Universidade da Região da Campanha
Centro de Ciências da Economia e Informática
Curso de Informática
Disciplina: Tópicos Especiais em Sistemas de Informação

Apostila de PHP

Prof. Cristiano Cachapuz e Lima

Sumário

1 Introdução 3
2 Exemplo de Script 3
3 Configuração (php.ini) 5
4 Sintaxe Básica 6
5 Variáveis 8
6 Tipos de dados 9
7 Constantes 12
8 Expressões 13
9 Operadores 14
10 Estruturas de Controle 18
11 Funções 22
12 Classes e Objetos 23
13 Referências 24
14 Matrizes 26
15 Inclusão de Arquivos 27
16 Cookies 28
17 Parâmetros 30
18 Formulários 31
19 Uploads 33
20 Envio de e-mails 34
21 Introdução ao MySQL 35
22 Exibição 35
23 Consulta e Ordenação 37
24 Inclusão e Atualização 38
25 Exclusão 40
Referências 40

1 Introdução

PHP é uma sigla recursiva que significa *PHP HyperText Preprocessor*. O PHP é uma linguagem de código-fonte aberto, muito utilizada na Internet e especialmente criada para o desenvolvimento de aplicativos *Web*.

Note como isso é diferente de scripts CGI escritos em outras linguagens como Perl ou C --- ao invés de escrever um programa com um monte de comandos para imprimir HTML, você escreve um arquivo HTML com algum código inserido para fazer alguma coisa (nesse caso, imprimir um pouco de texto). O código PHP é delimitado por tags iniciais e finais que lhe permitem pular pra dentro e pra fora do "modo PHP".

A melhor coisa em usar PHP está no fato de ele ser extremamente simples para um iniciante, mas oferece muitos recursos para o programador profissional.

Para testar scripts PHP é necessário um servidor com suporte a esta tecnologia. Normalmente, o mais utilizado é o Apache. O banco de dados mais utilizado com os scripts PHP é o MySQL. Um exemplo de pacote pronto para execução de um ambiente Apache + PHP + MySQL é o EasyPHP (http://www.easyphp.org). Qualquer editor de textos pode ser usado para escrever os scripts PHP (ex. bloco de notas ou a ferramenta *free* MPS PHP Designer em http://www.mpsoftware.dk).

As páginas PHP devem ser salvas no diretório raiz do servidor. Para testes locais com o EasyPHP, essa pasta é c:\Arquivos de programas\EasyPHP\www. Para acessar a página, deve-se abrir o *browser* Internet Explorer e digitar-se o nome do domínio (http://127.0.0.1) e o nome da página com extensão .php. Quando o EasyPHP está sendo executado, aparece um ícone com uma letra "e" ao lado do relógio do Windows.

2 Exemplo de Script

Para criar o primeiro exemplo, digite o seguinte código-fonte no seu editor e salve com o nome de `teste.php` dentro do diretório raiz do servidor.

```
<html>
 <head>
  <title>Teste PHP</title>
 </head>
 <body>
 <?php echo "<p>Alô Mundo</p>"; ?>
 </body>
</html>
```

Figura 1 – Primeiro script

No *browser*, digite o endereço `http://127.0.0.1/teste.php` e veja o resultado. Veja também o código fonte da página (Exibir → Código fonte). É interessante notar que os comandos PHP não aparecem porque o servidor interpreta todos os scripts antes de enviar a página para o *browser*.

O que PHP pode fazer ?

Qualquer coisa. O PHP é focado para ser uma linguagem de script do lado do servidor, portanto, você pode fazer qualquer coisa que outro programa CGI pode fazer, como: coletar

dados de formulários, gerar páginas com conteúdo dinâmico ou enviar e receber cookies. Mas o PHP pode fazer muito mais.

Esses são os maiores campos onde os scripts PHP podem se utilizados:

- Script no lado do servidor (server-side). Este é o mais tradicional e principal campo de atuação do PHP. Você precisa de três coisas para seu trabalho. O interpretador do PHP (como CGI ou módulo), um servidor *web* e um browser. Basta rodar o servidor web conectado a um PHP instalado. Você pode acessar os resultados de seu programa PHP com um browser, visualizando a página PHP através do servidor web.
- Script de linha de comando. Você pode fazer um script PHP funcionar sem um servidor *web* ou *browser*. A única coisa necessária é o interpretador. Esse tipo de uso é ideal para script executados usando o `cron` ou o Agendador de Tarefas (no Windows). Esses scripts podem ser usados também para rotinas de processamento de texto.
- Escrevendo aplicações GUI no lado do cliente (client-side). O PHP não é (provavelmente) a melhor linguagem para produção de aplicações com interfaces em janelas, mas o PHP faz isso muito bem, e se você deseja usar alguns recursos avançados do PHP em aplicações no lado do cliente poderá utilizar o PHP-GTK para escrever esses programas. E programas escritos desta forma ainda serão independentes de plataforma. O PHP-GTK é uma extensão do PHP, não disponível na distribuição oficial. Se você está interessado no PHP-GTK, visite o site http://gtk.php.net.

O PHP pode ser utilizado na maioria dos sistemas operacionais, incluindo Linux, várias variantes Unix (incluindo HP-UX, Solaris e OpenBSD), Microsoft Windows, Mac OS X, RISC OS, e provavelmente outros. O PHP também é suportado pela maioria dos servidores web atuais, incluindo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape and iPlanet Servers, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd, e muitos outros. O PHP pode ser configurado como módulo para a maioria dos servidores, e para os outros como um CGI comum.

Com o PHP, portanto, você tem a liberdade para escolher o sistema operacional e o servidor web. Do mesmo modo, você pode escolher entre utilizar programação estrutural ou programação orientada a objeto, ou ainda uma mistura deles. Mesmo não desenvolvendo nenhum recurso padrão de OOP (Object Oriented Programming, Programação Orientada a Objetos) na versão atual do PHP, muitas bibliotecas de código e grandes aplicações (incluindo a biblioteca PEAR) foram escritos somente utilizando OOP.

