Programação Web com PHP e MySQL

(mas com vários outros: CSS, XML, JavaScript, etc)

Imagem em: http://www.solontavares.com/selecao/node/44

Ribamar FS

Maio de 2015

http://ribafs.16mb.com

Sumário

1-HTML

Resumo de HTML

Select

<form>
<select name="carros">
<option selected value="volvo">Volvo (Este é o valor default – selected)
<option value="saab">Saab
<option value="fiat">Fiat
<option value="audi">Audi
</select>
</form>

TextArea
<trextarea rows=”10” cols=”30”>Valor default que será exibido</textarea>

CheckBox
<form name="input" action="html_form_action.php" method="get">
I have a bike: <input type="checkbox" name="Bike" checked>

I have a car: <input type="checkbox" name="Car">

<input type="submit" value="Submit">
</form>

Radio
<form name="input" action="html_form_action.php" method="get">
Male: <input type="radio" name="Sex" value="Male" checked>

Female <input type="radio" name="Sex" value="Female">

<input type ="submit" value ="Submit Now!">
</form>

Botão do tipo Submit envia o conteúdo do form para o action automaticamente.
Caso precisemos enviar para o action através de uma função em JavaScript, devemos usar um botão do tipo Button no lugar do tipo Submit.
Campo Desabilitado e Somente Leitura

<input type ="text" value ="Valor" disabled readonly>

MÉTODOS POST E GET (Formulários)
GET - Este método exibe os dados no Browser ao enviar ao Servidor
POST - Este método envia os dados para o servidor sem exibir no Browser (mais seguro, pois não exibe os dados na URL do browser).

Optgroup – Agrupar grandes listas separando por grupos

```
<select>
<optgroup label="Swedish Cars">
<option value ="volvo">Volvo</option>
<option value ="saab">Saab</option>
</optgroup>
<optgroup label="German Cars">
<option value ="mercedes">Mercedes</option>
<option value ="audi">Audi</option>
</optgroup>
</select>
```

<input type=”tipo” name=”nome” value=”valor default>

tipo:
hidden – campo oculto, normalmente para armazenar valores a serem transportados
password – campo tipo senha que mostra asteriscos ao digitar
reset – botão que limpa todos as campos ao ser clicado
file – tipo para upload

Formatação

Negrito
<s>~~Riscado~~</s>

<font size="3" color="red">
<FONT FACE = TipoDaFonte>Texto</FONT>
TipoDaFonte: times, arial, courier

_{$_{subscrito}$}

^{^{sobrescrito}}

<h1> ... <h6> - Títulos
<h1 align=”center”>Título 1 Centralizado</h1>

<HR> traça uma linha horizontal

Imagem

<img SRC="NomeImagem.Ext" ALT="Aqui o Texto de Dicas" height=50 width=50>
Height - Altura da imagem e Width - Largura

Tabela
<TABLE BORDER = 1>
<TH> Cabeçalho Coluna 1</TH><TH> Cabeçalho Coluna 2</TH>

<TR><TD> Linha1, Coluna1</TD><TD> Linha1, Coluna2</TD></TR>
<TR><TD> Linha2, Coluna1</TD><TD> Linha2, Coluna2</TD></TR>
</TABLE>

Código verde:
```
<tr><td colspan="4"></td><td></td></tr>
```
Código azul:
```
<tr><td colspan="3"></td><td></td><td></td></tr>
```

```
<TABLE BORDER=1>
<TR>
<TD rowspan=2>
my first table
</TD>
<TD>
my first table
</TD>
</TR>
<TR>
<TD>
my first table
</TD>
</TABLE>
```

my first table	my first table
	my first table

colspan – juntar colunas
rowspan – juntar linhas

ÂNCORAS (LINKS)
Abrindo o resultado em uma nova janela. O default, sem target, abre na atual.
W3Schools

FRAMES

View this page for the result
Código de "planets.htm":
<html>
<frameset cols = "25%, 25%,*">
<frame src ="venus.htm" />
<frame src ="sun.htm" />
<frame src ="mercur.htm" />
</frameset>
</html>

HTML RESPEITANDO A POSIÇÃO DE DIGITAÇÃO

O comando (tag) para que o HTML respeite a posição de digitação do texto é o <PRE>.

Exemplo:
<PRE> O que for digitado
Será visto na mesma
posição quando usarmos este comando.</PRE>

COMENTÁRIO
Para inserir comentários em páginas HTML use
<!-- para iniciar o comentário
e
--> para finalizar.

QUEBRAR LINHA

IMAGEM DE FUNDO NA PÁGINA

<BODY BACKGROUND = "http://www.ctonline.hpg.com.br/figura.jpg">
A figura é repetida preenchendo a página.

TABELAS DE CORES HTML

1 - http://www.ufpa.br/dicas/html-cor.htm
2 - http://orbita.starmedia.com/~edaurelio/a12.htm
3 - http://209.135.157.193/html/colortable.html
4 - http://www.geocities.com/ensinandohtml/cores_html.htm

Introdução ao HTML

Aqui teremos apenas um resumo básico dos principais comandos do HTML. Para uma referência completa dos comandos existem vários bons tutoriais na Internet. A intenção é abordar o básico necessário para o trabalho com PHP.

Wikipédia - http://pt.wikipedia.org/wiki/HTML

HTML (acrônimo para a expressão inglesa *HyperText Markup Language*, que significa *Linguagem de Marcação de Hipertexto*) é uma linguagem de marcação utilizada para produzir páginas na Web. Documentos HTML podem ser interpretados por navegadores. A tecnologia é fruto do "casamento" dos padrões HyTime e SGML.

HyTime é um padrão para a representação estruturada de hipermédia e conteúdo baseado em tempo. Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (como áudio, vídeo, etc.), conectados por hiper-ligações. O padrão é independente de outros padrões de processamento de texto em geral.

SGML é um padrão de formatação de textos. Não foi desenvolvido para hipertexto, mas tornou-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações.

História

Tim Berners-Lee criou o HTML original (e outros protocolos associados como o HTTP) em uma estação NeXTcube usando o ambiente de desenvolvimento NeXTSTEP. Na época a linguagem não era uma especificação, mas uma coleção de ferramentas para resolver um problema de Tim: a comunicação e disseminação das pesquisas entre ele e seu grupo de colegas. Sua solução, combinada com a então emergente internet pública (que tornaria-se a Internet) ganhou atenção mundial.

As primeiras versões do HTML foram definidas com regras sintáticas flexíveis, o que ajudou aqueles sem familiaridade com a publicação na Web. Atualmente a sintaxe do HTML é muito mais rígida, permitindo um código mais preciso. Através do tempo, a utilização de ferramentas para autoria de HTML aumentou, assim como a tendência em tornar a sintaxe cada vez mais rígida. Apesar disso, por questões históricas (retrocompatibilidade), os navegadores ainda hoje conseguem interpretar páginas web que estão longe de ser um código HTML válido.

A linguagem foi definida em especificações formais na década de 1990, inspiradas nas propostas originais de Tim Berners-Lee em criar uma linguagem baseada em SGML para a Internet. A primeira publicação foi esboçada por Berners-Lee e Dan Connolly, e publicada em 1993 na IETFcomo uma aplicação formal para o SGML (com uma DTD em SGML definindo a gramática). A IETF criou um grupo de trabalho para o HTML no ano seguinte, e publicou o HTML 2.0 em 1995. Desde 1996, as especificações HTML vêm sendo mantidas, com o auxílio de fabricantes de software, pela World Wide Web Consortium (W3C).[1] Apesar disso, em 2000 a linguagem tornou-se também uma norma internacional (ISO/IEC 15445:2000). A última especificação HTML lançada pela W3C foi a recomendação HTML 4.01, publicada no final de 1999. Uma errata ainda

foi lançada em 2001.
Desde a publicação do HTML 3.5 no final de 1997, o grupo de trabalho da W3C tem cada vez mais — e de 2002 a 2006, de forma exclusiva — focado no desenvolvimento do

XHTML, uma especificação HTML baseada em XML que é considerada pela W3C como um sucessor do HTML.[2][3][4] O XHTML faz uso de uma sintaxe mais rigorosa e menos ambígua para tornar o HTML mais simples de ser processado e estendido.

Essas três linguagens, se podemos chamar assim, o HTML, o JavaScript e o CSS, elas são linguagens do lado do cliente, ou seja, todas são processadas e interpretadas pelo navegador do cliente. Já o PHP é uma linguagem que é interpretada também, mas no lado do servidor, tudo acontece no computador remoto, onde está instalado o Apache que interpretará o código PHP.

Tutorial indicado
http://www.w3schools.com/html/default.asp

Testes online
http://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro

1 - Tags e Atributos

http://www.w3schools.com/html/default.asp
http://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro

Uma página em HTML é formada por tags, etiquetas ou elementos e atributos dos elementos.

Uma TAG: <BODY>
Um atribulo: BGCOLOR e "red" é o valor do atributo:

<BODY BGCOLOR="red">

2 - Head, title e body

Vejamos o uso das tags: Head, title e body
Ajuda muito indentar as tags de acordo com a hierarquia.

```
<html>
	<head>
		<title>Título desta página</title>
		<!-- Aqui também geralmente ficam os includes do CSS e do JavaScript
		assim também como as metatags. -->
	</head>
	<body>
		Aqui fica o corpo da página.
		Onde adicionamos texto, imagens, etc.
	</body>
</html>
```

3 - Tabelas

As tabelas são muito úteis, especialmente para mostrar dados vindos de bancos de dados.

As tabelas são como planilhas, que tem linhas, colunas e células.

As linhas ficam na horizontal e são formadas pela tag <tr> e </tr>.
Para iniciar uma tabela usamos a tag:

<table>

Dica: use sempre aspas duplas para delimitar os valores dos atributos, para que seu código seja compatível com XML.

O código de uma tabela com 2 linhas com 2 colunas e cabeçalho.

```
<table border="1" width="80%">
<tr>
        <th>Cabeçalho</th>
        <th>Outro Cabeçalho</th>
</tr>
<tr>
        <td>linha 1, coluna 1</td>
        <td>linha 1, coluna 2</td>
</tr>
<tr>
        <td>linha 2, coluna 1</td>
        <td>linha 2, coluna 2</td>
</tr>
</table>
```

```
<table border="1" width="80%">
<tr>
    <th>Cabeçalho</th>
    <th>Outro Cabeçalho</th>
</tr>
<tr>
    <td>linha 1, coluna 1</td>
    <td>linha 1, coluna 2</td>
</tr>
<tr>
    <td>linha 2, coluna 1</td>
    <td>linha 2, coluna 2</td>
</tr>
</table>
```

Agora como a tabela é mostrada no navegador:

Cabeçalho	Outro Cabeçalho
linha 1, coluna 1	linha 1, coluna 2
linha 2, coluna 1	linha 2, coluna 2

Algo importante nas tabelas é o uso de colspan e rowspan para realizar merge nas colunas e linhas.

Para que a linha dois tenha apenas uma célula, então usaremos o código:

```
<table border="1" width="80%">
<tr>
        <th>Cabeçalho</th>
        <th>Outro Cabeçalho</th>
</tr>
<tr>
        <td>linha 1, coluna 1</td>
        <td>linha 1, coluna 2</td>
</tr>
<tr>
        <td colspan="2" align="center" valign="top">colspan=2</td>
</tr>
</table>
```

Comentários

Para comentar uma ou mais linhas em HTML usamos o comentário como a seguir:

```
<!-- ... -->
<!-- Aqui fica o
comentário de várias
linhas
-->
```

Agora vamos exibir apenas uma célula na coluna 1:

```
<table border="1">
        <tr>
                <td rowspan="3">Célula 1</td>
                <td>Célula 2</td>
        </tr>
        <tr>
                <td>Célula 3</td>
        </tr>
        <tr>
                <td>Célula 4</td>
        </tr>
</table>
```

Veja este outro:

```
<table border="1">
  <tr>
    <td colspan="2">Célula 1</td>
    <td>Célula 2</td>
  </tr>
  <tr>
    <td>Célula 3</td>
    <td>Célula 4</td>
    <td>Célula 5</td>
  </tr>
</table>
```

Bom tutorial: http://pt-br.html.net/tutorials/html/

4 - Formulários

Formulários são usados para enviar informações de uma página para outra e essa outra pode enviar para um banco de dados ou para outra finalidade.

Veja um formulário básico, contendo apenas um campo texto e um botão do tipo submit.

<form name="frmAcesso" action="menu.php" method="POST">
Login<input type="text" name="login" />
Senha<input type="password" name="senha" />
<input type="submit" value="Enviar" />
</form>

Tipos de controles de formulários

input: text, hidden, submit, button, reset, image, password
text – caixa de texto
hidden – campo oculto
password – oculta senha
radio – radio button, onde aparecem vários botões e somente um fica selecionado
checkbox - onde aparecem vários botões e podemos selecionar vários
textarea – grande caixa de texto, onde temos cols para colunas e rows para linhas
select – caixa de seleção. Internamente um option para cada opção.

submit – este tipo de botão ao ser ativado envia o conteúdo do form sem confirmação alguma
button – este é inativo por padrão. Para que tenha alguma ação devemos usar o JavaScript
reset – este resetar todos os campos ao estado original
image – mais um tipo de botão, que exibe uma imagem para clicar
<label> Define um rótulo para um controle
<fieldset> Define um fieldset
<legend> Define um título para um fieldset

Exemplo de Formulário com muitos recursos:

```
<h1>Formulário de Envio da Dados</h1>
<form name="frmInserir" action="inserir.php" method="POST">
Nome<input type="text" name="nome" size="45" maxlength="45" TABINDEX="1"
value="João de Brito" READONLY><br>
Login<input type="text" name="login" size="12" maxlength="12" TABINDEX="2"><br>
Senha<input type="password" name="senha" size="32" maxlength="32"
TABINDEX="3"><br>
<input type="hidden" name="controle" value="35">
Sexo<br>
Masculino<input type="radio" name="sexo" value="m" TABINDEX="4"><br>
Feminino<input type="radio" name="sexo" value="f" TABINDEX="5" CHECKED><br>
Animal de Estimação<br>
Gato<input type="checkbox" name="estimacao" value="gato" TABINDEX="6"><br>
Cachorro<input type="checkbox" name="estimacao" value="cao" TABINDEX="7"><br>
Pássaro<input type="checkbox" name="estimacao" value="gato" TABINDEX="8"
CHECKED><br>
Observação<br>
<textarea cols="80" rows="10"  TABINDEX="9">Deixe seu comentário</textarea><br>
Cor Preferida<br>
<select name="cor" MULTIPLE SIZE="3" TABINDEX="10">
<option value="azul">Azul
<option value="vermelho" SELECTED>Vermelho
<option value="verde">Verde
</select><br>
<input type="submit" name="enviar" value="Enviar" TABINDEX="11">
<input type="reset" name="limpar" value="Limpar" TABINDEX="12">
</form>
```

Vejamos alguns detalhes importantes:

MULTIPLE – Permite selecionar mais de uma opção, segurando o SHIFT ou o CTRL.
SIZE – a seleção já inicia com a quantidade aberta. Sem ela vem fechada por padrão, com apenas uma opção e abre ao clicar.
READONLY e DISABLED – impedem digitação, mas o disabled além de não permitir não envia seu conteúdo para o action. Usa-se solto, como no exemplo abaixo:

```
Nome<input type="text" name="nome" size="45" maxlength="45" TABINDEX="1"
value="João Brito Cunha" READONLY><br>
```

TABINDEX – este atributo determina como será a ordem ao se pressionar a tecla TAB.
SELECTED – este indica qual a opção default do SELECT, a que estará selecionada ao iniciar a página.
VALUE – este indica o valor default de um controle do tipo INPUT.
HIDDEN – atributo para campos tipo INPUT utilizado para repassar variáveis com valores. Evite repassar dados sigilosos, pois pode ser visualizado ao pedir a visualização dos fontes da página
CHECKED – Este indica o valor default de controles do tipo RADIO e CHECKBOX.
ACTION – atributo do form que indica o arquivo para onde serão submetidos os valores dos campos do form.

O destino do action pode ser a si mesmo usando action="" ou um e-mail, usando:
action="mailto:ribafs@gmail.com"
também para uma função em JavaScript, assim:
action="javascript:NomeFuncao()"
E ainda passar parâmetros assim:
action="inserir.php?codigo=3"
METHOD – atributo do form que indica um dos métodos de envio: GET ou POST.
O método GET tem a limitação de 1024 caracteres e envia através da URL, portanto não é indicado para grandes valores nem para valores sigilosos.
O método POST não tem limitação de tamanho e envia as informações através do ambiente do HTTP, o que torna um pouco mais seguro que o método GET (primeiro contacta o servidor e depois envia as informações).

Quando temos um form para **envio de arquivos por upload** devemos adicionar mais um atributo:
enctype="multipart/form-data"

Exemplo:
```
<form method="POST" action="upload.php" enctype="multipart/form-data">
<input type="file" name="arquivo" accept="image/*" />
<input type="submit" value="Enviar"/>
</form>
```

Botão do tipo imagem
```
<input type="image" src="nomeimagem.png" border="0" align="right" />
```

Ótimo tutorial - http://www.htmlcodetutorial.com/

20 Melhores Práticas em Formulários para Iniciantes:
http://net.tutsplus.com/tutorials/html-css-techniques/20-html-forms-best-practices-for-beginners/

5 - Frames

O frame é um recurso muito poderoso, mas que atualmente está caindo em desuso pelos profissionais mais qualificados e que estão adotando o CSS para realizar a mesma função.

Exemplo:

```
<FRAMESET ROWS="20%,*">
   <FRAME SRC="frame1_title.html" NAME="TITLE">
   <FRAME SRC="frame1_body.html"  NAME="MAIN">

<NOFRAMES>NOFRAMES stuff
</NOFRAMES>

</FRAMESET>
```

Este exemplo significa que o arquivo *frame1_title.html* ocupará 20% da área superior (horizontal)

enquanto que o *frame1_body.html* ocupará os 80% restantes.

```
<FRAMESET COLS="13%,73%,*">
     <FRAME SRC="esquerda.html">
     <FRAME SRC="centro.html" MARGINHEIGHT=1>
     <FRAME SRC="direita.html"  MARGINHEIGHT=50>

<NOFRAMES>NOFRAMES stuff
</NOFRAMES>

</FRAMESET>
```

Neste exemplo é reservado 13% da área vertical da página para o arquivo esquerda.html, mais 73% para o centro.html e
o restante para o direita.html

Veja um exemplo de como usar o CSS para substituir os frames:
http://ribafs.org/portal/projetos-opensource/84-utilitarios-em-php/143-templates-usando-css

6 - Links

Um link interliga páginas ou elementos de páginas como imagens, vídeos, sons, etc.

Um link é criado com a tag <a (âncora) e o atributo href, como no exemplo abaixo:

Cursos Gratuitos Online

Da forma acima temos o link de destino http://cursos.ribafs.org, o texto do link Cursos Gratuitos Online e também temos o target, que indica que o link será aberto em uma nova seção do navegador.

target:
"_blank" (abre numa nova seção),
"_parent" - usado com frames, onde um arquivo de frameset está dentro de outro arquivo de frameset. "_self" – abre o documento na janela atual.
"_top" - carrega o documento linkado para o frame acima (top).

Podemos anular o href para criar o link através do JavaScrpt ou então apenas executar uma função:

Teste

Teste

#destino

Podemos indicar um link para uma região de uma página, para isso primeiro damos um nome para a região, com:
Região 1 da página

Então criamos um link que chama a região, assim:
Região 1

Também podemos criar um link para um e-mail, assim:
Enviar E-mail

Outro:
<a href="mailto:joao@gmail.com?
cc=pedro@yahoo.com&bcc=antonio@gmail.com&subject=Muitas
%20Felicidades&body=Um%20convite%20para%20uma%20grande
%20primavera!">Enviar e-mail!

7 - Imagens

Para exibir uma imagen em uma página usamos a sintaxe:

<img src="imagem.png">

É bom usar o atributo ALT que mostra um texto alternativo para quando o navegador não carrega a imagem, como é o caso de deficientes visuais (acessibilidade).

<img src="imagem.png" ALT="Minha imagem">

Também podemos usar outro atributo que controla as bordas da imagem:
<img src="imagem.png" ALT="Minha imagem" BORDER="0">

Outros atributos: alinhamento: left, right, middle, top e largura (witdh) e altura (height) e num texto:

Uma imagem
<img src="hackanm.gif"
align="bottom" width="48" height="48">
num texto.

Entidades HTML
Esta página contém tabela da representação em forma de entidades HTML (seqüências começando com "&" e terminando com ";") para caracteres do alfabeto grego e alguns outros símbolos, que são bastante usados em literatura técnica e matemática:
http://www.mspc.eng.br/info/htmlEnt10.shtml

8 - Listas

Em HTML podemos criar listas ordenadas e não ordenadas.

Listas não ordenadas (unordered - ul) - É uma lista de itens, marcados por pequenos círculos, veja um exemplo:

```
<ul>
        <li>Café</li>
        <li>Leite</li>
</ul>
```

Irá aparecer no navegador assim:

- Café
- Leite

Listas ordenadas (ordered - ul) – É também uma lista de itens, mas sendo que os itens são precedidos de números.

```
<ol>
        <li>Café</li>
        <li>Leite</li>
</ol>
```

Irá aparecer no navegador assim:

1. Café
2. Leite

9 - Metatags

Metatags são muito importantes para a visibilidade dos sites pelos instrumentos de busca, pois é dos itens que os mesmos levam em conta em suas buscas.

As metatags, são atributos que ficam dentro da tag <head> ... </head> e alguns deles são:

<meta http-equiv="content-type" content="text/html; charset=UTF-8" />

<meta http-equiv="content-language" content="pt-br" />

<meta name="description" content="Cursos Gratuítos Online, CSS, HTML, PHP, Joomla" />

<meta name="keywords" content="HTML, CSS, PHP, Joomla" />

Bom tutorial - http://www.brunodulcetti.com/blog/2006/10/18/quais-e-como-utilizar-as-meta-tags-na-sua-pagina.html

10 – Formtação

Através do HTML também podemos formatar o conteúdo, como:

negrito – bold ou ...
itálico
<u>sublinhado – underline</u>
^{sobrescrito – superscript}
_{subscrito – subscript}

Dica: recomenda-se atualmente que toda a formatação fique a cargo do CSS e não do HTML.

Assim como também podemos alinhar o texto de várias formas:

...
<left>... </left>
<right>... </right>

E também nas tags, como:

<h1 align="center">... </h1>

Aqui também vale a recomendação para ao invés usar CSS.

Para uma referência online

http://www.w3schools.com/tags/default.asp
http://www.html-reference.com/
HTML 4.01 - http://www.w3.org/TR/html4/

Lista de Tags por categoria
http://htmlhelp.com/reference/html40/olist.html

Lista alfabética de Tags
http://htmlhelp.com/reference/html40/alist.html

XHTML1
Online http://www.w3.org/TR/xhtml1/
PDF - http://www.w3.org/TR/xhtml1/xhtml1.pdf

Tutoriais diversos:
http://www.criarweb.com/manuais/2/

Mapa e Área

```
<map name="bigthings" id="bigthings">
  <area shape="rect" coords="35,4,205,108"
```

```
      href="http://en.wikipedia.org/wiki/Australia's_Big_Things"
      alt="Australia's Big Things (on Wikipedia)"/>
   ⋮
</map>
<p><img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
   border="0" usemap="#bigthings"/></p>
```

Teclas de atalho:

```
<map name="bigthings" id="bigthings">
  <area shape="rect" coords="35,4,205,108"
      href="http://en.wikipedia.org/wiki/Australia's_Big_Things"
      alt="Australia's Big Things (on Wikipedia)" accesskey="b"/>
    ⋮
</map>
<p><img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
   border="0" usemap="#bigthings"/></p>
```

Atributo ALT

```
<map name="bigthings" id="bigthings">
  <area shape="rect" coords="35,4,205,108"
      href="http://en.wikipedia.org/wiki/Australia's_Big_Things"
     alt="Australia's Big Things (on Wikipedia)"/>
   ⋮
</map>
<p><img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
border="0" usemap="#bigthings"/></p>
```

Paradas de Tabulação - tabindex

```
<map name="bigthings" id="bigthings">
  <area shape="rect" coords="35,4,205,108" tabindex="3"
      href="http://en.wikipedia.org/wiki/Australia's_Big_Things"
      alt="Australia's Big Things (on Wikipedia)"/>
   ⋮
</map>
<p><img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
border="0" usemap="#bigthings"/></p>
```

Destino do Link - target

```
<map name="bigthings" id="bigthings">
  <area shape="rect" coords="35,4,205,108" target="_top"
      href="http://en.wikipedia.org/wiki/Australia's_Big_Things"
      alt="Australia's Big Things (on Wikipedia)"/>
   ⋮
</map>
```

```
<p><img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
border="0" usemap="#bigthings"/></p>
```

Som de fundo

```
<bgsound balance="number" loop="number" src="uri" volume="number">
```

Embutindo multimídia

```
<embed alt="string" height=" { number |
percentage } " hidden=" { true | false } "
pluginspage="uri" src="uri" type="MIME type"
width="number">
</embed>
```

Imagens

```
<img alt="string" height=" { number | percentage } "
src="uri" width=" { number | percentage } " />

<img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"/>
```

Alinhamento

```
align=" { bottom | left | middle | right | top } "

<p><img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
align="right"/>Driving along, we spotted a giant prawn, so had to
stop and take a closer look.</p>
```

Espaço Horizontal - hspace

```
hspace="number"

<p><img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
align="left" hspace="10"/>Driving along, we spotted a giant prawn,
so we had to stop and take a closer look.</p>
```

Atributo ismap para mapas

```
<img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
ismap="ismap"/>
```

Espaços verticais - vspace

vspace="number"
<img src="giant-prawn.jpg" alt="The Giant Prawn at Ballina"
align="left" vspace="10"/>Driving along, we spotted a giant prawn,
so we had to stop and take a closer look.

HTML EM EXEMPLOS

TAG BODY

<body text="#00FFFF" bgcolor="#000000" link="#FFFF00" vlink="#FF0000"background="http://www.yourdomain.com/yourdirectory/background.gif" onLoad="document.form1.campo1.focus()"**>**

<body text="#00FFFF" bgcolor="#000000" link="#FFFF00" vlink="#FF0000"background="background.gif"**>**

QUEBRA DE LINHA

** <p** ALIGN="center"**>Este parágrafo ficará
** Centralizado na página** **

FORMATAÇÃO DE TEXTO

<B>**Negrito**</B> ou **negrito**

<I>*Itálico*</i>

<U>Sublinhado</U>

<S>~~Riscado~~</S>

^{Sobrescrito}

_{Subscrito}

<big> grande </big>

<small> pequeno </small>

COR DO TEXTO (Cores HTML 3.2)

** Texto Verde **

aqua	black	blue	fuchsia
gray	green	lime	maroon
navy	olive	purple	red
silver	teal	white	yellow

FONTES PARA TÍTULOS

<H1 align="center"> Título1</H1>
<H2 align="center"> Título2</H2>
<H3 align="center"> Título3</H3>
<H4 align="center"> Título4</H4>
<H5 align="center"> Título5</H5>
<H6 align="center"> Título6</H6>

Linha Horizontal

<HR width="40%" hight=2px>

TIPOS DE FONTES

Este texto será exibido com fonte Arial, caso esteja disponível

Verdana

Verdana

Arial

Arial

```
<font size=+2 face="Helvetica">Helvetica</font>
```

Helvetica

```
<font size=+2 face="Impact">Impact</font>
```

Impact

```
<font size=+2 face="Comic Sans MS">Comic Sans MS</font>
```

Comic Sans MS

FRAMES

index

```
<frameset cols="250,*" frameborder="1" framespacing="0">
      <frameset rows="85%,*" frameborder="1" framespacing="0">
            <frame src="navegacao.html" frameborder="1" framespacing="0" scrolling="yes"
name="navegacao">
      </frameset>
     <frame src="conteudo.html" frameborder="1" framespacing="0" name="conteudo">
</frameset>
```

conteudo.html

Conteúdo

navegacao.html

```
Navegação<br>
<a href="conteudohist.html" target="conteudo">História</a>
```

FORMULÁRIOS

```
<form method=post action="arquivo_ou_pagina" onSubmit=”retorno=confirm('Tem certeza
de que deseja excluir '+login.value+' ?');if (!retorno) return false;>
<p><b><u>N</u></b>ome<input TYPE="text" NAME="nome" SIZE="40"
MAXLENGTH="40" ACCESSKEY="n">
    <input TYPE="hidden" NAME="passar" SIZE="40" MAXLENGTH="40">
    Senha<input TYPE="password" NAME="senha" SIZE="10" MAXLENGTH="10"></p>
    <p>País
        <select NAME="pais">
            <option VALUE="br" SELECTED>Brasil
            <option VALUE="ch">China
            <option VALUE="pe">Peru
        </select>
   </p>
   <p>Receber nossas Notícias?
        <input TYPE="checkbox" NAME="noticias" VALUE="Yes" checked TABINDEX=”1”>
    </p>
   <p>Receber nosso site?
        <input TYPE="checkbox" NAME="site" VALUE="Yes" checked>
    </p>
    <p>Qual seu computador?
        <input TYPE="radio" NAME="tipo" VALUE="pentium" checked>Pentium
        <input TYPE="radio" NAME="tipo" VALUE="486DX">486 DX
```

```
        <input TYPE="radio" NAME="tipo" VALUE="486SX">486 SX
    </p>
    <p>Comentários: <br>
        <textarea NAME="comentarios" ROWS="10" COLS="70" wrap="virtual">
        Aqui seus comentários sobre nosso site
        </textarea>
    </p>
    <p>
        <input TYPE="submit" NAME="subEnviar" VALUE="Enviar">
        <input TYPE="reset" NAME="resLimpar" VALUE="Limpar">
    </p>
</form>
```

COMENTÁRIOS

<!-- Este é um comentário HTML (múltiplas linhas) -->

LINKS/ÂNCORAS

Texto do link

ABRIR LINK DA MESMA PÁGINA

1- Seção de Dicas

2 - Salto para a seção de Dicas

Link para um e-mail

Uploaf (file)

```
        <p>
        <input type="hidden" name="MAX_FILE_SIZE" value="100" />
        <input name="arquivo" type="file"/></p>
```

TABELAS

<table border="1" bordercolor="red">

```
<tr bgcolor="yellow"><td>row 1, cell 1</td><td>row 1, cell 2</td></tr>
<tr><td bgcolor="blue">row 2, cell 1</td><td>row 2, cell 2</td></tr>
</table>
```

Row1,cell1	Row1,cell2
Row2,cell1	Row2,cell2

<table border="1" width=600 bordercolordark="black">

```
<tr><td colspan=2>row 1, cell 1</td></tr>
<tr><td width=150 height=5>row 2, cell 1</td><td width=450>row 2, cell
2</td></tr>
</table>
```


<table border="1" bordercolorlight="red">

```
<tr><td ROWSPAN=2>row 1, cell 1</td><td>row 1, cell 2</td></tr>
<tr><td>row 2, cell 2</td></tr>
</table>
```


ALINHAMENTO VERTICAL NO TEXTO DAS CÉLULAS DA TABELA
<td valign="top"> </td>
<td valign="middle"></td>
<td valign="bottom"> </td>

IMAGENS
<img src="boat.gif" alt="Texto de Dicas">

Exibindo imagem ao invés de texto no Link

<img src="http://www.davesite.com/graphx/davesmll.gif" align="right" border="0" height="200" width="350" hspace="20" vspace="20">

TEXTAREA

Attribute for `<TEXTAREA ...>`
`WRAP = SOFT | HARD | OFF`

WRAP describes how the text in the text area should wrap at the end of lines. Until this attribute came along, browsers generally did not do word wrapping. If you typed a line that was longer than the display area, the line just kept going, hopefully with the display area scrolling along. This was not the way people are used to entering text, so Netscape added the WRAP attribute.
SOFT wraps long lines in the text area for easy editing, much like a word processor. It does not, however, send the carriage returns to the server. This type of wrap is probably the most useful because it is the easiest for the user to edit, but it does not actually change any of their data. HARD looks to the user like SOFT, but the carriage returns the user sees *are* sent to the server. OFF does not wrap at all; it displays and sends the text exactly as typed in.

this code	produces this
<TEXTAREA NAME="SOFT" COLS=25 ROWS=5 *WRAP=SOFT*>	The quality of mercy is not strain'd, it droppeth as the gentle rain from heaven upon the place beneath
<TEXTAREA NAME="HARD" COLS=25 ROWS=5 *WRAP=HARD*>	The quality of mercy is not strain'd, it droppeth as the gentle rain from heaven upon the place beneath
<TEXTAREA NAME="OFF" COLS=25 ROWS=5 *WRAP=OFF*>	The quality of mercy is not strain'd, it droppeth as the gentle rain from heaven upon the place beneath.

Default behaviour differs between browsers <TEXTAREA NAME="NONE" COLS=25 ROWS=5>	The quality of mercy is not strain'd, it droppeth as the gentle rain from heaven upon the place beneath.
	Submit

You may from time to time see other variations on WRAP, such as VIRTUAL or PHYSICAL. Netscape introduced these attributes a few years ago as proposed extensions to HTML 3.0, then abandoned them. Officially, the HTML 4.0 specs don't list WRAP, but Netscape and MSIE list WRAP = HARD | SOFT | OFF in their guides. It's best to stick these three values.

2-JavaScript

Definição

JavaScript é uma linguagem usada geralmente no lado do cliente (interpretada pelo navegador do usuário). Com ela podemos criar funções, rotinas e verdedeiros programinhas dentro de uma página. Veja o caso de um jogo online em JavaScript. Ele é todo carregado para o computador do cliente e então o mesmo poderá jogar.
Uma ótima função do JavaScript é deixar o site mais amigável, podendo jogar o foco no primeiro campo de um form, podendo mostrar uma mensagem quando o usuário clicar num botão ou disparar um certo evento. Uma pequena linguagem auxiliar mas com muitos recursos.

Wikipédia - http://pt.wikipedia.org/wiki/Javascript

JavaScript é uma linguagem de programação criada pela Netscape em 1995, que a princípio se chamava LiveScript, a Netscape após o sucesso inicial desta linguagem, recebe uma colaboração considerável da Sun, após esta colaboração, podemos dizer que o JavaScript é uma linguagem compatível com a linguagem Java, por esta razão, a semelhança dos nomes "JavaScript".

A linguagem foi criada para atender, principalmente, as seguintes necessidades:

- Validação de formulários no lado cliente (programa navegador);
- Interação com a página.

Assim, foi feita como uma linguagem de script. Javascript tem sintaxe semelhante à do Java, mas é totalmente diferente no conceito e no uso.

De acordo com seu sistema de tipos JavaScript é:

- fraca - sua tipagem é mutável;
- dinâmica - uma variável pode assumir vários tipos diferentes durante a execução;
- implícita - as variáveis são declaradas sem tipo.

1. É interpretada, ao invés de compilada;
2. Possui ótimas ferramentas padrão para listagens (como as linguagens de script, de modo geral);
3. Oferece bom suporte a expressões regulares (característica também comum a linguagens de script).

Sua união com o CSS é conhecida como DHTML. Usando o Javascript, é possível modificar dinamicamente os estilos dos elementos da página em HTML.
Dada sua enorme versatilidade e utilidade ao lidar com ambientes em árvore (como um documento HTML), foi criado a partir desta linguagem um padrão ECMA, o ECMA-262, também conhecido como ECMAScript. Este padrão é seguido, por exemplo, pela linguagem ActionScript da Adobe(Antigamente Macromedia, porém a empresa foi vendida à Adobe).
Além de uso em navegadores processando páginas HTML dinâmicas, o JavaScript é hoje usado também na construção do navegador Mozilla, o qual oferece para a criação de sistemas GUI todo um conjunto de ferramentas (em sua versão normal como navegador, sem a necessidade de nenhum software adicional), que incluem (e não apenas) um interpretador de Javascript, um comunicador Javascript <-> C++ e um interpretador de XUL, linguagem criada para definir a interface gráfica de aplicações.

O uso de JavaScript em páginas XHTML, pelo padrão W3C, deve ser informado ao navegador da seguinte forma:

```
<script type="text/javascript">

/* aqui fica o script */

</script>
```

Caso contrário, o navegador irá interpretar o script como sendo código HTML, escrevendo-o na página.

2 - Instruções, delimitador de instrução e comentários

http://www.w3schools.com/js/default.asp
http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro

Instrução – é um conunto de comandos suficientes para que o interpretador realize uma função, para que possa executar a instrução e acontecer algo significativo.

Delimitador de instruções – no JavaScript podemos usar o ponto e vírgula ou a quebra de linha.

Comentários – Em JavaScript existem dois tipos de comentários:

// - comentáro oriundo do C++ e para uma única linha
/* ... */ - comentário oriundo do C e para múltiplas linhas.

```
// Aqui comenta apenas esta linha

/*
Aqui pode comentar
várias
linhas
*/
```

3 - Script interno

Podemos criar scripts em JavaScript dentro de um arquivo em HTML, como a seguir:

```
<html>
	<head>
		<title>Título da Página</title>

		<script>
		function alerta(){
			alert('Olá Mundo!');
		}
		</script>
	</head>
	<body>
	<a href="#" onClick="alerta()">Clique a aqui</a>
```

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

Este link nos serve para executar a função em JavaScript.

4 - Uso nas tags do HTML

Também podemos executar o código JavaScript diretamente dentro das tags do HTML. Vejamos alguns exemplos usando o form em HTML anteriormente mostrado.

```
<body onLoad="document.frmInserir .login.focus()">
<h1>Formulário de Envio da Dados usando JavaScript nas Tags</h1>

<a href="#" onClick="alerta()">Clique a aqui</a>
<form name="frmInserir" action="inserir.php" method="POST"
onSubmit="if(login.value==''') {alert('Favor preencher o login');login.focus();return false;}">
Nome<input type="text" name="nome" size="45" maxlength="45" TABINDEX="1"
value="João de Brito" READONLY><br>
Login<input type="text" name="login" size="12" maxlength="12" TABINDEX="2"><br>
Senha<input type="password" name="senha" size="32" maxlength="32"
TABINDEX="3"><br>
<input type="hidden" name="controle" value="35">
<input type="submit" name="enviar" value="Enviar" TABINDEX="11">
<input type="reset" name="limpar" value="Limpar" TABINDEX="12">
</form>

</body>
```

Inicialmente jogamos o foco no campo login ao carregar o documento com:
onLoad="document.frmInserir .login.focus()"

Depois ao submeter o form (ao clicar no botão Enviar) nós verificamos se o campo login está vazio, caso esteja emitimos um alert, jogamos o foco novamente no campo login e cancelamos a ação com:
onSubmit="if(login.value==''') {alert('Favor preencher o login');login.focus();return false;}"

Mas a forma ideal de trabalhar com JavaScript é em um arquivo externo e realizando um include no HTML ou PHP, como a seguir.

