadas com ';SMTP' e ';sendmail' foram substituídas por linhas iniciadas com 'SMTP' e 'sendmail', ou seja, as linhs foram descomentadas. Além disso foram acrescentados ao final da linha o servidor de SMTP e o email padrão, smtp.brturbo.com e thiago_algo@brturbo.com , respectivamente.

No caso de você estar utilizando Linux (o que é altamentte recomendado) é preciso especificar o caminho do aplicativo sendmail (sendmail_path). Portanto abra o arquivo php.ini, que no meu caso se encontra em /etc/php4/apache/ e descomente a linha onde se encontra o 'sendmail_path'. Adicione ao final desta linha o caminho do aplicativo sendmail que no meu caso se encontra em /usr/lib/postfix/ e pronto. Veja como fica o bloco no meu caso:

```
[mail function]
; For Win32 only.
;SMTP =
;sendmail_from =

; For Unix only.  You may supply arguments as well (default: 'sendmail −t −i').
sendmail_path = /usr/lib/postfix/
```

15 Anexo B − Instalação e Configuração do PHP

15.1 Windows

15.1.1 Download dos itens necessários

Primeiro, faça o download dos programas a serem usados.

Quanto ao Apache, usaremos o Apache 1.3.*, visto que na documentação do PHP existe a seguinte recomendação: "Não use Apache 2.0 e PHP em um sistema de produção, seja no Unix ou no Windows" (http://www.php.net/manual/pt_BR/install.apache2.php).

- Apache: http://mirrors.uol.com.br/pub/apache/httpd/binaries/win32/apache_1.3.31−win32−x86−no_src.exe
- MySQL 4.1.8: http://dev.mysql.com/get/Downloads/MySQL−4.1/mysql−4.1.8−win−noinstall.zip/from/http://www.linorg.usp
- PHP 5.0.3: http://br2.php.net/get/php−5.0.3−Win32.zip/from/this/mirror

15.1.2 Instalação

Execute a instalação do Apache e o instale com as configurações padrões. Se quiser, pode escolher outro diretório para a instalação.

Extraia o MySQL em uma pasta qualquer. Recomendo dentro da pasta onde você instalou o Apache.
Ex: C:\Arquivos de Programas\Apache Group\Apache\mysql

Extraia o PHP 5 na pasta C:\php5

15.1.3 Configuração do PHP

Vá para a pasta c:\php5 e copie o arquivo php5ts.dll para a seguinte pasta, de acordo com o seu Windows:

− c:\windows\system (em Windows 9x/Me)
− c:\windows\system32 (em WindowsXP)
− c:\winnt\system32 (para Windows NT/2000)

Copie também o arquivo c:\php5\libmysql.dll para umas das pastas ditas acima, de acordo com o Windows em uso. Esse arquivo é necessário para o funcionamento do MySQL no PHP. Ainda no c:\php5, renomeie o arquivo "php.ini−dist" para "php.ini" e abra−o. Procure a linha extension_dir = "./" e a altere para extension_dir = "c:/php5/ext/", é o diretório onde ficam as extensões do php (MySQL, Curl, GD, etc).

Agora, localize a linha ;extension=php_mysql.dll e tire o ; do início dela. Se quiser também, já aproveite e faça o mesmo na linha ;extension=php_gd2.dll, caso queira a biblioteca GD para a manipulação de imagens.

Salve as alterações e mova o "php.ini" para a pasta:
− c:\windows (em Windows 9x/Me/XP)
− c:\winnt (para Windows NT/2000)

15.1.4 Configuração do Apache

Vá para a pasta onde você instalou o Apache e abra o arquivo conf/httpd.conf em qualquer editor de texto. (Ex: Bloco de Notas).

− Localize a linha #LoadModule unique_id_module modules/mod_unique_id.so e logo abaixo dela adicione:
LoadModule php5_module "c:/php5/php5apache.dll"

− Localize a linha AddModule mod_setenvif.c e logo abaixo, adicione:
AddModule mod_php5.c

− Localize AddType application/x−tar .tgz e logo abaixo, adicione:
AddType application/x−httpd−php .php
AddType application/x−httpd−php−source .phps

− Localize:

```
<IfModule mod_dir.c>
DirectoryIndex index.html
</IfModule>
```

E logo ao lado do index.html adicione:
index.php default.php main.php

15.1.5 Configuração do MySQL

Não há nada para se configurar no MySQL, você só deve iniciar o mesmo. Vá para a pasta que instalou o MySQL, então abra o bin/mysqld.exe, iniciando o servidor do MySQL.Lembre−se que sempre você terá de iniciar o MySQL.

Se não quiser ir na pasta toda vez que iniciar o computador, crie um atalho para o bin/mysqld.exe e coloque no "Iniciar, Programas, Inicializar (ou Iniciar, de acordo com o windows)".

15.1.6 Finalização

Agora, vá em "Iniciar, Programas, Apache HTTP Server, Control Apache Server, Restart", para reinicializar o Apache com as alterações feitas. Pronto! Agora você tem PHP5 + MySQL em seu Windows!.

Para efetuar um teste, crie um arquivo chamado phpinfo.php, com o conteúdo:

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

e o coloque na pasta htdocs dentro da pasta do Apache. Abra seu navegador e digite http://localhost/phpinfo.php. Se a página abrir com as informações do PHP, significa que tudo deu certo.

Observações: Lembrando que nos caminhos que mostrei, o C:\ deve ser substituído pela letra do HD em que está seu Windows e onde foram instalados os programas. Uso o C:\ no artigo, pois é a letra que é normalmente usada.

Caso queira register_globals no PHP, abra o c:\windows\php.ini, localize a linha "register_globals = Off" e arrume para "register_globals = On". Veja mais aqui:
http://www.php.net/manual/pt_BR/security.registerglobals.php

15.2 Linux

A instalação do conjunto PHP + Apache + MySQL varia de distribuição em distribuição Linux. Por isso deixarei este tópico sem mais explicações, ficando para vocês, desenvolvedores, a tarefa de procurarem

artigos ou faqs na Internet, que contêm informações sobre a instalação e configuração destes pacotes.