Com PHP você não está limitado a gerar somente HTML. As habilidades do PHP incluem geração de imagens, arquivos PDF e animações Flash (utilizando libswf ou Ming) criados dinamicamente. Você pode facilmente criar qualquer padrão texto, como XHTML e outros arquivos XML. O PHP pode gerar esses padrões e os salvar no sistema de arquivos, em vez de imprimi-los, formando um cache dinâmico de suas informações no lado do servidor.

Talvez a mais forte e mais significativa característica do PHP é seu suporte a uma ampla variedade de banco de dados. Escrever uma página que consulte um banco de dados é incrivelmente simples. Os seguintes bancos de dados são atualmente suportados:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

Adicionalmente, o PHP suporta ODBC (Open Database Connection, ou Padrão Aberto de Conexão com Bancos de Dados), permitindo que você utilize qualquer outro banco de dados que suporte esse padrão mundial.

O PHP também tem suporte para comunicação com outros serviços utilizando protocolos como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (em Windows) e incontáveis outros. Você pode abrir sockets de rede e interagir diretamente com qualquer protocolo. O PHP também suporta o intercâmbio de dados complexos WDDX, utilizado em virtualmente todas as linguagens de programação para *web*. Falando de comunicação, o PHP implementa a instanciação de objetos Java e os utiliza transparentemente como objetos PHP. Você ainda pode usar sua extensão CORBA para acessar objetos remotos.

O PHP é extremamente útil em recursos de processamento de texto, do POSIX Estendido ou expressões regulares Perl até como interpretador para documentos XML. Para acessar e processar documentos XML, são suportados os padrões SAX e DOM. Você ainda pode usar nossa extensão XSLT para transformar documentos XML.

Utilizando o PHP no campo do e-commerce, você poderá usar as funções específicas para Cybescash, CyberMUT, Verysign Payflow Pro e CCVS, práticos sistemas de pagamento online.

Por último mas longe de terminar, temos também outras extensões interessantes: funções para o search engine mnoGoSearch, funções para Gateway IRC, vários utilitários de compressão (gzip, bz2), calendário e conversões de datas, tradução, etc.

3 Configuração (php.ini)

As configurações do PHP ficam armazenadas em um arquivo denominado php.ini e que é carregado cada vez que o servidor é iniciado. No Windows, ele fica na pasta c:\Windows.

Exemplo:

```
[PHP]

;;;;;;;;;;;
; WARNING ;
;;;;;;;;;;;
; This is the default settings file for new PHP installations.
; By default, PHP installs itself with a configuration suitable for
; development purposes, and *NOT* for production purposes.
; For several security-oriented considerations that should be taken
; before going online with your site, please consult php.ini-
recommended
```

```
   ; and http://php.net/manual/en/security.php.


   ;;;;;;;;;;;;;;;;;;;
   ; About this file ;
   ;;;;;;;;;;;;;;;;;;;
   ; This file controls many aspects of PHP's behavior.  In order for PHP
to
   ; read it, it must be named 'php.ini'.  PHP looks for it in the
current
   ; working directory, in the path designated by the environment
variable
   ; PHPRC, and in the path that was defined in compile time (in that
order).
   ; Under Windows, the compile-time path is the Windows directory.  The
   ; path in which the php.ini file is looked for can be overridden using
   ; the -c argument in command line mode.
   ;
   ; The syntax of the file is extremely simple.  Whitespace and Lines
   ; beginning with a semicolon are silently ignored (as you probably
guessed).
   ; Section headers (e.g. [Foo]) are also silently ignored, even though
   ; they might mean something in the future.
```

Figura 2 – Trecho de exemplo do php.ini

Através de modificações neste arquivo é possível alterar várias opções no comportamento do PHP. Todas as linhas iniciadas por ponto-e-vírgula são comentários.

4 Sintaxe Básica

Tags especiais indicam ao PHP onde estão os blocos de código. A tag de abertura é formada por um sinal de "menor que" (<), um sinal de interrogação (?) e a sigla php. A tag de fechamento é formada por um ponto interrogação (?) e sinal de "maior que" (>).

Ex:

```
<?php
...
?>
```

Exercício: digite o código da figura 1 e salve no diretório raiz do servidor Apache. Veja o resultado da página através de seu carregamento no *browser*.

O sinal de ponto-e-vírgula (;) indica o final de um comando (ver figura 1). A próxima figura mostra outro exemplo.

```
   <html>
   <head>
   <title>Teste PHP</title>
   </head>
   <body>

   <?php
   $a = 10;
   $b = 15;
```

```
    $c = $a + $b;
    echo "$a mais $b é igual a $c";
    ?>

    </body>
    </html>
```

Figura 3 – Exemplo de código

Os comentários de mais de uma linha no PHP são obtidos através de /* e */. Os comentários de apenas uma linha são obtidos através de //.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
/*
O código abaixo soma duas variáveis
e exibe o valor encontrado
*/
$a = 10;
$b = 15;
$c = $a + $b;
echo "$a mais $b é igual a $c";
?>

</body>
</html>
```

Figura 4 – Exemplo de código com comentários de mais de uma linha

Os comentários não aparecem no *browser*.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$a = 10; //A variável $a recebe o valor 10
$b = 15; //A variável $b recebe o valor 15
//A variável $c recebe o valor da soma
$c = $a + $b;
//O resultado obtido é exibido
echo "$a mais $b é igual a $c";
?>

</body>
</html>
```

Figura 5 - Exemplo de código com comentários de uma linha

Palavras-chave do PHP

```
and         do          for         include     require     true
break       else        foreach     list        return      var
case        elseif      function    new         static      virtual
class       extends     global      not         switch      xor
continue    false       if          or          this        while
default
```

5 Variáveis

Variáveis armazenam valores. Pode-se referir a variáveis para obter seu valor ou para alterar seu conteúdo.