5 - Script externo

Criamos o arquivo, sem adicionar a tar incial <script> nem a final </script>, apenas com as funções e instruções e procedimentos, como no exemplo:

Vamos criar um arquivo chamado **funcoes.inc.js** com o conteúdo a seguir:

```
function login_foco(){
	document.frmInserir.login.focus();
}
```

```
function login_preencher(){
	if(document.frmInsert.login.value=='') {
		alert('Favor preencher o login');
		return false;
	}
}
```

Agora como fazer com que a página HTML saiba da existência das funções em JS e as use?
Devemos fazer o include do arquivo JS com as funções com a linha abaixo escrita entre as tags <head> e </head>:

```
<script type="text/javascript"  src="funcoes.inc.js"></script>
```

Veja o exemplo anterior mas agora usando um arquivo externo:

```
<head>
<script type="text/javascript"  src="funcoes.inc.js"></script>
<head>
<body onLoad="login_foco()">
<h1>Formulário de Envio da Dados usando JavaScript nas Tags</h1>

<a href="#" onClick="alerta()">Clique a aqui</a>
<form name="frmInserir" action="" method="POST" onSubmit="login_preencher()">
Nome<input type="text" name="nome" size="45" maxlength="45" TABINDEX="1"
value="João de Brito" READONLY><br>
Login<input type="text" name="login" size="12" maxlength="12" TABINDEX="2"><br>
Senha<input type="password" name="senha" size="32" maxlength="32"
TABINDEX="3"><br>
<input type="hidden" name="controle" value="35">
<input type="submit" name="enviar" value="Enviar" TABINDEX="11">
<input type="reset" name="limpar" value="Limpar" TABINDEX="12">
</form>
</body>
```

Obs.: Aqui tive problema quando usei o arquivo externo e criei a função login_preencher(), pois o return false não funcionou com meus testes no Firefox, já o trecho de código dentro da tag <form funcionou sem problemas.

6 - Variáveis

Nomes de variáveis em JavaScript podem ser curtos, com apenas uma letra como também podem ser descritivos, com várias letras e algarismos, sendo que não podem iniciar com um algarismo.

Exemplos:

```
var x;
var y = 4;
var nome = 'Antônio';
```

Veja que podemos apenas declarar a variável, sem qualquer valor como também já podemos declarar com valor.

Também podemos declarar sem o var, assim:
y = 4;
nome = 'Antônio';

7 - Operadores

Em JavaScript:

= é utilizado para atribuir valores (operador de atribuição)
+ é usado para somar valores ou para concatenar strings (operador aritmético)

Exemplos:

x = 5;
z = 3;
y = x + z;

Primeiro o lado direito do igual é somado e depois é atribuído a y.

Operadores aritméticos:

Operadores aritméticos

Operador	Descrição	Exemplo(s)
+	Soma valores.	`a = 2 + 3;` `b = b +1;`
-	Subtrai valores (como operador binário).	`x = x - 5;` `x = a - b`
-	Muda sinal (como operador unitário).	`x = -x;` `x = -(a + b);`
*	Multiplica valores.	`a = 2 * 3;` `b = c * 5;`
/	Divide valores.	`a = 50 / 3;` `b = b * 4;`
%	Resto da divisão.	`d = 5 % 3;` d assume valor 2.
++(var)	Incremento de 1 (antes).	Se x é 2, `y = ++x` faz x igual a 3 e depois y igual a 3.
(var)**++**	Incremento de 1 (depois).	Se x é 2, `y = x++` faz y igual a 2 e depois x igual a 3.
--(var)	Decremento de 1 (antes).	Se x é 2, `y = --x` faz x igual a 1 e depois y igual a 1.

(var)--	Decremento de 1 (depois).	Se x é 2, y = x- - faz y igual a 2 e depois x igual a 1.

Operadores de atribuição

Operador	Descrição	Exemplo(s)
=	Atribui o valor do operando esquerdo ao operando direito.	`x = 3;` `a = b + c;`
+=	Soma 2 valores e atribui o resultado ao primeiro valor.	`x += 3;` Se x era 1, passa para 4.
-=	Subtrai 2 valores e atribui o resultado ao primeiro.	`x -= 3;` Se x era 1, passa para -2.
***=**	Multiplica 2 valores e atribui o resultado ao primeiro.	`x *= 2;` Se x era 4, passa para 8.
/=	Divide 2 valores e atribui o resultado ao primeiro.	`x /= 2;` Se x era 4, passa para 2.
%=	Calcula o resto da divisão de 2 valores e atribui o resultado ao primeiro.	`x %= 2;` Se x era 3, passa para 1.

Operadores de comparação

Operador	Descrição	Exemplo(s), supondo a = 3 e b = 5
==	Verdadeiro se os operandos são iguais. Se não são do mesmo tipo, a linguagem tenta converter para a correta comparação.	`a == 3;` // retorna verdadeiro `a == b;` // retorna falso
!=	Verdadeiro se os operandos não são iguais. Se não são do mesmo tipo, a linguagem tenta converter para a correta comparação.	`a != 3;` // retorna falso `a != b;` // retorna verdadeiro
===	Verdadeiro se os operandos são iguais e do mesmo tipo.	`a === 3;` // retorna verdadeiro `a === "3";` // retorna falso
!==	Verdadeiro se os operandos não são iguais ou não são do mesmo tipo.	`a !== b;` // retorna verdadeiro `a !== "3";` // retorna verdadeiro
>	Verdadeiro se o operando esquerdo é maior que o direito.	`a > b;` // retorna falso `b > a;` // retorna verdadeiro
>=	Verdadeiro se o operando esquerdo é maior ou igual ao direito.	`a >= 3;` // retorna verdadeiro `b >= 7;` // retorna falso
<	Verdadeiro se o operando esquerdo é menor que o direito.	`a < b;` // retorna verdadeiro `b < a;` // retorna falso
<=	Verdadeiro se o operando esquerdo é menor ou igual ao direito.	`a <= 3;` // retorna verdadeiro `a <= 0;` // retorna falso

Operadores de strings

Operador	Descrição	Exemplo(s)
+	Concatenar strings.	str_1 = "Bom"; `str_2 = str_1 + " dia";` str_2 contém "Bom dia"

+=	Concatenar e atribuir o resultado ao operando da esquerda.	`str_1 = "Bom";`
		`str_1 += " dia";`
		
		str_1 contém "Bom dia"

Operadores lógico

Em geral são usados com expressões que retornam valores booleanos, isto é, verdadeiro ou falso.

Operador	Descrição	Exemplo(s), supondo a = 3 e b = 5
&&	E lógico: retorna verdadeiro se ambas as expressões são verdadeiras e falso nos demais casos	`a==3 && b<10` // retorna verdadeiro `a!=3 && b==5` // retorna falso
||	OU lógico: retorna verdadeiro se pelo menos uma das expressões é verdadeira e falso se todas são falsas	`a==3 || b<10` // retorna verdadeiro `a!=3 || b==5` // retorna verdadeiro `a==1 || b==3` // retorna falso
!	NÃO lógico: retorna verdadeiro se o operando é falso e vice-versa	

O operador + é usado em Strings

O operador + usado com strings concatena as strings.
str1 = "Cursos ";
str2 = "Gratuítos Online";
str = str1 + str2;

Dica: Ao adicionar string com número o resultado será um número.

str = 5 + "5"; // O resultado deverá ser 55
Exemplo:
<script>
var x=5;
y="5";
alert(x+y);
</script>

8 - Estruturas de controle

8.1 - if, else e else if

Sintaxe:

```
if (condição) {
        instruções a serem executadas se a condição for verdadeira;
}

if (condição) {
        instruções a serem executadas se a condição for verdadeira;
}else{
        instruções a serem executadas se a condição for false;
}

if (condição) {
        instruções a serem executadas se a condição for verdadeira;
}else if(condição2){
        instruções a serem executadas se a condição2 for verdadeira;
}else{
        instruções a serem executadas se a condição2 for falsa;
}
```

Exemplos:

```
var x = 2;
var y = 5;

if(x > y){
        alert('x é maior que y');
}

if(x > y){
        alert('x é maior que y');
}else{
        alert('y é maior que x');
}

if(x > y){
        alert('x é maior que y');
}else if(y>x){
        alert('y é maior que x');
}else{
        alert('y é menor que x');
}
```

Lembrando que o fluxo somente entrará em }else if(y>x){ se (x > y) for falso.

8.2 - for

Sintaxe:
```
for (var=valorinicial;var<=valorfinal;var=var+incremento){
        instruções a serem executadas tantas vezes quanto seja o laço;
}
```

Exemplo:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
        document.write("O número é " + i);
        document.write("<br />");
}
</script>
</body>
</html>
```

document.write() é uma função que escreve na tela.

8.3 - while

Sintaxe:
```
while (var<=valorfinal){
        instruções a serem executadas enquanto var for menor que valor final;
}
```

Exemplo:
```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5){
        document.write("O número é " + i);
        document.write("<br />");
        i++;
}
</script>
</body>
</html>
```

Observe que a variável precisou ser incrementada manualmente, caso contrário fica em loop infinito.

8.4 - do .. while

Sintaxe:
do{
	instruções a serem executadas;
}
while (var<=endvalue);

Observe a diferença marcante entre os laços while e do ... while. No primeiro não será executada nenhuma instrução caso a expressão inicial não seja verdadeira, já no laço do ... while as instruções são executadas pelo menos a primeira vez independente de a expressão ser ou não verdadeira. Devemos sempre ter essa diferença em mente ao usar os laços.

Exemplo:
```
<html>
<body>
<script type="text/javascript">
var i=0;
do{
	document.write("O número é " + i);
	document.write("<br />");
	i++;
}
while (i<=5);
</script>
</body>
</html>
```

8.5 - switch

Sintaxe:
```
switch (n){
	case 1:
		bloco1 a ser executado;
		break;
	case 2:
		bloco2 a ser executado;
		break;
	default:
		bloco a ser executado, caso n seja diferente de 1 e de 2;
}
```

Exemplo:
```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
```

```
var d=new Date();
oDia=d.getDay();
switch (oDia){
	case 5:
		document.write("Finalmente Sexta");
		break;
	case 6:
		document.write("Super Sábado");
		break;
	case 0:
		document.write("Domingo de alegria");
		break;
	default:
		document.write("Estou aguardando o final de semana!");
}
</script>
```

8.6 - break e continue

Os laços while, do...while, for e switch podem ser interrompidos usando-se os comandos brek e continue.

break – interrompe o laço e continua o código após o laço.

continue – este interrompe apenas a iteração atual, voltando para o início do laço para continuar a próxima iteração.

Observe que não é interação e sim iteração, ou seja, cada volta que dá o laço.

Exemplo de break:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++){
	if (i==3){
		break;
	}
	document.write("The number is " + i);
	document.write("<br />");
}
</script>
</body>
</html>
```

Exemplo de continue:

```
<html>
<body>
<script type="text/javascript">
```

```
var i=0
for (i=0;i<=10;i++){
        if (i==3){
                continue;
        }
        document.write("The number is " + i);
        document.write("<br />");
}
</script>
</body>
</html>
```

8.7 - for .. in

Este laço varre os elementos de um array ou as propriedades de um objeto.

Sintaxe:

```
for (variavel  in object){
        instruções a serem executadas;
}
```

Exemplo:

```
<html>
<body>
<script type="text/javascript">
var x;
var meuscarros = new Array();
meuscarros[0] = "Ford";
meuscarros[1] = "Volvo";
meuscarros[2] = "BMW";

for (x in meuscarros){
        document.write(meuscarros[x] + "<br />");
}
</script>
</body>
</html>
```

9 - Janelas de Diálogo

O JavaScript conta com 3 janelas especiais de diálogo: alert, confirm e prompt.

alert – exibe uma mensagem em uma caixa de diálogo com apenas um botão OK.

Exemplo:

```
alert('Algum texto!');
```

Não existe retorno neste diálogo.

confirm - neste diálogo é exibida uma mensagem e dois botões: OK e Cancel. Se pressionado OK retorna TRUE e se pressionado Cancel retorna FALSE.

Exemplo:
var retorno = confirm('Deseja realmente excluir?');

prompt – Este diálogo exibe uma mensagem e uma caixa de texto para receber um texto do usuário e dois botões OK e Cancel. É semelhante ao confirm, sendo que adicionado da caixa de texto.

```
var name=prompt("Favor entrar seu nome","João Brito Cunha");
if (nome!=null && nome!=""){
        document.write("Olá " + nome + "! Como você está?");
}
```

Confira também: http://www.quackit.com/javascript/tutorial/javascript_popup_boxes.cfm

Janela popup

Tipo especial de janela, onde podemos passar diversos parâmetros como largura, altura, distância da horizontal, da vertical, etc.

```
<head>
<script type="text/javascript">
<!--
function meuPopup() {
window.open( "http://www.google.com/", "minhaJanela",
"status = 1, height = 300, width = 300, resizable = 0" )
}
//-->
</script>
</head>

<body>
<form>
<input type="button" onClick="meuPopup()" value="PopUp!">
</form>
<p onClick="meuPopup()">Clique aqui também!</p>
</body>
```

Alguns parâmetros importantes:

- **dependent** – Subjanela fechará se a janela pai (a janela que abriu esta) fechar
- **fullscreen** – Exibe a janela no modo fullscreen no navegador
- **height** – A altura da nova janela em pixels
- **width** – A largura da nova janela em pixels
- **left** – Distância em pixel da margem esquerda da tela
- **top** – Distância em pixel da margem superior da tela
- **resizable** – Permite ao usuário que redimensione a jenela ou impedir de dimensionar. Atualmente não funciona no Firefox
- **status** – Exibir ou não a barra de status
- **menubar** – exibir ou não a janela de menu do navegador

- **scrollbar** – exibir ou não as barras de rolagem do navegador nas janelas
- **location** – exibir ou não a caixa de texto location do navegador

Tutorial com grande lista de parâmetros:
http://www.java2s.com/Tutorial/JavaScript/0380__Window/windowopen.htm

Exemplo:

```
window.open('window1.htm','the_first_window','location, toolbar, resizable');
```

JAVASCRIPT EM EXEMPLOS E FUNÇÕES ÚTEIS

Split - quebrar em partes separadas por um delimitador

```
document.getElementById("nomeElemento").value.split(",");
```

PROIBIR QUANTIDADE DE CARACTERES DIFERENTE

```
onBlur="if(this.value.length != 7) {alert('Somente 7 caracteres!')";
```

FOCO NO PRIMEIRO CAMPO DO FORMULÁRIO

```
<body onLoad="document.form1.campo1.focus()">
```

MUDAR DE CAMPO AO TECLAR ENTER

```
onKeyPress="if (event.keyCode == 13) numero.focus()"
// Ao teclar Enter vai para o campo numero (botão deve ser do tipo Button)
```

SOMENTE VALORES NUMÉRICOS DE 0-1 E NÃO NULOS

```
onBlur="if (isNaN(this.value) || (this.value >1) || (this.value=='') )
{alert('Somente número e menor ou igual a 1!')";
```

MAXIMIZAR JANELA

```
function maximizarJanela() {
   if (window.screen) {
     var aw = screen.availWidth;
     var ah = screen.availHeight;
     window.moveTo(0, 0);
     window.resizeTo(aw, ah);
  }
}
```

VALIDAR E-MAIL

```
function emailvalidation(entered, alertbox){
        // E-mail Validation by Henrik Petersen / NetKontoret
        // Explained at www.echoecho.com/jsforms.htm
        // Please do not remove this line and the two lines above.
        with (entered){
                apos=value.indexOf("@");
                dotpos=value.lastIndexOf(".");
                lastpos=value.length-1;
                if (apos<1 || dotpos-apos<2 || lastpos-dotpos>3 || lastpos-
dotpos<2) {
```

```
                        if (alertbox) {
                                alert(alertbox);
                        } return false;
                }else {
                        return true;
                }
        }
}
Uso:
<input name=campo2 onBlur="emailvalidation(campo2, 'E-mail incorreto')">
```

LIMITAR CARACTERES DE TEXTAREA

```
<BODY><table><tr><td>
<textarea onkeyup="max(this)" onkeypress="max(this)" rows="2" cols="35"
name="Area"></textarea><br>
<font id=Digitado color=red>0</font> Caracteres digitados   /   restam
<font id=Restante color=red>100</font>
</td></tr></table>
<SCRIPT LANGUAGE=javascript>
function max(txarea) {
total = 100;
tam = txarea.value.length;
str="";
str=str+tam;
Digitado.innerHTML = str;
Restante.innerHTML = total - str;
  if (tam > total){
    aux = txarea.value;
    txarea.value = aux.substring(0,total);
    Digitado.innerHTML = total
    Restante.innerHTML = 0
  }
}
</SCRIPT>
```

IMPRIMIR AUTOMATICAMENTE AO ABRIR

```
<script language="JavaScript">
self.print();
</script>
```

IMPRIMIR AO CLICAR

```
    Clique no botão para imprimir esta página.
    <form action="javascript:;">
      <input onclick="window.print()" type="button" value="Imprimir esta
página"/>
    </form>
```

ESTRUTURAS DE CONTROLE

if...else

```
bananas = 22
if (bananas == 6)
{
alert("Temos seis de bananas")
}
else if (bananas == 10)
{
alert("Temos dez bananas")
}
else
{
alert("Temos outra quantidade de bananas")
}
```

switch...case

```
farol = "amarelo"
```

```
switch (farol) {
case "vermelho":
alert("Pare")
break
case "amarelo":
alert("Atencao")
break
case "verde":
alert("Prossiga")
break
default:
alert("Cor ilegal")
}
letra = "e"
switch (letra) {
case "a":
case "e":
case "i":
case "o":
case "u":
alert("Vogal")
default:
alert("Outro caracter")
}
```

for

```
a = 2
for (i = 0; i < 2; i++)
{
a = i
}
alert(a)
```

while

```
numero = 0
while (numero < 10)
{
numero++
}
alert(numero)
```

do...while

```
numero = 0
do
{
numero++
}
while (numero < 10)
alert(numero)
```

EVENTOS MAIS COMUNS

Clik - Quando o usuário clica sobre um botão, um link ou outro elementos.
Load - Quando a página é carregada pelo browser. (<body onLoad="...")
Unload - Quando o usuário sai da página. (<body onUnload="...")
MouseOver - Quando o usuário coloca o ponteiro do mouse sobre um link ou outro elemento.
MouseOut - Quando o ponteiro do mouse não está sobre um link ou outro elemento.
Focus - Quando um elemento de formulário recebe o focu, isto é, está ativo.
Blur - Quando um elemento de formulário perde o focu, isto é, quando o deixa de estar ativo.
Change - Quando o valor de um campo de formulário é modificado.
Select - Quando o usuário seleciona um campo dentro de elemento de formulário.
Submit - Quando o usuário clica sobre o botão Submit para enviar um formulário.

MANIPULADORES DE EVENTOS COMUNS

OBJETO PROCEDIMENTOS DE EVENTOS DISPONÍVEIS
Janela onLoad, onUnload
Link hypertexto onClick, onmouseOver, on mouseOut
Elemento de texto onBlur, onChange, onFocus, onSelect
Elemento de zona de texto onBlur, onChange, onFocus, onSelect
Elemento botão onClick
Botão Radio onClick
Lista de selecção onBlur, onChange, onFocus
Botão Submit e Button onClick
Botão Reset onClick

ARRAYS

```
value=new Array;
for (number=1; number<=100; number=number+1)
{ value[number]=number*10};
```

Concatenação

```
var msgErro = new String("");
var msgHeader = new String("Erro: ");
var codigoErro = new String("X001");
msgErro = msgHeader.concat(codigoErro); // Saída: Erro: X001
```

TIPOS DE DADOS

Número - 3 ou 7.987, números Inteiro e flutuante. Inteiros podem ser positivos, 0 ou negativos;
Inteiros podem ser expressados como decimal (base 10), hexadecimal (base 16), e octal (base 8). Um decimal
integer literal consiste de uma sequência de dígitos sem um 0 inicial. Um 0 (zero) inicial em um inteiro
literal indica que este é um octal; iniciando com 0x (ou 0X) indica um hexadecimal. Inteiros Hexadecimais podem
incluir dígitos (0-9) e as letras a-f e A-F. Inteiros Octal podem incluir somente os dígitos 0-7.
Um floating-point pode conter qualquer ponto decimal, um "e" (uppercase ou lowercase), que é usado para representar
"potência de dez" em notação científica, ou ambas. A parte expoente é um "e" ou "E" seguido por um integer,
que pode ser sinalizado (precedido por "+" ou "-"). Um floating-point literal precisa ter ao menos um dígito e qualquer
quantidade de pontos decimais ou "e" (ou "E").

Boleano

O tipo boleano, boolean em inglês, serve para salvar ou sim ou um não, ou com outras palavras, um verdadeiro ou falso.
Utiliza-se para realizar operações lógicas, geralmente para realizar ações se o conteúdo de
uma variável é verdadeiro ou falso.
Se uma variável é verdadeira, então:
    Executo umas instruções Si no   Executo outras
Os dois valores que podem ter as variáveis boleanas são true ou false.

```
minhaBoleana = true
minhaBoleana = false
```

String

O último tipo de dados é o que serve para salvar um texto. Javascript só tem um tipo de dados para salvar texto
e nele, se podem introduzir qualquer número de caracteres. Um texto pode estar composto de números, letras e
qualquer outro tipo de caracteres e signos. Os textos se escrevem entre aspas, duplas ou simples.

```
meuTexto = "Miguel vai pescar"
meuTexto = '23%%$ Letras & *--*'
```

Objetos - myObj = new Object();
Null - Not the same as zero - no value at all. A null value is one that has no value and means nothing.
Undefined - A value that is undefined is a value held by a variable after it has been created, but before a value has been assigned to it.

NAVEGAR ENTRE PÁGINAS

```
<a href="#" onClick="history.back()">Voltar</a>
<a href="#" onClick="history.forward()">Avançar</a>
<a href="#" onClick="history.go(-3)">Voltar 3 páginas</a>
<a href="#" onClick="history.go(3)">Avançar 3 páginas</a>
```

REDIRECIONAMENTO

```
<a href="#" onClick="location='menu.html'">Menu</a>
```

Automaticamente

```
<head>
<meta http-equiv="refresh" content="10;url=http://www.vivaolinux.com.br/">
<!---- adcione somente esta linha no cabeçalho, o content corresponde ao tempo
em segundos e url será para onde será redirecionado o usuário ------>
</head>
<body>
Depois de 10 segundo você será redirecionado para o site <b>Viva o Linux</b>.
</body>
```

SOLICITANDO CONFIRMAÇÃO AO SUBMETER

```
<form action="" method="POST" onSubmit="if (confirm('Are you sure you want to
submit this data?'))
{alert('Thank you for your feedback.'); return true}
else
{return false}">

<INPUT TYPE=SUBMIT>
</form>
```

VARRER ELEMENTOS DO ARRAY FORMS

```
Os tipo texto terão conteúdos setados para vazio
var frm = window.document.forms[0]
for (var i = 0; i < frm.elements.length; i++) {
if (frm.elements[i].type == "text") {
frm.elements[i].value = ""
}
}
```

CONVERTER PARA MAIÚSCULAS

```
onKeyUp="campo.value = campo.value.toUpperCase()";
```

CHECAR CHECKBOX OU RADIO

```
document.forms[0].nomeElemento.chacked
document.formName.nomeelemento.checked
```

CONFIRM - Caixa de confirmação

```
if (confirm("Tem certeza de que deseja voltar ao menu?")) {
location.href = "menu.php";
} else {
alert("Você desistiu de ir ao menu!");
```

```
}
```

PROMPT - Caixa de diálogo onde o usuário pode entrar algum texto

```
var resposta = prompt("Qual o seu nome?","José")
if (resposta) {
alert("Olá, " + resposta + "!")
}
```

EXPRESSÃO CONDICIONAL (TERNÁRIO)

```
int x = 3, y = 4, maximo;
maximo = (x > y) ? x : y;
alert('Máximo '+maximo);
```

EXEMPLO DE USO DO FOCUS NO FIREFOX

```
<script>
function checkPassword(userPass) {
if (userPass.value !='riba') {
alert("Please enter your password again.")
userPass.focus() userPass.select() }
}
</script>
<body>
Entre com sua senha
<INPUT TYPE="password" NAME="userPass" onBlur="checkPassword(this)">
</body>
```

Outro:

```
No primeiro campo -> onBlur="if(this.value.length != 7) {alert('Somente 7
caracteres!'); foco=1}else{foco=0}";
No campo seguinte -> onFocus="if(foco==1) campoanterior.focus()"
```

DETECTANDO A RESOLUÇÃO E CORES DO USUÁRIO

```
<body>
<script language="JavaScript1.2">
<!--
document.writeln("Sua resolução está configurada para: " + screen.width + "x" +
screen.height +"");
document.writeln(" e " + screen.colorDepth + " bit de cores");
//-->
</script>
```

SUBSTRINGS

```
variavel="ribamar";
alert(variavel.substring(3,6));
Retorna do 3 (quarto) ao 6-1 (5, sexto) -> ama
```

RETORNAR CARACTERE EM POSIÇÃO

```
variavel="ribamar";
alert(variavel.charAt(4)); // retorna "m"
```

LOCALIZAR SUBSTRING EM STRING

```
variavel="ribamar";
alert(variavel.indexOf("ama")); // retorna 3, que é aposição do a inicial
```

SOBRESCREVER CARACTERES DE STRING

Vide item "REMOVER MÁSCARA DE MOEDA"

PERMITIR SOMENTE NUMÉRICOS

```
<input type="text" name="teste" size="18" maxlength="18" onkeypress="return
```

```
validaTecla(this, event)">
<script language="JavaScript">
<!--
function isNum( caractere ){
var strValidos = "0123456789"
if ( strValidos.indexOf( caractere ) == -1 )
return false;
return true;
}
function validaTecla(campo, event){
var BACKSPACE= 8;
var key;
var tecla;
CheckTAB=true;
if(navigator.appName.indexOf("Netscape")!= -1)
tecla= event.which;
else
tecla= event.keyCode;
key = String.fromCharCode( tecla);
//alert( 'key: ' + tecla + ' -> campo: ' + campo.value);
if ( tecla == 13 )
return false;
if ( tecla == BACKSPACE )
return true;
return ( isNum(key));
}
</script>
```

ADICIONAR ÍCONE AO LADO DA URL
Criar um arquivo chamado "favicon.ico" e deixá-lo no raiz do site.

FUNÇÕES IMPORTANTES
parseInt(valor)
parseFloat(valor)
Math.ceil(valor) - retorna próximo maior que valor
Math.floor(valor) - retorna próximo menor que valor
Math.round(valor) - retorna valor inteiro arredondado de valor
Math.pow(base,expoente) - potência
Math.max(nr1,nr2,nr3) - retorna máximo da lista
Math.min(nr1,nr2,nr3) - retorna mínimo da lista

MANIPULANDO DATAS
Fri Feb 09 21:17:02 2006
getDate() - Obtém o dia do mês (numérico de 1 a 31)
getDay() - Obtém o dia da semana (0 a 6)
getMonth() - Obtém o mês (numérico de 0 a 11)
getYear() - Obtém o ano
getHours() - Obtém a hora (numérico de 0 a 23)
getMinutes() - Obtém os minutos (numérico de 0 a 59)
getSeconds() - Obtém os segundos (numérico de 0 a 59)
DataToda = new Date()
DiaHoje = DataToda.getDay()

TIMEOUT

```
<head>
<script type="text/javascript">
function timeout(){
setTimeout("alert('Este alert abriu 10 segundos após você clicar no botão')",
10000)
}
```

```
</script>
</head>
<body>
<form>
<input type="button" onclick="timeout()" value="Set timeout">
</form>
```

ÚLTIMA ATUALIZAÇÃO

```
<script language="JavaScript" type="text/javascript">
<!--
var a;
a=new Date(document.lastModified);
lm_year=a.getYear();lm_year=((lm_year<1000)?((lm_year<70)?2000:1900):0)+lm_year;
lm_month=a.getMonth()+1;lm_month=((lm_month<10)?'0':'')+lm_month;
lm_day=a.getDate();lm_day=((lm_day<10)?'0':'')+lm_day;
lm_hour=a.getHours();lm_hour=((lm_hour<10)?'0':'')+lm_hour;
lm_minute=a.getMinutes();lm_minute=((lm_minute<10)?'0':'')+lm_minute;
lm_second=a.getSeconds();lm_second=((lm_second<10)?'0':'')+lm_second;
document.write("Última Alteração em: " + lm_day+'.'+lm_month+'.'+lm_year+'
'+lm_hour+':'+lm_minute+':'+lm_second);
// -->
</script>
```

CAMPO DE FRAME

```
var getframefieldvalue = parent.frames[1].document.myform.myfield.value
```

JANELA POPUP

```
win = window.open("hello.html","","width=150,height=150");
```

MARCAR/DESMARCAR TODOS OS CHECKBOX

```
<head>
<title>Exemplo: Marcar todos os checkboxes de uma página - DOM e
Javascript</title>
<script type="text/javascript">
function controlaMarcacao (marcar) {
var itens = document.getElementsByTagName("input");
var total = itens.length;
for (var k = 0; k < total; k++) {
if (itens.item(k).type != "checkbox") {
continue;
}
itens.item(k).checked = marcar;
}
}
</script>
</head>
<body>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p><input type="checkbox" /></p>
<p>
<input type="button" id="movendoObjetos" value="Marcar todos"
onclick="controlaMarcacao (true);" onkeypress="controlaMarcacao (true);" />
```

```
<input type="button" id="movendoObjetos" value="Desmarcar todos"
onclick="controlaMarcacao (false);" onkeypress="controlaMarcacao (false);" />
</p>
</body>
```

ARREDONDAR NÚMEROS COM DECIMAIS

```
}
function round(number,decPlace) {
decPlace = (!decPlace ? 2 : decPlace);
return Math.round(number * Math.pow(10,decPlace)) / Math.pow(10,decPlace);
}
```

FORMATAR NÚMEROS

```
function numberFormat(amount) {
var rawNumStr = round(amount) + '';
rawNumStr = (rawNumStr.charAt(0) == '.' ? '0' + rawNumStr : rawNumStr);
if (rawNumStr.charAt(rawNumStr.length - 3) == '.') {
      return rawNumStr
} else if (rawNumStr.charAt(rawNumStr.length - 2) == '.') {
      return rawNumStr + '0';
} else { return rawNumStr + '.00'; }
}
```

FORMATAR MOEDA AO DIGITAR

```
<script language="JavaScript">
function FormataValor(campo,tammax,teclapres) {
var tecla = teclapres.keyCode;
vr = document.form[campo].value;
vr = vr.replace( "/", "" );
vr = vr.replace( "/", "" );
vr = vr.replace( ",", "" );
vr = vr.replace( ".", "" );
vr = vr.replace( ".", "" );
vr = vr.replace( ".", "" );
vr = vr.replace( ".", "" );
tam = vr.length;
if (tam < tammax && tecla != 8){ tam = vr.length + 1; }
if (tecla == 8 ){ tam = tam - 1; }
if ( tecla == 8 || tecla >= 48 && tecla <= 57 || tecla >= 96 && tecla <= 105 ){
if ( tam <= 2 ){
document.form[campo].value = vr; }
if ( (tam > 2) && (tam <= 5) ){
document.form[campo].value = vr.substr( 0, tam - 2 ) + ',' + vr.substr( tam - 2,
tam ); }
if ( (tam >= 6) && (tam <= 8) ){
document.form[campo].value = vr.substr( 0, tam - 5 ) + '.' + vr.substr( tam - 5,
3 ) + ',' + vr.substr( tam - 2, tam ); }
if ( (tam >= 9) && (tam <= 11) ){
document.form[campo].value = vr.substr( 0, tam - 8 ) + '.' + vr.substr( tam - 8,
3 ) + '.' + vr.substr( tam - 5, 3 ) + ',' + vr.substr( tam - 2, tam ); }
if ( (tam >= 12) && (tam <= 14) ){
document.form[campo].value = vr.substr( 0, tam - 11 ) + '.' + vr.substr( tam -
11, 3 ) + '.' + vr.substr( tam - 8, 3 ) + '.' + vr.substr( tam - 5, 3 ) + ',' +
vr.substr( tam - 2, tam ); }
if ( (tam >= 15) && (tam <= 17) ){
document.form[campo].value = vr.substr( 0, tam - 14 ) + '.' + vr.substr( tam -
14, 3 ) + '.' + vr.substr( tam - 11, 3 ) + '.' + vr.substr( tam - 8, 3 ) + '.' +
vr.substr( tam - 5, 3 ) + ',' + vr.substr( tam - 2, tam );}
```

```
}
for (var ct = 0; ct < document.form.elements.length; ct++) {
if (document.form.elements[ct].name == document.form.elements[campo].name) {
if ( !teclapres.shiftKey && tecla == 9 && document.form.elements[ct+1] &&
document.form.elements[ct+1].name == "senhaConta" && document.applets['tclJava']
){
document.applets['tclJava'].setFocus();
}
}
}
}
</script>
</head>
<body>
<form method="POST" name="form">
<p><input type="Text" name="valor" size="23" maxlength="17"
onKeyDown="FormataValor('valor', 13, event)"></p>
</form>
```

REMOVER MÁSCARA DE MOEDA

```
<input type=text name=nome
onBlur="n=nome.value;n=n.replace('.','');n=n.replace(',','.');nome.value=n">
```

TECLAS DE ATALHO PARA FORMS

```
<H2>TECLAS DE ATALHO</H2>
<FORM>
<u>C</u>ódigo<input type=text name=codigo accesskey="c"><br>
<u>N</u>ome<input type=text name=nome accesskey="n"><br>
<u>E</u>-mail<input type=text name=email accesskey="e"><br>
<input type=submit value=Enviar>
</form>
```

CÓDIGOS DE TECLAS

```
<HEAD>
<script type="text/javascript">
function init() {
document.onkeydown = showKeyDown;
document.onkeyup = showKeyUp;
document.onkeypress = showKeyPress;
}
function showKeyDown(evt) {
evt = (evt) ? evt : window.event;
document.getElementById("pressKeyCode").innerHTML = 0;
document.getElementById("upKeyCode").innerHTML = 0;
document.getElementById("pressCharCode").innerHTML = 0;
document.getElementById("upCharCode").innerHTML = 0;
restoreModifiers("");
restoreModifiers("Down");
restoreModifiers("Up");
document.getElementById("downKeyCode").innerHTML = evt.keyCode;
if (evt.charCode) {
document.getElementById("downCharCode").innerHTML = evt.charCode;
}
showModifiers("Down", evt);
}
function showKeyUp(evt) {
```

```
evt = (evt) ? evt : window.event;
document.getElementById("upKeyCode").innerHTML = evt.keyCode;
if (evt.charCode) {
document.getElementById("upCharCode").innerHTML = evt.charCode;
}
showModifiers("Up", evt);
return false;
}
function showKeyPress(evt) {
evt = (evt) ? evt : window.event;
document.getElementById("pressKeyCode").innerHTML = evt.keyCode;
if (evt.charCode) {
document.getElementById("pressCharCode").innerHTML = evt.charCode;
}
showModifiers("", evt);
return false;
}
function showModifiers(ext, evt) {
restoreModifiers(ext);
if (evt.shiftKey) {
document.getElementById("shift" + ext).style.backgroundColor = "#ff0000";
}
if (evt.ctrlKey) {
document.getElementById("ctrl" + ext).style.backgroundColor = "#00ff00";
}
if (evt.altKey) {
document.getElementById("alt" + ext).style.backgroundColor = "#0000ff";
}
}
function restoreModifiers(ext) {
document.getElementById("shift" + ext).style.backgroundColor = "#ffffff";
document.getElementById("ctrl" + ext).style.backgroundColor = "#ffffff";
document.getElementById("alt" + ext).style.backgroundColor = "#ffffff";
}
</script>
</head>
<body onload="init()">
<h1>Keyboard Event Handler Lab</h1>
<hr />
<form>
<table border="2" cellpadding="2">
<tr>
<th></th>
<th>onKeyDown</th><th>onKeyPress</th><th>onKeyUp</th>
</tr><tr>
<th>Key Codes</th>
<td id="downKeyCode">0</td><td id="pressKeyCode">0</td><td id="upKeyCode">0</td>
</tr><tr>
<th>Char Codes (IE5/Mac; NN6)</th>
<td id="downCharCode">0</td><td id="pressCharCode">0</td><td
id="upCharCode">0</td>
</tr><tr>
<th rowspan="3">Modifier Keys</th><td><span
id="shiftDown">Shift</span></td><td><span id="shift">Shift</span></td><td><span
id="shiftUp">Shift</span></td>
</tr><tr>
<td><span id="ctrlDown">Ctrl</span></td><td><span
id="ctrl">Ctrl</span></td><td><span id="ctrlUp">Ctrl</span></td>
</tr><tr>
<td><span id="altDown">Alt</span></td><td><span
```

```
id="alt">Alt</span></td><td><span id="altUp">Alt</span></td>
</tr>
</table></form>


TECLA PRESSIONADA
<body onKeyPress =" show_key(event.which)">
<form method="post" name="my_form">
The key you pressed was:
<input type="text" name="key_display" size="2">
</form>
<script language="JavaScript">
function show_key ( the_key )
{
if ( ! the_key )
{
the_key = event.keyCode;
}
document.my_form.key_display.value
= String.fromCharCode ( the_key );
}
</script>



COMPLETAR COM ZEROS À ESQUERDA
<script language="Javascript">
function preenche(campo, tamanho){
//captura do texto
var strText = campo.value;
/*verifica se o campo está vazio e pede
a confirmação do valor*/
if (strText == "" ) {
//apresenta a caixa de confirmação
if (confirm("Texto vazio - preencher com " + tamanho + " zeros? ")) {
//preeche com a quantidade de zeros informada no
//parâmetro tamanho
for (i=0; i<tamanho;i++)
campo.value += "0";
}else{
//a pessoa clicou no cancelar, voltando o foco para
//o campo
campo.focus();
}
}else{ //há alguma string no campo
//tamanho da string
var intTamStr = strText.length;
/*verifica se o tamanho da string eh
menor ou igual ao tamanho que eh pedido
na funcão*/
if (intTamStr <= tamanho){//executa a adaptação do texto
//quantos zeros serão incluídos no texto
var intTam = parseInt(tamanho) - intTamStr;
//preenchimento do campo
for (i=0; i<intTam; i++){
strText = "0" + strText;
}
//atribuição da variável ao campo
```

```
campo.value = strText;
}else{
// o texto eh maior do que eh pedido na função
alert("Este campo pode ter no máximo \n" + tamanho + " caracteres.");
campo.focus();
}
}
}
</script>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FORM METHOD=POST ACTION="asparena.asp" name="fTest">
<INPUT TYPE="text" NAME="primeiro" onBlur="preenche(this, 10)";>
<INPUT TYPE="submit">
</FORM>
```