No PHP elas são representadas por um cifrão ($) mais o nome da variável. Os nomes de variáveis válidos são iniciados por letras ou por um subscrito (_). Existe diferenciação entre nomes de variáveis maiúsculas e minúsculas.

Ex: $a, $_A, $_a

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$a = 10;
$A = 20;
echo "O valor de 'a' é $a e o de 'A' é $A";
?>

</body>
</html>
```

Figura 6 – Exemplo de código com variáveis minúscula e maiúscula

Quando a variável é declarada dentro de uma função, ela só estará disponível para o código desta função. O código a seguir gera um erro devido a essa característica.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

function soma($a)
{
  $b = $a + 5;
}

soma(10);

echo "o valor de 'b' é $b";
```

```
?>

</body>
</html>
```

Figura 7 – Declaração de variável dentro de função

Para evitar este tipo de problema, deve-se definir a variável como global. O código a seguir resolve o problema do código anterior. Compare os resultados dos dois scripts.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

function soma($a)
{
  global $b;
  $b = $a + 5;
}

soma(10);

echo "o valor de 'b' é $b";

?>

</body>
</html>
```

Figura 8 – Declaração de variável global

6 Tipos de dados

O PHP suporta vários tipos de dados:

Inteiro – Números inteiros (isto é, números sem ponto decimal)

Números de dupla precisão – Números reais (isto é, números que contêm um ponto decimal)

String – Texto entre aspas simples (´ ´) ou duplas (“ “)

Booleanos – armazenam valores verdadeiros ou falsos, usados em testes de condições

Array – Grupo de elementos do mesmo tipo

Objeto – Grupo de atributos e métodos

Recurso – Uma origem de dados externa

Nulo – Nenhum valor

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>
```

```
<?php

$a = True;

if ($a)
{
  echo "Verdadeiro";
}
else
{
  echo "Falso";
}

?>

</body>
</html>
```

Figura 9 – Código com dados booleanos

Teste o código anterior alterando o valor da variável para False.

Pode-se armazenar valores inteiros, positivos ou negativos. Pode-se usar também valores hexadecimais.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$a = 0x1A; //Coresponde ao decimal 26
$b = -16;
$c = $a + $b;

echo "a + b = $c";

?>

</body>
</html>
```

Figura 10 – Código com variáveis hexadecimal e valor negativo

Os valores de ponto flutuante são representados através de ponto (.).

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$preco = 11.90;
```

```
$soma = $preco * 4;

echo "Quatro revistas W custam R$ $soma<br>";

?>

</body>
</html>
```

Figura 11 – Código com variável de ponto flutuante

As strings são armazenadas dentro de aspas duplas (“) ou simples (‘).

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$texto1 = 'Esse é o primeiro texto.<br>';
$texto2 = "Esse é o segundo texto.";

echo $texto1;
echo $texto2;

?>

</body>
</html>
```

Figura 12 – Código com strings entre aspas simples e duplas

As variáveis do tipo matriz ou array permitem o armazenamento de diversos elementos referenciados por uma mesma referência. Maiores detalhes serão vistos na seção 14.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$frutas = array(
  1 => "Laranja",
  2 => "Maçã",
  3 => "Uva");

echo "<li> $frutas[1]<br>";
echo "<li> $frutas[2]<br>";
echo "<li> $frutas[3]<br>";

?>
```

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

Figura 13 – Código com matriz

7 Constantes

Constantes são identificadores para valores simples. O seu conteúdo não muda durante a execução do código. Elas são criadas com a função `define` e, por convenção, são escritas com letras maiúsculas e não usam o cifrão no início.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
define("CONSTANTE", "Alô mundo.");
echo CONSTANTE;
?>

</body>
</html>
```

Figura 14 – Código com constante

O PHP implementa algumas constantes, a maioria são matemáticas. O código seguinte demonstra o uso da constante `M_PI`.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

function calculaAreaCirculo($raio)
{
  return M_PI * pow($raio, 2);
}

$meuRaio = 5;
$area = calculaAreaCirculo($meuRaio);

echo "<b>Raio</b> = $meuRaio<br>";
echo "<b>Área</b> = $area";

?>

</body>
</html>
```

Figura 15 – Código com a constante M_PI

8 Expressões

Tudo que tem um valor pode ser considerado uma expressão. O código a seguir demonstra na prática.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$b = ($a = 5);
echo "O valor de 'b' é $b";

?>

</body>
</html>
```

Figura 16 – Código com uso de expressão

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$b = $a = 5;
echo "O valor de 'b' é $b";

?>

</body>
</html>
```

Figura 17 – Variação do código anterior

Expressões de comparação retornam valores booleanos, sendo vazio representando verdadeiro e um representando falso. As expressões de comparação são usadas em declarações condicionais para determinar se um bloco de código será executado ou não.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$valor = (5 < 10);
echo "O valor da expressão '5 > 10' é $valor";
```

```
?>

</body>
</html>
```

Figura 18 – Código com expressão de comparação

9 Operadores

São usados para efetuarem operações sobre as variáveis e constantes. Os operadores do PHP são:

+ soma
- subtração
* multiplicação
/ divisão
^ exponenciação
% módulo, resto da divisão
++ acrescenta um a uma variável
-- subtrai um de uma variável
+= soma um valor a uma variável e lhe atribui o resultado

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$x = 2;
echo($x + 2);
echo "<br>";
$x = 2;
echo(5 - $x);
echo "<br>";
$x = 4;
echo($x * 5);
echo "<br>";
$x = 15;
echo($x / 5);
echo "<br>";
$x = 10;
echo($x % 8);
echo "<br>";
$x = 5;
$x++;
echo($x);
echo "<br>";
$x = 5;
$x--;
echo($x);
echo "<br>";
$x = 8;
echo($x);
```

```
echo "<br>";
$x = 8;
$x = $x + 10;
echo($x);
echo "<br>";
$x = 8;
$x += 10;
echo($x);
?>

</body>
</html>
```

Figura 19 – Código com diversas operações matemáticas

Há também os operadores de comparação. Uma comparação sempre gera um dos dois valores possíveis: vazio, que corresponde a falso, e 1, que corresponde a verdadeiro.

= = é igual a
! = não é igual a
> é maior que
< é menor que
>= é maior ou igual a
<= é menor ou igual a

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$x = 5;

$resultado = ($x == 8);

if($resultado == 1)
{
  echo "verdadeiro";
}
else
{
  echo "falso";
}

echo "<br>";

$x = 5;

$resultado = ($x != 8);

if($resultado == 1)
{
  echo "verdadeiro";
}
else
```

```
{
  echo "falso";
}

echo "<br>";

$x = 5;

$resultado = ($x > 8);

if($resultado == 1)
{
  echo "verdadeiro";
}
else
{
  echo "falso";
}

echo "<br>";

$x = 5;

$resultado = ($x > 8);

if($resultado == 1)
{
  echo "verdadeiro";
}
else
{
  echo "falso";
}

echo "<br>";

$x = 5;

$resultado = ($x >= 8);

if($resultado == 1)
{
  echo "verdadeiro";
}
else
{
  echo "falso";
}

echo "<br>";

$x = 5;

$resultado = ($x <= 8);

if($resultado == 1)
{
```

```
  echo "verdadeiro";
}
else
{
  echo "falso";
}
?>

</body>
</html>
```

Figura 20 – Código com operadores de comparação

Operadores lógicos

and ou && - operador lógico "e", apenas retornando verdadeiro quando as duas condições envolvidas no teste forem verdadeiras

or ou || operador lógico "ou", retornando verdadeiro quando uma ou as duas condições envolvidas no teste forem verdadeiras

! operador lógico "não", invertendo o resultado de um teste

xor – operador lógico "ou exclusivo" que determina se uma de duas condições é verdadeira mas não ambas. Se ambas forem verdadeiras, o teste final será falso

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$x = 6;
$y = 3;

$resultado = ($x < 10 && $y > 1);

if($resultado == 1)
{
  echo "verdadeiro";
}
else
{
  echo "falso";
}

echo "<br>";

$x = 6;
$y = 3;

$resultado = ($x == 5 || $y == 5);

if($resultado == 1)
{
  echo "verdadeiro";
}
```

```
else
{
  echo "falso";
}

echo "<br>";

$x = 6;
$y = 3;

$resultado = (!($x == $y));

if($resultado == 1)
{
  echo "verdadeiro";
}
else
{
  echo "falso";
}
?>

</body>
</html>
```

Figura 21 – Código com operadores lógicos

10 Estruturas de Controle

No PHP, as estruturas de controle são formadas por declarações condicionais e de *looping*:

if – executa uma ação se uma condição for atendida. O bloco de comandos a ser executado deve ser escrito entre chaves;

else – pode-se colocar um conjunto de comandos alternativos caso o teste do if seja negativo. A declaração else deve vir logo após o bloco de código relacionado ao if (ver figura 24). O comando if também pode ser usado após a declaração else (figura 25).

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$x = 20;

if ($x > 10)
{
  echo("O valor da variável é maior que 10.");
}
```

```
?>

</body>
</html>
```

Figura 22 – Código com declaração condicional if verdadeiro

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$x = 5;

if ($x > 10)
{
  echo("O valor da variável é maior que 10.");
}
?>


</body>
</html>
```

Figura 23 – Código com declaração condicional if falso

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$x = 5;

if ($x > 10)
{
  echo("O valor da variável é maior que 10.");
}
else
{
  echo("O valor da variável é menor que 10.");
}
?>

</body>
</html>
```

Figura 24 – Código com declaração condicional if e else

```
<html>
<head>
<title>Teste PHP</title>
</head>
```

```
<body>

<?php
$cor = "branco";

if ($cor == "vermelho")
{
  echo("A variável contém o valor 'vermelho'.");
}
else if ($cor == "azul")
{
  echo("A variável contém o valor 'azul'.");
}
else if ($cor == "amarelo")
{
  echo("A variável contém o valor 'amarelo'.");
}
else
{
  echo("O valor da variável não foi identificado.");
}
?>

</body>
</html>
```

Figura 25 – Código com declaração condicional if e else if

switch / case – forma de testar uma dentre várias possibilidades. A declaração default executa caso nenhuma das opções for verdadeira (figura 26). A declaração break faz com que o restante do código não seja executado caso o teste seja verdadeiro.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$d = getdate();

switch ($d['wday'])
{
case 5:
  echo("Finalmente Sexta");
  break;
case 6:
  echo("Super Sábado");
  break;
case 0:
  echo("Domingo Sonolento");
  break;
default:
  echo("Estou esperando pelo fim da semana");
}
?>
```

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

Figura 26 – Código com declaração condicional switch e case

for – estrutura de looping que executa um bloco de código quantas vezes for indicado em uma variável. Deve-se definir a variável que será testada no looping, uma condição de teste e o incremento (ou decremento) da variável de controle.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

for ($i = 1; $i < 10; $i++)
{
  echo("Linha $i <br>");
}

?>

</body>
</html>
```

Figura 27 – Código com looping definido pelo comando if

while – estrutura de looping que não necessita de um número determinado de iterações. Ele é executado enquanto uma condição for verdadeira.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$i = 1;

while ($i < 10000)
{
  echo($i);
  $i *= 2;
  echo(" vezes 2 é igual a $i <br>");
}

?>

</body>
</html>
```

Figura 28 – Código com declaração condicional while

do-while– outra forma de looping que executa um bloco de código, testa uma condição e repete novamente o bloco de código (ou não).