Exibir/Ocultar um controle

```
<style type="text/css">
//#coef {position:absolute; width:200px; background-color:#66FF66; border:1px
solid #000000; padding:10px;}
#coef {position:absolute; width:100px; background-color:#FFFFFF;}

#coef2 {position:absolute; left: 500px; width:102px; background-color:#FFFFFF; }

</style>
<script type="text/javascript">

var ns4 = (document.layers);
var ie4 = (document.all && !document.getElementById);
var ie5 = (document.all && document.getElementById);
var ns6 = (!document.all && document.getElementById);

function exibe(par_id){
if(ns4){document.layers[par_id].visibility = "show";}
else if(ie4){document.all[par_id].style.visibility = "visible";}
else if(ie5 || ns6){document.getElementById(par_id).style.visibility =
"visible";}}

function oculta(par_id){
if(ns4){document.layers[par_id].visibility = "hide";}
else if(ie4){document.all[par_id].style.visibility = "hidden";}
else if(ie5 || ns6){document.getElementById(par_id).style.visibility =
"hidden";}}
</script>

<center>
<a href="#" onclick="exibe('coef2')">Exibe</a>
<a href="#" onclick="oculta('coef2')">Oculta</a>
<br><br><br>
<div id="coef" align=left><b>Coeficiente</b></div></center>
<div id="coef2"><b>DMT</b></div></center>
</center>
```

Dicas Seletas de JavaScript

1 - Aceitar Somente Vogais Minúsculas

Observe como usar as funções em outros exemplos.

```
function isVogal( caractere ) {
var strValidos = "aeiou"
if ( strValidos.indexOf( caractere ) == -1 )
  return false;
return true;
}
```

2 - Aceitar Alfabeto Maiúsculo e Minúsculo com Acentos, Espaço, Ponto ...

```
function isAlfabeto( caractere ) {
var strValidos =
"ABCDEFGHIJKLMNOPQRSTUVWXYZÃÁÉÍÓÜÊÔabcdefghijklmnopqrstuvwxyzáéíóãêôüaeio
u-/ ."
if ( strValidos.indexOf( caractere ) == -1 )
  return false;
return true;
}
```

3 - Formatando Valores Monetários

```
<SCRIPT language="JavaScript">
<!-- Begin
```

```
function formatCurrency(num) {
num = num.toString().replace(/\$|\,/g,"""");
if(isNaN(num)) num = "0";
cents = Math.floor((num*100+0.5)%100);
num = Math.floor((num*100+0.5)/100).toString();
if(cents < 10) cents = "0" + cents;
for (var i = 0; i < Math.floor((num.length-(1+i))/3); i++)
num =
num.substring(0,num.length-(4*i+3))+""."+num.substring(num.length-(4*i+3));
return (""R$"" + num + "","" + cents);
}
// End -->
</script>
```

4 - Abrir Página de Acordo com Resolução do Monitor do Visitante

```
<html><head>
<script LANGUAGE="JavaScript">
<!--
document.write('<font face=verdana size=2 color=black>sua resolução é ' + screen.width
+ ' x ' + screen.height + '');
if (screen.width == 640) {
document.write('<br><font face=verdana size=2 color=black>redirecionamento para
página correspondente...');
alert("sua resolução é 640 x 480 será aberta uma nova página correspondente a sua
resolução");
// página que abre para está resolução //
open('pagina640.html');
}
if (screen.width == 800) {

document.write('<br><font face=verdana size=2 color=black>redirecionamento para
página correspondente...');

alert("sua resolução é 800 x 600 será aberta uma nova página correspondente a sua
resolução");

// página que abre para está resolução //

open('pagina800.html');

}

if (screen.width == 1024) {

document.write('<br><font face=verdana size=2 color=black>redirecionamento para
página correspondente...');

alert("sua resolução é 1024 x 768 será aberta uma nova página correspondente a sua
resolução");

// página que abre para está resolução //
```

```
open('pagina1024.html');

}

if (screen.width == 1280) {

document.write('<br><font face=verdana size=2 color=black>redirecionamento para página correspondente...');

alert("sua resolução é 1280 x 960 será aberta uma nova página correspondente a sua resolução");

// página que abre para está resolução //

open("pagina1280.html");

}

if (screen.width == 1600) {

document.write('<br><font face=verdana size=2 color=black>redirecionamento para página correspondente...');

alert("sua resolução é 1600 x 1200 será aberta uma nova página correspondente a sua resolução");

// página que abre para está resolução //

open("pagina1600.html");

}

//-->

</script> </head> </html>
```

5 - Voltar para página anterior

```
<a href="javascript:history.back()">Voltar</a>
```

6 - Passar para a Próxima Página

```
<a href="javascript:history.forward()">Próxima</a>
```

7 - Maximizar Janela

```
function maximizarJanela() {
   if (window.screen) {
     var aw = screen.availWidth;
     var ah = screen.availHeight;
     window.moveTo(0, 0);
     window.resizeTo(aw, ah);
  }
}
```

8 - Máscara para formatar telefones em formulários HTML no formato (31)2211-1122

```
<SCRIPT LANGUAGE="JavaScript">
<!-- Begin
var n;
var p;
var p1;
function ValidatePhone(){
p=p1.value
if(p.length==2){
        //d10=p.indexOf('(')
        pp=p;
        d4=p.indexOf('(')
        d5=p.indexOf(')')
        if(d4==-1){
                pp="("+pp;
        }
        if(d5==-1){
                pp=pp+")";
        }
        //pp="("+pp+")";
        document.form1.telefone.value="";
        document.form1.telefone.value=pp;
}
if(p.length>2){
        d1=p.indexOf('(')
        d2=p.indexOf(')')
        if (d2==-1){
                l30=p.length;
                p30=p.substring(0,4);
                //alert(p30);
                p30=p30+")"
                p31=p.substring(4,l30);
                pp=p30+p31;
                //alert(p31);
                document.form1.telefone.value="";
                document.form1.telefone.value=pp;
        }
        }
if(p.length>5){
        p11=p.substring(d1+1,d2);
        if(p11.length>3){
        p12=p11;
        l12=p12.length;
        l15=p.length
        //l12=l12-3
        p13=p11.substring(0,3);
        p14=p11.substring(3,l12);
        p15=p.substring(d2+1,l15);
```

```
        document.form1.telefone.value="";
        pp="("+p13+")"+p14+p15;
        document.form1.telefone.value=pp;
        //obj1.value="";
        //obj1.value=pp;
        }
        l16=p.length;
        p16=p.substring(d2+1,l16);
        l17=p16.length;
        if(l17>3&&p16.indexOf('-')==-1){
                p17=p.substring(d2+1,d2+5);
                p18=p.substring(d2+5,l16);
                p19=p.substring(0,d2+1);
                //alert(p19);
        pp=p19+p17+"-"+p18;
        document.form1.telefone.value="";
        document.form1.telefone.value=pp;
        //obj1.value="";
        //obj1.value=pp;
        }
}
//}
setTimeout(ValidatePhone,100)
}
function getIt(m){
n=m.name;
//p1=document.forms[0].elements[n]
p1=m
ValidatePhone()
}
function testphone(obj1){
p=obj1.value
//alert(p)
p=p.replace("(","")
p=p.replace(")","")
p=p.replace("-","")
p=p.replace("-","")
//alert(isNaN(p))
if (isNaN(p)==true){
alert("Check phone");
return false;
}
}
//  End -->
</script>
```

No formulário, inclua a seguinte função no <input> :

```
<input type="text" size="20" name="telefone"
value="" onclick="javascript:getIt(this)" maxlength="13">
```

9 - Rotinas de Validação em JavaScript

Algumas rotinas básicas para validação de campos em formulários de diversos tipos (data, string, números, email).

```
<html><body>
<form method="post" action="formulario.htm" onsubmit="return validateForm(this)">
<TABLE BORDER=1>
<TR>
<TD>Nome:</TD><TD><input name="nome" size="32" ></TD>
</TR>
<TR>
<TD>Email:</TD><TD><input name="email" size="32" ></TD>
</TR>
<TR>
<TD>Data:</TD><TD><input name="data" size="16" ><i>(mm/dd/yyyy)</i></TD>
</TR>
<TR>
<TD>Valor:</TD><TD><input name="valor" size="4" >
<input type="submit" value="Enviar" name="submit">
</TD>
</TR>
</TABLE>
</form>

<script Language="JavaScript">

function isEmailAddr(email){
  var result = false;
  var theStr = new String(email);
  var index = theStr.indexOf("@");
  if (index > 0){
    var pindex = theStr.indexOf(".",index);
    if ((pindex > index+1) && (theStr.length > pindex+1))
          result = true;
  }
  return result;
}

function validRequired(formField,fieldLabel){
          var result = true;
          if (formField.value == ""){
                    alert('O campo "' + fieldLabel +'" deve ser preenchido.');
                    formField.focus();
                    result = false;
          }

          return result;
}
```

```
function validEmail(formField,fieldLabel,required){
	var result = true;

	if (required && !validRequired(formField,fieldLabel))
		result = false;

	if (result && ((formField.value.length < 3) || !isEmailAddr(formField.value)) )	{
		alert("O email foi digitado de forma incorreta");
		formField.focus();
		result = false;
	}

  return result;

}

function validInt(formField,fieldLabel,required){
	var result = true;
	if (required && !validRequired(formField,fieldLabel)) result = false;
	if (result){
		var num = parseInt(formField.value);
		if (isNaN(num)){
			alert('Por favor, preencha o campo "' + fieldLabel +'" com um número
válido.');
			formField.focus();
			result = false;
		}
	}
	return result;
}

function validateForm(theForm){
   if (!validRequired(theForm.nome,"Nome")) return false;
   if (!validEmail(theForm.email,"Email",true)) return false;
   if (!validDate(theForm.data,"Data",true)) return false;
   if (!validNum(theForm.valor,"Valor",true)) return false;
   return true;
}

</script>

</body></html>
```

Fonte: http://lib.seven.com.br/ampliar.asp?codcat=18&codartigo=415

10 - Rotinas a serem usados dentro dos campos do form

onChange="this.value=this.value.toUpperCase();" //Converte para maiúsculas
onBlur="if (isNaN(txtSiafi.value)) alert('Somente algarismos (0 a 9) são aceitos!');
s.focus();"
onKeyPress="if (event.keyCode == 13) numero.focus()" // Ao teclar Enter vai para o campo numero (botão Button)

Funções:

```
function isNumeric(pNum){
// http://www.superpro.com.br
	if (pNum==''){
		return false;
	}
	for (var i = 0; i < pNum.length; i++){
		var ch = pNum.substring(i, i + 1);
		if (ch < '0' || '9' < ch){
			return false;
		}
	}
	return true;
}

function IsDate(pData){
// http://www.superpro.com.br
	if(pData.length<10 || pData.length>10){
		alert('Data inválida\nInforme a data no formato (dd/mm/aaaa)');
		return false;
	}
	var ano = '' + pData.substring(6,10);
	var mes = '' + pData.substring(3,5);
	var dia = '' + pData.substring(0,2);

	if(dia>'31'){
		alert('Data inválida');
		return false;
	}
	if(mes>'12'){
		alert('Data inválida');
		return false;
	}
	if(mes=='02'){
		if(ano%4!=0 && dia>'28'){
			alert('Data Inválida');
			return false;
		}
		else{
			if(dia>'29'){
```

```
                        alert('Data Inválida');
                        return false;
                }
        }
        if(mes<='07'){
                if(mes%2==0 && dia>'30'){
                        alert('Data inválida');
                        return false;
                }
        }
        else{
                if(mes>'09'){
                        if(mes%2!=0 && dia>'30'){
                                alert('Data inválida');
                                return false;
                        }
                }
        }

        return true
}
function UCase(campo){
// http://www.superpro.com.br
        return campo.toUpperCase()
}

function LCase(campo){
// http://www.superpro.com.br
        return campo.toLowerCase()
}

function RTrim(campo){
// http://www.superpro.com.br
        y=true;
        while(y==true){
                x = campo.length;
                if(Right(campo,1)==' '){
                        campo = Left(campo,x-1);
                        y=true;
                }
                else{
                        y=false
                }
        }
        return campo;
}

function LTrim(campo){
// http://www.superpro.com.br
```

```
        y=true;
        while(y==true){
                x = campo.length-1;
                if(Left(campo,1)==' '){
                        campo = Right(campo,x);
                        y=true;
                }
                else{
                        y=false;
                }
        }
        return campo;
}

function Trim(campo){
// http://www.superpro.com.br
        return RTrim(LTrim(campo));
}
```

11- Exemplo de Máscara e Validação de Datas

```
<head>
<script>
function mascara_data(recebedata,ind){
// ind é o número correspondente ao índice do campo no form
// Form com 4 campos: ind=0 para o primeiro e ind=3 para o último
     var mydata = "";
     mydata = mydata + recebedata;
     if (mydata.length == 2){
        mydata = mydata + "/";
        document.forms[0].elements[ind].value = mydata;
     }
     if (mydata.length == 5){
        mydata = mydata + "/";
        document.forms[0].elements[ind].value = mydata;
     }
     if (mydata.length == 10){
        verifica_data(ind);
     }
}

function verifica_data (ind) {
   dia = (document.forms[0].elements[ind].value.substring(0,2));
   mes = (document.forms[0].elements[ind].value.substring(3,5));
   ano = (document.forms[0].elements[ind].value.substring(6,10));
   situacao = "";

   // verifica o dia valido para cada mes
   if ((dia < 01)||(dia < 01 || dia > 30) && (  mes == 04 || mes == 06 || mes == 09 || mes ==
11 ) || dia > 31) {
```

```
        situacao = "falsa";
    }

    // verifica se o mes e valido
    if (mes < 01 || mes > 12 ) {
        situacao = "falsa";
    }

    // verifica se e ano bissexto
    if (mes == 2 && ( dia < 01 || dia > 29 || ( dia > 28 && (parseInt(ano / 4) != ano / 4)))) {
        situacao = "falsa";
    }

    if (document.forms[0].elements[ind].value == "") {
        situacao = "falsa";
    }
    if (situacao == "falsa") {
        alert("Data inválida!");
        document.forms[0].elements[ind].focus();
    }
}
</script>
</head>

<body bgcolor=#FFFFC0 onload=document.frmferias.periodo1inicio.focus();>
<form name=frmferias method=post action=escala.php>
P E R Í O D O S : (dd/mm/aaaa)</b><BR><BR>
<b>Primeiro <input type="text" name="periodo1inicio"
OnKeyUp="mascara_data(this.value,3)">
<input type=button value=Confirmar>
</form>
</body>
```

12 - Detectar Browser

Basta copiar para a página

```
<script LANGUAGE="JavaScript">
<!--
var nombre = navigator.appName
 if (nombre == "Microsoft Internet Explorer"){
          alert("Seu browser é o Internet Explorer ou Compatível!");
          url=("explorer.htm");
 } else {
         alert("Seu browser é o Firefox, Mozilla, Netscape ou Compatível!");
         url=("netscape.htm")
         window.location=url;
}
//-->
</script>
```

13 - Combo (Select) com Estados Braisleiros

```
<html>
<body>
<!--ESTADOS BRASILEIROS EM SELECT-->
<style TYPE="text/css">
.estados { color= #000000; font-family: verdana, arial, courier; font-size: 8pt; }
</STYLE>

<select name="UF" class="estados" width=8>
<option value="RJ" select>Rio de Janeiro </option>
<option value="AC">Acre </option>
<option value="AL">Alagoas </option>
<option value="AM">Amazonas </option>
<option value="AP">Amap&aacute; </option>
<option value="BA">Bahia </option>
<option value="CE">Cear&aacute; </option>
<option value="DF">Distrito Federal </option>
<option value="ES">Esp&iacute;rito Santo
</option>
<option value="GO">Goias </option>
<option value="MA">Maranh&atilde;o </option>
<option value="MT">Mato Grosso </option>
<option value="MS">Mato Grosso do Sul
</option>
<option value="MG">Minas Gerais </option>
<option value="PA">Par&aacute; </option>
<option value="PB">Paraiba </option>
<option value="PR">Paran&aacute; </option>
<option value="PE">Pernambuco </option>
<option value="PI">Piaui </option>
<option value="RN">Rio Grande do Norte</option>
<option value="RS">Rio Grande do Sul </option>
<option value="RO">Rond&ocirc;nia </option>
<option value="RR">Roraima </option>
<option value="SC">Santa Catarina </option>
<option value="SE">Sergipe </option>
<option value="SP">S&atilde;o Paulo </option>
<option value="TO">Tocantins </option>
<option value="XX">Outros</option>
</select>

</body>
</html>
```

14 - Formatar Moeda (Real)

```
<html>
<head>
<SCRIPT language="JavaScript">
<!-- Begin

function numero2moeda(num) {
  num = num.toString().replace(/\$|\,/g,'');
  if(isNaN(num))
        num = "0";
        sign = (num == (num = Math.abs(num)));
        num = Math.floor(num*100+0.50000000001);
        cents = num%100;
        num = Math.floor(num/100).toString();
  if(cents<10)
        cents = "0" + cents;
        for (var i = 0; i < Math.floor((num.length-(1+i))/3); i++)
        num = num.substring(0,num.length-(4*i+3))+'.'+
        num.substring(num.length-(4*i+3));
        return (((sign)?'':'-') + 'R$' + num + ',' + cents);
}
// End -->
</script>

<title>Moeda</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<BODY>
<center>
<form name=currencyform>
Enter um número e clique no botão: <input type=text name=input size=15
value="1000434.23">
<input type=button value="Convert"
onclick="this.form.input.value=numero2moeda(this.form.input.value);">
</form>
</center>

<p><center>
<font face="arial, helvetica" size="-2">Free JavaScripts provided<br>
by <a href="http://javascriptsource.com">The JavaScript Source</a></font>
</center><p>
</body>
</html>
```

15 - Limitar Caracteres Digitrados em Textarea

```
<HTML><HEAD> <TITLE></TITLE> </HEAD> <BODY>
<table><tr><td>
<textarea onkeyup="max(this)" onkeypress="max(this)" rows="2" cols="35"
name="Area"></textarea><br>
<font id=Digitado color=red>0</font> Caracteres digitados   /   restam <font
id=Restante color=red>100</font>
</td></tr></table>

<SCRIPT LANGUAGE=javascript>
function max(txarea) {
total = 100;
tam = txarea.value.length;
str="";
str=str+tam;
Digitado.innerHTML = str;
Restante.innerHTML = total - str;
  if (tam > total){
    aux = txarea.value;
    txarea.value = aux.substring(0,total);
    Digitado.innerHTML = total
    Restante.innerHTML = 0
  }
}
</SCRIPT>
</BODY></HTML>
```

16 - Formatação do CNPJ

```
<HTML><HEAD><TITLE>CNPJ Formatação</title></head><body>
<input type="text" name="cnpj" size="18" maxlength="18" onBlur="FormataCNPJ(this)"
onkeypress="return validaTecla(this, event)">
<script language="JavaScript">

<!--
function isNum( caractere ) {
var strValidos = "0123456789"
if ( strValidos.indexOf( caractere ) == -1 )
return false;
return true;
}

function validaTecla(campo, event) {
var BACKSPACE= 8;
var key;
var tecla;

CheckTAB=true;
if(navigator.appName.indexOf("Netscape")!= -1)
```

```
tecla= event.which;
else
tecla= event.keyCode;

key = String.fromCharCode( tecla);
//alert( 'key: ' + tecla + ' -> campo: ' + campo.value);

if ( tecla == 13 )
return false;
if ( tecla == BACKSPACE )
return true;
return ( isNum(key));
}

function FormataCNPJ( el )
{
vr = el.value;
tam = vr.length;
if ( vr.indexOf(".") == -1 ) {
if ( tam <= 2 )
el.value = vr;
if ( (tam > 2) && (tam <= 6) )
el.value = vr.substr( 0, 2 ) + '.' + vr.substr( 2, tam );
if ( (tam >= 7) && (tam <= 10) )
el.value = vr.substr( 0, 2 ) + '.' + vr.substr( 2, 3 ) + '.' + vr.substr( 5, 3 ) + '/';
if ( (tam >= 11) && (tam <= 18) )
el.value = vr.substr( 0, 2 ) + '.' + vr.substr( 2, 3 ) + '.' + vr.substr( 5, 3 ) + '/' + vr.substr( 8, 4 )
+ '-' + vr.substr( 12, 2 ) ;
}
return true;
}
//-->
</script>
</body></html>
```

17 - Combo (Select) com Países do Mundo

Brasil como Default (selected)

```
<select name=".co"size=1 >
<option selected value="br" >Brasil
<option value="af" >Afeganistão
<option value="za" >África do Sul
<option value="al" >Albânia
<option value="de" >Alemanha
<option value="dz" >Algéria
<option value="ad" >Andorra
<option value="ao" >Angola
<option value="ai" >Anguilla
<option value="aq" >Antártida
```

<option value="ag" >Antígua e Barbuda
<option value="an" >Antilhas Holandesas
<option value="sa" >Arábia Saudita
<option value="ar" >Argentina
<option value="am" >Armênia
<option value="aw" >Aruba
<option value="au" >Austrália
<option value="at" >Áustria
<option value="az" >Azerbaijão
<option value="bs" >Bahamas
<option value="bh" >Bahrain
<option value="bd" >Bangladesh
<option value="bb" >Barbados
<option value="by" >Belarus
<option value="be" >Bélgica
<option value="bz" >Belize
<option value="bj" >Benin
<option value="bm" >Bermuda
<option value="bo" >Bolívia
<option value="bt" >Butão
<option value="ba" >Bósnia-Herzegovina
<option value="bw" >Botsuana
<option value="bn" >Brunei
<option value="bg" >Bulgária
<option value="bf" >Burkina Faso
<option value="bi" >Burundi
<option value="cv" >Cabo Verde
<option value="kh" >Camboja
<option value="cm" >Camarões
<option value="ca" >Canadá
<option value="kz" >Casaquistão
<option value="td" >Chade
<option value="cl" >Chile
<option value="cn" >China
<option value="cy" >Chipre
<option value="co" >Colômbia
<option value="km" >Comoros
<option value="cg" >Congo
<option value="kp" >Coréia do Norte
<option value="kr" >Coréia do Sul
<option value="ci" >Costa do Marfim
<option value="cr" >Costa Rica
<option value="hr" >Croácia
<option value="cu" >Cuba
<option value="dk" >Dinamarca
<option value="dj" >Djibouti
<option value="dm" >Dominica
<option value="sv" >El Salvador
<option value="ec" >Equador
<option value="eg" >Egito

<option value="ae" >Emirados Árabes Unidos
<option value="er" >Eritréia
<option value="es" >Espanha
<option value="sk" >Eslováquia
<option value="si" >Eslovênia
<option value="us" >Estados Unidos da América
<option value="ee" >Estônia
<option value="et" >Etiópia
<option value="fj" >Fiji
<option value="ph" >Filipinas
<option value="fi" >Finlândia
<option value="fr" >França
<option value="ga" >Gabão
<option value="gm" >Gâmbia
<option value="gh" >Gana
<option value="ge" >Geórgia
<option value="gi" >Gibraltar
<option value="gd" >Granada
<option value="gr" >Grécia
<option value="gl" >Groelândia
<option value="gp" >Guadalupe
<option value="gu" >Guam
<option value="gt" >Guatemala
<option value="gf" >Guiana Francesa
<option value="gn" >Guiné
<option value="gw" >Guiné-Bissau
<option value="gq" >Guiné Equatorial
<option value="gy" >Guiana
<option value="ht" >Haiti
<option value="hn" >Honduras
<option value="hk" >Hong Kong
<option value="hu" >Hungria
<option value="ye" >Iêmen
<option value="ky" >Ilhas Cayman
<option value="ck" >Ilhas Cook
<option value="fk" >Ilhas Falkland
<option value="fo" >Ilhas Faroe
<option value="hm" >Ilhas Heard e McDonald
<option value="mp" >Ilhas Mariana do Norte
<option value="mh" >Ilhas Marshall
<option value="gs" >Ilhas S. Georgia e S. Sandwich
<option value="sj" >Ilhas Svalbard e Jan Mayen
<option value="tc" >Ilhas Turks e Caicos
<option value="vi" >Ilhas Virgens
<option value="vg" >Ilhas Virgens Britânicas
<option value="in" >Índia
<option value="id" >Indonésia
<option value="is" >Islândia
<option value="ir" >Irã
<option value="iq" >Iraque

<option value="ie" >Irlanda
<option value="il" >Israel
<option value="it" >Itália
<option value="yu" >Iugoslávia (ex-)
<option value="jm" >Jamaica
<option value="jp" >Japão
<option value="jo" >Jordânia
<option value="ki" >Kiribati
<option value="kw" >Kuwait
<option value="la" >Laos
<option value="lv" >Látvia
<option value="ls" >Lesoto
<option value="lb" >Líbano
<option value="lr" >Libéria
<option value="ly" >Líbia
<option value="li" >Liechtenstein
<option value="lt" >Lituânia
<option value="lu" >Luxemburgo
<option value="mo" >Macau
<option value="mk" >Macedônia, antiga Iugoslávia
<option value="mg" >Madagascar
<option value="my" >Malásia
<option value="mw" >Malaui
<option value="mv" >Maldivas
<option value="ml" >Mali
<option value="mt" >Malta
<option value="ma" >Marrocos
<option value="mq" >Martinica
<option value="mu" >Maurício
<option value="mr" >Mauritânia
<option value="yt" >Mayotte
<option value="mx" >México
<option value="fm" >Micronésia
<option value="mz" >Moçambique
<option value="md" >Moldova
<option value="mc" >Mônaco
<option value="mn" >Mongólia
<option value="ms" >Montserrat
<option value="mm" >Myanmar
<option value="na" >Namíbia
<option value="nr" >Nauru
<option value="np" >Nepal
<option value="nl" >Holanda
<option value="ni" >Nicarágua
<option value="ne" >Niger
<option value="ng" >Nigéria
<option value="nu" >Niue
<option value="no" >Noruega
<option value="nc" >Nova Caledônia
<option value="nz" >Nova Zelândia

<option value="om" >Oman
<option value="pw" >Palau
<option value="pa" >Panamá
<option value="pg" >Papua Nova Guiné
<option value="pk" >Paquistão
<option value="py" >Paraguai
<option value="pe" >Peru
<option value="pf" >Polinésia Francesa
<option value="pl" >Polônia
<option value="pr" >Porto Rico
<option value="pt" >Portugal
<option value="qa" >Qatar
<option value="ke" >Quênia
<option value="uk" >Reino Unido
<option value="cf" >República Centro-Africana
<option value="do" >República Dominicana
<option value="cz" >República Tcheca
<option value="re" >Reunião
<option value="ro" >Romênia
<option value="rw" >Ruanda
<option value="ru" >Rússia
<option value="eh" >Saara Ocidental
<option value="vc" >Saint Vincent e Grenadines
<option value="ws" >Samoa
<option value="as" >Samoa Americana
<option value="sm" >San Marino
<option value="sh" >Santa Helena
<option value="lc" >Santa Lúcia
<option value="st" >São Tomé e Príncipe
<option value="sn" >Senegal
<option value="sl" >Serra Leão
<option value="sc" >Seychelles
<option value="sg" >Singapura
<option value="sy" >Síria
<option value="so" >Somália
<option value="lk" >Sri Lanka
<option value="sd" >Sudão
<option value="sr" >Suriname
<option value="sz" >Suazilândia
<option value="se" >Suécia
<option value="ch" >Suíça
<option value="th" >Tailândia
<option value="tw" >Taiwan
<option value="tz" >Tanzânia
<option value="tp" >Timor Leste
<option value="tg" >Togo
<option value="tk" >Tokelau
<option value="to" >Tonga
<option value="tt" >Trinidad e Tobago
<option value="tn" >Tunísia

```
<option value="tm" >Turcomenistão
<option value="tr" >Turquia
<option value="tv" >Tuvalu
<option value="ua" >Ucrânia
<option value="ug" >Uganda
<option value="uy" >Uruguai
<option value="uz" >Usbequistão
<option value="vu" >Vanuatu
<option value="va" >Vaticano
<option value="ve" >Venezuela
<option value="vn" >Vietnã
<option value="zr" >Zaire
<option value="zm" >Zâmbia
<option value="zw" >Zimbábue
<option value="outro">
</select>
```

18 - Imprimir Automaticamente ao Abrir Página

```
<script language="JavaScript">
        self.print();
</script>
```

19 - Imprimir ao Clicar

```
<html><head><title></title></head><body>
   <p>
          Clique no botão para imprimir esta página.
   </p>
   <form action="javascript:;">
     <input onclick="window.print()" type="button" value="Imprimir esta página"/>
   </form>
</body></html>
```

20 - Abrir Janela com Determinado Tamanho

```
<html>
<head>
<script type="text/javascript">
//Abrir janela com determinado tamanho
function resizeWindow(x,y)
{
        top.window.resizeTo(x,y)
        /*
                Usamos o objecto top para aceder à janela que ocupa
```

```
                o topo da hierarquia. Se não o fizéssemos estaríamos
                a actuar sobre a subjanela que contém o exercício,
                que por estar dentro de uma moldura não no daria o
                resultado que queremos.
        */
}
</script>
   <title></title>
</head>
<body>
        <a href="javascript: resizeWindow(640,480)">Dar à janela o tamanho 640 x
480</a><br/>
        <a href="javascript: resizeWindow(780,580)">Dar à janela o tamanho 780 x
580</a><br/>
        <a href="javascript: resizeWindow(800,600)">Dar à janela o tamanho 800 x
600</a><br/>
        <a href="javascript: resizeWindow(1024,768)">Dar à janela o tamanho 1024 x
768</a><br/>
        O opera não consegue executar este exemplo.
  </body>
</html>
```

21 - Agendar Execução de Tarefa

```
<html>
<head>
<script type="text/javascript">
//Agendar execução de uma função
function avisar(){
        alert('AVISO MUITO IMPORTANTE!!!\n\nJá passou um segundo desde que fez
clique.')
}

function agendar(){
        setTimeout("avisar()", 1000)
}
</script>
   <title></title>
</head>
<body>
  <br/>
  <a href="javascript:agendar()">Agendar a apresentação de um aviso</a>
</body></html>
```

JavaScript Avançado

http://www.w3schools.com/js/default.asp
http://www.w3schools.com/js/tryit.asp?filename=tryjs_intro

O JavaScript captura e tem resposta para vários tipos de eventos das págnas:
– click do mouse
– página ou imagem sendo carregada ou descarregada
– mouse se movendo sobre uma região de página
– selecionando ou alterando campo de form
– submetendo um form
– pressionando uma tecla

Os eventos oferecem uma grande oportunidade de interação com os osuários.
Podemos jogar o foco no primeiro campo de um formulário para facilitar a vida do usuário de forma que ele já abre o site começa a digitar no campo correto. Exemplo:

<body onLoad="document.frmInserir.login.focus();">

Com isso o campo login do form frmInserir receberá o foco logo que a página que contém o form seja carregara.

Podemos interagir com o usuário quando ele clica em um link, em um botão, quando ele escolhe um item de uma lista, quando ele clica no botão submit e de muitas outras formas.

http://www.mspc.eng.br/info/jscriptContrEv.shtml (Página de Marco Soares)

onAbort | onBlur | onChange | onClick | onDblClick | onDragDrop | onError | onFocus | onKeyDown | onKeyPress | onKeyUp | onLoad | onMouseDown | onMouseMove | onMouseOut | onMouseOver | onMouseUp | onMove | onReset | onResize | onSelect | onSubmit | onUnload |

onAbort

Executa um código JavaScript quando o usuário interrompe o carregamento de uma imagem, por exemplo, com um clique no botão Parar. Deve ser usado com o objeto imagem (IMG) de HTML.

Exemplo:

```
<IMG NAME="foto" SRC="minha_foto.gif" onAbort="alert('Você não viu minha foto')">
```

onBlur

Executa um código JavaScript quando um elemento, janela ou quadro, perde o foco. Usado para Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, window, na forma `onBlur="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<INPUT TYPE="text" VALUE="" NAME="nome"
onBlur="verificarNome(this.value)">
```

Onde "verificarNome(n)" é uma função (a ser desenvolvida) para verificar se o usuário inseriu um nome válido.

onChange

Executa um código JavaScript quando um campo perde o foco e tem seu valor modificado. Usado para FileUpload, Select, Text, Textarea, na forma `onChange="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<INPUT TYPE="text" VALUE="" NAME="nome"
onChange="verificarNome(this.value)">
```

Onde "verificarNome(n)" é uma função (a ser desenvolvida) para verificar se o usuário inseriu um nome válido.

onClick

Executa um código JavaScript quando um objeto é clicado. Usado para Button, document, Checkbox, Link, Radio, Reset, Submit, na forma `onClick="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo 1:

```
<A HREF="http://www.abc.com.br/" onClick="return confirm('Abre
ABC?')">Empresa ABC</A>
```

Se o usuário clica no link, abre a caixa de confirmação. Se, nessa caixa, o botão Cancelar é clicado, o link não é aberto.

Exemplo 2:

```
<INPUT TYPE="button" VALUE="Resultado" onClick="calcular(this.form)">
```

Ao clicar no botão, é chamada a função "calcular(a)" (a ser desenvolvida).

onDblClick

Executa um código JavaScript quando um duplo clique é dado no objeto. Usado para document, Link, na forma `onDblClick="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<form>
<INPUT Type="button" Value="Teste" onDblClick="alert('Foi dado um duplo
clique')">
</form>
```

onDragDrop

Executa um código JavaScript quando um objeto é arrastado para a janela do navegador. Usado para window, na forma `onDragDrop="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

onError

Executa um código JavaScript quando um erro ocorre no carregamento de uma janela ou imagem. Usado para Image, window, na forma `onError="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

O evento só ocorre em caso de erro de sintaxe no código ou erro em tempo de execução do mesmo. Não reporta erros do navegador.

Se é forçado para nulo, suprime mensagens de erro.

Exemplo 1:

```
<IMG NAME="foto" SRC="foto.gif" ALIGN="left" BORDER="2" onError="null">
```

Exemplo 2:

```
<SCRIPT language="javascript"_>
window.onerror=null
</SCRIPT>
<IMG NAME="foto" SRC="foto.gif" ALIGN="left" BORDER="2">
```

O primeiro exemplo suprime mensagens de erro apenas no carregamento da imagem foto.gif. O

segundo exemplo suprime para toda a janela.

onFocus

Executa um código JavaScript quando o objeto recebe o foco. Usado para Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, window, na forma `onFocus="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<INPUT TYPE="text" VALUE="" NAME="nome"
onFocus="verificarNome(this.value)">
```

Onde "verificarNome(n)" é uma função (a ser desenvolvida) para verificar se um nome válido existe no campo toda vez que esse campo recebe o foco.

onKeyDown

Executa um código JavaScript quando uma tecla é pressionada. Usado para document, Image, Link, Text, Textarea, na forma `onKeyDown="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<script language="javascript">
function semArroba(e){
var carac = String.fromCharCode(e.which);
if (carac == '@')
return false;
}
document.form1.nome.onkeydown = semArroba;
</script>

<form NAME="form1">
<INPUT TYPE="text" VALUE="" NAME="nome">
</form>
```

O código dado evita que o usuário insira o caractere "@" no campo "nome".

onKeyPress

Executa um código JavaScript quando o usuário pressiona ou mantém pressionada uma tecla. Usado para document, Image, Link, Text, Textarea, na forma `onKeyPress="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<script language="javascript">

function rolar(e){
var carac = String.fromCharCode(e.which);
if (carac == 'c')
self.scrollBy(0,10);
else if(carac == 'b')
self.scrollBy(0,-10);
else
return false;
}

document.captureEvents(Event.KEYPRESS);

document.onkeypress = rolar;

</script>
```

O código dado rola a tela 10 pixels acima se o caractere "c" for pressionado e abaixo se "b" for pressionado (o evento KeyPress é repetido continuamente enquanto o usuário mantém a tecla pressionada).

onKeyUp

Executa um código JavaScript quando o usuário libera uma tecla que foi pressionada. Usado para document, Image, Link, Text, Textarea, na forma `onKeyUp="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<script language="javascript">

function teclaLiberada(e){
var carac = String.fromCharCode(e.which);
alert("Você liberou a tecla " + carac);
}
document.onkeyup=teclaLiberada;
document.captureEvents(Event.KEYUP);

</script>
```

O código dado exibe uma mensagem com o caractere da tecla liberada.

onLoad

Executa um código JavaScript quando o navegador termina o carregamento de uma janela, de todos os quadros dentro de um FRAMESET, de uma imagem. Usado para Image, Layer, window, na forma `onLoad="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Se é usado para uma imagem gif animada, é executado para cada parte da animação.

Exemplo 1:

```
<script language="javascript">

function nomeImagem(img){
alert('Carregada imagem ' + img.name);
}

</sript>

<IMG NAME="foto" SRC="foto.gif" ALIGN="left" BORDER="2"
onLoad="nomeImagem(this)">
```

O código exibe o nome da imagem exibida.

Exemplo 2:

```
<BODY onLoad="window.alert("Este é o nosso site!")>
```

O código exibe uma mensagem sempre que um usuário abre a página.

onMouseDown

Executa um código JavaScript quando o usuário pressiona um botão do mouse. Usado para Button, document, Link, na forma `onMouseDown="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

onMouseMove

Executa um código JavaScript quando o usuário move o cursor com o mouse. Usado na forma `onMouseMove="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Notar que não é associado a nenhum evento em particular por ser bastante freqüente.

onMouseOut

Executa um código JavaScript quando o usuário move o cursor de dentro para fora de uma determinada área (mapa de imagem ou link). Usado para Layer, Link, na forma `onMouseOut="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

onMouseOver

Executa um código JavaScript quando o usuário move o cursor de fora para dentro de uma determinada área (mapa de imagem ou link). Usado para Layer, Link, na forma `onMouseOver="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<A HREF="http://www.abc.com.br/" onMouseOver="window.status='Loja ABC - Tudo para seu micro'; return true">Empresa ABC</A>
```

O código dado exibe a mensagem na barra de status do navegador sempre que o usuário passa o cursor sobre o link.

onMouseUp

Executa um código JavaScript quando o usuário libera um botão do mouse. Usado para Button, document, Link, na forma `onMouseUp="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

onMove

Executa um código JavaScript quando o usuário move uma janela. Usado para window, na forma `onMove="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

onReset

Executa um código JavaScript quando o usuário clica o botão Reset de um formulário. Usado para Form, na forma `onReset="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<FORM NAME="form1" onReset="alert('Considerado meio padrão de envio')">
Meio de envio:
<INPUT TYPE="text" NAME="forma" VALUE="Via aérea" SIZE="2"_>
<INPUT TYPE="reset" NAME="padrao" VALUE="Limpar">
</FORM>
```

O código dado avisa que o meio padrão será considerado se o usuário clicar no botão "Limpar" do formulário.

onResize

Executa um código JavaScript quando o usuário redimensiona uma janela ou um frame. Usado para window, na forma `onResize="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<script language="javascript">

function abrirJanela(){
var janela;
janela=window.open("janela.htm",null,"width=200, height=200,
resizable=yes,  menubar=no, location=no");
janela.captureEvents(Event.RESIZE);
janela.onresize=informar;
}

function informar(){
alert("Janela redimensionada para largura: " + this.outerWidth + "e
altura: " +this.outerHeight);
this.focus();
}

</script>
```

O código dado informa a nova largura e a nova altura sempre que a janela é redimensionada.

onSelect

Executa um código JavaScript quando o usuário seleciona um texto em uma caixa. Usado para Text, Textarea, na forma `onSelect="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

onSubmit

Executa um código JavaScript quando o usuário clica o botão Submeter de um formulário. Usado para Form, na forma `onSubmit="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<FORM NAME="form1" onSubmit="return verificarDados(this)">
.
</FORM>
```

No código dado, a função a ser desenvolvida, "verificarDados(a)", deve retornar True se os dados são válidos e False caso contrário.

onUnload

Executa um código JavaScript quando o usuário sai da janela. Usado para window, na forma `onUnload="algumaCoisa"`, onde "algumaCoisa" é uma função ou código de JavaScript.

Exemplo:

```
<BODY onUnload="terminar()">
```

No código dado, a função a ser desenvolvida, "terminar()", deve fazer alguma ação que for necessária quando o usuário sair da janela.

Funções Internas

Funções de strings (http://www.mspc.eng.br/info/jscriptString.shtml)

O objeto String | A propriedade length (comprimento) | anchor | big | blink | bold | charAt | charCodeAt | concat | fixed | fontcolor |fontsize | fromCharCode | indexOf | italics | lastIndexOf | link | match | replace | search | slice | small | split | strike | sub | substr | substring | sup | toLowerCase | toString | toUpperCase

Introdução

Obs: aqui é mantido o termo inglês *string* que, em computação, significa uma seqüência de caracteres. Em JavaScript e em várias outras linguagens, strings são literalmente definidas entre aspas. Exemplos: "Bom dia", "A0028", "100" são strings. Mas 100 (sem aspas) é o número 100.

Resumo das funções de strings

Funções genéricas: charAt (caractere de uma dada posição) | charCodeAt (valor ASCII do caractere de uma dada posição) | concat (combinar duas ou mais strings) | fromCharCode (forma string a partir dos valores ASCII) | indexOf (posição da primeira ocorrência de uma string em outra) | lastIndexOf (posição da última ocorrência de uma string em outra) | length (comprimento da string) | match (expressão regular em uma string) | replace (substituir ocorrências em uma string) | search (procurar ocorrência em uma string) | slice (extrair parte de uma string) | split (separar uma string em uma array de strings) | substr (extrair parte de uma string) | substring (extrair parte de uma string) | toLowerCase (converter para minúsculas) | toString (string do objeto) | toUpperCase (converter para maiúsculas)

Funções que substituem marcas de HTML: anchor (âncora de hipertexto) | big (fonte grande) | blink (piscar) | bold (negrito) | fixed (fonte de tamanho fixo) | fontcolor (cor da fonte) | fontsize (tamanho da fonte) | italics (itálico) | link (link de hipertexto) | small (fonte pequena) | strike (fonte tachada) | sub (subescrito) | sup (sobrescrito)

O objeto String

É criado através de:

```
new String(str).
```

Onde str é uma string qualquer. Notar a diferença entre string literal e string objeto. Exemplos:

`str_1 = "abc"` (string literal) e
`str_2 = new String("abc")` (string objeto).

Na prática, quaisquer das funções aqui dadas podem ser usadas com strings literais. A linguagem cria um objeto temporário e o descarta após a execução.

A propriedade length

Indica o comprimento (número de caracteres) da string.

Exemplo (a variável *len* assume valor 7);

```
str = "Bom dia";
len = str.length;
```

anchor(nome)

Cria uma âncora no documento HTML com o atributo *NAME* igual a *nome*.

Exemplo:

```
str = "Início da página";
document.write(str.anchor("inicio_pagina"));
```

Produz o mesmo resultado da seguinte linha HTML:

<A NAME="inicio_pagina">Início da página</A>

big()

Faz a string aparecer como se estivesse entre as marcas <BIG> e </BIG> de HTML.

Exemplo:

```
str = "Bom dia";
document.write(str.big());
```

Produz o mesmo resultado da seguinte linha HTML:

<BIG>Bom dia</BIG>

blink()

Faz a string piscar como se estivesse entre as marcas <BLINK> e </BLINK> de HTML (Netscape somente).

Exemplo:

```
str = "Bom dia";
document.write(str.blink());
```

Produz o mesmo resultado da seguinte linha HTML:

<BLINK>Bom dia</BLINK>

bold()

Faz a string aparecer em negrito como se estivesse entre as marcas <B> e </B> de HTML.

Exemplo:

```
str = "Bom dia";
document.write(str.bold());
```

Produz o mesmo resultado da seguinte linha HTML:

<B>Bom dia</B>

charAt(ndx)

Retorna o caractere da posição especificada por *ndx*, um inteiro entre 0 e comprimento da string menos 1.

Exemplo (a variável str_2 será "o"):

```
str_1 = "Bom dia";
str_2 = str_1.charAt(1);
```

charCodeAt([ndx])

Retorna o valor ASCII do caractere na posição especificada por *ndx*, um inteiro entre 0 e comprimento da string menos 1. Se não indicado, o valor 0 é assumido.

Exemplo (a variável val será 97, o valor ASCII de "a"):

```
str = "Bom dia";
val = str.charCodeAt(6);
```

concat(str2, str3 [, ..., strN])

Combina duas ou mais strings, retornando uma nova. str2, ..., strN são as strings a

combinar.

Exemplo (a variável nova_str contém "Bom dia"):

```
str = "Bom ";
nova_str = str.concat("dia");
```

fixed()

Faz a string ser exibida com uma fonte de largura fixa, como se estivesse entre as marcas <TT> e </TT> de HTML.

Exemplo:

```
str = "Bom dia";
document.write(str.fixed());
```

Produz o mesmo resultado da seguinte linha HTML:

<TT>Bom dia</TT>

fontcolor(cor)

Exibe a string na cor especificada, como se estivesse entre as marcas <FONT COLOR=cor> e </FONT> de HTML.

O parâmetro *cor* é uma string literal da cor reconhecida pelo navegador ou na forma RGB hexadecimal. Por exemplo, "FF0000" para vermelho.

Exemplo:

```
str = "Bom dia";
document.write(str.fontcolor("blue"));
```

Produz o mesmo resultado da seguinte linha HTML:

<FONT COLOR="blue">Bom dia</FONT>

fontsize(sz)

Exibe a string no tamanho de fonte especificado, como se estivesse entre as marcas <FONT SIZE=sz> e </FONT> de HTML. O parâmetro *sz* é o tamanho da fonte.

Exemplo:

```
str = "Bom dia";
document.write(str.fontsize("3"));
```

Produz o mesmo resultado da seguinte linha HTML:

<FONT SIZE="3">Bom dia</FONT>

fromCharCode(car1, ..., carN)

Retorna uma string (não objeto String) com os caracteres dados pelos valores ASCII *car1, ..., carN*. Deve ser sempre usado na forma String.fromCharCode(...) em vez de um objeto string criado.

Exemplo (retorna "abc"):

```
String.fromCharCode(97,98,99);
```

Observação: os eventos *KeyDown*, *KeyPress* e *KeyUp* contém os códigos ASCII da tecla usada. Para saber o símbolo da tecla, pode-se usar a propriedade *which*:

```
String.fromCharCode(KeyDown.which);
```

indexOf(iStr [, iniNdx])

Retorna o índice da primeira ocorrência de *iStr*, começando de *iniNdx* ou de 0 se ele não é dado. Se *iStr* não é encontrado, retorna -1. É sensível a letras maiúsculas e minúsculas.

Exemplos:

`"Bom dia".indexOf("Bom")` retorna 0.
`"Bom dia".indexOf("Bon")` retorna -1.
`"Bom dia".indexOf("d")` retorna 4.
`"Bom dia".indexOf("bom")` retorna -1.

italics()

Faz a string ser exibida em itálico, como se estivesse entre as marcas <I> e </I>
de HTML.

Exemplo:

```
str = "Bom dia";
document.write(str.italics());
```

Produz o mesmo resultado da seguinte linha HTML:

<I>Bom dia</I>

lastIndexOf(iStr [, iniNdx])

Retorna o índice da última ocorrência de *iStr* ou -1 se não encontrado. A procura é feita na ordem inversa (direita para esquerda), começando do índice *iniNdx* ou do comprimento da string se ele não é indicado.

Caracteres são indexados da esquerda para a direita, a partir de 0. A procura é sensível a letras maiúsculas e minúsculas.

Exemplos:

`"Boa noite".lastIndexOf("Boa")` retorna 0.
`"Boa noite".lastIndexOf("o")` retorna 5.
`"Boa noite".lastIndexOf("b")` retorna -1.

link(hRef)

Cria um link de hipertexto HTML para a URL (absoluta ou relativa) dada por *hRef*.

Exemplo:

```
str = "Página inicial MSPC";
url = "http://www.mspc.eng.br";
document.write(str.link(url);
```

Produz o mesmo resultado da seguinte linha HTML:

<A HREF="http://www.mspc.eng.br">Página inicial MSPC</A>

match(rExp)

Empregado para determinar o resultado de expressões regulares para uma string. *rExp* é o nome da expressão regular, podendo ser uma variável ou literal.

Para encontrar ocorrências simples, é melhor usar a função search.

Exemplo (retorna "O,o"):

```
str = "Sistema Operacional";
document.write(str.match(/o/gi);
```

replace(rExp, nStr)

Procura ocorrências da expressão regular em uma string e substitui por nStr. *rExp* é o nome da expressão regular, podendo ser uma variável ou literal. *nStr* pode ser substituída por uma função.

O conteúdo do objeto string original não é alterado. Uma nova string é retornada.

Exemplo (retorna "Boa noite"):

```
str = "Boa tarde";
re = /'tarde/gi;
str_1 = str.replace(re,"noite");
document.write(str_1);
```

search(rExp)

Procura um dado formato especificado pela expressão regular *rExp*, que pode ser uma variável ou literal. Se encontrado, retorna o índice da expressão na string. Caso contrário, retorna -1.

Exemplo:

```
strMail = new String("fulano@nome.com.br");
re = /^[^@]+@[^@]+.[a-z]{2,}$/i;
if(strMail.search(re) == -1)
     document.write("O endereço de email não é válido");
```

slice(iniNdx [, fimNdx])

Extrai parte de uma string, retornando uma nova. *iniNdx* é o índice base zero inicial e *fimNdx* é o índice base zero final. Se não especificado, a operação se dá até o final da string.

Valor negativo para *fimNdx* indica deslocamento a partir do final. Por exemplo, slice(1,-1) extrai do segundo até o penúltimo caracter.

Exemplo (escreve "tarde"):

```
str = "Boa tarde";
str_1 = str.slice(4);
document.write(str_1);
```

small()

Faz a string ser exibida em fonte pequena, como se estivesse entre as marcas <SMALL> e </SMALL> de HTML.

Exemplo:

```
str = "Bom dia";
document.write(str.small());
```

Produz o mesmo resultado da seguinte linha HTML:

<SMALL>Bom dia</SMALL>

split([sep][, lim])

Separa uma string em um conjunto (array) de strings, usando como separador o caractere indicado por *sep*. O parâmetro opcional *lim* é um inteiro que limita o número de separações.

Exemplo (no resultado, str_2[0] contém "AA", str_2[1] contém "BB", str_2[2] contém "CC" e str_2[3] contém "DD"):

```
str_1 = new String("AA,BB,CC,DD");
str_2 = str.split(",");
```

strike()

Faz a string ser exibida com uma linha atravessada, como se estivesse entre as marcas <STRIKE> e </STRIKE> de HTML.

Exemplo:

```
str = "Bom dia";
document.write(str.strike());
```

Produz o mesmo resultado da seguinte linha HTML:

<STRIKE>Bom dia</STRIKE>

sub()

Faz a string ser exibida na forma subscrita, como se estivesse entre as marcas _e de HTML.

Exemplo:

```
str = "Bom dia";
document.write(str.sub());
```

Produz o mesmo resultado da seguinte linha HTML:

_{Bom dia}

substr(ini [, compr])

Extrai parte de uma string, começando no índice base zero dado por *ini*. Se este é negativo, significa índice a partir do final da string. O parâmetro opcional *compr* é número de caracteres a extrair a partir do índice dado. Se omitido, a operação se dá até o final da string.

Exemplo (resulta "dia"):

```
str = "Bom dia";
document.write(str.substr(4,3));
```

substring(ndx1, ndx2)

Extrai parte de uma string, começando do índice base zero *ndx1* e terminando, mas não incluindo, no índice base zero *ndx2*.

Exemplo (resulta "Bom"):

```
str = "Bom dia";
document.write(str.substring(0,3));
```

sup()

Faz a string ser exibida na forma sobrescrita, como se estivesse entre as marcas ^e de HTML.

Exemplo:

```
str = "Bom dia";
document.write(str.sup());
```

Produz o mesmo resultado da seguinte linha HTML:

^{Bom dia}

toLowerCase()

Retorna a string original com os caracteres minúsculos. Não altera a original.

Exemplo (resulta "bom dia"):

```
str = "Bom dia";
document.write(str.toLowerCase());
```

toString()

Retorna a string do objeto especificado.

Exemplo (resulta "Bom dia"):

```
str = new String("Bom dia");
document.write(str.toString());
```

toUpperCase()

Retorna a string original com os caracteres maiúsculos. Não altera a original.

Exemplo (resulta "BOM DIA"):

```
str = "Bom dia";
document.write(str.toUpperCase());
```

Funções matemáticas
http://www.mspc.eng.br/info/jscriptMath.shtml

Funções com Data e Hora

http://www.mspc.eng.br/info/jscriptDataHora.shtml

Observações

1-) Parâmetros indicados entre parênteses são opcionais.

2-) UTC (Coordinated Universal Time) é o padrão atual de tempo de acordo com os fusos horários. É continuação moderna do GMT (Greenwich Mean Time). Algumas vezes são considerados sinônimos, mas rigorosamente não são.

3-) Algumas funções (obsoletas ou de pouco uso) de data e hora não são apresentadas.

O objeto Date()

Permite trabalhar com datas e horas. Pode ser criado por uma das formas abaixo:

```
new Date()
new Date(ms)
new Date(string_de_data)
new Date(nAno, nMês, nDia [, nHora, nMin, nSeg, nMs] )
```

A data é dada em milissegundos, a partir de 00:00:00 h GMT de 01 de janeiro de 1970.

Se nenhum parâmetro é fornecido (primeira forma), são assumidas a data e hora locais, isto é, do computador que executa o script.

Parâmetros:

ms: milissegundos desde a data inicial dada acima.

string_de_data: uma seqüência de caracteres em um formato suportado. Exemplo: "Mon, 27 Dec 2004 12:00:00 GMT".

nAno, nMês, nDia, etc: valores inteiros representando partes da data. Exemplo: Janeiro 0, Fevereiro 1, etc.

Exemplo de aplicação 1: a variável *intervalo* armazena o tempo de execução em milissegundos das *outras instruções*.

```
tempo_1 = new Date();
// outras instruções
tempo_2 = new Date();
intervalo = tempo_2 - tempo_1
```

Exemplo de aplicação 2: este é um exemplo dinâmico, que atualiza o objeto Date() a cada segundo, permitindo um relógio na tela em vez da simples hora e data do carregamento da página.

O script abaixo deve estar entre <head> e </head> da página:

```
<script language="JavaScript">

function DataHora(){
var data = new Date();
tempo.innerHTML = data;
setTimeout("DataHora()",1000)
}
</script>
```

O código seguinte deve estar dentro de <body>:

```
onLoad="DataHora()"
```

E este no corpo da página (entre <body> e </body>):

```
<span id=tempo></span>
```

E o resultado é:

É evidente que os nomes aparecem em inglês. Para outra língua, mais códigos são necessários.

getDate()

Retorna um número inteiro entre 1 e 31, que representa o dia do mês do objeto Date.

Exemplo: a variável *dia* contém dia do mês da data atual.

```
data = new Date();
dia = data.getDate();
```

getDay()

Retorna um número inteiro do dia da semana. Domingo 0, segunda 1, terça 2, etc.

Exemplo: a variável *dia_semana* contém o dia da semana da data atual.

```
data = new Date();
dia_semana = data.getDay();
```

getFullYear()

Retorna o ano do objeto Date em números absolutos, por exemplo 1998.

Exemplo: a variável *ano* contém o ano da data atual.

```
data = new Date();
```

```
ano = data.getFullYear();
```

getHours()

Retorna a hora do objeto Date, um número inteiro entre 0 e 23.

Exemplo:

```
data = new Date();
hora = data.getHours();
```

getMilliseconds()

Retorna os milissegundos do objeto Date, um inteiro entre 0 e 999.

Exemplo:

```
data = new Date();
ms = data.getMilliseconds();
```

getMinutes()

Retorna os minutos do objeto Date, um inteiro entre 0 e 59.

Exemplo:

```
data = new Date();
min = data.getMinutes();
```

getMonth()

Retorna o mês do objeto Date, um inteiro entre 0 e 11 (0 janeiro, 1 fevereiro, etc).

Exemplo:

```
data = new Date();
mes = data.getMonth();
```

getSeconds()

Retorna os segundos do objeto Date, um número inteiro entre 0 e 59.

Exemplo:

```
data = new Date();
seg = data.getSeconds();
```

getTime()

Retorna o número de milissegundos da data conforme informado no primeiro tópico. Em geral usado para especificar a data e hora de um outro objeto Date.

Exemplo:

```
certo_dia = new Date("May 15, 1998");
outro_dia = new Date();
outro_dia.setTime(certo_dia.getTime());
```

getTimezoneOffset()

Retorna a diferença, em minutos, entre a hora local e a hora GMT.

Exemplo:

```
data = new Date();
dif_gmt_horas = data.getTimezoneOffset() / 60;
```

Date.parse(string_de_data)

Retorna o número de milissegundos de uma seqüência de caracteres (string) de data, desde 01 de janeiro de 1970 00:00:00 h (hora local).

A string de data deve estar dentro do padrão aceito. Exemplo: Tue, 28 Dec 2004 11:00:00 GMT-0330. Se não especificado o fuso horário, o horário local é considerado.

Deve sempre ser usado na forma Date.parse() e não com um objeto Date criado. Em geral usado com setTime() para especificar uma nova data para o objeto.

Exemplo:

```
data = new Date();
data.setTime(Date.parse("Apr 15, 2004"));
```

setDate(nDia)

Especifica um dia do mês para um objeto Date. nDia deve ser um inteiro entre 1 e 31, de acordo com o mês corrente.

Exemplo:

```
certo_dia = new Date("May 15, 1998");
certo_dia.setDate(25);
```

setFullYear(nAno [, nMes, nDia])

Especifica um ano (em 4 dígitos) para um objeto date existente. nAno é um inteiro de 4 dígitos representando o ano, nMes um inteiro de 0 a 11 para o mês (0 janeiro, 1 fevereiro, etc) e nDia um inteiro de 1 a 31 para o dia do mês. Se é dado um valor para nDia, é obrigatório um valor para nMes.

Se um parâmetro é especificado fora da faixa, a função tenta atualizar os demais de forma coerente. Exemplo: se um valor de 15 é dado para nMes, o ano é aumentado de 1 e o valor 4 é usado para nMes.

Exemplo:

```
certo_dia = new Date();
certo_dia.setFullYear(1995);
```

setHours(nHora [, nMin, nSeg, nMs])

Especifica a hora para um objeto Date. nHora é um inteiro entre 0 e 23. Os demais parâmetros são opcionais: nMin (inteiro entre 0 e 59), nSeg (inteiro entre 0 e 59) e nMs (inteiro entre 0 e 999

para os milissegundos). Se nSeg é fornecido, nMin também deve ser dado. Se nMs é especificado, nMin e nSeg também devem ser.

Se um parâmetro é especificado fora da faixa, a função tenta atualizar os demais de forma coerente. Exemplo: se 70 é dado para nMin, a hora é aumentada de 1 e 10 é usado para nMin.

Exemplo:

```
certo_dia = new Date();
certo_dia.setHours(10);
```

setMilliseconds(nMs)

Especifica os milissegundos para um objeto Date. nMs deve ser um inteiro entre 0 e 999.

Se nMs é dado fora da faixa, o objeto Date é atualizado de forma coerente. Exemplo: se nMs é 1100, os segundos são aumentados de 1 e 100 é usado para os milissegundos.

Exemplo:

```
certo_dia = new Date();
certo_dia.setMilliseconds(500);
```

setMinutes(nMin [, nSeg, nMs])

Especifica os minutos para um objeto Date. nMin deve ser um inteiro entre 0 e 59. Os demais parâmetros são opcionais: nSeg (inteiro entre 0 e 59) e nMs (inteiro entre 0 e 999 para os milissegundos). Se nSeg é especificado, nMin também deve ser. Se nMs é dado, nMin e nSeg também devem ser.

Se um parâmetro é dado fora da faixa, a função tenta atualizar os demais de forma coerente. Exemplo: se nSeg é 80, os minutos são aumentados de 1 e 20 é usado para os segundos.

Exemplo:

```
certo_dia = new Date();
certo_dia.setMinutes(25);
```

setMonth(nMes [, nDia])

Especifica o mês de um objeto Date. nMes deve ser um inteiro entre 0 e 11 (0 janeiro, 1 fevereiro,

etc). nDia é um parâmetro opcional para o dia do mês (inteiro entre 0 e 31).

Se um parâmetro é dado fora da faixa, a função tenta atualizar o objeto de forma coerente. Exemplo: se 12 é dado para o mês, o ano é aumentado de 1 e 1 é usado para o mês.

Exemplo:

```
certo_dia = new Date();
certo_dia.setMonth(4);
```

setSeconds(nSeg [, nMs])

Especifica os segundos de um objeto Date. nSeg deve ser um inteiro entre 0 e 59. nMs é um parâmetro opcional para os milissegundos (inteiro entre 0 e 999).

Se um parâmetro é dado fora da faixa, a função tenta atualizar o objeto de forma coerente. Exemplo: se 70 é dado para nSeg, os minutos são aumentados de 1 e 10 é usado para os segundos.

Exemplo:

```
certo_dia = new Date();
certo_dia.setSeconds(15);
```

setTime(nMs)

Especifica um valor para o objeto Date. nMs é um inteiro correspondente ao número de milissegundos desde 01 de janeiro de 1970 00:00:00 h.

Exemplo:

```
certo_dia = new Date("May 15, 1998");
outro_dia = new Date();
outro_dia.setTime(certo_dia.getTime());
```

toLocaleString()

Retorna uma seqüência de caracteres (string) de data, com formato definido pelas configurações do sistema operacional.

Exemplo:

```
<script language="JavaScript">
var d = new Date();
document.write(d.toLocaleString());
</script>
```

E o resultado desse script é:

Dom 04 Out 2009 16:36:38 BRT

Se o seu sistema operacional não é inglês configurado para o padrão americano, notar a diferença com o exemplo dado no primeiro tópico, O objeto Date().

toUTCString()

Retorna uma string formatada de acordo com a convenção UTC. Pode variar de acordo com o sistema operacional.

Exemplo:

```
data = new Date();
var str = data.toUTCString();
```

Date.UTC(nA, nM, nD [, nHora, nMin, nSeg, nMs])

Retorna o número de milissegundos desde 01 de janeiro de 1970 00:00:00 h, hora universal.

nA: ano depois de 1900. nM: inteiro de 0 a 11 para o mês. nD: inteiro de 1 a 31 para o dia do mês. nHora: inteiro de 0 a 23 para as horas. nMin: inteiro de 0 a 59 para os minutos. nSeg: inteiro de 0 a 59 para os segundos. nMs: inteiro de 0 a 999 para os milissegundos.

Se um parâmetro for dado fora da faixa, a função tenta ajustar os demais de forma coerente. Exemplo: se 12 é usado para mês, ano é incrementado de 1 e 1 é usado para mês.

Deve sempre ser usado na forma Date.UTC(...) e não com um objeto Date criado.

Exemplo:

```
data_utc = new Date(Date.UTC(2004, 11, 15, 12, 0, 0));
```

Funções de Arrays
http://www.mspc.eng.br/info/jscriptArray.shtml

Funções Definidas pelo Usuário (programador)

Sintaxe:
function *nomefuncao*(*var1,var2,...,varX*){
	código da função;
}

Exemplo:

```
<html><head>
<script type="text/javascript">
function olamundo(){
	alert('Olá Mundo do JavaScript!');
}
</script>
</head>
<body><form>
<input type="button" value="Clique!" onClick="olamundo()" />
</form></body></html>
```

Redireciona a página de acordo com a Resolução
Autor: Leandro Alexandre ® <contato@leandro.adm.br> no Viva o Linux

```
<script language="JavaScript">
<!--
function resolucao(){
if (screen.width == "640" && screen.height == "480"){
window.location.href="index_640x480.html";
}
if (screen.width == "800" && screen.height == "600"){
window.location.href="index_800x600.html";
}
if (screen.width == "1024" && screen.height == "768"){
window.location.href="index_1024x768.html";
}
if (screen.width == "1280" && screen.height == "1024"){
window.location.href="index_1280x1024.html";
}
}
resolucao();
//-->
</script>
```

Coloque entre as <head> </head>
http://www.vivaolinux.com.br/scripts/verFonte.php?codigo=3775

Boas Práticas em JavaScript

- Use = = = ao invés de = =
	= = = compara se têm o mesmo valor e se do mesmo tipo.

= = compara se são do mesmo valor apenas

- Cuidado com operações lógicas. Tenha sempre segurança do que está fazendo
- Use Ferramentas – console de Erros do Firefox, a extensão Firibug e o JS Lint (http://www.jslint.com/) para debugar o JavaScript.
- Use sempre o seu código em JavaScript na parte de baixo da página e o CSS na parte de cima. Veja que o CSS é necessário para a formatação e importante desde o início, já o JS deve ficar na parte de baixo para um melhor desempenho.
- Ao criar um for, declare as variáveis fora do for.
- Reduza o uso de variáveis gloais
- Comente seu código
- Para uma longa lista de variáveis, use vírgula ao invés de vários var, como a seguir: var var1 = valor1, var2 = valor2, var3 = valor3, var4 = valor4;
- Nunca esqueça de usar o ponto e vírgula (;) para finalizar instruções.
- Leia, leia, leia e leia muito, em especial as novidades da área.

Referências:

Gráficos com JavaScript - http://thedesignmag.com/16-javascript-graph-resources-for-creating-stylish-chart.html
http://net.tutsplus.com/tutorials/html-css-techniques/30-html-best-practices-for-beginners/
http://net.tutsplus.com/tutorials/javascript-ajax/24-javascript-best-practices-for-beginners/
http://net.tutsplus.com/tutorials/html-css-techniques/30-css-best-practices-for-beginners/
http://net.tutsplus.com/tutorials/html-css-techniques/20-html-forms-best-practices-for-beginners/
http://www.javascriptajax.com.br/javascript-core/caracteristicas-do-javascript.html
http://dzineblog.com/2009/08/getting-creative-with-javascript-30-websites-with-cool-slider-and-scrollers.html
http://www.javaworld.com/javaworld/jw-09-1996/jw-09-javascript.html

3-CSS

Introdução ao CSS

1 - Introdução

O que é CSS?
CSS é a abreviatura para **C**ascading **S**tyle **S**heets. - Folha de Estilos em Cascata

O que eu posso fazer com CSS?

CSS é uma linguagem para estilos que define o layout de documentos HTML. Por exemplo, CSS controla fontes, cores, margens, linhas, alturas, larguras, imagens de fundo, posicionamentos e muito mais. Aguarde e você verá!

HTML pode ser (in)devidamente usado para definir o layout de websites. Contudo CSS proporciona mais opções e é mais preciso e sofisticado. CSS é suportado por todos os navegadores atuais.

Depois de estudar algumas poucas lições deste tutorial, você estará em condições de projetar uma folha de estilos, usando CSS para dar um grande visual ao seu website.

Qual é a diferença entre CSS e HTML?

HTML é usado para estruturar conteúdo. CSS é usado para formatar conteúdo estruturado.

OK, isto soa um tanto técnico e confuso. Mas, por favor continue lendo. Tudo fará sentido em breve.

Em tempos passados quando a Madonna era virgem e um sujeito chamado Tim Berners Lee inventou a World Wide Web, a linguagem HTML era usada somente para estruturar textos. Um autor podia marcar seus textos definindo "isto é um cabeçalho " ou "isto é um parágrafo" usando tags HTML tais como <h1> e <p>.

À medida que a Web ganhava popularidade, os designers começavam a sentir a necessidade de encontrar meios de construir layout para os documentos online. Para suprir estas necessidades os fabricantes de navegadores (àquela época a Netscape e a Microsoft) inventaram novas tags HTML tais como, por exemplo a tag <font> que se diferenciava das tags originais do HTML pelo fato de destinar-se à layout — e não à estrutura.

Isto adicionalmente teve o efeito de disvirtuar o emprego de tags inicialmente projetadas para estrutura como por exemplo a tag <table> que passaram a ser empregadas para layout. Muitas destas novas tags para layout como a tag <blink> eram suportadas somente por um determinado tipo de navegador. A frase "Você precisa do navegador X para visualizar esta página" tornou-se comum nos websites.

CSS foi inventada para solucionar esta situação, colocando à disposição dos web designers meios sofisticados de projetar layouts suportados por todos os navegadores. E ao mesmo tempo a separação dos estilos de apresentação da marcação dos conteúdos torna a manutenção dos sites bem mais fácil.

Quais são os benefícios do uso de CSS?

CSS é uma revolução no mundo do web design. Os benefícios concretos do uso de CSS incluem:

- controle do layout de vários documentos a partir de uma simples folha de estilos;
- maior precisão no controle do layout;
- aplicação de diferentes layouts para servir diferentes mídias (tela, impressora, etc.);
- emprego de variadas, sofisticadas e avançadas técnicas de desenvolvimento.

Tradução do inglês de http://www.pt-br.html.net/tutorials/css/lesson1.asp
Referência
http://www.w3schools.com/css/default.asp

Testar online
http://www.w3schools.com/css/tryit.asp?filename=trycss_ex

Sintaxe:
seletor {propriedade: valor;}

Exemplo:
body {color:black;}

Mudará a cor do texto do body para preto.
O que mostra que quando aplicamos CSS ao HTML, o CSS mudará (sobrescreverá) o comportamento do HTML.

Existem as classes e os IDs. Veja como os criamos:

```
.nome_classe {
        margin-left: 1.0in;
}

#nome_id {
        margin-left: 1.0in;
}
```

Classe inicia com ponto (.).
ID inicia com cerquilha (#).

Separando em Camadas com CSS

Uma das funções importantes do CSS é permitir que separemos em camadas o conteúdo do site (HTML) da formatação (CSS) e também o JavaScript em outra camada.
Cada camada em arquivo separado:
conteúdo - .html
formatação - .css
JavaScript - .js

Suprindo Deficiências do HTML

Outra função muito importante do CSS é suprir algumas deficiências do HTML. Por exemplo, o HTML com seus recursos não permite mudar a cor de fundo de uma caixa de texto de um input de formulário e o CSS consegue. Assim vários outros recursos permitidos pelo CSS, como mudar a cor de fundo de uma textarea, de um botão, cor das bordas de uma caixa de texto, cor ao mover o mouse, etc. Alterar a posição de um elemento da página para o lado de outro elemento.

Exemplo de CSS que altera o comportamento do input e do select e também quando perde o foco:

```
input {
	background-color: #6D5B52;
	border-style: 1px solid;
	border-color: black gray gray black;
	color: white;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 11px;
}

input:focus {
	background-color: #6D5B52;
	border-style: 1px solid;
	border-color: red red red red;
	color: white;
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 11px;
}

select {
	background-color: #6D5B52;
	font:12px verdana, arial, helvetica, sans-serif;
	color:white;
}

select:focus {
	background-color: #3C1103;
	font:12px verdana, arial, helvetica, sans-serif;
	color:white;
}
```

Quando no CSS usamos o nome de controles de formulários, o seu comportamento e formatação será substituído pelo comportamento adicionado no CSS (que prevalecerá).

2 - Script interno

Este **cria CSS dentro da página HTML.**

```
<html>
  <head>
    <title>Exemplo</title>
    <style type="text/css">
      body {background-color: #FF0000;}
    </style>
  </head>
  <body>
    <p>Esta é uma página com fundo vermelho</p>
  </body>
</html>
```

Uso nas tags do HTML
Aqui criamos dentro das tags HTML (na tag body).

```
<html>
  <head>
    <title>Exemplo</title>
  </head>
  <body style="background-color: #FF0000;">

    <p>Esta é uma página com fundo vermelho</p>
  </body>
</html>
```

Script externo
Este deixa o código CSS num outro arquivo e é o mais indicado para quando temos grandes códigos CSS.

```
<html>
  <head>
    <title>Meu documento</title>
    <link rel="stylesheet" type="text/css" href="style/style.css" />

  </head>
  <body>
```

4 - Uso do CSS nas Tags HTML

Assim como o JavaScript podemos usar o CSS nas tags HTML, como no exemplo abaixo:

Código <input type="text" name="codigo" value="<?php print $codigo;?>" READONLY style="background-color:gray;">

O CSS foi usado para deixar o fundo da caixa de texto com cor cinza, simulando desabilitado, coisa que não se consegue com HTML.

Ouro exemplo:

<h1style="color:red;">Título em cor vermelha</h1>

5 - Uso do CSS em Arquivos Externos

Este é o uso recomendado do CSS, em arquivos externos, para separas as camadas,

ficando no arquivo CSS tudo que diz respeito à formatação e ao posicionamento e no HTML o conteúdo.

Assim, se quisermos alterar o conteúdo não haverá problema com a formatação e posicionamento e vice-versa.

Para criar um arquivo externo em CSS, não podemos adicionar as tags <style> ... </style>, apenas o conteúdo como no exemplo seguinte.

style.css

```
body {
margin: 0;
font-family: verdana, arial, helvetica, sans-serif;
color: #F9E8E5;
/* background-color: #C5F5BF; */
background-image: url("../../images/bg_brown.jpg");
border-top: 15px #082A08 solid; /* Barra acima do Título */
font-size: 14px;
}

#main-title {
margin-bottom: 10px;
margin-left: 0;
margin-right: 0;
padding-top:5px;  /* Corrigir no original, que era 15 */
padding-bottom: 8px;
padding-left: 25px;
padding-right: 0;
background-color: #571B07;
border-top: 1px black solid;
border-bottom: 1px black solid;
color: white;
font-weight: bold;
font-size: 18px;
font-family: Verdana, Helvetica, Arial, sans-serif;
}
```

Para usar este arquivo em um HTML efetuamos o "include" entre as tags <head> e </head>, assim:

```
<head>
	<title>Título da Página</title>
	<link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

6 - Cores

Nesta lição você aprenderá como aplicar cores de primeiro plano e cores de fundo no seu website. Abordaremos ainda os métodos avançados de controle e posicionamento de imagens de fundo. Serão explicadas as seguintes propriedades CSS:

- color
- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background

Cor do primeiro plano: a propriedade 'color'

A propriedade `color` define a cor do primeiro plano de um elemento.

Considere, por exemplo, que desejamos que todos os cabeçalhos de primeiro nível no documento sejam na cor vermelha. O elemento HTML que marca tais cabeçalhos é o elemento <h1>. O código a seguir define todos os <h1> na cor vermelha.

```
h1 {
        color: #ff0000;
}
```

- Ver exemplo

As cores podem ser definidas pelo seu valor hexadecimal como no exemplo acima (#ff0000), com uso do nome da cor ("red") ou ainda pelo seu valor rgb (rgb(255,0,0)).

A propriedade 'background-color'

A propriedade `background-color` define a cor do fundo de um elemento.

O elemento <body> contém todo o conteúdo de um documento HTML. Assim, para mudar a cor de fundo da página, devemos aplicar a propriedade `background-color` ao elemento <body>.

Você pode aplicar cores de fundo para outros elementos, inclusive para cabeçalhos e textos. No exemplo abaixo foram aplicadas diferentes cores de fundo para os elementos <body> e <h1>.

```
body {
        background-color: #FFCC66;
}

h1 {
        color: #990000;
        background-color: #FC9804;

}
```

- Ver exemplo

Notar que foram aplicadas duas propriedades ao elemento <h1> separadas por um ponto e vírgula.