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
$i = 1;

do
{
  echo ("Linha $i <br>");
  $i++;
}
while ($i < 10)

?>

</body>
</html>
```

Figura 29 - Código com declaração condicional do-while

11 Funções

Uma função é um bloco de código reutilizável que é executado devido a um evento ou pela chamada de outra função. Deve-se usar a declaração function para criar uma função. Os parâmetros usados pela função são declarados entre parênteses. Os comandos a serem executados pela função devem estar entre chaves (figura 30). A declaração return retorna um valor quando a função é chamada. Esta declaração não é necessária se a função não retorna nenhum valor.

Para se chamar uma função, deve-se escrever seu nome e indicar os parâmetros entre parênteses.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
function soma($valor1, $valor2)
{
  $resultado = $valor1 + $valor2;
  return ($resultado);
}


$x = soma(7, 8);
```

```
echo($x);
?>

</body>
</html>
```

Figura 30 – Exemplo de uma função

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
function escreveTexto()
{
  echo("Já sei criar funções!");
}

escreveTexto();
?>

</body>
</html>
```

Figura 31 – Segundo exemplo de declaração de função

12 Classes e Objetos

Uma classe é uma coleção de atributos e métodos. O código a seguir define uma classe chamada CarrinhoDeCompras, que é uma matriz associativa com os artigos do carrinho e duas funções: uma para adicionar e outra para remover os itens.

Classes são tipos, isto é, "rascunhos" para a criação de objetos. Deve-se utilizar o operador new para criar uma variável do tipo CarrinhoDeCompras.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
class CarrinhoDeCompras
{
    var $items;  // Items no carrinho de compras

    // Adiciona $num artigos do $artnr ao carrinho

    function adiciona_item ($artnr, $num)
    {
        $this->items[$artnr] = $num;
    }
```

```
    // Retira $num artigos do $artnr do carrinho

    function remove_item ($artnr, $num)
    {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>

</body>
</html>
```

Figura 32 – Código de definição de uma classe

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php
class CarrinhoDeCompras
{
    var $itens;  // Itens no carrinho de compras

    // Adiciona $num artigos do $artnr ao carrinho

    function adiciona_item ($artnr, $num)
    {
        $this->itens[$artnr] = $num;
    }

    // Retira $num artigos do $artnr do carrinho

    function remove_item ($artnr, $num)
    {
        if ($this->itens[$artnr] > $num) {
            $this->itens[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}

$carrinho = new CarrinhoDeCompras;
$carrinho->adiciona_item("Banana", 12);
?>

</body>
</html>
```

Figura 33 – Definição de classe e instanciação de um objeto do tipo CarrinhoDeCompras

13 Referências

Referências, em PHP, significa acessar o mesmo conteúdo de variável através de vários nomes. Isto não é parecido como os ponteiros em C: aqui temos apelidos numa tabela simbólica. Note que no PHP nomes de variáveis e conteúdo de variável são tratados diferentemente, então um mesmo conteúdo pode ter nomes diferentes. A analogia mais próxima é a dos arquivos e seus nomes em sistemas UNIX: nomes de variáveis são o caminho completo dos arquivos, enquanto o conteúdo da variável são os dados desse arquivo. Assim, referências pode ser tomadas como os links físicos dentro do sistema de arquivos UNIX.

No PHP, pode-se acessar valores de variáveis através de diferentes nomes. Pode-se usar o sinal de igual seguido do sinal de "e comercial" (&).

Referências PHP permitem fazer duas variáveis se referirem ao mesmo conteúdo. Ou seja:

```
<?php
$a =& $b
?>
```

aqui `$a` e `$b` apontam para a mesma variável.

> **Nota:** `$a` e `$b` são completamente iguais aqui, mas não porque `$a` está apontando para `$b` ou vice versa, mas sim que `$a` e `$b` apontam para o mesmo lugar.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$a =& $b;

$b = 100;

echo $a;

?>

</body>
</html>
```

Figura 34 – Código de exemplo de atribuição de valor a duas variáveis (a e b)

O comando unset remove uma referência previamente declarada, mas ela mantém o último valor recebido.

```
<html>
<head>
```

```
<title>Teste PHP</title>
</head>
<body>

<?php

$a =& $b;

$b = 100;

unset($b);

$b = 200;

echo $a;

?>

</body>
</html>
```

Figura 35 – Código de remoção de uma referência

14 Matrizes

Matrizes são variáveis que armazenam mais de um valor simultaneamente. Uma matriz no PHP é atualmente um mapa ordenado. Um mapa é um tipo que relaciona *valores* para *chaves*. Este tipo é otimizado de várias maneiras, então você pode usá-lo como um array real, ou uma lista (vetor), *hashtable* (que é uma implementação de mapa), dicionário, coleção, pilha, fila, etc.

As referências aos elementos da matriz podem ser declaradas como valores numéricos ou strings (figura 37).

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$funcionarios = array(0 => "José",
                      1 => "João",
                      2 => "Maria",
                      3 => "Pedro",
                      4 => "Carla");

echo "<b>Funcionários</b>";
echo "<ul>";
echo "<li>" . $funcionarios[0];
echo "<li>" . $funcionarios[1];
echo "<li>" . $funcionarios[3];
echo "</ul><p>";
echo "<b>Funcionárias</b>";
```

```
echo "<ul>";
echo "<li>" . $funcionarios[2];
echo "<li>" . $funcionarios[4];
echo "</ul>";

?>

</body>
</html>
```

Figura 36 – Código com declaração de matriz

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$siglas = array("SP" => "São Paulo",
                "RJ" => "Rio de Janeiro",
                "MG" => "Minas Gerais");

echo $siglas["SP"];

?>

</body>
</html>
```

Figura 37 – Código com declaração de matriz e referência através de string

15 Inclusão de Arquivos

O comando `include` permite a inclusão de outros arquivos php dentro do script que está sendo executado. Pode-se criar uma função que imprime a data atual e pode-se reusá-lo sem precisar reescrever o código cada vez que for necessário. No exemplo a seguir, pode-se chamar o primeiro script de cabecalho.php e o próximo script o inclui através do comando include.

```
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?php

$meses = array(1  => "Janeiro",
               2  => "Fevereiro",
               3  => "Março",
```

```
                4  => "Abril",
                5  => "Maio",
                6  => "Junho",
                7  => "Julho",
                8  => "Agosto",
                9  => "Setembro",
                10 => "Outubro",
                11 => "Novembro",
                12 => "Dezembro");

$hoje = getdate();
$dia = $hoje["mday"];
$mes = $hoje["mon"];
$nomeMes = $meses[$mes];
$ano = $hoje["year"];

echo "Olá. Hoje é dia $dia de $nomeMes de $ano."

?>

</body>
</html>
```