Images de fundo [background-image]

A propriedade CSS `background-image` é usada para definir uma imagem de

fundo.

Usaremos a imagem de uma borboleta para exemplificar a aplicação de imagens de

fundo. Você pode fazer o download da imagem mostrada abaixo e usá-la nos seus

experimentos (clique com o botão direito do mouse sobre a imagem e escolha "salvar imagem como") ou você poderá usar uma outra imagem qualquer ao seu gosto.

Para inserir uma imagem de fundo na página basta aplicar a propriedade `background-image` ao elemento <body> e especificar o caminho para onde está gravada a imagem.

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.gif");
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

- Ver exemplo

NB: Notar como foi especificado o caminho para a imagem usando **url("butterfly.gif")**. Isto significa que a imagem está localizada no mesmo diretório da folha de estilos. Pode ser escolhido um outro diretório para gravar as imagens e o caminho seria **url("../images/butterfly.gif")** ou até mesmo hospedá-la na Internet: **url("http://www.html.net/butterfly.gif")**.

Imagem de fundo repetida [background-repeat]

No exemplo anterior você observou que a imagem da borboleta repetiu tanto na vertical como na horizontal cobrindo toda a tela? A propriedade `background-repeat` controla o comportamento de repetição da imagem de fundo.

A tabela a seguir mostra os quatro diferentes valores para `background-repeat`.

Value	Description	Example
`Background-repeat: repeat-x`	A imagem se repete na horizontal	Ver exemplo
`background-repeat: repeat-y`	A imagem se repete na vertical	Ver exemplo
`background-repeat: repeat`	A imagem se repete na tanto na horizontal como na vertical	Ver exemplo
`background-repeat: no-repeat`	A imagem não se repete	Ver exemplo

Por exemplo, o código mostrado a seguir é para que a imagem não se repita na tela:

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.gif");
        background-repeat: no-repeat;
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

- Ver exemplo

Image de fundo fixa [background-attachment]

A propriedade `background-attachment` define se a imagem será fixa ou se irá rolar juntamente com o elemento que a contém.

Uma imagem de fundo fixa permanece no mesmo lugar e não rola com a tela ao contrário da imagem que não é fixa e rola acompanhando o conteúdo da tela.

A tabela a seguir mostra os quatro diferentes valores para `background-attachment`. Veja os exemplos para constatar a diferença entre imagem fixa e imegem que rola.

Value	Description	Example
`Background-attachment: scroll`	A imagem rola com a página	Ver exemplo
`Background-attachment: fixed`	A imagem é fixa	Ver exemplo

Por exemplo, o código abaixo fixa a imagem na tela.

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.gif");
        background-repeat: no-repeat;
        background-attachment: fixed;
}

h1 {
        color: #990000;
        background-color: #FC9804;
}
```

- Ver exemplo

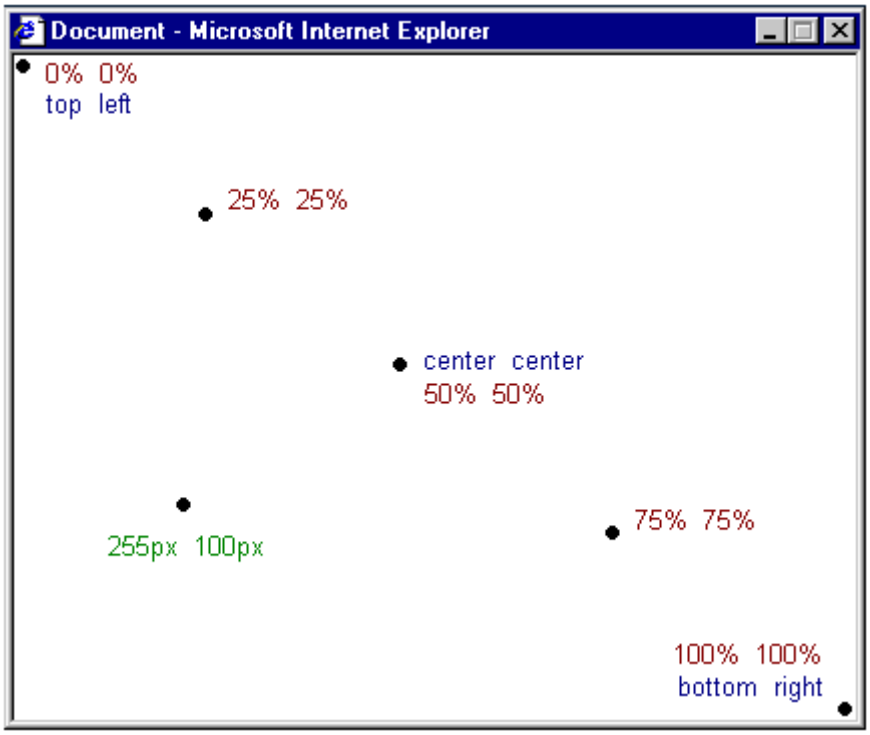
Posição da imagem de fundo [background-position]

Por padrão uma imagem de fundo é posicionada no canto superior esquerdo da tela. A propriedade `background-position` permite alterar este posicionamento padrão e colocar a imagem em qualquer lugar na tela.

Existem várias maneiras de definir o posicionamento da imagem na tela definindo valores

para `background-position`. Todas elas se utilizam de um sistema de coordenadas. Por exemplo, os valores '100px 200px' posiciona a imagem a 100px do topo e a 200px do lado esquerdo da janela do navegador.

As coordenadas podem ser expressas em percentagem da largura da janeja, em unidades fixas (pixels, centímetros, etc.) ou pode-se usar as palavras top, bottom, center, left e right. A figura a seguir ilustra o modelo de coordenadas:



Na tabela a seguir são mostrados alguns exemplos .

Value	Description	Example
`background-position: 2cm 2cm`	A imagem é posicionada a 2 cm da esquerda e 2 cm para baixo na página	Ver exemplo
`background-position: 50% 25%`	A imagem é centrada na horizontal e a um quarto (25%) para baixo na página	Ver exemplo
`background-position: top right`	A imagem é posicionada no canto superior direito da página	Ver exemplo

No exemplo de código a seguir a imagem é posicionada no canto inferior direito da página:

```
body {
        background-color: #FFCC66;
        background-image: url("butterfly.gif");
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-position: right bottom;
}
```

```
h1 {
        color: #990000;
        background-color: #FC9804;
}
```

- Ver exemplo

Compilando [background]

A propriedade `background` é uma abreviação para todas as propriedades listadas anteriormente.

Com `background` você declara várias propriedades de modo abreviado, economizando digitação e alguns bites, além de tornar a folha de estilo mais fácil de se ler e entender.

Por exemplo, observe as cinco linhas a seguir:

```
background-color: #FFCC66;
background-image: url("butterfly.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
```

Usando `background` você consegue o mesmo resultado, abreviando como mostrado abaixo:

```
background: #FFCC66 url("butterfly.gif") no-repeat fixed right bottom;
```

A declaração abreviada deve seguir a seguinte ordem:

`[background-color]` | `[background-image]` | `[background-repeat]` | `[background-attachment]` | `[background-position]`

Se uma das propriedades não for declarada ela assume automaticamente o seu valor default. Por exemplo, a propriedade `background-attachment` e `background-position` não foram declaradas no código mostrado a seguir:

```
background: #FFCC66 url("butterfly.gif") no-repeat;
```

Exemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <title>My first styled page</title>
  <style type="text/css">
  body {
    font-family: Georgia, "Times New Roman",
          Times, serif;
    color: purple;
    background-color: #d8da3d }
```

```
  h1 {
    font-family: Helvetica, Geneva, Arial,
          SunSans-Regular, sans-serif }
  </style>
</head>

<body>
```

7 - Como Trocar o Cursor do Mouse com CSS

http://networking.mydesigntool.com/viewtopic.php?tid=466&id=31

cursor:default

cursor:pointer

Melhorando o estilo de tabelas com CSS

http://networking.mydesigntool.com/viewtopic.php?tid=468&id=31

As duas propriedades não declaradas assumirão o valor default que como você já sabe são: a imagem rola na tela e será posicionada no canto superior esquerdo (que são os valores default para as propriedades não declaradas).

Sumário

Nesta lição você aprendeu técnicas que não são possíveis com uso de HTML. A brincadeira continua na próxima lição onde examinaremos as possibilidades de estilização das fontes.

Bom tutorial em: http://www.pt-br.html.net/tutorials/css/
http://maujor.com/tutorial/css3-modulo-template-layout.php
http://www.echoecho.com/css.htm

Uso de sprites com CSS:
http://ribafs.org/portal/colaboracoes/1-ribamar-fs/123-sprites-em-css

Templates com CSS
http://ribafs.org/portal/projetos-opensource/84-utilitarios-em-php/143-templates-usando-css

8 - Comentários em CSS

Para comentar código em CSS devemos usar somente o comentário tipo C:

```
/*
Estas linhas não serão interpretadas
pelo navegador.
*/
```

A intenção aqui foi apenas de mostrar uma pequena porção dos recursos do CSS, que são muitos.
Para mais detalhes consulte os sites citados ou outros bons tutoriais existentes na Internet.

9 - Dicas úteis para CSS

```
*{
        margin: 0; padding: 0;
}

body {
        font: 12px "Lucida Grande", Sans-Serif;
        background: #ccc;
}

h1, h2, h3     {
        font: 28px Georgia, Serif;
        border-bottom: 1px dotted #ccc;
        margin: 0 0 10px 0;
}

input[type="text"] {
        padding: 3px;
        border: 1px solid #666;
        width: 350px;
        margin: 0 0 15px 0;
}
```

10 - Recomendações para HTML, JavaScript e CSS

- Deixe o código fácil de ser lido
- Mantenha-o consistente
- Organize bem o código
- Use bem a indentação em funções, classes, ids, trechos de código
- Crie o HTML antes
- Não crie classes grandes, mas várias pequenas
- Use o correto doctypy: http://www.alistapart.com/articles/doctype/
- Comente seu código
- No CSS use
  - margin: 0 auto; // top, bottom e left, right valores, respectivamente. para centrar divs, parágrafos ou outros elementos em seu layout.
- Use a extensão do Firefox, o Firebug para ganhar produtividade
- Use no CSS: text-transform: lowercase;
- Valide seu HTML (http://validator.w3.org/) e seu CSS (http://jigsaw.w3.org/css-validator/)
- Para tamanho da página use pixel e para tamanho de fontes use ems.
- Use bastante as listas

– Maior compatibilidade entre os navegadores. Inicie o CSS com: * {margin:0;padding:0;}
– Use um reset CSS: http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded/
http://www.webdesignerdepot.com/2009/08/250-resources-to-help-you-become-a-css-expert/

11 - Funções em PHP que retornam CSS

```
<?php
// Funções em PHP que retornam CSS
/* Cores
RGB = "rgb(255,255,0)"
HEXADECIMAL = "#FFFF00"
NOME = "yellow"
*/
function cor_fundo($cor){
        return $fundo = "style=\"background-color:$cor\"";
}

function cor_fonte($cor){
        return "style=\"color:$cor\"";
}
print "<table><tr ". cor_fundo('black') .."><td ".
cor_fonte('white').">TESTE</td></tr></table>";

function imagem_fundo($imagem){
        return "background-image: url(\"$imagem\")";
}

function imagem_fundo($imagem){
        return "style=\"background-image: url('$imagem')\"";
}
print "<body ".imagem_fundo('/dnocs.jpg').">"; // Esta imagem deve estar num diretório
disponível ao apache

function random_color()
{
          $chars = "abcdef0123456789";

        srand((double)microtime() * 1000000);

          $color = '';

          for ($i=0; $i<6; $i++)
          {
             $num = rand(0, 15);
             $tmp = substr($chars, $num, 1);

             $color .= $tmp;
          }
```

```
        return "#" . $color;
}
//print random_color();

?>
```

Referências

Mastering CSS, Part 1: Styling Design Elements
http://www.smashingmagazine.com/2009/08/03/mastering-css-styling-design-elements/
http://www.smashingmagazine.com/2009/08/10/mastering-css-advanced-techniques-and-tools/
40 CSS Web Form Style Tutorials For Web Developers
http://www.websiteoptimization.com/speed/tweak/forms/css.html
Fancy Form Design Using CSS de Cameron Adams na sitepoint (http://sitepoint.com)
pForm - http://www.phpform.org/
http://nidahas.com/2006/12/06/forms-markup-and-css-revisited/
http://www.alistapart.com/articles/prettyaccessibleforms/
Layouts e Templates CSS Free - http://www.smashingmagazine.com/2007/01/12/free-css-layouts-and-templates/
Layouts CSS - http://www.dynamicdrive.com/style/layouts/category/C11/
CSS Reference - http://xhtml.com/en/css/reference/
CSS Reference - http://www.w3schools.com/CSS/CSS_reference.asp

Resumo do CSS

Cor para a tag H1

```
<style type="text/css">
H1 { color: green }
</style>
```

Fonte para parágrafo

```
<style type="text/css">
p { font-family: "comic sans ms" }
</style>
```

**Estilo em **

```
<div style="BACKGROUND-COLOR: yellow">Lançado o Joomla! 1.0.3 </div>
```

Alinhamento, cor e fonte para parágrafo

```
<style type="text/css">
p {
        text-align: center;
        color: green;
        font-family: arial;
        background-color: red;
}
</style>
```

Agrupando seletores

```
h1, h2, h3, h4, h5, h6 {
        color: green;
}
```

Seletores de classe

```
<style type="text/css">
        p.direita  { text-align: right }
        p.centro   { text-align: center }
</style></head><body>
<p class="direita">
        Este parágrafo está alinhado a direita.
</p>
<p class="centro">
        Este parágrafo está alinhado ao centro.
</p>
```

Genérico (utilizar em qualquer tag)

```
<style type="text/css">
        .centro   { text-align: center }
</style>
</head>
<body>
<h2 class="centro">
        Cabeçalho alinhado ao centro
```

```
</h2>
<p class="centro">
        Este parágrafo também está alinhado ao centro.
</p>
```

Seletor de ID

```
<style type="text/css">
p#para1{
        text-align: center;
        color: red
}
</style>
</head>
<body>
<p id="para1">
        Este parágrafo está alinhado ao centro e
        tem cor encarnada.
</p>
```

Soltos:

```
P {margin-left: 20px }        (px = 1 pixel)
body { background-image: url (“images/nome.gif”) }
font-size: 8pt (1pt = 1/72 polegadas)
text-align: left
border: solid thin red

<p style="color: blue; margin-left: 20px">
Isto é um parágrafo formatado com o atributo style
</p>

<p style="font-family: 'sans-serif'">
Neste parágrafo o tipo de letra é "sans-serif"
</p>
<p style="font-family: 'comic sans ms'">
Neste parágrafo o tipo de letra é "comic sans ms"
</p>
```

Include do CSS

```
<link rel=”stylesheet” type=”text/css” href=”includes/includes.inc.css”>


<style type="text/css">
p{
        color: blue;
        font-family: cursive;
        font-size:16px
}
.verde{
```

```
        color:green
}
</style>
```

background-position: center center (top left ou 0% 0%)

background-repeat: no-repeat (repeat-x ou repeat-y, horizontal ou vertical)

background-attachment: fixed ou scroll (imagem fixa ou rolante com o scroll)

Espaçamento entre letraas

p { letter-spacing: 12px}

p { letter-spacing: -0.5 }

Alinhamento
text-align: left, right, center, justify

Decoração do Texto

text-decoration: none (underline, overline, line-through e blink)

Text-transform

text-transform: uppercase (none, capitalize, lowercase)

Fontes

```
<style type="text/css">
h3 {font-family: times}
p {font-family: courier}
p.sansserif {font-family: sans-serif}
</style>
   <title></title>
  </head>
  <body>
   <h3>Isto é um cabeçalho de nível 3</h3>
   <p>Isto é um parágrafo</p>
   <p class="sansserif">Isto é um parágrafo com uma class diferente</p>
```

Tamanho das Letras

```
<style type="text/css">
h1 {font-size: 150%}
h2 {font-size: 130%}
p {font-size: 100%}
</style>
```

Estilo das Letras

```
<style type="text/css">
h1 {font-style: italic}
h2 {font-style: normal}
p {font-style: oblique}
</style>
```

```
<style type="text/css">
p.normal {font-weight: normal}
p.thick {font-weight: bold}
p.thicker {font-weight: 900}
</style>

    <title></title>
  </head>
  <body>
    <p class="normal">Insto é um parágrafo com letra normal</p>
    <p class="thick">Insto é um parágrafo com letra mais grossa</p>
    <p class="thicker">Insto é um parágrafo com letra muito grossa</p>
```

```
<style type="text/css">
p{
        font: italic small-caps 900 13px arial
}
</style>
```

Espessura do Traço
font-weight: bold (bolder, lighter, 100, 200, 300, … 800)

Bordas das palavras

```
<style type="text/css">
p.dotted {border-style: dotted}
p.dashed {border-style: dashed}
p.solid {border-style: solid}
p.double {border-style: double}
p.groove {border-style: groove}
p.ridge {border-style: ridge}
p.inset {border-style: inset}
p.outset {border-style: outset}
</style>
```

```
<style type="text/css">
p.soliddouble {border-style: solid double}
p.doublesolid {border-style: double solid}
p.groovedouble {border-style: groove double}
p.three {border-style: solid double groove}
</style>
```

Cores das Bordas

```
<style type="text/css">
p.stl1{
border-style: solid;
border-color: #0000ff
}
p.stl2{
border-style: solid;
border-color: #ff0000
#0000ff
}
p.stl3{
border-style: solid;
border-color: #ff0000
#00ff00 #0000ff
}

p.stl4{
border-style: solid;
border-color: #ff0000
#00ff00 #0000ff
rgb(250,0,255)
}
</style>
```

Borda inferior

```
<style type="text/css">
p {
border-bottom: medium solid #ff0000
}
</style>
```

Borda Lateral

```
<style type="text/css">
p {
border-left: medium solid #ff0000
}
</style>
```

Borda superior

```
<style type="text/css">
p {
border-top: medium solid #ff0000
}
</style>

<style type="text/css">
```

```
p {
border: medium double #ff00ff
}
</style>
```

Largura e Altura de Imagem

```
<style type="text/css">
img.normal {
height: auto;
width: auto
}

img.grande {
height: 64px;
width: 64px
}

img.pequeno{
height: 16px;
width: 16px
}
</style>
  </head>
  <body>
    <img class="normal" height="32" width="32" src="2569.gif"/><br/><br/>
    <img class="grande" height="32" width="32" src="2569.gif"/><br/><br/>
    <img class="pequeno" height="32" width="32" src="2569.gif"/>
```

Posicionamento

```
<style type="text/css">
h1{
position: absolute;
top: 100px;
left: 100px
}
p{
position: absolute;
top: 200px;
left: 100px
}
</style>  </head>  <body>
    <h1>Um Cabeçalho</h1>
    <p> O <b>cabeçalho</b> está colocado 100px abaixo do topo do
        documento e 100px à direita da extremidade
        esquerda. O <b>parágrafo</b> está colocado 200px abaixo
        do topo do documento e 100px à direita da extremidade
        esquerda.
    </p>
```

Alinhas Imagem na Vertical

```
<style type="text/css">
img.top {vertical-align:text-top}
img.bottom {vertical-align:text-bottom}
</style>

   <title></title>
  </head>
  <body>
   <p> Isto é uma imagem <img class="bottom" src="2569.gif"/>
       dentro de um parágrafo.
   </p>

   <p> Isto é a mesma imagem <img class="top" src="2569.gif"/>
       dentro de outro parágrafo.
   </p>
```

Quando Texto não cabe

```
<style type="text/css">
img.top {vertical-align:text-top}
img.bottom {vertical-align:text-bottom}
</style>

   <title></title>
  </head>
  <body>
   <p> Isto é uma imagem <img class="bottom" src="2569.gif"/>
       dentro de um parágrafo.
   </p>

   <p> Isto é a mesma imagem <img class="top" src="2569.gif"/>
       dentro de outro parágrafo.
   </p>
```

Elemento por traz de outro

```
<style type="text/css">
img.x
{
position:absolute;
left:0;
top:0;
z-index:-1
}
</style>

   <title></title>
  </head>
```

```
  <body>
    <h1>Isto é um cabeçalho</h1>
    <img class="x" src="euros.jpg"/>
    <p>O valor por omissão da propriedade z-index é 0.
        A imagem tem o z-index com valor -1, pelo que
        a sua prioridade é inferior. Por isso ela
        aparece atrás do restante conteúdo.
    </p>
```

Cortando Imagem

```
<style type="text/css">
img
{
position:absolute;
clip: rect(0 200 100 0)
}
</style>

    <title></title>
  </head>
  <body>
    <p>A propriedade "CLIP" permite-nos cortar a imagem.
        Neste exemplo as dimensões da imagem são 300x200,
        mas os valores dados à propriedade "CLIP" fazem com
        que apenas seja apresentada uma secção com 200x100.
    </p>

    <p><img width="300" height="200" src="caminho.jpg"/></p>
```

Formatação de Listas

```
<style type="text/css">
ul.disc
{       list-style-type: disc
}

ul.circle
{       list-style-type: circle
}

ul.square
{       list-style-type: square
}

ul.none
{       list-style-type: none
}
</style>
```

```
    <title></title>
  </head>

  <body>
    <ul class="disc">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
    </ul>

    <ul class="circle">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
    </ul>

    <ul class="square">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
    </ul>

    <ul class="none">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
    </ul>

<style type="text/css">
ol.decimal {   list-style-type: decimal      }
ol.lroman {    list-style-type: lower-roman        }
ol.uroman {   list-style-type: upper-roman        }
ol.lalpha {     list-style-type: lower-alpha }
ol.ualpha {    list-style-type: upper-alpha}
</style>
    <title></title>
  </head>

  <body>
    <ol class="decimal">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
    </ol>

    <ol class="lroman">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
```

```
    </ol>

    <ol class="uroman">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
    </ol>

    <ol class="lalpha">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
    </ol>

    <ol class="ualpha">
      <li>Rebuçados</li>
      <li>Pastilha elástica</li>
      <li>Caramelos</li>
    </ol>

<style type="text/css">
ul
{
list-style: square inside url("seta.gif")
}
</style>

    <title></title>
  </head>

  <body>
    <ul>
      <li>Rebuçados</li>
      <li>Pastilha Elástica</li>
      <li>Caramelos</li>
    </ul>
```

Cores de Links

```
<style type="text/css">
a.stl1:link {color: #ff0000}
a.stl1:visited {color: #0000ff}
a.stl1:hover {color: #ffcc00}

a.stl2:link {color: #ff0000}
a.stl2:visited {color: #0000ff}
a.stl2:hover {font-size: 150%}

a.stl3:link {color: #ff0000}
```

```
a.stl3:visited {color: #0000ff}
a.stl3:hover {background: #66ff66}

a.stl4:link {color: #ff0000}
a.stl4:visited {color: #0000ff}
a.stl4:hover {font-family: fixedsys}

a.stl5:link {color: #ff0000; text-decoration: none}
a.stl5:visited {color: #0000ff; text-decoration: none}
a.stl5:hover {text-decoration: underline}
</style>

   <title></title>
</head>

<body>
<p>
  Passe com o ponteiro do rato sobre as ligações e
  veja como elas reagem de maneira diferente.
</p>
<p style="font-weight: bold">
  <a class="stl1" href="ex_css.htm">Esta muda de cor</a><br/>
  <a class="stl2" href="ex_css.htm">Esta muda o tamanho das letras</a><br/>
  <a class="stl3" href="ex_css.htm">Esta muda a cor de fundo</a><br/>
  <a class="stl4" href="ex_css.htm">Esta muda o tipo de letra</a><br/>
  <a class="stl5" href="ex_css.htm">Esta muda a decoração do texto</a>
</p>
```

Destacando primeira letra

```
<style type="text/css">
p      { font-size: 12pt }
p:first-letter   { font-size: 200%; float: left }
</style>
</head>
<body>
<p>
A primeira letra deste parágrafo é maior<br>
do que as restantes.
</p>
<p>
A primeira letra deste parágrafo é maior<br>
do que as restantes.
</p>

<style type="text/css">

div:first-letter
{
```

```
color: red;
font-size:xx-large
}
</style>
   <title></title>
  </head>

  <body>
   <p><b>Nota:</b> O Internet Explorer 5.0 não
        suporta o pseudo-elemento "first-letter".</p>
   <div>
        O pseudo-elemento "first-letter" permite dar um estilo
        especial ao texto fazendo com que a primeira letra de
        um bloco tenha um estilo diferente.
   </div>
  </body>
```

Controlando Quebra de Página

table { page-break-after: always }

auto, avoid (evitar as mudasças de página após o elemento)
always – sempre muda de página
left – mudança de página depois do elemento até alcançar uma página do lado esquerdo em branco (tipo livro, lado esquerdo e direito são diferentes)
right - mudança de página depois do elemento até alcançar uma página do lado direito em branco (tipo livro, lado esquerdo e direito são diferentes)

table { page-break-before: always }

table { page-break-inside: always }
auto (deixa a quebra de linha de acordo com o padrão do documento (ao fim da página).
avoid (evitar as mudanças de página dentro do elemento)

```
<html>
<head>
<title>Propriedades “page-break-after” e “page-break-before”</title>
</head>
<body>
<img src="s2slogo.gif" width=100 height=100>
<div style="page-break-after:always">Page Marker #1</div>

<img src="internetdotcomlogo.gif" width=120 height=151>

<div style="page-break-before:always">Page Marker #2</div>
<img src="internetdotcomlogo.gif" width=120 height=151>
</body>
</html>
```

<STYLE>
.pbreak
{
PAGE-BREAK-AFTER: always
}
</STYLE>

e depois colocar esse divzinho onde vc quer quebrar

```
Exemplo de Relatório:

<style type="text/css">
   @page { size:landscape; }
</style>
```

```
<html>
  <head>
    <title>Meine Seite</title>
  </head>
  <body>
    <div class="Menu">
      <a href="index.htm>Home</a>
      <a href="eineSeite.htm>Eine Seite</a>
      <a href="eineAndere.htm>Eine andere</a>
    </div>
    <div class="Inhalt">
      <h1>Eine &Uuml;berschrift</h1>
      <p>Ein Text</p>
    </div>
  </body>
 </html>
```

```
.Menu {
   font-family: Arial, MS Sans Serif, Verdana;
   font-size: 12px;
   color: #0000FF;
}
   .Inhalt {
   font-family: Arial, MS Sans Serif, Verdana;
   font-size: 14px;
   color: #999999;
}
```

```
<html>
 <head>
 <title>Meine Seite</title>
 <style type="text/css" media="screen,projection">
   .Menu {
```

```
    font-family: Arial, MS Sans Serif, Verdana;
    font-size: 12px;
    color: #0000FF;
    }
    .Inhalt {
    font-family: Arial, MS Sans Serif, Verdana;
    font-size: 14px;
    color: #999999;
    }
  </style>
  <style type="text/css" media="print">
    /*Der allgemeine Bereich Menü wird versteckt*/
    .Menu {
    visibility: hidden;
    display: none;
    }
    /*Die Schriftfarbe wird auf schwarz gesetzt*/
    .Inhalt {
    font-family: Arial, MS Sans Serif, Verdana;
    font-size: 14px;
    color: black;
    }
  </style>
  </head>
  <body>
  <div class="Menu">
    <a href="index.htm>Home</a>
    <a href="eineSeite.htm>Eine Seite</a>
    <a href="eineAndere.htm>Eine andere</a>
  </div>
  <div class="Inhalt">
    <h1>Eine &Uuml;berschrift</h1>
    <p>Ein Text</p>
  </div>
  </body>
</html>
```

Múltiplas fontes

Você pode utilizar na tag <FONT> múltiplas fontes especificando a ordem de preferência. Ou seja, se a primeira fonte indicada por você não constar no computador do visitante, a segunda será a escolhida, caso não tenha passa a ser a terceira e assim por diante. Veja o exemplo:

<FONT FACE="Arial, Comic Sans MS, Helvetica" SIZE="12" COLOR="purple">

Ou na declaração CSS:
H1 {font-family: "Arial, Comic Sans MS, Helvetica" ... }

Fonte incorporada

Utilizando as fontes incorporadas, o visitante de seu site fará o download e a instalação da fonte utilizada por você em seu site caso ele ainda não a tenha em seu micro. A sintaxe para incorporação é a seguinte:

<STYLE TYPE="text/css"><!--
@font-face {font-family: nome_da_fonte;
font-style: estilo_da_fonte;
font-weight: densidade_da_fonte;
src: url_da_fonte}
--></STYLE>
OBS: Essa declaração deve estar entre as tags <HEAD>...</HEAD> em sua página HTML.

Referência:
Livro Eletrônico de Helder José
http://www.artifice.web.pt/ebooks.html

Dicas Seletas de CSS

1 - Formatar todas as tags <P> de um documento

```
<head>
        <style type=”text/css”>
                P {font-family:Arial; color:blue;}
        </style>
</head>

<body>
        <P> Parágrafo com o estilo acima </P>
</body>
```

2- Estilo dentro de tags

<H1 style=”color:red”> Título vermelho </H1>

H1, H2, H3 {color:blue} Os 3 títulos em azul

3 - Include do CSS

O arquivo css não pode conter nenhuma tag HTML.

```
---estilos.css---
h1{     font-family: sans-serif;
        color: #666666;
}

p{
        font-family: cursive;
}
```

```
<html><head>
<link href="estilos.css" type="text/css" rel="stylesheet"/>
<title>Exemplo</title>
</head>
<body>
```

```
<h1>Este cabeçalho foi formatado com uma folha de estilos.</h1>
<p>Este parágrafo também!</p>
</body></html>
```

4 - Posicionando com Estilo

```
<div id=”quadro1” style=”position:absolute; left:300px;top:100px;width:150px;height:130px”>

<P>Conteúdo do quadro1</P>
</div>

<div id=”quadro2” style=”position:absolute; left:300px;top:300px;width:150px;height:130px”>

<P>Conteúdo do quadro2</P>
</div>
```

5 - Posicionamento absoluto com sobreposição de elementos e relativo

```
<html>
  <head>
    <title></title>
  </head>
  <body>
    <p>
        Este parágrafo está na posição normal (relativa.)
    </p>
    <p style="position: absolute; top: 10px; left: 30px">
        Este parágrafo está numa posição absoluta.
    </p>
    <p style="position: relative">
        Este parágrafo está numa posição relativa.
    </p>
  </body>
</html>
```

6 - Cursor com Rastro

```
<html>
<head>
<script type="text/javascript">
//Cursor com rastro
var i=0
var txt=new Array("rastoP","rastoH","rastoP","rasto-","rasto5")

function cursor(intervalo, event)
{
pos=i*8+15
if (intervalo=='primeiro')
```

```
        i=0
if(i==0)
{
      xpos=event.clientX
      ypos=event.clientY
      document.getElementById(txt[i]).style.visibility="visible"
      document.getElementById(txt[i]).style.position="absolute"
      document.getElementById(txt[i]).style.left=xpos+15
      document.getElementById(txt[i]).style.top=ypos+15
}
else
{
      document.getElementById(txt[i]).style.visibility="visible"
      document.getElementById(txt[i]).style.position="absolute"
      document.getElementById(txt[i]).style.left=xpos+pos
      document.getElementById(txt[i]).style.top=ypos+pos
}
i++
if (i==txt.length)
      i-=1
setTimeout("cursor('seguinte')",10)
}
</script>
<title></title>
</head>
<body onmousemove="cursor('primeiro', event)">
   <h1>Mova o cursor sobre este exemplo</h1>
<span id="rastoP" style="VISIBILITY: hidden">P</span>
   <span id="rastoH" style="VISIBILITY: hidden">H</span>
   <span id="rastoP" style="VISIBILITY: hidden">P</span>
   <span id="rasto-" style="VISIBILITY: hidden">-</span>
   <span id="rasto5" style="VISIBILITY: hidden">5</span>
   <br/><br/><br/><br/>
  </body>
</html>
```

7 - Botões com Estilo

```
<head>
<LINK REL=stylesheet HREF="estilobotoes.css" TYPE="text/css">
</head>

<body>
  <form name="form1" method="post" id="form1" action="">
   <div align="center">
   <input name="btnMenu" type="submit" class="btnMenu" value="Menu" id="btnMenu"
onClick="window.open('menu.php');window.close()">
   <input name="btnMy" type="submit" class="btnMenu" value="MySQL" id="btnMy"
onClick="form1.action='mylogin.php';form1.submit()">
```

```
   <input name="btnPg" type="submit" class="btnMenu" value="PostgreSQL" id="btnPg"
onClick="form1.action='pglogin.php';form1.submit()">
   </div>
</form>

</body>

------------estilobotoes.css-----------------
.btnMenu {
background-color: greenyellow ;
color: blue;
font-size: 8pt;
font-family: verdana;
cursor: hand;
}
------------estilobotoes.css-----------------
```

8 - Gerando botões com estilo CSS

http://downloads.scriptfacil.com/site.script.codigo.ver.php?url=http://www.scriptfacil.com/scripts/311/gera.html

9 - Menus CSS

Colaboração: Gian Franco B. Barcellini para a lista
Dicas-L (http://www.dicas-l.unicamp.br/)

Com CSS (Cascading Style Sheets), se dá para fazer muita coisa.
A complexidade das soluções desejadas vão também do trivial
ao extremamente complexo.

Para quem não tem tempo de estudar em profundidade as técnicas
de confecção de páginas com CSS e deseja acrescentar menus bonitos
aos seus sites, nada melhor do que visitar o
Free Menu Designs (http://e-lusion.com/design/menu/), onde
estão disponíveis para download 9 tipos de menus.

Muito fácil de usar e adaptar.

BORDAS

border-color: red

border-width: thin(medium, thick, lenght)

border-width: 6px (pt, pc, cm, em, mm, %)

border-top-width (top, right, bottom, left)

border-top-color

border-top border-style: none (hidden, dotted, dashed, solid, double, groove, ridge, inset, border-right outset)

border-bottom

border-left

border (todas as 4 bordas)

Através de um arquivo externo

```
<html>
<head>
<link rel=STYLESHEET href="styles/stylesheets.css" type="text/css">
<title>...</title>
</head>
<body>...
```

Diretamente na página:

```
<html><head>
<style type="text/css">
<!--
P { font-size: 10pt; font-family: "Verdana, Arial, Sans-Serif"; color: #000066 }
H1 { font-size: 16pt; font-family: "Impact, Arial, Sans-Serif"; color: #990000 }
-->
</style>
...</head>
<body>...
```

Exemplo:

```
<html><head>
<style type="text/css">
body {background-color: yellow}
h1 {background-color: #00ff00}
h2 {background-color: transparent}
p {background-color: rgb(250,0,255)}
</style></head><body>

<h1>Este é o header 1</h1>
<h2> Este é o  header 2</h2>
```

```
<p> Este é um parágrafo</p>
</body></html>
```

Imagem de fundo:

```
<head>
<style type="text/css">
body
{
background-image:
url('bgdesert.jpg')
}
</style></head>
```

Setando a forma de um elemento

```
<head>
<style type="text/css">
img
{
position:absolute;
clip:rect(0px 50px 200px 0px)
}
</style></head><body>
<p>The clip property is here cutting an image:</p>
<p><img border="0" src="bookasp20.gif" width="120" height="151"></p>
</body>
```

Imagem na frente de texto:

```
<head><style type="text/css">
img.x
{
position:absolute;
left:0px;
top:0px;
z-index:1
}
```

```
</style></head><body>
<h1>This is a Heading</h1>
<img class="x" src="bulbon.gif" width="100" height="180">
<p>Default z-index is 0. Z-index 1 has higher priority.</p>
</body>
```

Como Atributo de Tags:

```
<div style="margin-left: 0.5in; font-size: 10pt">
Este deve ser um bloco indentado com algum
<span style="font-weight: bold; background: #FFFF00"> texto selecionado</span>
dentro dele
</div>
```

Podemos criar arquivos que reescrevem as TAGs originais com CSS:

Criar um arquivo chamado “template.ini.css”:

```
BODY, TD {
        background-color: #E8FFFF;
        color: Green;
        font-family: "Times New Roman", "Bookman Old Style", serif;
        font-size: 12px;
        text-align: left;
}
H2 {
        font-size: 14pt;
}
H3 {
        font-size: 15pt;
        font-style: italic;
        font-family: sans-serif;
}
H4 {
   font-family: sans-serif;
   font-size: small;
}
A {
        color: Navy;
}
A:Hover {
```

```
        color: Lime;
}
HR {
        height: 1pt;
}
```

Então fazemos a inclusão no arquivo HTML ou outro a ser usado, como explicado acima.

Ao utilizar uma dessas TAGs reescritas elas assumirão o comportamento adotado no CSS.

4-XML

XML Tutorial
http://www.w3schools.com/xml/

Introduction To SimpleXML With PHP
http://www.phpro.org/tutorials/Introduction-To-SimpleXML-With-PHP.html

Exemplo com o Joomla :

joomla.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<extension version="3.4" type="file" method="upgrade">

	<name>files_joomla</name>

	<author>Joomla! Project</author>

	<authorEmail>admin@joomla.org</authorEmail>

	<authorUrl>www.joomla.org</authorUrl>

	<copyright>(C) 2005 - 2015 Open Source Matters. All rights reserved</copyright>

	<license>GNU General Public License version 2 or later; see LICENSE.txt</license>

	<version>3.4.1</version>

	<creationDate>March 2015</creationDate>

	<description>FILES_JOOMLA_XML_DESCRIPTION</description>


	<scriptfile>administrator/components/com_admin/script.php</scriptfile>


	<update>

		<schemas>

			<schemapath
type="mysql">administrator/components/com_admin/sql/updates/mysql</schemapath>

			<schemapath
type="sqlsrv">administrator/components/com_admin/sql/updates/sqlsrv</schemapath>

			<schemapath
type="sqlazure">administrator/components/com_admin/sql/updates/sqlazure</schemapath>

			<schemapath
```

type="postgresql">administrator/components/com_admin/sql/updates/postgresql</schemapath>

</schemas>

</update>

<fileset>

<files>

<folder>administrator</folder>

<folder>bin</folder>

<folder>cache</folder>

<folder>cli</folder>

<folder>components</folder>

<folder>images</folder>

<folder>includes</folder>

<folder>language</folder>

<folder>layouts</folder>

<folder>libraries</folder>

<folder>logs</folder>

<folder>media</folder>

<folder>modules</folder>

<folder>plugins</folder>

<folder>templates</folder>

<folder>tmp</folder>

<file>htaccess.txt</file>

<file>web.config.txt</file>

<file>LICENSE.txt</file>

<file>README.txt</file>

<file>index.php</file>

```
        </files>
    </fileset>

    <updateservers>
        <server type="collection">http://update.joomla.org/core/list.xml</server>
        <server type="collection">http://update.joomla.org/jed/list.xml</server>
    </updateservers>
</extension>
```

6-AJAX

Introdução ao Ajax para Aplicações Web PHP

https://netbeans.org/kb/docs/php/ajax-quickstart_pt_BR.html

AJAX Tutorial

http://www.w3schools.com/ajax/default.ASP

Tutorial básico sobre AJAX

http://www.htmlstaff.org/ver.php?id=7873

7-jQuery

Por onde começar com a jQuery

Que tal fazer um curso?

Este guia é uma introdução à biblioteca jQuery. Ter um conhecimento sobre javascript e modelo de objeto de documento (DOM) são requisitos necessários. Ele começa do básico e tenta explicar os detalhes quando necessário. Ele abordará um exemplo simples alô mundo, seletores e eventos básicos, AJAX, efeitos e o desenvolvimento de plugins.
Este guia não contém exemplos "prontos para clicar". A intenção de prover apenas os "códigos para copiar" é um convite para que você mesmo faça os testes. Copie um exemplo, veja o que ele faz, e modifique-o.

Conteúdo

Configuração

Para começar, nós precisamos de uma cópia da bliblioteca jQuery. Ainda que a versão mais recente possa ser encontrada aqui, este guia fornece um pacote básico que você poderá baixar.

Kit para começar com o jQuery

Faça o download deste arquivo e extraia o seu conteúdo. Abra o starterkit.html e o custom.js com o seu editor preferido e o starterkit.html com o navegador.

Agora nós temos tudo o que precisamos para inciar o notório exemplo do "Alô mundo".

Links interessantes para este capítulo:

- Kit para começar
- Downloads do jQuery

Olá jQuery

Como quase tudo o que fazemos quando estamos utilizando o jQuery lê ou manipula um modelo de objeto de documento (DOM), precisamos nos certificar que começamos adicionado eventos etc tão logo o DOM esteja pronto.

Para fazer isso, nós registramos o evento *ready (pronto)* para o documento.

```
$(document).ready(function() {
        // faça alguma coisa quando o DOM estiver pronto
});
```

Colocar um *alert* nesta função não faz muito sentido, pois o *alert* não requer que o DOM esteja carregado. Então vamos tentar algo mais sofisticado: Mostrar um *alert* quando clicarmos o link.

```
$(document).ready(function() {
        $("a").click(function() {
                alert("Olá mundo!");
        });
});
```

Dessa forma o *alert* será exibido assim que você clicar no link.

Vamos dar uma olhada no que estamos fazendo: `$("a")` é um seletor do jQuery, neste caso, ele seleciona todos os elementos a. O `$` por si só é um *alias* para a "classe" jQuery, por outro lado o `$()` constrói um novo objeto jQuery. A função `click()` que chamamos depois é um método do objeto jQuery. Ele liga o evento clique a todos os elementos selecionados (neste caso, um único elemento a) e executa a função fornecida quando o evento ocorre.
Isto é similar ao seguinte código:

```
<a href="#" onclick="alert('Olá mundo')">Link</a>
```

A diferença é bem óbvia: Nós não precisamos escrever onclick para todo elemento. Nós temos uma separação clara de estrutura (HTML) e comportamento (JS), assim como separamos estrutura e formatação utilizando CSS.

Com isso em mente, exploramos seletores e eventos um pouco mais a fundo.

Links interessantes para este capítulo:

- Base do jQuery
- Expressões do jQuery
- Eventos Básicos do jQuery

Me encontre: Usando seletores e eventos

O jQuery provê duas maneiras de selecionar elementos. A primeira utiliza uma combinação de seletores CSS e XPath passados como uma string para o construtor do jQuery (ex. `$("div > ul a")`). A segunda utiliza vários métodos do objeto jQuery. Ambas podem ser combinadas.

Para testar alguns desses seletores, vamos selecionar e modificar a primeira lista ordenada no nosso kit para começar.

Primeiramente queremos selecionar a própria lista. A lista tem um ID "listaOrdenada". No javascript clássico, você pode selecioná-la usando `document.getElementById("listaOrdenada")`. Com o jQuery, nós fazemos isso assim:

```
$(document).ready(function() {
        $("#listaOrdenada").addClass("vermelho");
});
```

O kit para começar fornece uma folha de estilos com a classe "vermelho" que simplesmente adiciona um fundo vermelho. No entanto, quando você recarrega a página no seu navegador, você verá que a primeira lista tem o fundo vermelho. A segunda lista

permanece inalterada.

Agora vamos adicionar mais classes para os elementos filhos desta lista.

```
$(document).ready(function() {
        $("#listaOrdenada > li").addClass("azul");
});
```

Isto seleciona todos os `lis` filhos do elemento com id listaOrdenada e adiciona a classe "azul".

Agora alguma coisa mais sofisticada: Nós queremos adicionar e remover a classe quando o usuário passar o mouse sobre o elemento li, mas somente no último elemento da lista.

```
$(document).ready(function() {
        $("#listaOrdenada li:last").hover(function() {
                $(this).addClass("verde");
        }, function() {
                $(this).removeClass("verde");
        });
});
```

Existem diversos outros seletores similires à sintaxe CSS e XPath. Mais exemplos e a lista de todas as expressões disponíveis podem ser encontrados aqui.

Para todo evento *onxxx* disponível, como onclick, onchange, onsubmit, existem um equivalente no jQuery. Alguns outros eventos, como ready e hover, são métodos convenientes para algumas tarefas.

Você pode encontrar uma lista completa com todos os eventos suportados no Visual jQuery, na seção de *Events*.

Com estes seletores e eventos você já pode fazer muita coisa, mas tem muito mais.

```
$(document).ready(function() {
        $("#listaOrdenada").find("li").each(function(i) {
            $(this).html( $(this).html() + " BAM! " + i );
        });
    });
```

O `find()` permite que você faça uma pesquisa mais a fundo nos descendentes dos elementos já selecionados, apesar de `$("#listaOrdenada).find("li")` ser praticamente o mesmo que `$("#listaOrdenada li")`. O `each()` faz a iteração sobre cada elemento e permite um processamento mais profundo. A maioria dos métodos, como o `addClass()`, utiliza o `each()` internamente. Neste exemplo, o `html()` é utilizado para recuperar o texto de cada elemento li, adicionar algum texto a ele e definí-lo como o texto do elemento.

Uma outra tarefa que você frequentemente terá que lidar é chamar métodos em elementos DOM que não são suportados pelo jQuery. Pense em um formulário que você gostaria de resetar depois que enviou com sucesso via AJAX.

```
$(document).ready(function() {
        // use isto para resetar um único formulário
        $("#reset").click(function() {
                $("#form")[0].reset();
        });
});
```

Este código seleciona o elemento com o ID "form" e chama o `reset()` no primeiro elemento selecionado. Caso exista mais de um form, você pode fazer dessa maneira:

```
$(document).ready(function() {
        // use isto para resetar todos os formulários de uma só vez
        $("#reset").click(function() {
                $("form").each(function() {
                        this.reset();
                });
        });
});
```

Isto deve selecionar todos os formulários no seu documento, fazer a iteração sobre eles e chamar o `reset()` para cada um.

Outro problema que você pode encontrar é não selecionar alguns elementos. O jQuery provê o `filter()` e o `not()` para isto. Enquanto o `filter()` reduz a seleção para os elementos que atendem à expressão de filtro, o `not()` faz exatamente o contrário, removendo todos os elementos que atendem a expressão. Imagine uma lista desordenada onde você quer selecionar todos os elementos li que não possuam um filho ul.

```
$(document).ready(function() {
        $("li").not("[ul]").css("border", "1px solid black");
});
```

Isto seleciona todos os elementos li e remove todos os elementos da seleção que possuam um elemento ul como filho. Sendo assim todos os elementos li ficarão com uma borda, com exceção daqueles que possuam um filho ul. A sintaxe `[expressão]` é vinda do XPath e pode ser utilizada para filtrar elementos e atributos filhos. Talvez você queira selecionar todas as âncoras que possuam o attributo name:

```
$(document).ready(function() {
        $("a[@name]").background("#eee");
});
```

Isto adiciona uma cor de fundo para todos os elementos âncora com o atributo name.

Mais comum que selecionar as âncoras pelo nome, você pode precisar selecionar as âncoras pelo atributo "href". Isto pode ser um problema uma vez que os navegadores se comportam inconsistentemente quando retornam o que eles pensam que o valor do "href" é. Para selecionar apenas uma parte do value, podemos utilizar o seletor contém "*=" ao invés do igual ("="):

```
$(document).ready(function() {
        $("a[@href*=/content/gallery]").click(function() {
                // faça alguma coisa com todos os links que apontam para algum
lugar em /content/gallery
        });
});
```

Até agora, usamos todos os seletores para selecionar os filhos ou filtrar a seleção atual. Existem situações onde você irá precisar selecionar os anteriores ou próximos elementos, conhecidos como *siblings* (filhos do mesmo pai). Pense em uma página de um FAQ, onde todas as respostas estão escondidas em um primeiro momento e são exibidas quando a questão é clicada. O código do jQuery para isso:

```
$(document).ready(function() {
        $('#faq').find('dd').hide().end().find('dt').click(function() {
         var resposta = $(this).next();
         if (resposta.is(':visible')) {
             resposta.slideUp();
         } else {
             resposta.slideDown();
         }
     });
});
```

Aqui estamos usando um pouco de encadeamento para reduzir o tamanho do código e ganhar performance, uma vez que '#faq' é selecionada apenas uma única vez. Usando o `end()`, o primeiro `find()` é desfeito, assim podemos começar a procurar com o próximo `find()` no nosso elemento #faq, ao invés de procurar no filho dd.

Com o acionamento do evento click, a função passada ao método `click()`, utilizamos $(this).next() para encontrar o próximo *sibling* a partir do dt atual. Isto nos permite selecionar rapidamente a resposta seguinte à questão clicada.

Além dos *siblings*, você também pode selecionar os elementos pais (também conhecidos como ancestrais para os mais familiarizados com o XPath). Talvez você pode querer realçar o parágrafo que é o pai do link que o usuário passar o mouse. Tente isso:

```
$(document).ready(function() {
        $("a").hover(function() {
            $(this).parents("p").addClass("realcar");
        }, function() {
            $(this).parents("p").removeClass("realcar");
        });
    });
```

Para todos os elementos âncoras que o usuário passar o mouse, o parágrafo pai é procurado e a classe "realcar" é adicionada e removida.

Vamos voltar um passo atrás de continuarmos: o jQuery faz com que o código fique menor e tornando-o mais fácil de ler e dar manutenção. O código seguinte é um atalho para a notação $(document).ready(callback):

```
$(function() {
        // código a ser executado quando DOM está pronto
});
```

Aplicado ao exemplo do Alô Mundo!, ficaria assim:

```
$(function() {
        $("a").click(function() {
                alert("Alô Mundo!");
        });
});
```

Agora com o básico em mãos, queremos explorar outros aspectos, começando com o AJAX.

Links interessantes para este capítulo:

- Documentação da API do jQuery

- Visual jQuery - Uma documentação categorizada e navegável da API do jQuery
- Expressões do jQuery: CSS
- Expressões do jQuery: XPath
- Expressões do jQuery: Customizadas
- Eventos especiais no jQuery
- jQuery DOM Traversing

Me avalie: Usando AJAX

Nesta parte nós escreveremos uma pequena aplicação AJAX que permite que o usuário avalie alguma coisa, assim como é feito no youtube.com.

Precisamos de um pouco de código do servidor para isso. Meu exemplo utiliza um arquivo php que lê o parâmetro "avaliacao" e retorna o número de avaliações e a média de avaliação. Dê uma olhada no rate.php para o código server-side.

Nós não queremos que este exemplo funcione sem AJAX, mesmo que isto seja possível, então nós geramos a marcação necessária com o jQuery e a adicionamos à div com o ID "avaliacao".

```
$(document).ready(function() {
        // gera a marcação
        var ratingMarkup = ["Por favor avalie: "];
        for(var i=1; i <= 5; i++) {
                ratingMarkup[ratingMarkup.length] = "<a href='#'>" + i +
"</a> ";
        }
        // adiciona a marcação ao container e aplica o acionador de click às
âncoras
        $
("#avaliacao").append( ratingMarkup.join('') ).container.find("a").click(functio
n(e) {
                e.preventDefault();
                // envia a requisição
                $.post("rate.php", {avaliacao: $(this).html()}, function(xml) {
                        // formata o resultado
                        var resultado = [
                                "Obrigado por avaliar, média atual: ",
                                $("media", xml).text(),
                                ", número de votos: ",
                                $("contador", xml).text()
                        ];
                        // saída do resultado
                        $("#avaliacao").html(resultado.join(''));
                } );
        });
});
```

Este pedaço de código gera cinco elementos âncoras adicionando-os ao elemento container com o id "avaliacao". Depois disso, para cada âncora no container, um observador do click é adicionado. Quando a âncora é clicada, uma requisição post é enviada para o rate.php com o conteúdo da âncora como parâmetro. O resultado retornado como um XML é então adicionado ao container, substituindo as âncoras.

Se você não possui um servidor web com o PHP instalado em mãos, você pode dar uma olhada no exemplo on-line.

Para ver um exemplo muito bom de um sistema de avaliação que funciona até mesmo sem javascript, visite softonic.de e clique em "Kurz bewerten!"

Mais documentação sobre os métodos AJAX do jQuery pode ser encontrada aqui ou no Visual jQuery na categoria AJAX.

Um problema muito comum encontrado quando se carrega um conteúdo por AJAX é o seguinte: Quando se adiciona controladores de evento ao seu documento que deveriam ser aplicados também ao conteúdo carregado, você deve aplicar estes controladores depois que o conteúdo é carregado. Para prevenir a código duplicado, você pode delegar a uma função. Exemplo:

```
// Vamos usar o atalho
$(function() {
        var addClickHandlers = function() {
                $("a.cliqueParaCarregarConteudo").click(function() {
                        $("#target").load(this.href, addClickHandlers);
                });
        };
        addClickHandlers();
});
```

Agora a função addClickHandlers é aplicada uma vez quando o DOM está pronto e depois toda vez que o usuário clicar um link com a classe `cliqueParaCarregarConteudo` e o conteúdo tiver terminado de ser carregado.

Por favor observe que a função "addClickHandlers" é definida como uma variável local, ao invés de uma função global (como `function addClickHandlers() {...}`). Procure seguir esta prática para prevenir conflito com outras variáveis ou funções globais.

Com AJAX podemos explorar muita coisa "Web 2.0". Mas até agora estamos devendo efeitos bacanas.

Links interessantes para este capítulo:

- Módulo AJAX do jQuery
- API do jQuery: Contém descrição e exemplos para adicionar e todos os outros métodos do jQuery
- ThickBox: Um plugin para o jQuery que o utiliza para aprimorar o famoso lightbox

Me anime: Usando Efeitos

Animações simples com o jQuery podem ser feitas com o `show()` e o `hide()`

```
$(document).ready(function() {
        $("a").toggle(function() {
                $(".algumacoisa").hide('slow');
        }, function() {
                $(".algumacoisa").show('fast');
        });
});
```

Você pode criar qualquer combinação das animações com o `animate()` como por exemplo um *slide* com fade:

```
$(document).ready(function() {
        $("a").toggle(function() {
                $(".algumacoisa").animate({
```

```
                        height: 'hide',
                        opacity: 'hide'
                }, 'slow');
        }, function() {
                $(".algumacoisa").animate({
                        height: 'show',
                        opacity: 'show'
                }, 'slow');
        });
});
```

Efeitos muito mais sofisticados podem ser feitos com a coleção de plugins Interface. No site você encontrará demonstrações e a documentação.

Além da coleção Interface, que está no topo da lista de plugins do jQuery, existem muitos outros. O próximo capítulo lhe mostrará como usar o plugin *tablesorter*.

Links interessantes para este capítulo:

- Módulo de efeitos do jQuery
- Plugin Interface

Me ordene: Usando o plugin tablesorter

O plugin tablesorter permite que você ordene as tabelas no lado do cliente. Você inclui o jQuery e o pluguin e informa ao pluguin quais tabelas deseja ordernar.

Para experimentar este exemplo, adicione esta linha ao starterkit.html (abaixo da inclusão do jquery):

```
<script src="lib/jquery.tablesorter.js" type="text/javascript"></script>
```

Depois de incluir o plugin, você poderá chamá-lo assim:

```
$(document).ready(function() {
        $("#large").tableSorter();
});
```

Tente clicar nos cabeçalhos da tabela e veja se ela fica ordenada em ordem ascendente no primeiro clique e descendente no segundo.

A tabela deve utilizar algum realce nas linhas, podemos adicioná-los passando algumas opções:

```
$(document).ready(function() {
        $("#large").tableSorter({
                stripingRowClass: ['odd','even'],       // Nome da classe a ser
utilizada para a divisão das linhas como um array.
                stripRowsOnStartUp: true                // Divide as linhas
quando o tableSorter iniciar.
        });
});
```

Existem mais exemplos e documentação sobre as opções disponíveis no site do tablesorter.

A maioria dos plugins podem ser utilizados como este: Inclua o arquivo do pluguin e chame o método do plugin em alguns elementos, passando os parâmetros opcionais para

customizar o plugin.

Uma lista atualizada dos plugins disponíveis pode ser encontrada no site do jQuery.

Quando você utilizar o jQuery com mais frequência, você perceberá que será melhor empacotar o seu próprio código como um plugin, seja para o reuso para você mesmo ou para a sua empresa, ou para compartilhá-lo com a comunidade. O próximo capítulo dá algumas dicas sobre como estruturar um plugin.

Links interessantes para este capítulo:

- Plugins para o jQuery
- Tablesorter Plugin

Me plugue: Escrevendo seus próprios plugins

Escrever seus próprios plugins para o jQuery é muito fácil. Se você seguir as regras abaixo, será fácil para outros integrarem o seu plugin também.

1. Encontre um nome para o seu plugin, vamos chamar o nosso exemplo "foobar".
2. Crie um arquivo chamado jquery.[nomedoseuplugin].js, ex. jquery.foobar.js
3. Crie um ou mais métodos no plugin extendendo o objeto jQuery, ex.:

```
jQuery.fn.foobar = function() {
        // faça alguma coisa
};
```

4. Opcional: Crie um objeto com funções de ajuda, ex.:

```
jQuery.fooBar = {
        height: 5,
        calculateBar = function() { ... },
        checkDependencies = function() { ... }
};
```

   Você poderá então chamar essa função de ajuda pelo seu plugin:

```
jQuery.fn.foobar = function() {
        // faça alguma coisa
        jQuery.foobar.checkDependencies(value);
        // faça alguma outra coisa
};
```

5. Opcional: Crie configurações padrões que possam ser alteradas pelo usuário, ex.:

```
jQuery.fn.foobar = function(options) {
        var settings = {
                value: 5,
                name: "pete",
                bar: 655
        };
        if(options) {
                jQuery.extend(settings, options);
        }
};
```

   Você pode então chamar o plugin sem opções, usando as configurações padrões:

```
$("...").foobar();
```

Ou com algumas opções:

```
$("...").foobar({
        value: 123,
        bar: 9
});
```

Se você lançar o seu plugin, você deverá prover alguns exemplos e uma documentação. Existem diversos plugins disponíveis como ótimas referências.

Agora você já deve ter uma idéia básica de como escrever um plugin. Vamos utilizar este conhecimento e escrever nosso próprio plugin.

Uma coisa que muita gente, tentando manipular formulários com o jQuery, pede, é marcar e desmarcar radio buttons ou checkboxes. Eles acabam com um código como este:

```
$("input[@type='checkbox']").each(function() {
        this.checked = true;
        // ou, para desmarcar
        this.checked = false;
        // ou, para inverter
        this.checked = !this.checked;
});
```

Sempre que você possuir um *each* no seu código, você pode querer reescrever isto com um plugin, quase sem alterações:

```
$.fn.check = function() {
        return this.each(function() {
                this.checked = true;
        });
};
```

Este plugin agora pode ser utilizado:

```
$("input[@type='checkbox']").check();
```

Agova você pode escrever plugins para as duas funções uncheck() e toggleCheck() também. Mas ao contrário nós estendemos nosso plugin para aceitar algumas opções.

```
$.fn.check = function(modo) {
        var modo = modo || 'on'; // se modo não está definido, use 'on' como
padrão
        return this.each(function() {
                switch(modo) {
                case 'on':
                        this.checked = true;
                        break;
                case 'off':
                        this.checked = false;
                        break;
                case 'toggle':
                        this.checked = !this.checked;
                        break;
                }
        });
};
```

Definindo um valor padrão para as opções, permite que o usuário omita a opção ou passe

"on", "off", e "toggle", ex.:

```
$("input[@type='checkbox']").check();
$("input[@type='checkbox']").check('on');
$("input[@type='checkbox']").check('off');
$("input[@type='checkbox']").check('toggle');
```

Com mais de uma configuração opcional, este método torna-se complicado, pois o usuário deverá passar valores nulos se quiser omitir o primeiro parâmetro e usar somente o segundo.

O uso do *tablesorter* no último capítulo demonstra o uso de um objeto literal também resolve este problema. O usuário omite todos os parâmetros ou passa um objeto com um par chave/valor para cada configuração que deseja sobrescrever.

Como um exercício, você poderia tentar reescrever o código do quarto capítulo como um plugin. O esqueleto do plugin deve ser similar a isto:

```
$.fn.rateMe = function(options) {
        var container = this; // ao invés de selecionar um container estático
com  $("#rating"), nós agora utilizamos o contexto do jQuery

        var settings = {
                url: "rate.php"
                // coloque mais padrões aqui
                // lembre-se de colocar uma vírgula (",") depois de cada par,
mas não depois do último!
        };

        if(options) { // verifica se as opções estão presentes antes de estender
as configurações
                $.extend(settings, options);
        }

        // ...
        // resto do código
        // ...

        return this; // se possível, retorne "this" para não quebrar a corrente
});
```

Próximos passos

Se você tem interesse em desenvolver mais em javascript, você deve dar uma olhada numa extensão do Firefox chamada FireBug. Ela provê um console (ótimo para substituir os alerts), um debugger e outras coisas úteis para o seu desenvolvimento diário em javascript.

Se você tem problemas que não consegue resolver, idéias que gostaria de compartilhar ou apenas necessidade de expressar sua opinião sobre o jQuery, sinta-se à vontade para postar na lista de discussão do jQuery.

Para qualquer coisa relacionada a este guia, envie um email para mim ou poste um comentário no meu blog.

O que restou...

Muito obrigado a John Resig por esta ótima biblioteca! Obrigado à comunidade do jQuery

por proporcionar ao John café e tudo mais!

Tradução para o português do Brasil por Carlos Pires (carlos.pires arroba 2km.com.br) - Última atualização: 10-04-2007

Introdução à jQuery

Tradução livre do original em inglês
http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery

Este guia é uma introdução à biblioteca jQuery. Um conhecimento básico de JavaScript e do DOM (document object model) é necessário para uma melhor compreensão da jQuery. Este tutorial cobre um simples exemplo tipo hello world, seletor e eventos básicos, AJAX, FX e uso e autoria de plugins.

Este guia não contém nenhum exemplo do tipo "clique me". A intenção é oferecer somente código "copy me" para convidar você para testar isto por si mesmo. Copie um exemplo, veja o que ele faz e modifique até entender bem.

Conteúdo

1 Configuração

Para começar precisamos de uma cópia da biblioteca jQuery, que pode ser obtida aqui:
http://docs.jquery.com/Downloading_jQuery

Na página acima tem também uns links que podem ser usados sem baixar (CDN).

Aqui o link direto para a versão de hoje (18/07/2012), pacote mínimo:
http://code.jquery.com/jquery-1.7.2.min.js

Aqui o pacote completo:
http://code.jquery.com/jquery-1.7.2.js

Também precisaremos do jQuery Starterkit:
http://docs.jquery.com/images/c/c7/Jquery-starterkit.zip

Que já traz um exemplo com HTML e CSS para trabalharmos.

Siga os passos:

-Baixe o starterkit e descompacte.
-Após baixar a versão mínima da jQuery, renomeie para jquery.js.
-Copie jquery.js para a pasta starterkit

-Agora edite os scripts starterkit.html e o custom.js para trabalharmos
-Também abra o starterkit.html mas no navegador para visualizar as alterações

Agora temos tudo de que precisamos para o nosso hello world.
Observe que o script starterkit.html inclue a jquery.js.

Precisamos estar certos de que o DOM está pronto para receber nossos eventos. Para isso sempre iniciamos o código com a função abaixo (veja que o script custom.js):

```
jQuery(document).ready(function() {
        // do something here
});
```

Com ela nós registramos u ready event para o documento.

Adicionar um alert na função não faz muito sentido tendo em vista que o alert não precisa do DOM para ser carregado. Então faremos algo mais sofisticado: mostrar o alert quando clicar no link.

Agora atualize o manupulador do $(document).ready no arquivo custom.js para que fique assim:

```
$(document).ready(function() {
  $("a").click(function() {
    alert("Olá mundo!");
  });
});
```

Veja a página quickstartkit.html e atualize. Depois clique em algum link. Em qualquer link o alert aparece, ou seja, todos os links receberam o alert.

Entendendo:

$("a") – este é um seletor. No nosso caso ele seleciona todos os elementos **a**.

$ - em si é um alias para a classe jQuery
$() - constrói um novo objeto jQuery.
click() -é uma função chamada depois é um método do objeto jQuery.
        Ele faz a ligação do evento click a todos os elementos
        selecionados (neste caso apenas o elemento âncora) e executa
        a função provida quando o evento ocorrer.

Isto é semelhante ao seguinte código:

```
<a href="" onclick="alert('Olá Mundo')">Link</a>
```

A diferença é clara: não precisamos escrever um evento onClick para cada elemento.

Temos uma clara separação entre a estrutura (HTML) e o comportamento (JavaScript), justamente como separamos estrutura e apresentação usando o CSS.

Veja as APIs da jQuery

http://api.jquery.com/category/core/

http://api.jquery.com/category/traversing/

Usando Seletores e Eventos
jQuery oferece duas aproximações para selecionar elementos. A primeira usa uma combinação de seletores do CSS e Xpath passados como uma string para o construtor da jQuery (p.e.: `$("div > ul a")`). A segunda usa vários métodos do objeto jQuery. Ambas as aproximações podem ser combinadas.

Para testar alguns destes seletores nós selecionamos e modificamos a primeira lista ordenada do starterkit.html.

Para começar iremos selecionar a lista em si. A lista tem um ID "orderedlist". No JavaScript clássico podemos selecionar isso usando `document.getElementById("orderedlist")`. Na jQuery nós fazemos isso assim:

```
$(document).ready(function() {
   $("#orderedlist").addClass("red");
 });
```

Deixe o custom.js com o conteúdo acima e atualize o html para ver o resultado.
Verá que as 3 linhas da lista ficaram com fundo vermelho. Mas a segunda lista não mudou.

Agora vamos mexer com a classe, alterando os elementos filhos da lista. Substitua o conteúdo de custom.js por:

```
$(document).ready(function() {
   $("#orderedlist > li").addClass("blue");
 });
```

Agora verá que as 3 linhas ficaram com a cor da fonte azul.

Agora vamos mudar a cor da fonte para verde quando o usuário passar o ponteiro do mouse sobre a última linha da lista (hover) e quando retirar o ponteiro a cor voltará ao normal. Mude para este código:

```
$(document).ready(function() {
   $("#orderedlist li:last").hover(function() {
     $(this).addClass("green");
   },function(){
     $(this).removeClass("green");
   });
 });
```

Existem muitos outros seletores similares para a sintaxe CSS e XPath. Muitos exemplos e uma lista com todas as expressões está disponível aqui:
http://docs.jquery.com/DOM/Traversing/Selectors

Para cada um dos eventos disponíveis (como onclick, onchange, onsubmit) existe um equivalente jQuery. Uma completa lista com todos os eventos:
http://api.jquery.com/category/events/

Com estes seletores e eventos você pode fazer um monte de coisas e muito mais.

Exemplo:

```
$(document).ready(function() {
   $("#orderedlist").find("li").each(function(i) {
     $(this).append( " BAM! " + i );
   });
 });
```

Entendendo:

find() - permite procurar pelos descendentes dos elementos já selecionados, sendo que `$("#orderedlist").find("li")` é parecido com `$("#orderedlist li").`

`each() - iterage sobre cada elemento e permite futuro processamento. Mais métodos como o addClass() usam o each().`

`append() - neste exemplo é usado para adicionar algum texto a si e configurar este como texto para o final de cada elemento.`

`Outra tarefa que muitas vezes precisamos executar é chamar métodos em elementos DOM que não são abrangidos pelo jQuery. Pense em um formulário que você gostaria de reinicializar depois de apresentado com sucesso via AJAX.`

```
$(document).ready(function() {
   // use this to reset a single form
   $("#reset").click(function() {
     $("form")[0].reset();
   });
 });
```

O zero em ("forms")[0] faz com que resete o primeiro campo. Para resetar o segundo use 1 e assim por diante.

Caso tenha mais de um form use:

```
$(document).ready(function() {
   // use this to reset several forms at once
   $("#reset").click(function() {
     $("form").each(function() {
       this.reset();
     });
   });
 });
```

Isso selecionará todos os forms com seu documento, iteragindo sobre ele e então chamado o reset() para cada um. Notar que em uma função .each() esta refere para o elemento atual. Também note que, desde que a função reset () pertence ao elemento form e não ao objeto jQuery, não podemos simplesmente chamar $("form").reset() para resetar todos os forms da página.

Um desafio adicional é selecionar apenas alguns elementos de um grupo de semelhantes ou idênticos. jQuery fornece filter() e not() para isso. Enquanto filter() reduz a seleção para os elementos que se encaixam na expressão de filtro, not() faz o contrário e remove todos os elementos que se encaixam na expressão. Pense em uma lista desordenada onde você quer selecionar todos os elementos li que não têm filhos ul.

```
$(document).ready(function() {
   $("li").not(":has(ul)").css("border", "1px solid black");
 });
```

Isto seleciona todos oe elementos li que têm um elemento ul como filho e remove todos os elementos da seleção. Portanto todos os elementos li recebem uma borda exceto o que tem um filho ul.

A sintaxe [expressão] é tomada a partir do XPath e pode ser usado para filtrar por atributos. Talvez você queira selecionar todas as âncoras que têm um atributo nome:

```
$(document).ready(function() {
   $("a[name]").css("background", "#eee" );
 });
```

Isto adiciona cor de fundo para todos os elementos âncora que têm um atributo name.

Mais frequentemente do que selecionar as âncoras pelo nome, você pode precisar selecionar as âncoras pelo atributo "href". Este pode ser um problema como navegadores se comportam inconsistentemente quando retornando o que eles acham que o "href" é o valor (Nota: Este problema foi corrigido recentemente na jQuery, disponível em todas as versões depois da 1.1.1). Para corresponder a apenas uma parte do valor, podemos usar o contains select "* =" em vez de um igual ("="):

```
$(document).ready(function() {
   $("a[href*='/content/gallery']").click(function() {
     // do something with all links that point somewhere to /content/gallery
   });
 });
```

Até agora, todos os seletores foram usados para selecionar filhos ou filtrar a seleção atual. Há situações onde você precisa selecionar os elementos anteriores ou seguintes, conhecidos como irmãos. Pense em uma página de FAQ, onde todas as respostas estão primeiro escondidas, e então mostradas quando a questão é clicada. O código jQuery para isso:

```
$(document).ready(function() {
   $('#faq').find('dd').hide().end().find('dt').click(function() {
     $(this).next().slideToggle();
   });
 });
```

Aqui usamos alguns encadeamentos para reduzir o tamanho do código e obter um melhor desempenho, como '# faq' é selecionada somente uma vez. Usando end(), o primeiro find() é desfeito, para que possamos iniciar a pesquisa com o próximo find() no nosso elemento #faq, ao invés dos filhos dd.

Dentro do manipulador click, a função passada para o método click(), utilizamos $(this).next () para encontrar o próximo irmão a partir do dt atual. Isso nos permite selecionar rapidamente a resposta seguinte à questão clicada.

Além de irmãos, você também pode selecionar os elementos pais (também conhecidos como ancestrais para os mais familiarizados com o XPath). Talvez você queira destacar o parágrafo que é o pai do link que o usuário passa o ponteiro do mouse (hover). Tente isto:

```
$(document).ready(function(){
   $("a").hover(function(){
     $(this).parents("p").addClass("highlight");
   },function(){
     $(this).parents("p").removeClass("highlight");
   });
 });
```

Para todos os elementos âncoras que pairavam o mouse, o parágrafo pai é procurado e uma classe de "destaque" adicionada e removida.

Deixa ir um passo para trás antes de continuar: jQuery gosta muito de criar um código mais curto e, portanto, mais fácil de ler e manter. O seguinte é um atalho para o $ (document) pronto notação (callback).:

```
$(function() {
   // code to execute when the DOM is ready
 });
```

Aplicado para o exemplo Hello World teremos:

```
$(function() {
   $("a").click(function() {
     alert("Hello world!");
   });
 });
```

Rate me (Dê sua nota): Usando AJAX

Nesta parte, vou escrever um aplicativo pequeno com AJAX, que permite ao usuário votar em alguma coisa, assim como é feito no youtube.com.

Nós precisamos de algum código do servidor para isso. Meu exemplo usa um arquivo php que lê o "rating" parâmetro e retorna o número de votos e a classificação média. Dê uma olhada no rate.php para o código do lado do servidor.

Nós não queremos que este exemplo funcione sem AJAX, embora possa ser possível, portanto, gerar a marcação necessária com jQuery e anexá-lo a uma div container com uma identificação de "rating".

```
$(document).ready(function() {
   // generate markup
   $("#rating").append("Favor votar: ");
```

```
  for ( var i = 1; i <= 5; i++ ) {
    $("#rating").append("<a href='#'>" + i + "</a> ");
  }

  // add markup to container and apply click handlers to anchors
  $("#rating a").click(function(e) {
    // stop normal link click
    e.preventDefault();

    // send request
    $.post("rate.php", {rating: $(this).html()}, function(xml) {
      // format and output result
      $("#rating").html(
        "Grato por votar, média atual: " +
        $("average", xml).text() +
        ", número de votos: " +
        $("count", xml).text()
      );
    });
  });
});
```

Este trecho de código gera cinco elementos âncoras e os acrescenta para o elemento de contêiner com o "rating" id. Depois, para cada âncora no interior do recipiente, um manipulador de clique é adicionado. Quando a âncora é clicado, um pedido é enviado para pós rate.php com o conteúdo da âncora como um parâmetro. O resultado devolvido como um XML é então adicionado ao recipiente, substituindo as âncoras.

Se você não tiver um servidor web com PHP instalado em mãos, você pode ver um exemplo online. Para um exemplo muito bom de um sistema de classificação que funciona mesmo sem JavaScript, visite softonic.de e clique em "Kurz bewerten!"

Mais documentação dos métodos de Ajax jQuery pode ser encontrada em Documentação Ajax ou jQuery Visual arquivado em Ajax.

Um problema muito comum encontrado ao carregar conteúdo por Ajax é esta: Ao adicionar manipuladores de eventos para o documento que deve também ser aplicado ao conteúdo carregado, você tem que aplicar esses manipuladores depois que o conteúdo é carregado. Para evitar duplicação de código, você pode delegar para uma função. Exemplo:

```
function addClickHandlers() {
  $("a.remote", this).click(function() {
    $("#target").load(this.href, addClickHandlers);
  });
}
$(document).ready(addClickHandlers);
```

Agora addClickHandlers é chamado uma vez quando o DOM está pronto e, em seguida, toda vez que um usuário clicar em um link com a classe remota e o conteúdo tiver terminado o carregamento.

Observe a consulta $("a.remote", this), este é passado como um contexto: Para o evento document ready, este se refere ao documento, e, por isso, procura o documento inteiro por âncoras com classe remota. Quando addClickHandlers é usado como uma chamada de retorno para load(), este refere-se a um elemento diferente: No exemplo, o elemento

com alvo id. Isso impede que o evento click seja aplicado novamente e novamente para as mesmas ligações, causando um acidente, eventualmente.

Outro problema comum com callbacks são parâmetros. Você especificou o seu retorno, mas precisa passar um parâmetro extra. A maneira mais fácil de conseguir isso é quebrando o retorno dentro de outra função:

```
// get some data
var foobar = ...;

// specify handler, it needs data as a paramter
function handler(data) {
  //...
}

// add click handler and pass foobar!
$('a').click(function() {
  handler(foobar);
});

// if you need the context of the original handler, use apply:
$('a').click(function() {
  handler.apply(this, [foobar]);
});
```

Com este simples Ajax podemos cobrir um monte da "Web 2.0". Agora que vimos alguns exemplos básicos de Ajax, vamos adicionar alguns efeitos simples e animações para a página.

Links interessantes para este capítulo:

- jQuery Ajax Documentation
- jQuery API - Contém descrição e exemplos para adicionar e todos os outros métodos jQuery
- ThickBox – Um plugin jQuery que usa jQuery para enfatizar o famoso lightbox

Anime-me: Usando Efeitos

```
$(document).ready(function(){
   $("a").toggle(function(){
     $(".stuff").hide('slow');
   },function(){
     $(".stuff").show('fast');
   });
});
```

Você pode criar qualquer combinação de animações com animate(), por exemplo, um slide com fade:

```
$(document).ready(function() {
      $("a").toggle(function() {
            $(".stuff").animate({
                  height: 'hide',opacity: 'hide' //remove the ","}, 'slow');
      }, function() {
            $(".stuff").animate({
```

```
                height: 'show',opacity: 'show'}, 'slow');
    });
});
```

Efeitos muito mais sofisticados pode ser conseguidos com o plugin de interface collection (http://interface.eyecon.ro/). O site oferece demos e documentação. Enquanto interface está no topo da lista de plugins do jQuery, existem muitos outros. A parte seguinte mostra como usar o plugin tablesorter.

Links Interessantes para este Capítulo

- jQuery Effects Documentation
- Interface plugin

Sort me: Usando o Plugin Tablesorter

O plugin tablesorter permite a classificação de tabelas no lado do cliente. Você inclui o jQuery, e o plugin e diz ao plugin quais tabelas você deseja classificar.

Para testar este exemplo precisamos fazer o download do plugin tablesorter:
http://tablesorter.com/__jquery.tablesorter.zip

Descompactar e copiar o arquivo jquery.tablesorter.js para o nosso diretório /var/www/starterkit. Então adicionar esta linha para o index.php do nosso starterkit:

```
<script src="jquery.tablesorter.js"></script>
```

Agora pode testar com isso:

```
$(document).ready(function(){
   $("#large").tablesorter();
 });
```

Agora clique nos títulos da grande tabela jQuery Tablekit abaixo.
Verá que suas células serão ordenadas.

Mais exemplos e documentação sobre as opções disponíveis no site:
http://tablesorter.com/docs/

A maioria dos plugins pode ser usados como este: Incluir o arquivo de plugin e chamar o método do plugin em alguns elementos, passando algumas configurações opcionais para personalizar o plugin.

Uma lista atualizada de plugins pode ser encontrada no site:
http://docs.jquery.com/Plugins

Quando você estiver usando jQuery com mais frequência, você pode achar útil empacotar o seu próprio código como um plugin, ou reutilizá-lo para você ou sua empresa, ou compartilhá-lo com a comunidade. O próximo capítulo dá algumas dicas sobre como estruturar um plugin.

Plug me: Criando seus próprios plugins

Escrever seus próprios plugins para o jQuery é muito fácil. Se você seguir as seguintes regras, será fácil para os outros integrarem o seu plugin.

Nomeando plugins
Procure manter jquery no início do nome do plugin e o nome em seguida, por exemplo:

jquery.teste.js

Adicionando um método customizado

Criar um ou mais métodos para o plugin extendendo o objeto jQuery, por exemplo:

```
jQuery.fn.teste = function() {
   // do something
 };
```

Poderemos acessar com:

```
$(...).teste();
```

Configurações default

Criando configurações default, que podem ser alteradas pelo usuário, por exemplo:

```
jQuery.fn.teste = function(options) {
   var settings = jQuery.extend({
     value: 5, name: "pete", bar: 655
   }, options);
 };
```

Podemos então chamar o plugin sem opções, usando as default:

```
$("...").teste();
```

Ou com algumas opções:

```
$("...").teste({ value: 123, bar: 9 });
```

Documentação

Se você liberar seu plugin você pode prover alguns exemplos e documentação também.
Aqui existem muitos plugins disponíveis como uma grande referência:
http://docs.jquery.com/Plugins

Agora você tem uma básica ideia de como se escreve um plugin. Deixe usar este conhecimento para criar um plugin.

Checkbox Plugin

Muitas pessoas, tentando manipular formulários com jQuery, pedem, como marcar e desmarcar botões de rádio ou checkboxes. Eles acabam em um código como este:

```
$(":checkbox").each(function() {
   this.checked = true;
   this.checked = false; // or, to uncheck
   this.checked = !this.checked; // or, to toggle
 });
```

Sempre que você tem um each() em seu código, você pode querer reescrever isso como um plugin, bastante simples:

```
jQuery.fn.check = function() {
   return this.each(function() {
     this.checked = true;
   });
 };
```

Este plugin pode ser usado assim:

```
$(":checkbox").check();
```

Agora você pode escrever plugins para ambos uncheck() e toggleCheck() também. Mas ao invés disso nós estendemos nosso plugin para aceitar algumas opções.