Figura 38 – Script que é salvo com o nome de cabecalho.php

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
include("cabecalho.php");
?>

</body>
</html>
```

Figura 39 – Código com inclusão de arquivo externo chamado cabecalho.php

16 Cookies

Cookies são formas de armazenar informações a respeito de uma sessão dentro do disco rígido do usuário cliente. O comando setcookie armazena um cookie com as informações que se desejam recuperar em seguida. Quando não for declarado um tempo de vida, o cookie se auto-destrói quando a sessão é encerrada (quando o browser for fechado).

```
<?php

if (isset($HTTP_POST_VARS['usuario'])) {

  $user = $HTTP_POST_VARS['usuario'];

  setcookie("usuario", $user);
  $mensagem = "Usuário $user conectado.<p>";
```

```
}
else
{

  $mensagem = "Digite o seu nome de usuário<p>";

}

?>
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?
echo $mensagem;
?>

<form method="post" action="teste.php">
Nome de Usuário: <input type="text" name="usuario">
<br>
<input type="submit" value="Enviar">
</form>

</body>
</html>
```

Figura 40 – Código que cria um cookie com o nome do usuário

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php

$user = $_COOKIE["usuario"];

echo "O usuário $user está conectado.";

?>

</body>
</html>
```

Figura 41 – Código que recupera os dados do cookie criado anteriormente

O código a seguir demonstra o uso de um cookie com "tempo de vida" definido em 3600 segundos, isto é, uma hora. Após uma hora decorrida de sua criação, ele é removido.

```
<?php

if (isset($_POST['usuario'])) {
```

```
  $user = $_POST['usuario'];

  setcookie("usuario", $user, time() + 3600); // Expira em uma hora

  $mensagem = "Usuário $user conectado.<p>";

}
else
{

  $mensagem = "Digite o seu nome de usuário<p>";

}

?>
<html>
<head>
<title>Teste PHP</title>
</head>
<body>

<?
echo $mensagem;
?>

<form method="post" action="teste.php">
Nome de Usuário: <input type="text" name="usuario">
<br>
<input type="submit" value="Enviar">
</form>

</body>
</html>
```

Figura 42 – Código que cria um cookie com o nome do usuário que dura uma hora

17 Parâmetros

O uso de parâmetros facilita a programação porque permite a passagem de dados entre o browser e o script ou entre scripts. A passagem de parâmetros entre o browser e o script é feita dentro da URL, por exemplo e é manipulada pela função $_GET.

Nesse exemplo a seguir, cada um dos links envia um valor diferente para a página que é aberta (teste.php). Para enviar um parâmetro, a sintaxe inclui um sinal de interrogação, o nome da variável, um sinal de igual e o valor da variável.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php

if (isset($_GET["valor"]))
```

```
{
  $valor = $_GET["valor"];
  echo "Você clicou no link $valor <p>";
}
else
{
  echo "Clique em um dos links abaixo:<p>";
}

?>

<a href="teste.php?valor=1">link 1</a><br>
<a href="teste.php?valor=2">link 2</a><br>
<a href="teste.php?valor=3">link 3</a><br>
<a href="teste.php?valor=4">link 4</a><br>
<a href="teste.php?valor=5">link 5</a><br>

</body>
</html>
```

Figura 43 – Código com passagem de parâmetro

Caso exista necessidade de se passar mais de um parâmetro, deve-se separá-los através de "e comercial" (&), conforme figura 44.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
if (isset($_GET["nome"]) && isset($_GET["sobrenome"])) {
  $nome = $_GET["nome"];
  $sobrenome = $_GET["sobrenome"];
  echo "O nome selecionado foi $nome $sobrenome<p>";
}
else
{
  echo "Selecione um nome<p>";
}

?>

<a href="teste.php?nome=Pedro&sobrenome=Silva">Pedro Silva</a><br>
<a href="teste.php?nome=Maria&sobrenome=Santos">Maria Santos</a><br>

</body>
</html>
```

Figura 44 – Código com passagem de mais de um parâmetro

18 Formulários

Os valores enviados através de um formulário podem ser recuperados pela variável pré-definida $_POST. Através dela é possível obter os dados que foram enviados através do

método POST do HTML, bastando indicar o nome do campo do formulário. No comando action do formulário, deve-se indicar a página PHP que irá receber os valores. O mesmo documento pode conter o código e o formulário (figura 45).

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php

if (isset($_POST["pnome"]) && isset($_POST["snome"]))
{
  $pnome = $_POST["pnome"];
  $snome = $_POST["snome"];

  echo "Olá $pnome $snome.<p>";
}
else
{
  echo "Digite o seu nome.<p>";
}

?>

<form method="post" action="teste.php">
Primeiro Nome: <input type="text" name="pnome">
<br>
Sobrenome: <input type="text" name="snome">
<br><br>
<input type="submit" value="Enviar">
</form>

</body>
</html>
```

Figura 45 – Código com formulário enviando dados através do método POST

Se for usado o método GET, os dados podem ser visualizados na URL do browser. Para recuperar estes dados, deve-se usar a variável pré-definida $_GET. Executar os códigos das figuras 45 e 46 e analisar o comportamento do browser.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php

if (isset($_GET["pnome"]) && isset($_GET["snome"]))
{
  $pnome = $_GET["pnome"];
  $snome = $_GET["snome"];
```

```
  echo "Olá $pnome $snome.<p>";
}
else
{
  echo "Digite o seu nome.<p>";
}

?>

<form method="get" action="teste.php">
Primeiro Nome: <input type="text" name="pnome">
<br>
Sobrenome: <input type="text" name="snome">
<br><br>
<input type="submit" value="Enviar">
</form>

</body>
</html>
```

Figura 46 – Código com formulário enviando dados através do método GET

19 Uploads

O PHP permite que sejam enviados arquivos para o servidor. Deve-se modificar o arquivo de configuração (php.ini):

```
file_uploads   = On
upload_tmp_dir ="C:\Arquivos de programas\EasyPHP\tmp\"
upload_max_filesize = 2M
```

Para o próximo exemplo, o diretório de upload deve ser trocado para C:\temp. É necessário reiniciar o servidor a cada modificação em algum arquivo de configuração.