```
jQuery.fn.check = function(mode) {
   // if mode is undefined, use 'on' as default
   var mode = mode || 'on';

   return this.each(function() {
     switch(mode) {
       case 'on':
         this.checked = true;
         break;
       case 'off':
         this.checked = false;
         break;
       case 'toggle':
         this.checked = !this.checked;
         break;
     }
   });
 };
```

Ao fornecer uma opção default, o usuário pode omitir a opção ou passar um "on", "off" ou "toggle", por exemplo.:

```
$(":checkbox").check();
 $(":checkbox").check('on');
 $(":checkbox").check('off');
 $(":checkbox").check('toggle');
```

Configurações Opcionais
Com mais de uma configuração opcional, esta abordagem fica complicado, porque o usuário deve passar valores nulos se quiser omitir o primeiro parâmetro e usar somente o segundo.

A utilização do tablesorter no capítulo anterior demonstrou o uso de um objeto literal para resolver este problema. O usuário pode omitir todos os parâmetros ou passar um objeto com um par chave/valor para cada configuração que deseja substituir.

Para um exercício, você poderia tentar reescrever o código votação da quarta seção como um plugin. O esqueleto do plugin deve ficar assim:

```
jQuery.fn.rateMe = function(options) {
  // instead of selecting a static container with
  // $("#rating"), we now use the jQuery context
  var container = this;

  var settings = jQuery.extend({
    url: "rate.php"
    // put more defaults here
  }, options);

  // ... rest of the code ...

  // if possible, return "this" to not break the chain
  return this;
});
```

Rodar o plugin com:

```
$(...).rateMe({ url: "test.php" });
```

Próximos Passos
Se você planeja desenvolver mais com JavaScript, você deve considerar usar a extensão do Firefox chamada FireBug. Ela fornece um console (bom para substituir alertas), um depurador e outras coisas úteis para o desenvolvimento JavaScript diariamente.

Se você tem problemas que não possa resolver, ideias que você quer compartilhar ou apenas necessidade de expressar sua opinião sobre jQuery, sinta-se livre para postar a lista de discussão jQuery (http://jquery.com/discuss/).

Para qualquer coisa relacionada a este guia poste um comentário no meu blog ou contacte-me diretamente (http://bassistance.de/index.php/2006/09/12/jquery-getting-started-guide/).

Muito obrigado a John Resig por esta grande biblioteca! Graças à comunidade jQuery por fornecer a John bastante café e tudo mais!

Curso em 56 vídeo-aulas
https://www.youtube.com/watch?v=fq5SRwGAd8I&list

8-Flash

Definição

Adobe Flash (antes: Macromedia Flash), ou simplesmente Flash, é um software primariamente de gráfico vetorial - apesar de suportar imagens bitmap e vídeos - utilizado geralmente para a criação de animações interativas que funcionam embutidas num navegador web e também por meio de desktops, celulares, smartphones, tablets e televisores. O produto era desenvolvido e comercializado pela Macromedia, empresa especializada em desenvolver programas que auxiliam o processo de criação de páginas web.

Costuma-se chamar apenas de flash os arquivos gerados pelo Adobe Flash, ou seja, a animação em si. Esses arquivos são de extensão ".swf" (de Shockwave Flash File). Eles podem ser visualizados em uma página web usando um navegador que o suporta (geralmente com plug-in especial) ou através do Flash Player, que é um leve aplicativo somente-leitura distribuído gratuitamente pela Adobe. Os arquivos feitos em Flash são comumente utilizados para propaganda animada (banners) em páginas web, mas evidentemente não limitando-se a isso, pois existem diversos jogos e apresentações dos mais variados tipos utilizando a tecnologia. Até mesmo sites inteiros podem ser feitos em '.swf'.

Em versões recentes (a partir da 5), a Macromedia expandiu a utilização do Flash para além de simples animações, mas também para uma ferramenta de desenvolvimento de aplicações completas. Isso graças aos avanços na linguagem ActionScript, que é a linguagem de programação utilizada em aplicações de arquivos flashes (.swf). A terceira versão desta linguagem acaba de ser lançada, tornando mais fácil e rápido criar aplicações para web, além de contar com recursos bem mais poderosos.

Uma nova plataforma, chamada Apollo, está sendo lançada pela Adobe e tem como objetivo solidificar o desenvolvimento da linguagem ActionScript, seja através do Flash, do Adobe Flex ou de outros programas. (Wikipédia)

Como fazer um formulário em FLASH com PHP
http://ajuda.uolhost.com.br/index.php?p=resposta&res=904#rmcl

Integração Flash PHP e MySQL em 3 artigos
http://www.mxstudio.com.br/flash/integracao_mysql___php___flash___parte_3/

GUIA RÁPIDO DE USO DO FLASH 8 PROFISSIONAL

O Flash é uma ferramenta que pode tornar um site bem mais atraente e interativo.

Para isso deve ser usado por profissionais de design com talento e com muita cautela, pois também pode tornar o site pesado para carregar.

Usado com bom senso é um grande aliado, pois traz recursos muito atraentes e difíceis de conseguir com outra ferramenta.

Este pequeno guia não é um tutorial do Flash, mas apenas um guia de uso do Flash para edição de filmes (fontes .FLA).

- Caso o arquivo que vá abrir seja de versão anterior, inicie fazendo uma cópia do mesmo para preservar o original, pois normalmente ao ser editado e salvo em versão mais rcente não mais poderá ser aberto na versão mais antiga.

- Abra o arquvo no Flash e tecle Ctrl+Enter para executar no visualizador. Depois feche para voltar ao Flash.

- Ao selecionar um objeto (movie clip, button ou graphic) podemos alterar suas propriedades e seu ActionScript. Para exibir a janela de propriedades tecle Ctrl+F3 e para exibir a do ActionScript para alterar ou adicionar algun código tecle F9.

- Temos duas ferramentas de seleção (seta preta - Selection Tool - tecla de atalho V e seta branca - Subselection Tool - tecla de atalho A). A selection com um duplo clique seleciona todas as camadas ou componentes de um objeto e pode alterar as propriedades de todos de uma vez, já a Subselection não tem esta função, por exemplo, selecione um segmento de reta e ao clica em um dos extremos pode apenas mover um extremo e deixar o outro fixo. Já a selection moverá o segmento de reta por inteiro.

- Observe que por default a Timeline mostra Scene 1. Ao editar um button (com um duplo clique), ao lado de Scene aparece Symbol 2, pois entramos no modo de edição de Símbolo. Logo abaixo 4 fases do botão, Up, Over, Down e Hit. Up é quando o mouse ao clicar no botão inicia o clique, o down é quando o mouse ao clicar termina o clique e solta o botão. o Over é quando o mouse é movido passando sobre o botão e o Hit é a porção de área clicável do botão. Para mover de uma região desta para outra arraste o mouse para a outra.

- Algumas vezes certos textos e outros objetos não aparecem no Flash (apenas no visualizador) por mais que selecionemos todos os objetos. Nestes casos tecle Ctrl+L para exibir a biblioteca (Library) com todos os objetos do arquivo. Com um clique sobre o nome ou o ícone selecionamos. Com um duplo clique sobre o ícone selecionamos para edição no centro.

- Ao mostrar o editor de símbolos com um botão teclando Enter ele mostra o comportamento e as demais fases.

- Adicionando ação a um botão

- Primeiro convertemos o objeto em Symbol - Button.

- Depois Selecionamos o objeto e teclamos F9

- Então clicamos no sinal de + da barra de ActionScripts

- Selecionamos Global Functions - Movie Clip Control - on e duplo clique em press

Ficará assim:

```
on (press) {

}
```

- Insira uma linha em branci acima do fecha chaves e Novamente clicar no sinal de +

```
on (press) {

}
```

- Global Functions - Browser/Network - getURL e então digite a URL:

Ficará assim:

```
on (press) {
	getURL("http://ribafs.net", "_blank")

}
```

Como também podemos abrir um arquivo local - getURL ("3Produtividade/index.php","_blank");

- Um recurso importante é fazer o Flash gerar a página HTML que abre um arquivo .SWF.

Vá em File - Publish Settings. Na aba Formats, Type marque HTML.

Agora para publicar (gerar o HTML) basta teclar Shift+F12 ou File - Publish.

- Ctrl+3 exibe a área central de edição de cenários para caber na tela.

Inserir Flash em HTML

Inserir Animação em Flash no HTML

```
<object width="550" height="400">
	<param name="movie" value="media/video/pegadinha.swf" />
	<PARAM NAME=quality VALUE=high>
	<PARAM NAME=bgcolor VALUE=#333399>
	<embed quality=high bgcolor=#333399 WIDTH="320" HEIGHT="240"
NAME="Yourfilename" ALIGN="" allowfullscreen="true" width="550" height="400"
src="media/video/pegadinha.swf"></embed>
</object>
```

2-PHP

1-PHP Básico
2-Avançado
3-PHPOO

1-PHP Básico

Tutorial Php - Iniciante Funções e conhecimentos básicos

Hello there, the angel from my nightmare...

Tá bom, tá bom, não foi um começo nada muito bom, mas acredite, vai piorar . Estamos aqui reunidos por um motivo em comum, não? Queremos aprender PHP, ou não queremos, mas **temos**. Tudo beleza então, é só efetuar um depósito de R$ 10,00 na conta XYZ-0 que eu lhe envio o Suco de Laranja Mastering PHP, é só beber duas vezes ao dia, que você aprende PHP em 78 horas.

É infelizmente não é um jeito fácil dessa maneira que irá realmente aprender PHP. Será um caminho árduro e cheio de muralhas pela frente, muralhas mesmo, pedras são pequenas para atrapalhar... Mas eu ficar contando historinhas ou piadinhas não vai ajudar em muita coisa, então, vamos ao que interessa.

Introdução

Caso você já tenha tido a oportunidade de ler outro tutorial, provavelmente já viu que PHP é uma linguaguem Server-Side, ou seja, que é executada no, e somente, no servidor. Diferente do Javascript por exemplo que é uma linguaguem Client-Side, executada no cliente (internauta).

Dessa forma, há coisas fora do nosso alcance para realizar no PHP. Como por exemplo, alterar a cor de um botão ao internauta pressionar a letra "b". Isso está sendo executado aonde? No servidor ou no cliente? Já pensando em nosso fórum, faça essa pergunta para você mesmo antes de criar um tópico. Isso é ou não é relacionado à PHP?

Como vamos ver desde ponto em diante, comentários em alguns scripts, vamos ver as possiveis formas de adicionar um comentário no PHP.

```
<?php
// Comentário de 1 linha
# Comentário de 1 linha
/* Comentário em bloco
segue comentado até finalizar o bloco
com */
?>
```

O que vamos ver nesse tutorial?

Antes de prosseguir, vamos deixar claro que, estou considerando que você já está com um servidor web com o PHP rodando em sua máquina, ou com um servidor online. Configurações padrões do php.ini 4.3.0+ (register_globals = off).

Vou tentar, através desse tutorial, abranguer os primeiros passos dado ao PHP, elevando um pouco a dificuldade em cada passo que damos. Não é minha intenção passa funções relacionadas à banco de dados nesse primeiro tutorial, mas para um segundo, isso seria o foco. Então vamos parando com o *lero-lero* e começar a ralar

"Adeus Mundo!"
Por que sempre o famoso "Olá Mundo!"? Vamos estar nos escondendo dele ao tentar aprofundar em PHP. Quanto mais você se interessa em aprender, mais festas deixará de participar, menos garotas irá conhecer e mais sóbrio irá ficar (estou em dúvida se isso é um fato bom ou ruim).

- Nossa, mas ele sempre tira tempo para outra piadinha horrível...

Tááááá booooooooom! Todas as páginas, para serem executados os códigos/instruções PHP, precisam ter sua extensão .php (.php3 está ficando para trás). Sabemos qual a extensão das páginas para serem consideradas PHP, e agora, como eu crio um código PHP? Por padrão, um código PHP é iniciado com a tag **<?php** e encerrado com **?>**. Não vamos nos ater à outras tags, mas só para conhecimento existem as opções:

```
<? ?>
<% %>
<script languague="php"> </script>
```

Certo, vamos criar nossa página "tut01.php" e "limpe" o código fonte dela, não vamos trabalhar com html ainda, somente com a saída PHP. Vamos iniciar um bloco de código PHP e fazer uma saída para o navegador.

```
<?php
echo "Adeus Mundo!";
?>
```

Como diria meu amigo, Jack Estripador, vamos por partes, ou melhor, por linhas.
Linha 01: <?php => Iniciamos o bloco PHP;
Linha 02: echo "Adeus Mundo!"; => *echo* é uma função do PHP para *imprimir* uma saída no navegador. Escrevemos a saída entres aspas pois não estamos trabalhando com variáveis ainda. Utilizamos o ";" no final de cada linha de comando, caso contrário um erro é gerado;
Linha 03: ?> => Encerramos o bloco PHP.

Salvamos essa página e publicamos no servidor. A única saída que temos no navegador será "Adeus Mundo!" (sem as aspas). O código fonte também omite qualquer outra saída, sendo assim, o PHP somente apresentará para o usuário o que for mandado apresentar.

- Certo, mas por que o Joãzinho usa o *print*? Não tem diferença?

Embora muitos digam que não, tem sim. Claro, nada que seja muito utilizado em casos "normais", mas que há uma diferença, há. Tempo de execução, exatamente iguais, sem diferença nisso. Digamos que a única visivel seja, *print* retorna para uma variável TRUE ou FALSE. (lembrando, essa parte é somente para tirar a dúvida de algumas pessoas em relação a comparação de *echo* e *print*, não há necessidade de decorar nada daqui).

```
<?php
$print = print "Teste";
// A saída será Teste, mas agora apresente a variável
```

```
echo $print;
// Saída 1 ou TRUE
?>
```

Eba! Já sei apresentar um conteúdo no navegador
Agora vamos começar a complicar um pouco mais. Vamos ver como apresentar um mais número de saídas no mesmo bloco, e logo em seguida, começaremos a utilizar algumas variáveis.

```
<?php
echo "Nome:";
echo "Luciano";
?>
```

Mas preciso realmente preciso utilizar duas vezes *echo*?!?! Não é necessário, para isso, nós utilizamos a concatenação "." (em outras linguaguens é mais comum ver o símbolo "+" para concatenação, mas no PHP é o "." mesmo).

```
<?php
echo "Nome:" . "Luciano";
?>
```

Você se pergunta, qual a vantagem disso? Poderia colocar os dois textos juntos. Sim, realmente poderia, mas é uma introdução à concatenação. Agora veremos a utilização de variáveis.

O que diabos são variáveis?
São apenas blocos de memória para armazenamento de alguma informação. Texto, números, arquivos, etc... Novamente, diferente de outras linguaguens, no PHP você não é obrigado a definir o tipo de variável que deseja utilizar (int, float, bool, ...). Vamos à alguns exemplos:

```
<?php
$minha_variavel = "Meu texto";
?>
```

Vamos seguir o exemplo do nosso amigo Jack Estripador, por partes. Como podem ver, para declarar uma variável iniciamos o texto com o caracter "$", ele é que indica que o texto a seguir será uma variável. Algumas observações quanto à variáveis:

- Você não pode iniciar um nome de variável com números, ex: $1. Mas pode utilizar nas demais partes do nome, ex: $a1;
- O PHP é case-sensitive (há diferenças entre $nome, $Nome e $NOME);
- Utilize sempre variáveis com nomes indicando o seu conteúdo, para simplificar a manutenção de terceiros e até mesmo a sua.

Tenho uma variável, o que faço com ela?
Digamos que a base de todas as linguaguens de programação são variáveis, pois, se não precisa-se de conteúdos variáveis, por que programaria? Simplesmente utilizava HTML.

Vamos à um outro código:

```
<?php
$nome = "Luciano";
$idade = 19;
?>
```

Declaramos duas variáveis no exemplo acima, uma chamada $nome, com o valor "Luciano" e outra, chamada $idade, com o valor 19. Por que não foi utilizado aspas para inserir o valor 19? Simples, variáveis do tipo texto (text) precisam ser inseridas entre aspas, já as numerais (int, float, ...) você simplesmente precisa informar o número.

Beleza, agora temos duas variáveis e não fizemos nada com elas...
Calma lá! Vamos juntar tudo que vimos até agora:

```
<?php
$nome = "Luciano";
$idade = 19;
echo "Olá, meu nome é " . $nome . " e tenho " . $idade . " anos.";
?>
```

Aqueceu a mente agora? Jack neles:

$nome = "Luciano" => Simplesmente declaramos uma variável, como haviamos feito anteriormente;
$idade = 19 => Mesmo que o anterior, declaração de outra variável;
echo ..... => Complicou? Bom, temos a primeira parte "Olá, meu nome é " que é a apresentação de um texto estático para o navegador, utilizamos o "." para adicionar outro valor à apresentação, *$nome* aqui apresentamos o valor da variável $nome, veja que não é escrito no navegador a palavra "$nome", mas sim "Luciano", adicionamos outro bloco de texto estático, o valor da variável $idade, e por fim, " anos.".

É isso então? Boa sorte no PHP e comece a trabalhar!

.....
É, infelizmente não é só isso não, temos muuuuuuuuito mais desafios pela frente. Que tal algumas funções básicas que podemos fazer trabalhando com variáveis:

strlen(TEXTO)

```
<?php
$nome = "Luciano";
echo "Seu nome tem " . strlen($nome) . " letras.";
?>
```

A funcionalidade da função *strlen* é de contar o número de caracteres que temos em determinada string (texto). Utilizamos a concatenação em uma função, como podem ver,

podemos concatenar textos estáticos, variáveis e funções.

substr(TEXTO, INICIO, [ANDAR])

```
<?php
$nome = "Luciano";
echo "A primeira letra de seu nome é " . substr($nome, 0, 1);
?>
```

A função *substr* tem por objetivo "cortar" parte de um texto para a apresentação. O primeiro parâmetro passamos o texto a ser cortado, por segundo o caracter onde deve ser dado o inicio do corte (lembrando que começa do 0, não do 1), e por final, mas não obrigatório, quantos caracteres devem ser "andados", caso não passe nenhum valor, será até o final do texto.

trim(TEXTO)

```
<?php
$variavel = "   OI     ";
echo trim($variavel);
?>
```

trim elimina os espaços em branco do inicio e do final de um texto.

ucfirst(TEXTO)

```
<?php
$nome = "luciano";
echo ucfirst($nome);
?>
```

ucfirst capitaliza (existe essa palavra?!) a primeira letra de um texto.

strtoupper(TEXTO)

```
<?php
$nome = "Luciano";
echo strtoupper($nome);
?>
```

strtoupper transforma todas as letras de um texto em maiuscula.

strtolower(TEXTO)

```
<?php
$nome = "Luciano";
echo strtolower($nome);
?>
```

strtolower é a função inversa de *strtoupper*. Transforma todos os caracteres em minusculo.

str_replace(PESQUISA, SUBSTITUI, TEXTO)

```
<?php
$texto = "isso foi xxxxxx mesmo";
echo str_replace("xxxxxx", "****", $texto);
?>
```

str_replace procura por uma combinação de letras e a substitui em determinado texto. Outras opções nessa função podem ser vistas com a utilização de matrizes.

Captei! Vamos para Matrizes
Digamos que o monstro para muitos programadores iniciantes. Matrizes são grupos de valores em uma única variável. No PHP temos algumas maneiras para iniciar uma matriz:

```
<?php
$matriz_a = array("João", "Maria");
$matriz_b[] = "João";
$matriz_b[] = "Maria";
?>
```

Ambas teriam o mesmo conteúdo. Na primeiro forma ($matriz_a) declaramos que a variável é do tipo *array* (matriz) e em seguida passamos os valores para ela. Já na segunda, colocamos cada valor separamente. Caso não seja definido um índice para a matriz, ela se auto-inicia do 0 e também se incrementa de acordo com a necessidade.

```
<?php
$matriz = array("Oi", "Tchau");
echo $matriz[0]; // Oi
echo $matriz[1]; // Tchau
?>
```

Criamos nossa matriz, e em seguida apresentamos os valores dela, como não definimos índice em nenhum dos casos, ela se iniciou no 0 e foi se incrementando.

Mas como definir um índice?
Vamos ver nas duas formas:

```
<?php
$matriz = array(5 => "João", "indice" => "Maria");
echo $matriz[5]; // João
echo $matriz["indice"]; // Maria
?>
```

```
<?php
$matriz[5] = "João";
$matriz["indice"] = "Maria";
echo $matriz[5];
echo $matriz["indice"];
?>
```

Certo, mas matriz não tem vantagem nenhum em relação às variáveis normais...
10 minutos e você mudará completamente de idéia. Vamos criar uma lista de nomes em uma variável e apresenta-las:

```
<?php
$nome1 = "Luciano";
$nome2 = "João";
$nome3 = "Maria";
echo $nome1;
echo $nome2;
echo $nome3;
?>
```

Divertido, não? Imagine fazer dessa forma para 500 nomes? Eu não queria ser programador nesses casos... Agora vejamos com matrizes:

```
<?php
$nomes = array("Luciano", "João", "Maria");
foreach( $nomes as $nome ) {
  echo $nome;
}
?>
```

NÃO!!! Não desista agora. Nosso amigo Jack vai explicar melhor tudo isso:

$nome = array(...) => Criamos nosso array, nada de novo
foreach($nomes as $nome) { => Opa, algo novo aí. Para quem conhece inglês já deve ter imaginação do que isso faz, *foreach* (para cada):

```
para cada( $nomes como $nome )
```

Ele irá passar por TODOS os valores da matriz e copiar o seu valor para a variável $nome. Interessante não? É como se executá-se-mos o código tantas vezes quanto há valores na matriz.

Utilizamos "{" e "}" para determinar o inicio e fim do nosso comando *foreach*. Tudo que está entre eles será repetido.

Vamos à outro exemplo com o foreach:

```
<?php
$nomes = array(5 => "Luciano", 10 => "João", "indice" => "Maria"); // Criamos a
matriz definindo seus índices
```

```
foreach( $nomes as $indice => $valor ) {
  echo $indice . " = " . $valor;
}
?>
```

ixi, fedeu? Calma lá, o que temos de novo nessa parte? **($nome as $indice => $valor)**, dessa vez, ao invéz de passar somente o valor de cada índice na matriz, estamos passando o nome do índice. Lembrando, a primeir váriavel receberá o nome do índice e a segunda o seu valor.

Matrizes, basicamente é isso, sua definição e utilização. Vamos passar por algumas funções que envolvem matrizes e dar mais valor à essas pérolas da programação.

Lá vem bomba... Condicional
Qual a vantagem de ter um conteúdo dinâmico em seu site se ele não conseguir decidir qual caminho tomar? Vamos ter uma introdução às condicionais agora *if else*

```
<?php
$nome = "Luciano";
if( $nome == "Luciano" ) {
  echo "Você realmente é o Luciano";
}
?>
```

Como vimos anteriormente, "{" e "}" servem para definir um inicio e um fim a um comando. *if* estará se perguntando SE o valor da variável $nome for igual à "Luciano". Se sim, o código entre "{" e "}" é executado, caso contrário, nenhuma ação é tomada.

Outro detalhe é, para definir valores à variáveis, utilizamos "=", para comparar valores "==". E uma terceira opção seria "===", que compara também o tipo da variável, além de seu valor (utilizada para diferenciar 0 de FALSE, por exemplo).

Agora veremos para não deixar passar em branco nossa condicional, a utilização do comando *else*:

```
<?php
$idade = 19;
if( $idade >= 18 ) {
  echo "Maior de idade";
} else {
  echo "Menor de idade";
}
?>
```

Nossa comparação dessa vez não consulta por um valor igual, mas sim, ">=" (maior ou igual). Somente aplicada em números. Caso o valor da variável $idade não seja >= à 18, a instrução *else* é executada.

Com isso já é possivel fazer uma pesquisa em uma matriz, por exemplo:

```
<?php
$nomes = array("João", "Maria", "Carlos", "Ana", "Paulo", "Bruna");
foreach( $nomes as $nome ) {
  if( $nome == "Ana" ) {
    echo "Ana encontrada";
  }
}
?>
```

O que temos no script acima? Criamos uma matriz com uma lista de nomes, em seguida, varremos toda a matriz, passando cada valor para a variável $nome. Dentro do loop para cada valor na matriz, verificamos se o valor do $nome é igual à "Ana" se for, apresentamos a mensagem "Ana encontrada".

Alguns erros comuns nesses casos:

```
<?php
$nomes = array("João", "Maria", "Carlos", "Ana", "Paulo", "Bruna");
foreach( $nomes as $nome ) {
  if( $nome == "Ana" ) {
    echo "Ana encontrada";
  } else {
    echo "Ana não encontrada";
  }
}
?>
```

Não que esse esteja errado, mas provavelmente não trará a saída desejada, pois, repare bem, em cada loop do comando *foreach* ele executa a condicional, pegamos o primeiro caso, "João", como "João" é diferente de "Ana", ele executa o comando *else*, apresentando, dessa forma, várias vezes "Ana não encontrada". Para executar da maneira desejada esse script, precisamos de uma variável conhecida pelo nome de *flag*.

Pronto, só faltava uma bandeira mesmo no PHP...
Nada disso, *flag* é o nome dado à uma variável que tem apenas um valor para comparação, na maioria das vezes *TRUE* ou *FALSE* (VERDADEIRO ou FALSO). Vejamos como fica nosso exemplo dessa forma:

```
<?php
$nomes = array("João", "Maria", "Carlos", "Ana", "Paulo", "Bruna");
$flag = false;
foreach( $nomes as $nome ) {
  if( $nome == "Ana" ) {
    $flag = true;
  }
}
if( $flag ) {
  echo "Ana encontrada";
} else {
  echo "Ana não encontrada";
}
```

```
?>
```

Jack?!?!

Criamos a matriz, definimos nossa $flag como false (falso), pois "Ana" não foi encontrada dentro da matriz ainda, varremos a matriz passando o valor para a variável $nome, se $nome for igual a "Ana" alteramos nossa $flag para true (verdadeiro). Terminando completamente o loop, executamos nossa condicional, repare que não temos realmente uma comparação a ser feita, por que isso? pois na verdade toda comparação retorna TRUE ou FALSE, ex: 1 == 2 FALSE, 5 == 5 TRUE, e assim por diante. Como nossa variável já tem um valor boleano (TRUE ou FALSE), ela mesmo se encarregará de retornar a resposta para o comando *if*. Se $flag contér TRUE apresentamos "Ana encontrada", caso contrário, "Ana não encontrada".

Imagine agora, se estivessemos procurando por "João", era uma repetição sem sentido até o final da matriz, já que "João" é o primeiro nome, mas como consigo parar um loop?

```
<?php
$nomes = array("João", "Maria", "Carlos", "Ana", "Paulo", "Bruna");
$flag = false;
foreach( $nomes as $nome ) {
  if( $nome == "João" ) {
    $flag = true;
    break;
  }
}
if( $flag ) {
  echo "João encontrado";
} else {
  echo "João não encontrado";
}
?>
```

Nossa única diferença, o comando *break*, que simplesmente corta o loop no momento que for encontrado o nome que procuramos.

Outras formas de Loop
Vamos ver algumas outras formas de loop para utilizar não somente em arrays. Vamos para a primeira delas, *while*:

```
<?php
$contagem = 1;
while( $contagem <= 10 ) {
  echo $contagem;
  $contagem = $contagem + 1;
}
?>
```

O comando *while* (enquanto) executa determinada função enquanto a sua condicional for

verdadeira. Jack, venha cá:

while($contagem <= 10) => Enquanto contagem for menor ou igual à 10, apresentamos o valor de $contagem no navegador e alteramos o valor de $contagem para $contagem + 1, ou seja, um a mais que o seu valor anterior.

Sim, muitas vezes vocês vão conseguir fazer os chamados loops infinitos, ou seja, ele vai ficar em execução até estourar o tempo limite do PHP.

Um exemplo de loop infinito (não execute =D):

```
<?php
$contagem = 1;
while( $contagem <= 10 ) {
  echo $contagem;
}
?>
```

Veja que fazemos praticamente a mesma ação que a acima, mas esquecemos de aumentar o valor da variável $contagem, ou seja, ela sempre terá seu valor igual à 1, nunca passará de 10.

Outro comando para loop que é disponivel no PHP é o *for*, que, basicamente, é o mesmo que o while, mas não precisamos controlar nossa contagem, como no exemplo acima.

```
<?php
for( $contagem = 1; $contagem <= 10; $contagem++ ) {
  echo $contagem;
}
?>
```

De certa forma mais simples que o *while*. Sua sintaxe é a seguinta:

for(INICIALIZACAO; CONDICIONAL; INCREMENTO)

Ou seja, na primeira parte iniciamos a $contagem, depois fazemos nossa condicional verificando se $contagem é <= à 10 e por último, aumentamos o valor da variável $contagem. Repare que dessa vez, utilizamos $contagem++ para aumentar a variável.

```
<?php
$numero = 5;
$numero++; // $numero é aumentado em 1
$numero--; // $numero é diminuido em 1
$numero += 10; // $numero é aumentado em 10
$numero -= 10; // $numero é diminuido em 10
$numero = 5;
$a = $numero++; // $a recebe $numero (5) e então $numero é aumentado em 1
$numero = 5;
$a = ++$numero; // $numero é aumentado em 1 e então passa o novo valor (6) para
$a
$numero = 5;
$a = $numero--; // $a recebe $numero (5) e então $numero é diminuido em 1
```

```
$numero = 5;
$a = --$numero; // $numero é diminuido em 1 e então passa o novo valor (4) para
$a
?>
```

Há também outros comandos para loop como *do... while* / *while... do* Mas que teria a mesma sintexa do comando while, deixaremos eles de fora por enquanto.

Ahhhh, acabou
Exato, infelizmente acabou... Tudo precisa de um começo, um meio e um fim. Bom, estamos quase em 20%  Estou preparando outros tutoriais envolvendo funções de maior utilidade para o desenvolvimento de sites, mas espero que esse pequeno tutorial tenha servidor como uma introdução para quem procura aprender mais sobre PHP e as maravilhas que pode fazer com ele.

Qualquer dúvida, estou inteiramente a disposição para tentar lhe ajudar. Basta "perguntar"

Quanto à liberdade para apresentar esse tutorial em outros fóruns, sites de download ou qualquer outro site de intuito educacional, sinta-se livre, escrevi esse tutorial pensando em ajudar quem está iniciando em PHP, quanto maior o número de atinguidos, melhor. Peço somente, que me reconheçam como autor dele, beleza então?

Tutorial

TUTORIAL DE PHP

Este tutorial teve como principal fonte de informações o manual oficial do PHP.

1 - INTRODUÇÃO

O Que é PHP?

É uma linguagem de script, Open Source, de uso geral, no lado do servidor, embutível no HTML e especialmente voltada para o desenvolvimento Web. Originalmente chamou-se PHP (Personal Home Page) mas depois teve seu nome alterado pela comunidade para PHP: Hypertext Processor, um acrônimo recursivo.

Um pequneno script em PHP

```
<?php
echo "Script em PHP!";
?>
```

Popularidade do PHP

A popularidade do PHP entre as demais linguagens é muito boa.
Segundo estatísticas do site:
http://www.drews.cx/2006/03/23/php-usage-stats-go-up-again/
Onde cita as 4 mais populares linguagens: C, C++, Java e PHP. Nesta ordem.
Observação: todas oriundas do C.
Na prática, entre as linguagens Web o PHP fica em segundo lugar.
Veja a lista geral:
http://www.tiobe.com/index.htm?tiobe_index
Em março/2006 ele já está presente em mais de 20 milhões de domínios ao redor do planeta (http://www.php.net/usage.php).

Estatísticas sobre o PHP:

http://www.nexen.net/chiffres_cles/phpversion/php_statistics_for_april_2006.php

História do PHP

- Criado em 1995 por Rasmus Lerdorf, na forma de scripts Perl para coletar estatísticas online de seu currículo. Com um interpretador em C e comunicação com SGBDs.

- Versão 2 aparece em novembro de 1997, quando recebe seu nome inicial e é enriquecido com um interpretador de formulários (FI) - PHP/FI. Teve seu código fonte disponibilizado para a comunidade. Contava com 50.000 domínios que o utilizavam (em torno de 1% dos existentes na época).

- Versão 3 sai em seguida, logo em julho de 1998. Similar ao PHP atual, esta versão foi totalmente reescrita por Andi Gutmans e Zeev Suraski, programadores israelenses. Inicia o suporte à orientação a objetos e a sua extensibilidade, que atraiu muitos programadores. Rebatizado de PHP: Hypertext Processor. Já estava presente em 10% dos servidores web da Internet.

- Versão 4 sai em maio de 2000. Melhora de performance, suporte a muitos servidores

web, a session, entre outros. Já está presente em 20% dos domínios da Internet.

- Versão 5 sai em julho de 2004. Seu foco principal é a orientação a objetos, que corrije deficiências e traz novos e amplos recursos para a orientação a objetos.

Delimitadores do PHP
Para que o interpretador reconheça quando tem que interpretar o script em PHP e quando não, usa-se delimitadores para
que quando os encontre ele interprete corretamente. Quando ele encontra o delimitador <?php ele começa a processar como PHP e quando encontra ?> ele pára de processar e tudo o mais envia para o browser como está. Existem outros tipos de delimitadores do PHP mas estes <?php ... ?> são os recomendados, por serem os mais abrangentes e terem suporte inclusive a XML.

Forças do PHP
- Sua simplicitade é muito importante para o programador que se inicia no PHP.

- Seus recursos atendem ao programador experiente.

- Sua documentação rica em conteúdo e exemplos facilita a solução de problemas, além da busca online.

- O suporte de maneira simples à maioria dos SGBDs do
mercado atende às necessidades de sites dinâmicos com acesso a bancos de dados.

O Que é Possível Realizar em PHP?
- Em sendo uma linguagem de uso geral, podemos realizar praticamente qualquer atividade realizada com outra linguagem. Ele roda no lado servidor, como aplicação Web e também roda no lado cliente, com o PHP-GTK.

- Existem edições para os principais sistemas operacionais (Linux, BSDs, Windows, Mac OS X, etc).

- Suportado pela maioria dos servidores Web, especialmente pelo Apache.

- Bom suporte à orientação a objetos, em especial com a chegada da versão 5.

- Comunicação com Java.

- Suporte aos principais SGBDs do mercado (MySQL, PostgreSQL, Oracle, SQL Server, etc), atualmente são mais de vinte os suportados.

- Suporte a LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (em Windows somente), a IRC, compressão (gzip, bz2, rar), PDF, GD, Flash, XML, etc..

2 - TUTORIAL SIMPLES

Para continuar você deve ter um servidor web instalado e com suporte a PHP e, caso queira usar bancos de dados, instale um SGBD.

Os arquivos em PHP somente funcionam corretamente no navegador, se estivem no diretório reconhecido pelo servidor web, que é o DocumentRoot. Considerando que o PHP e Apache tenham sido instalados com o Xampp e que o diretório web seja o default, este fica em:

C:\Arquivos de Programas\Xampp\htdocs

Além disso qualquer script que contenha algo em PHP deve ter a extensão .php para que seja corretamente processado pelo Apache.

Criar um arquivo chamado olamundo.php na pasta
C:\Arquivos de Programas\Xampp\htdocs, com o conteúdo:

<?php

echo "Olá PHP!";

?>

Startar o Apache e abrir no browser assim:

http://127.0.0.1/olamundo.php

Visualizar resultado de script PHP no Browser

Após ter executado o olamundo.php no Browser solicite a visualização do
código fonte da página exibida.

Veja que nada chega de PHP. O PHP é processado no servidor e envia apenas o
resultado em HTML para o browser.

Alerta sobre Editores e Processadores de Texto com PHP
Para criar scripts PHP evite o uso de processadores de texto e de editores como o
Bloco de Notas, pois deixam sujeira no código e dificultam o salvamento do arquivo.

Preferentemente use uma IDE especializada em PHP, que inclusive
tem outros recursos úteis, a exemplo do PHPEclipse ou um editor de texto como o
freeware Win32pad.

Informações sobre o PHP
Para obter diversas informações úteis sobre o PHP instalado no seu servidor, execute um
script com a função:

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

Checar navegador em uso

```
if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
    echo "Seu navegador é o Internet Explorer!";
} else {
    echo "Seu navegador não é o IE!";
}
```

strstr - procura substrings.

Teste com echo e print:

```
<?php
// Teste de echo e print
//print "primeiro, ", "segundo"; // Gera erro de parser
echo "primeiro, ", "segundo, ", "terceiro"; //Funciona
// Ou seja, print pode trabalhar apenas com uma string de cada vez e echo com várias
?>
```

print retorna valor enquanto que echo não retorna.