Todas as informações sobre o arquivo enviado ficam armazenadas na variável $_FILES. O comando que trata o envio do arquivo é `move_uploaded_file`. No exemplo a seguir, o usuário envia um arquivo de no máximo 30 Kb.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php

if (isset($_FILES['arquivo']['name'])){

  $uploaddir = 'c:\\temp\\';
  $arquivo = $uploaddir. $_FILES['arquivo']['name'];

  if (move_uploaded_file($_FILES['arquivo']['tmp_name'], $arquivo)) {
    print "O arquivo foi gravado com sucesso.";
```

```
  }
  else
  {
    print "Erro. O arquivo não foi enviado.";
  }
}
?>

<form enctype="multipart/form-data" action="teste.php" method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="30000">
Enviar este arquivo: <input name="arquivo" type="file">
<input type="submit" value="Envia Arquivo">
</form>

</body>
</html>
```

Figura 47 – Código com upload de arquivo e armazenamento na pasta C:\temp

20 Envio de e-mails

O PHP permite que se enviem e-mails de forma automatizada. Para isso, deve-se ajustar o arquivo de configuração (php.ini) para se indicar o servidor SMTP:

```
[mail function]
SMTP           =     localhost   ;for win32 only
sendmail_from  =     me@localhost.com  ;for win32 only
;sendmail_path =                       ;for unix only, may supply
arguments as well (default is 'sendmail -t -i')
```

A opção SMTP indica o endereço ou número IP do servidor SMTP. A opção sendmail_from indica o endereço do remetente da mensagem. É necessário reiniciar o servidor a cada modificação em algum arquivo de configuração. O próximo exemplo envia uma mensagem para o destinatário.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
$destinatario = "cristiano@urcamp.tche.br";
$assunto      = "Como enviar e-mais via PHP";
$mensagem    = "
<h2>Envio de e-mails via PHP</h2>

<p>Depois que o servidor está configurado, é muito simples enviar e-mails
com o PHP, usando apenas a função mail(). Você deve indicar como
parâmetros o destinatário, o assunto, e a mensagem. Para enviar
cabeçalhos adicionais, como informações sobre o formato da mensagem, há
um quarto parâmetro.</p>
";
$cabecalho  = "
MIME-Version: 1.0\r\n
Content-type: text/html; charset=iso-8859-1\r\n";
```

```
mail($destinatario, $assunto, $mensagem, $cabecalho);

echo "e-mail enviado com sucesso";
?>

</body>
</html>
```

Figura 48 – Código com envio de e-mail

PHP e MySQL

21 Introdução ao MySQL

O MySQL é o gerenciador de banco de dados mais usado com o PHP. Existem muitas funções pré-definidas para manipulação de conexões com bancos de dados.

A função mysql_connect tenta uma conexão com um servidor MySQL. Deve-se passar como parâmetros: o nome do servidor (ou número IP) onde o MySQL está sendo executado, o nome de usuário e a senha deste usuário. O comando alternativo die trata um possível fracasso na conexão.

A função mysql_selectdb seleciona qual base será selecionada dentro do banco de dados que foi conectado. O comando alternativo die trata um possível fracasso na seleção da base, podendo ser incluída uma mensagem customizada.

A função mysql_query faz consultas à base previamente selecionada. Deve-se passar, como parâmetros, os comandos SQL apropriados. Novamente, o comando alternativo die pode tratar um não sucesso na consulta.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
$link = mysql_connect("127.0.0.1", "root", "")
  or die("Não foi possível conectar");

mysql_select_db("teste")
  or die("Não foi possível selecionar o banco de dados");

$consulta = "SELECT * FROM Clientes";
$resultado = mysql_query($consulta)
  or die("Falha na execução da consulta");

echo "Consulta executada com sucesso";
?>

</body>
</html>
```

Figura 49 – Código com conexão a um banco de dados MySQL, seleção de uma base teste e consulta todos os registros da tabela Clientes

22 Exibição

Para que os registros da consulta sejam exibidos, deve-se usar a função mysql_fetch_assoc, que retorna uma matriz com a linha atual e move para a próxima. Para se imprimir todos os resultados de uma query, é necessária a construção de uma estrutura de repetição (while) até que a função mysql_fetch_assoc não retorne nenhum valor (vazio). Para melhorar a apresentação dos resultados, é possível usar tags HTML que incluam os dados dentro de tabelas, por exemplo.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
$link = mysql_connect("127.0.0.1", "root", "")
  or die("Não foi possível conectar");

mysql_select_db("teste")
  or die("Não foi possível selecionar o banco de dados");

$consulta = "SELECT * FROM Clientes";
$resultado = mysql_query($consulta)
  or die("Falha na execução da consulta");

$linha = mysql_fetch_assoc($resultado);

$NomeDaEmpresa = $linha["NomeDaEmpresa"];
$NomeDoContato = $linha["NomeDoContato"];

echo "<b>Nome da empresa:</b> $NomeDaEmpresa<br>";
echo "<b>Nome do contato:</b> $NomeDoContato";

?>

</body>
</html>
```

Figura 50 – Código com impressão de um registro a partir de uma consulta

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
$link = mysql_connect("127.0.0.1", "root", "")
  or die("Não foi possível conectar");

mysql_select_db("teste")
  or die("Não foi possível selecionar o banco de dados");
```

```
$consulta = "SELECT * FROM Clientes";
$resultado = mysql_query($consulta)
  or die("Falha na execução da consulta");

while ($linha = mysql_fetch_assoc($resultado))
{

  $NomeDaEmpresa = $linha["NomeDaEmpresa"];
  $NomeDoContato = $linha["NomeDoContato"];

  echo "<b>Nome da empresa:</b> $NomeDaEmpresa<br>";
  echo "<b>Nome do contato:</b> $NomeDoContato<p>";
}

?>

</body>
</html>
```

Figura 51 – Código com impressão de todos registros a partir de uma consulta

23 Consulta e Ordenação

Pode-se fazer consultas atendendo a certos critérios, que vão fazer com que a consulta seja refinada. No próximo exemplo, desejamos selecionar apenas os clientes de São Paulo. Nesse caso, a cláusula que deve ser alterada é a que faz a consulta SQL.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php

$link = mysql_connect("127.0.0.1", "root", "")
  or die("Não foi possível conectar");

mysql_select_db("teste")
  or die("Não foi possível selecionar o banco de dados");

$consulta = "SELECT NomeDaEmpresa, NomeDoContato
               FROM Clientes
               WHERE Cidade = 'São Paulo'";

$resultado = mysql_query($consulta)
  or die("Falha na execução da consulta");

while ($linha = mysql_fetch_assoc($resultado))
{

  $NomeDaEmpresa = $linha["NomeDaEmpresa"];
  $NomeDoContato = $linha["NomeDoContato"];

  echo "<b>Nome da empresa:</b> $NomeDaEmpresa<br>";
```

```
  echo "<b>Nome do contato:</b> $NomeDoContato<p>";
}

?>

</body>
</html>
```

Figura 52 – Código com impressão de determinados registros que satisfazem uma condição (select ... from ... where ...)

No caso de ordenação, a cláusula order by deve ser anexada à query SQL.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
$link = mysql_connect("127.0.0.1", "root", "")
  or die("Não foi possível conectar");

mysql_select_db("teste")
  or die("Não foi possível selecionar o banco de dados");

$consulta = "SELECT NomeDaEmpresa, NomeDoContato
               FROM Clientes
               WHERE Cidade = 'São Paulo'
            ORDER BY NomeDoContato";

$resultado = mysql_query($consulta)
  or die("Falha na execução da consulta");

while ($linha = mysql_fetch_assoc($resultado))
{

  $NomeDaEmpresa = $linha["NomeDaEmpresa"];
  $NomeDoContato = $linha["NomeDoContato"];

  echo "<b>Nome da empresa:</b> $NomeDaEmpresa<br>";
  echo "<b>Nome do contato:</b> $NomeDoContato<p>";
}

?>

</body>
</html>
```

Figura 53 – Código com impressão de determinados registros que satisfazem uma condição, ordenados por um dos atributos (select ... from ... where ... order by ...)

24 Inclusão e Atualização

Para se incluir dados em uma tabela MySQL, deve-se usar o comando INSERT. No exemplo a seguir, a inclusão de dados é estática. Para se criar um aplicativo que permita

inclusão, seria necessário adaptar o script para receber dados via formulário e incluí-los no banco de dados.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
$link = mysql_connect("127.0.0.1", "root", "")
  or die("Não foi possível conectar");

mysql_select_db("teste")
  or die("Não foi possível selecionar o banco de dados");

$CodigoDoCliente = "EELTD";
$NomeDaEmpresa = "Editora Europa";
$NomeDoContato = "Rodolfo Melo";
$Cidade = "São Paulo";

$consulta = "INSERT INTO Clientes
               (CódigoDoCliente, NomeDaEmpresa, NomeDoContato, Cidade)
             VALUES
               ('$CodigoDoCliente', '$NomeDaEmpresa', '$NomeDoContato',
'$Cidade')";
$resultado = mysql_query($consulta)
  or die("Falha na execução da consulta");

echo "Dados adicionados com sucesso";

?>

</body>
</html>
```

Figura 54 – Código com inclusão de um registro em uma tabela do banco de dados

O comando UPDATE altera um registro de uma tabela. No exemplo a seguir, o registro cujo código é “EELTD” passa a ter o nome “Robinson Melgar”.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
$link = mysql_connect("127.0.0.1", "root", "")
  or die("Não foi possível conectar");

mysql_select_db("teste")
  or die("Não foi possível selecionar o banco de dados");

$CodigoDoCliente = "EELTD";
$NomeDoContato = "Robinson Melgar";
```

```
$consulta = "UPDATE Clientes
                SET NomeDoContato = '$NomeDoContato'
              WHERE CódigoDoCliente = '$CodigoDoCliente'";

$resultado = mysql_query($consulta)
  or die("Falha na execução da consulta");

echo "Dados alterados com sucesso";

?>

</body>
</html>
```

Figura 55 – Código com alteração de dados via comando update

25 Exclusão

O comando SQL DELETE remove um registro de uma tabela. A cláusula WHERE delimita a condição para que a remoção seja executada.

```
<html>
<head>
<title>Página PHP</title>
</head>
<body>

<?php
$link = mysql_connect("127.0.0.1", "root", "")
  or die("Não foi possível conectar");

mysql_select_db("teste")
  or die("Não foi possível selecionar o banco de dados");

$CodigoDoCliente = "EELTD";

$consulta = "DELETE FROM Clientes
                  WHERE CódigoDoCliente = '$CodigoDoCliente'";

$resultado = mysql_query($consulta)
  or die("Falha na execução da consulta");

echo "Registro excluído com sucesso";

?>

</body>
</html>
```

Figura 56 – Código com remoção de registros

Referências

BAKKEN, S. S. et al. **PHP Manual**. Disponível em: <http://br.php.net/get/php_manual_pt_BR.chm/from/this/mirror>. Acesso em: 21 out. 2003.

EDITORA Europa. Curso de PHP. **www.com.br**, São Paulo, n. 40, set. 2003. CD-ROM.