```
<?php
$print = print "";
```

```
echo "Retorno do print: " . $print;
?>
```

Agora teste esse:

```
<?php
   $echo = echo "";
   echo "Retorno do echo: " . $echo;
?>
```

Irá receber um erro fatal (do parser).

3 - CONFIGURAÇÕES

php.ini

register_globals = off (por questões de segurança)
No Xampp o php.ini traz register_globals ativo por default.

GD, pgsql, mysql e outras extensões que queira usar deverá descomentar no php.ini, seção Extensions.

httpd.conf
Para poder alterar o diretório web default deverá mudar dois parâmetros no arquivo httpd.conf do Apache:

DocumentRoot “/opt/lampp/htdocs”
<Directory "/opt/lampp/htdocs">

Os diretórios acima são para a edição for Linux do Xampp. Para o Windows observe que o Apache utiliza, não barras, que são utilizadas neste SO, mas sim contra-barras e as vezes contra-barras duplas.

No caso queremos mudar o diretório default para /home/www, então alteramos o httpd.conf para que fique assim:

DocumentRoot “/home/www”
<Directory "/home/www">

No Windows fica assim:
DocumentRoot "c:/Arquivos de Programas/Xampp/htdocs"
<Directory "<c:/Arquivos de Programas/Xampp/htdocs">

Que devem ficar assim:
DocumentRoot "c:/1www"
<Directory "<c:/1www">

Configuração das extensões suportadas. Altere a linha:
DirectoryIndex index.html index.html.var index.php index.php3 index.php4

Veja que acima configurou-se para perceber inclusive arquivos com as extensões .php3 e .php4.
Caso não fossem incluídas, arquivos com extensão .php3 e .php4 não poderiam ser abertos neste servidor.

4 - REFERÊNCIA DA LINGUAGEM

Separador de Instruções
O ponto e vírgula ; é o separador de instruções em PHP (como em Perl e em C).
Sempre que uma instrução terminar devemos digitar um ponto e vírgula (echo "ola";).

Comentários
Em PHP podemos usar 3 tipos de comentários (/* ... */, // e #) mas
devemos utilizar somente os dois primeiros e o primeiro é o mais eficiente, que é o /* ... */, que veio do C, já que # está em processo de obsolescência. Ou seja, devemos usar // ou /* ... */, de preferência este último.

5 - TIPOS DE DADOS

O PHP suporta os oito tipos primitivos:

- boolean, integer, float e string (básicos)
- array e object (compostos)
- resource e NULL (especiais)

float é sinônimo de double em PHP.

Tipo de Variável
O tipo de uma variável em PHP não é controlado pelo programador, depende do valor da variável e é avaliado em tempo de execução. Não é permitido ao programador declarar o tipo de variáveis.

Funções que Retornal o Tipo

gettype

todas as is_type

Casting
(tipo) variavel;

Boleanos
Pode ser TRUE ou FALSE (case-insensitive)

Exemplos:

```
<?php
echo gettype((bool) "")."<br>";        // bool(false)
echo gettype((bool) 1)."<br>";         // bool(true)
echo gettype((bool) -2)."<br>";        // bool(true)
echo gettype((bool) "foo")."<br>";     // bool(true)
echo gettype((bool) 2.3e5)."<br>";     // bool(true)
echo gettype((bool) array(12))."<br>"; // bool(true)
echo gettype((bool) array())."<br>";   // bool(false)
?>
```

Convertendo Explicitamente para Boleano

Usa-se o modificador (bool) ou (boolean).

Valores que são considerados FALSE

FALSE, 0, 0.0, "0", "", array vazio, objeto sem membros e NULL (e variáveis indefinidas).

Os demais são TRUE

Inteiros
Em PHP, inteiro é um número do conjunto matemático dos Inteiros (Z), que contem os negativos, o zero e os positivos.

Em PHP os inteiros podem ser decimais, octais ou hexadecimais.

octal - precedido por 0.
hexadecimal - precedido por 0x.

Exemplos:

```
<?php
$a =1234;
echo $a."<br>"; // número decimal
$a =-123;
echo $a."<br>"; // um número negativo
$a =0123;
echo $a."<br>"; // número octal (equivalente a 83 em decimal)
$a =0x1A;
echo $a."<br>"; // número hexadecimal (equivalente a 26 em decimal)
?>
```

O tamanho dos inteiros depende da plataforma e é de 32 bits com sinal. O PHP não suporta inteiros sem sinal.

Overflow - caso seja especificado um número inteiro além dos limites, será interpretado como flutuante.

Convertendo Explicitamente para Inteiro

Usar o modificador (int) ou (integer).

Ou com a função intval().

De boleanos - FALSE será retornado como 0 e TRUE como 1.

De flutuantes - ao converter para inteiros serão truncados

De strings - A string será avaliada como um ponto flutuante se contiver qualquer um dos caracteres '.', 'e', ou 'E'. Em outros casos, ela será avaliada como um inteiro.

De outros tipos - não têm precisão, exatidão, portanto é melhor evitar.

Alerta:
```
echo (int) ((0.1 + 0.7 ) * 10); // Exibirá 7 ao invés do esperado 8
```

Ponto Flutuante
É o float, double ou real.

Exemplos:

1.234 ou 1.2e3 ou 7E-10

```
<?php
$a = 1.234;
echo $a."<br>";
```

```
$b = 1.2e3;
echo $b."<br>";
$c = 7E-4;
echo $c;
?>
```

O tamanho de um float depende também da plataforma e é de 64bits no formato IEEE(*).

Nunca compare números em ponto flutuante em igualdades, sob pena de cometer erros.

* - (Wikipedia - http://pt.wikipedia.org)O **Instituto de Engenheiros Eletricistas e Eletrônicos** ou **IEEE** (pronuncia-se I-3-E) é uma organização profissional sem fins lucrativos, fundada nos Estados Unidos. É a maior (em número de sócios) organização profissional do mundo. O IEEE foi formado em 1963 pela fusão do Instituto de Engenheiros de Rádio (IRA) com o Instituto Americano de Engenheiros Elétricistas (AIEE). O IEEE tem filiais em muitas partes do mundo, sendo seus sócios engenheiros eletricistas, engenheiros da computação, cientistas da computação, profissionais de telecomunicações etc. Sua meta é promover conhecimento no campo da engenharia elétrica, eletrônica e computação. Um de seus papéis mais importantes é o estabelecimento de padrões para formatos de computadores e dispositivos.

Strings
Em PHP um caractere ocupa um byte. Até a versão 5 o PHP não tem suporte a UNICODE, mas está previsto este suporte para a próxima versão (Fonte: Wikipedia - http://www.wikipedia.org).

Não há limite para o tamanho de uma string em PHP.

Especificando Strings:
- apóstrofos (chamados de aspas simples ')
- aspas (chamadas de aspas duplas ")
- heredoc (<<<)

Exemplos:

```
<?php
echo 'isto é uma string comum';
echo 'Você pode incluir novas linhas em strings,
dessa maneira que estará
tudo bem';

// Imprime: Arnold disse uma vez: "I\'ll be back"
echo 'Arnold once said: "I\'ll be back"';

// Imprime: Você tem certeza em apagar C:\*.*?
echo 'Você tem certeza em apagar C:\\*.*?';

// Imprime: Você tem certeza em apagar C:\*.*?
echo 'Você tem certeza em apagar C:\*.*?';

// Imprime: Isto não será substituido: \n uma nova linha
echo 'Isto não será substituido: \n uma nova linha';

// Imprime: Variaveis $também não $expandem
```

```
echo 'Variaveis $também não $expandem';

echo '<br>------------<br>';

$str = <<<EOD
Exemplo de uma string
distribuída em várias linhas
utilizando a sintaxe heredoc.
EOD;

/* Exemplo mais complexo, com variáveis */
class foo
{
   var $foo;
   var $bar;

   function foo()
   {
       $this->foo = 'Foo';
       $this->bar = array('Bar1', 'Bar2', 'Bar3');
   }
}

$foo = new foo();
$name = 'Meu nome';

echo <<<EOT
Meu nome é "$name". Eu estou imprimindo $foo->foo.
Agora, eu estou imprimindo {$foo->bar[1]}.
Isto deve imprimir um 'A' maiúsculo: \x41
EOT;

?>
```

6 - VARIÁVEIS

Em PHP as variáveis são iniciadas por um sinal de dólar $.

Variáveis Predefinidas

São as que já vêm definidas no próprio PHP. A função phpinfo() mostra também estas variáveis.

Com o PHP 4.2 o valor default da diretiva register_globals passou a ser off. Com isso as variáveis passaram a ser acessadas de forma diferente e muitos scripts deixaram de funcionar quanto tiveram o PHP atualizado e outros novos scripts não funcionaram devido esperar uma semelhante a anteerior.

On Off
==========

$DOCUMENT_ROOT $_SERVER['DOCUMENT_ROOT'];
$HOME $_ENV['HOME'];

$GLOBALS
$_SERVER
$_GET
$_POST
$_REQUEST
$_SESSION

Obs.: Agora, como o default do PHP é register_globals = Off, faz-se necessário usar $_POST['nomecampo'], para receber o valor de um campo de form em script PHP.

Escopo de Variáveis
O escopo de uma variável é o contexto onde ela foi definida e geralmente o escopo é local.

$a = 1;
include ("teste.php");

// $a estará disponível, será vista por teste.php, pois foi definida antes.

include ("teste.php");
$a = 1;

// Aqui, como $a foi definida após a inclusão, não será visto pelo teste.php

Escopo de variáveis em funções
A palavra-chave global pode preceder uma variável para tornar seu escopo global, como também $GLOBALS[].

Exemplos:

```
<?php
$a = 1; /* escopo global */

function Teste(){
   echo $a; /* referencia uma variável do escopo local (não
definida) */
}

Teste();
?>

<?php
$a = 1; /* escopo global */

function Teste(){
    global $a;
    echo $a; /* referencia a variável do escopo global */
}

Teste();
?>
```

Usando *$GLOBALS* no lugar de global

```
<?php
$a = 1;
$b = 2;

function Soma(){
   $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Soma();
echo $b;
?>
```

Utilizando Variáveis Estáticas
Variáveis estáticas guardam o valor de variáveis entre execuções de funções.
Ao fechar o programa, ao fechar ou atualizar o browser, o valor da variável se perde.

Exemplos:

```
<?php
function Teste (){
  $a = 0;
  echo $a;
  $a++;
}

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

function Teste2(){
  static $a = 0;
  echo $a;
  $a++;
}

for ($x=1;$x<10;$x++){
   Teste2();
}

echo "<br><br>";

// Função recursiva
function Teste3()
{
  static $count = 0;

  $count++;
  echo $count;
```

```
   if ($count < 10) {
       Teste3 ();
   }
   $count--;
}

for ($x=1;$x<5;$x++){
    Teste3();
    if ($x < 4) echo " - ";
}

echo "<br><br>";
//Declarando variáveis static

function foo(){
   static $int = 0;          // correro
   //static $int = 1+2;        // errado (é uma expressão)
   //static $int = sqrt(121);  // wrong  (é uma expressão também)

   $int++;
   echo $int;
}

foo();

?>
```

Variáveis Variáveis
São variáveis cujos nomes podem ser criados dinamicamente.

Variável comun -> $variavel;
Variável variável -> $$variavelvariavel;

Ela torna o valor de uma variável e o trata como se fosse o nome de uma variável.
Obs.: variáveis variáveis não podem ser utilizadas com os arrays superglobais.

Determiando o Tipo das Variáveis
gettype
is_array, is_float, is_int, is_object, is_string, is_numeric

7 - CONSTANTES
O valor de uma constante não pode ser alterado durante a execução do script.
Convenciona-se usar o nome de constantes com todas as letras em maiúsculas.

define ("NOME", "valor");

Exemplos:

```
<?php
// Nomes de constantes válidos
```

```
define("FOO",     "alguma coisa");
define("FOO2",    "alguma outra coisa");
define("FOO_BAR", "mais alguma outra coisa")

// Nomes de constantes inválidas
define("2FOO",    "alguma coisa");

// Isto é válido, mas deve ser evitado:
// O PHP pode vir a fornercer uma constante mágica
// que danificará seu script
define("__FOO__", "alguma coisa");

?>
```

Obs.: Somente dados escalares (bolean, integer, float, e string) podem ser armazanados nos valores de constantes.

A função constant() retorna o valor de uma constante.
A função get_defined_constantes() retorna todas as constantes definidas.
Enquanto que defined() checa se uma constante foi definida.

```
define (NOME, valor);
define ("PESO", 70);
print "O peso vale " . PESO . " KG";
<?php
switch (PHP_OS){
   case "WIN32":
      echo "Sistema Windows";
      break;
   case "Linux":
      echo "Sistema Linux";
      break;
   case "OS/2":
      echo "Sistema OS/2";
      break;
   default:
      echo "Sistema não é Windows, Linux nem OS/2";
      break;
}
?>
<?php
if (strtoupper(substr(PHP_OS, 0, 3)) === 'WIN') {
   echo 'Este é um servidor usando Windows!';
} else {
   echo 'Este é um servidor que não usa Windows!';
}
?>
```

Constantes Mágicas
Essas são as cinco constantes "mágicas", que mudam dependendo de onde elas são utilizadas. Por exemplo, o valor de **__LINE__** depende do número da linha em que é utilizada em seu script. Essas constantes especiais são insensíveis ao caso:

Algumas constantes "mágicas" do PHP

Nome	Descrição
__LINE__	A linha atual do script.
__FILE__	O caminho completo e nome do arquivo. Se utilizado dentro de um include, o nome do arquivo incluído será retornado.
__FUNCTION__	O nome da função (Acrescentado no PHP 4.3.0). A partir do PHP 5 esta constante retorna o nome da função como ela foi declarada (sensível a maiúsculas e minúsculas). No PHP 4 sempre retorna o nome em minúsculas.
__CLASS__	O nome da classe (Acrescentado no PHP 4.3.0). A partir do PHP 5 esta constante retorna o nome da função como ela foi declarada (sensível a maiúsculas e minúsculas). No PHP 4 sempre retorna o nome em minúsculas.
__METHOD__	O nome do método de classe. (Acrescentado no PHP 5.0.0). O nome do método é retornado como foi declarado (sensível a maiúsculas e minúsculas).

Veja também **get_class()**, **get_object_vars()**, **file_exists()**, e **function_exists()**.

8 - EXPRESSÕES

Uma expressão é qualquer coisa que tem um valor, normalmente na forma de constantes ou variáveis.

```
$a = 5;
```

Temos acima uma equação formada por duas expressões, a da esquerda composta por uma variável e a da direita composta por uma constante.

Funções são expressões cujo valor é igual ao seu valor de retorno.
O PHP é uma linguagem orientada as expressões.
Atribuições são analisadas da direita para a esquerda.

```
$b = $a = 5; // $a recebe 5 e então $b recebe 5
```

Exemplo:

```
<?php
function dobro($i){
   return $i*2;
}
$b = $a = 5;echo $b."<br>";        /* atribui o valor cinco às variáveis $a e $b */
$c = $a++;echo $c."<br>";          /* pós-incremento, atribui o valor original de $a
                    (5) para $c */
$e = $d = ++$b;echo $e."<br>";     /* pré-incremento, atribui o valor incrementado de
                    $b (6) a $d e $e */

/* neste ponto, tanto $d quanto $e são iguais a 6 */

$f = dobro($d++);echo $f."<br>";  /* atribui o dobro do valor de $d antes
                    do incremento, 2*6 = 12 a $f */
$g = dobro(++$e);echo $g."<br>";  /* atribui o dobro do valor de $e depois
                    do incremento, 2*7 = 14 a $g */
```

```
$h = $g += 10;echo $h."<br>";      /* primeiro, $g é incrementado de 10 e termina com o
                valor 24. o valor da atribuição (24) é
                então atribuído a $h, e $h termina com o valor
                24 também. */
?>
```

9 - OPERADORES

Um operador é algo que alimentado com um ou mais valores devolve outro valor. Assim as funções e outros construtores que retornam valor são operadores.

Tipos de Operadores
- unários - operam apenas em um valor (!, ++, ...)
- binários - operam em dois valores (+, - , ...)
- ternário - selecionar entre 2 valores, dependendo de um terceiro. Emglobá-los entre parênteses facilita o entendimento.

Precedência de Operadores

A precedência de um operador em relação a outro diz que um operador deve ser executado antes do outro.

Exemplo:
1 + 5 * 3

Dependendo da precedência dos operadores + ou * teremos resultados diferentes.
Se o operador + tiver precedência superior ao * então resolveremos assim:

Somamos 1 + 5 e somente depois multiplicamos por 3 -> (1+5)*3 = 6*3 = 18

Este resultado está incorreto, pois o operador * tem precedência superior ao + e deveria ser assim:
1+(5*3) = 1+15=16

Para facilitar a percepção das precedências usa-se parêntesis:

1 + (5 * 3) = Sempre devemos resolver antes o que estiver entre parêntesis.
1 + 15 = 16 (Neste caso fica mais claro).

Veja a tabela contendo a precedência de dos operadores do PHP, da maior precedência para a menor:

*** e /** têm precedência sobre **+ e -.**

Obs.: Para uma relação completa das precedências veja o manual oficial no site do PHP.

Operadores Aritméticos

São os operadores correspondentes às quatro operações comuns da matemática adicionados ao operador módulo:

• soma (+)
• subtração (-)
• multiplicação (*)
• divisão (/)
• módulo (%)

O módulo é o resto de um número dividido por outro.

Exemplos de operadores aritméticos:

```
$a = 9;
$b = 4;
echo "\$a + \$b = ".($a + $b); // 13
echo "\$a + \$b = ".$a + $b; // Retorna 4, pois após o ponto é considerado
strings
echo "\$a - \$b = ".($a - $b); // 5
echo "\$a * \$b = ".$a * $b; // 36
echo "\$a / \$b = ".$a / $b; // 2.25 - Divisão Quociente de $a por $b.
echo "\$a % \$b = ".$a % $b; // 1 - Resto de $a por $b
```

Operadores de Atribuição
O operador básico de atribuição é o sinal de igualdade =.

Exemplo:

$x = $y + 3; // O que representa: 3 será adicionado a $y e o resultado será atribuído a $x
$a = 3;
$a += 5; // Que é semelhante a $a = $a + 5;
$b = "Bom ";
$b .= "Dia!"; // Similar a $b = $b . "Dia!";

Exemplos:

```
<?php
$a = 3;
$a += 5; // $a recebe 5 e soma com seus 3, tipo: $a = $a + 5;
echo "\$a vale " . $a;

$b = "Bom ";
$b .= "Dia!"; // $b fica com "Bom Dia!", como em $b = $b . "Dia!";
echo "\$b vale " . $b;

$c=2;
$a -= $c;    //$a = $a - $c    Subtração
echo "\$a -= \$c vale " . $a;
$a *= $c;    //$a = $a * $c    Multiplicação
echo "\$a *= \$c vale " . $a;
$a /= $c;    //$a = $a / $c    Divisão
echo "\$a /= \$c vale " . $a;
$resto = $a % $c;    //$a = $a % $c    Módulo (resto)
echo "Resto de $a % $c vale: " . $resto;
?>
```

Operadores de Controle de Erro
Representado pelo símbolo de arroba @. Quando precede uma expressão ele abafa as mensagens de erro.
Observação - Somente funciona em expressões, ou seja, em qualquer construção que retorne algo.

Recomendação - Deixar todas as mensagens de erro originais do PHP livres em ambientes de desenvolvimento (de testes). Somente utilizar @ em ambiente de produção, após executar os testes.

Exemplos:

```
<?php
/* Erro intencional */
$a = 6;
$b = 0;

echo "Camuflando erro de divisão por zero";
$c = @($a / $b);
?>
```

Operadores de Execução

Existem algumas funções de execução de programas (shell_exec e outras) como também existe um operador, que é formado por dois sinais de crase``.

Nota: Caso safe_mode esteja desativado no php.ini como também shell_exec() então o operador de execução também fica dasativado.

Exemplo:

```
<?php
// Em PHP a crase ` é um operador de execução de arquivos do SO
// Como em scripts bash

// Exibir todos os arquivos do diretório atual, inclusive os
ocultos
if (PHP_OS == "WINNT"){

    $output = `dir/o/p`;
    echo "<pre>$output</pre>";
} elseif (PHP_OS == "Linux){
    $output = `ls -la`;
    echo "<pre>$output</pre>";
}else{
    echo "Você está usando um SO diferente de Linux e de Windows!"
}
?>
```

Operadores de Incremento e de Decremento

Os operadores de pré e pós-incremento/decremento são suportados pelo PHP.

++$a (pré-incremento) - Primeiro incrementa $a de 1, depois retorna $a incrementado

```
$a++ (pós-incremento) - Primeiro retorna $a, depois incrementa $a
de 1
--$a (pré-decremento) - Primeiro decrementa $a de 1, depois
retorna $a decrementado
$a-- (pós-decremento) - Primeiro retorna $a, depois decrementa $a
de 1
```

Exemplos:

```
<?php
echo "<h3>Pós-incremento</h3>";
$a = 5;
echo "\$a = ".$a."<br><br>";
echo "\$a++ deve ser: " . $a++ . "<br />\n";
echo "\$a deve ser: " . $a . "<br />\n";

echo "<h3>Pré-incremento</h3>";
$a = 5;
echo "++\$a deve ser: " . ++$a . "<br />\n";
echo "\$a deve ser: " . $a . "<br />\n";

echo "<h3>Pós-decremento</h3>";
$a = 5;
echo "\$a-- deve ser: " . $a-- . "<br />\n";
echo "\$a deve ser: " . $a . "<br />\n";

echo "<h3>Pré-decremento</h3>";
$a = 5;
echo "--\$a deve ser: " . --$a . "<br />\n";
echo "\$a deve ser: " . $a . "<br />\n";
?>
```

Operadores Lógicos

Utilizados para comparar duas expressões e o resultado será TRUE ou FALSE.

Exemplos:

```
<?php
$a = true; $b = FALSE; // true e false são insensitivos

echo ($a and $b)? "T<br>":"F<br>";    //E    Verdadeiro (TRUE) se tanto $a quanto $b são
verdadeiros.
echo ($a or $b)? "T<br>":"F<br>";    //OU    Verdadeiro se $a ou $b são verdadeiros.
echo ($a xor $b)? "T<br>":"F<br>";    //XOR    Verdadeiro se $a ou $b são verdadeiros,
mas não ambos.
echo (! $a)? "T<br>":"F<br>";        //NÃO    Verdadeiro se $a não é verdadeiro.
echo ($a && $b)? "T<br>":"F<br>";    //E    Verdadeiro se tanto $a quanto $b são
verdadeiros.
```

```
echo ($a || $b)? "T<br>":"F<br>";    //OU    Verdadeiro se $a ou $b são verdadeiros.
?>
```

Operadores de String

Strings em PHP são concatenadas com o operador ponto final ".".

Exemplos:

```
<?php
$a = "Olá ";
$b = $a . "mundo do PHP!";
echo $b;

$a = "Olá ";
$a .= "meu mundo!";

echo "<br>" . $a;
?>
```

Convertendo strings em números

```
<?php
$foo = 1 + "10.5";echo $foo."<br>";              // $foo é float (11.5)
$foo = 1 + "-1.3e3";echo $foo."<br>";              // $foo é float (-1299)
$foo = 1 + "bob-1.3e3";echo $foo."<br>";           // $foo é integer (1)
$foo = 1 + "bob3";echo $foo."<br>";                // $foo é integer (1)
$foo = 1 + "10 Small Pigs";echo $foo."<br>";       // $foo é integer (11)
$foo = 4 + "10.2 Little Piggies";echo $foo."<br>"; // $foo é float (14.2)
$foo = "10.0 pigs " + 1;echo $foo."<br>";          // $foo é float (11)
$foo = "10.0 pigs " + 1.0;echo $foo."<br>";        // $foo é float (11)
?>
```

Operações com Strings

```
<?php
// Pega o primeiro caracter da string
$str = 'Isto é um teste.';
$first = $str{0};
echo $first."<br>";
// Pega o terceiro caracter da string
$third = $str{2};
echo $third."<br>";
// Pega o último caracter da string
$str = 'Isto ainda é um teste.';
$last = $str{strlen($str)-1};
echo $last."<br>";
// Modifica o ultimo caracter da string
$str = 'Olhe o mal';
echo $str{strlen($str)-1} = 'r';
?>
```

Operadores com Arrays

Exemplo	Nome	Resultado
$a + $b	União	União de $a e $b.
$a == $b	Igualdade	**TRUE** se $a e $b tem os mesmos elementos.
$a === $b	Identidade	**TRUE** se $a e $b tem os mesmos elementos na mesma ordem.
$a != $b	Desigualdade	**TRUE** se $a não é igual a $b.
$a <> $b	Desigualdade	**TRUE** se $a não é igual a $b.
$a !== $b	Não identidade	**TRUE** se $a não é identico a $b.

O operador + acrescenta o array da direita no array da esquerda, contudo, chaves duplicadas NÃO são sobrescritas.

Exemplos:

```
<?php
$a = array("a" => "maçã", "b" => "banana");
$b = array("a" =>"pêra", "b" => "framboesa", "c" => "morango");

$c = $a + $b; // Uniao de $a e $b
echo "União de \$a e \$b: \n";
var_dump($c);

$c = $b + $a; // União de $b e $a
echo "União de \$b e \$a: \n";
var_dump($c);
?>
```

Observar que na união de $a+$b o valor de "b" é banana em $a, ele não foi sobrescrito por framboesa de $b.
Assim como framboesa "b" em $b não foi substituído por banana de $a na união de $b+ $a.

10 - ESTRUTURAS DE CONTROLE

As principais estruturas para controlar o fluxo dos scripts PHP.
if, else, elseif, while, do... while, for, break, consinue e switch.
Um script PHP é formado por instruções (cada instrução termina com ;).

Uma instrução pode ser:
- uma atribuição
- uma chamada de função
- um laço (loop)
- uma instrução condicional
- até uma instrução nula (;;).

As instruções podem ser agrupadas com chaves, formando blocos.

if - uma construção muito importante e versátil. Permite a execução condicional de fragmentos de código.

Sintaxe:

```
if (expressao) {
   instrucoes;
}
```

expressao - é avaliada como TRUE ou FALSE
- Se TRUE as instrucoes serão executadas
- Se FALSE as instrucoes serão ignoradas

Exemplos:

```
<?php
$a = 5;
$b = 3;
if ($a > $b){
   echo "a é maior que b";
}
?>
```

else - executa instrucoes2 caso expressao seja FALSE.

Sintaxe:

```
if (expressao) {
   instrucoes;
} else {
   instrucoes2;
}
```

Exemplos:

```
<?php
$a = 3;
$b = 5;
if ($a > $b){
   echo "a é maior que b";
} else {
   echo "a NÃO é maior que b";
}
?>
```

elseif - é uma combinação de if com else. Caso a expressao do if seja FALSE então o elseif testa a expresao2, se esta for TRUE instrucoes2 serão executadas, caso contrário instrucoes3 serão executadas.

Sintaxe:

```
if (expressao) {
    instrucoes;
} elseif (expressao2) {
    instrucoes2;
} else {
    instrucoes3;
}
```

```
<?php
$a = 3;
$b = 3;
if ($a > $b){
   echo "a é maior que b";
} elseif ($a==$b){
   echo "a é igual a b";
} else {
   echo "a NÃO é maior que b nem igual a b";
}
?>
```

while - executa instruções várias vezez, enquanto uma expressão for verdadeira (TRUE).

Sintaxe:

```
while (expressao) {
    instrucoes;
}
```

Em cada iteração expressao é avaliada, se TRUE instrucoes serão executadas, se FALSE, serão ignoradas.

Exemplos:

```
<?php

$i = 1;
while ($i <= 10) {
   echo $i++;  /* o valor impresso será
                  $i depois do acréscimo
                  (post-increment) */
}

?>
```

do ... while - Semelhante ao while, diferindo no fato da expressão condicional ficar ao final das iterações.
No do ... while a primeira iteração sempre executará as instruções incondicionalmente.

Sintaxe:

```
do {
    instrucoes;
} while (expressoes);    // Atentar para o ponto e vírgula ao final
```

Exemplos:

```
<?php
// Executa pelo menos uma vez incondicionalmente
$i = 6; $fator = 2; $minimo = 10;
do {
   if ($i < 5) {
       echo "\$i não é grande o suficiente";
       break;
   }
   $i *= $fator;
   if ($i < $minimo) {
       break;
   }
   echo "\$i está Ok e vale " . $i;

} while (0);

/* Exemplo simples
$i = 0;
do {
   echo $i;
} while ($i > 0);
*/
?>
```

for - laço composto por 3 instruções. A primeira é a de inicialização da variável, a segunda será avaliada a cada iteração e a terceira é a de incremento. As 3 separadas por ponto e vírgula. Todas as 3 são opcionais e caso a segunda seja nula o loop será infinito.

Sintaxe:

```
for (expr1; expr2; expr3) {
    instrucoes;
}
```

- expr1 será avaliada uma única vez incondicionalmente;
- expr2 será avaliada no início de cada iteração:
    - Se TRUE o laço continua e as instruções serão executadas
    - Se FALSE o laço termina.
- expr3 é avaliada ao final de cada iteração, no caso incrementa a variável

Exemplos:

```
<?php
/* exemplo 1 - Controla o fluso no for*/
echo "<br><br>1- ";
for ($i = 1; $i <= 10; $i++) {
   echo $i;
}
echo "<br><br>2- ";
/* exemplo 2 - Controle o fluxo no if interno*/
for ($i = 1; ; $i++) {
   if ($i > 10) {
       break;
   }
   echo $i;
}
echo "<br><br>3- ";
/* exemplo 3 - Controle o fluxo no if interno*/
$i = 1;
for (; ; ) {
   if ($i > 10) {
       break;
   }
   echo $i;
   $i++;
}
echo "<br><br>4- ";
/* exemplo 4 */
for ($i = 1; $i <= 10;$i++);
    echo $i;
?>
```

foreach - laço para varrer os elementos de uma matriz (array).

Atentar para o fato de que o foreach não requer reset() antes, pois quando é executado o ponteiro interno do array já é zerado automaticamente.

Sintaxe:

```
foreach (expressao_array as $valor){
 instrucoes;
}
foreach (expressao_array as $chave => $valor) {
 instrucoes;
}
```

Exemplos:

```
<?php
//Você pode ter notado que os seguintes itens são funcionalmente idênticos:
$arr = array("um", "dois", "três");
reset ($arr); // Aponta para o primeiro elemento
while (list(, $value) = each ($arr)) {
   echo "Valor: $value<br />";
}
foreach ($arr as $value) {
   echo "Valor: $value<br />";
}

//Os seguintes também são funcionalmente idênticos:
$arr = array("one", "two", "three");
reset($arr);
while (list($key, $value) = each ($arr)) {
   echo "Chave: $key; Valor: $value<br />";
}
foreach ($arr as $key => $value) {
   echo "Chave: $key; Valor: $value<br />";
}

/* exemplo foreach 1: somente valores */
$a = array(1, 2, 3, 17);
foreach ($a as $v) {
   echo "Valor atual de \$a: $v.<br>";
}

/* exemplo foreach 2: valores (com as chaves impressas para ilustração) */
$a = array(1, 2, 3, 17);
$i = 0; /* para exemplo somente */
foreach ($a as $v) {
   echo "\$a[$i] => $v.<br>";
   $i++;
}

/* exemplo foreach 3: chaves e valores */
```

```
$a = array (
   "um" => 1,
   "dois" => 2,
   "três" => 3,
   "dezessete" => 17
);
foreach ($a as $k => $v) {
   echo "\$a[$k] => $v.<br>";
}
?>
```

break - cancela a execução de laços: for, foreach, while, do ... while ou switch. break n é suportado, onde n é o número de estruturas a serem canceladas.

Sintaxe:

```
break;

break n;
```

Exemplos:

```
<?php
$arr = array('um', 'dois', 'três', 'quatro', 'PARE', 'cinco');
while (list (, $val) = each ($arr)) {
   if ($val == 'PARE') {
      break;    /* Você poderia colocar 'break 1;' aqui. */
   }
   echo "$val<br />";
}
echo "<br />";
/* Utilizando o argumento opcional. */

$i = 0;
while (++$i) {
   switch ($i) {
   case 5:
      echo "No 5<br />";
      break 1;  /* Sai somente do switch. */
   case 10:
      echo "No 10; saindo<br />";
      break 2;  /* Sai do switch e while. */
   default:
      break;
   }
}
?>
```

continue - sai da atual iteração de um loop para continuar na próxima iteração.

Sintaxe:

```
continue;
continue n;
```

Enquanto o break encerra definitivamente um laço, o continue encerra somente a iteração atual.

Exemplos:

```
<?php
$arr = array(1,2,3,4,5,6,7,8);
while (list ($key, $value) = each ($arr)) {
   if (!($key % 2)) { // pula itens pares, ou seja, processa somente ímpares
      continue;
   }
   echo ($value);
}
echo "<br>";
$i = 0;
while ($i++ < 5) {
   echo "Fora<br />";
   while (1) {
      echo "  Meio<br />";
      while (1) {
         echo "  Dentro<br />";
         continue 3;
      }
      echo "Isto nunca será exibido.<br />";
   }
   echo "Nem isso.<br />";
}

// Outro exemplo
  for ($i = 0; $i < 5; ++$i) {
     if ($i == 2)
        continue
     print "$i\n";
  }
?>
```

switch - similar a uma série de construções if seguidas.

Sintaxe:

```
switch (variavel) {
    case valor1:
```

```
        instrucoes;
        break;
    case valor2:
        instrucoes2;
        break;
    case valorN;
        instrucoesN;
        break;
    default:
        instrucoesDefault;
        break;
}
```

Exemplos:

```
<?php

$i = 1;

// Estrutura com if
if ($i == 0) {
   echo "\$i igual a 0";
} elseif ($i == 1) {
   echo "\$i igual a 1";
} elseif ($i == 2) {
   echo "\$i igual a 2";
}
echo "<br>";
// Estruturas com switch
switch ($i) {
   case 0:
      echo "\$i igual a 0";
      break;
   case 1:
      echo "\$i igual a 1";
      break;
   case 2:
      echo "\$i igual a 2";
      break;
}
echo "<br>";
$i = 2;
// Executará todos, falta o break
switch ($i) {
   case 0:
      echo "\$i igual a 0";
   case 1:
      echo "\$i igual a 1";
   case 2:
      echo "\$i igual a 2";
```

```
}
echo "<br>";
// Simulando intervalos
switch ($i) {
  case 0:
  case 1:
  case 2:
     echo "\$i é menor que 3 mas não negativo";
     break;
  case 3:
     echo "\$i é 3";
}
echo "<br>";
// Valor default
switch ($i) {
  case 0:
     echo "\$i igual a 0";
     break;
  case 1:
     echo "\$i igual a 1";
     break;
  case 2:
     echo "\$i igual a 2";
     break;
  default:
     echo "\$i não é igual a 0, 1 ou 2";
}
?>
```

Estruturas em Obsolescência
Devemos evitar, pois em futuras versões não mais serão utilizados.
Funções com nomes antigos, tipo:
pg_numrown, pg_fecharray e similares.
Caso se faça uma busca por estas funções no site do PHP nada será retornado.
As funções atuais tem nomes com espaço separando palavras, como:
pg_num_rown, pg_fech_array e similares.
Algumas funções agora tem outro nome, por exemplo:
pg_exec agora deve ser substuida por pg_query
Algumas estruturas de controle usavam os dois pontos:
while: ... endwhile, if: ... endif, devemos preferir:
while(){ ... }, if(){ ... }
Evitar o uso do comentário tipo shell #.
Estruturas do tipo 4HTTP_POST_VARS e similares foram substituídas por: $_POST e similares.

Formatando Números
number_format
number_format -- Formata um número com os milhares agrupados
string number_format (float number [, int decimals])

string number_format (float number, int decimals, string dec_point, string thousands_sep)
number_format() retorna uma versão formatada de number. Esta função aceita um, dois ou quatro parâmetros (não três):
Se apenas um parâmetro é dado, number será formatado sem decimais, mas com uma virgula (",") entre cada grupo de milhar.
Se dois parâmetros são dados, number será formatado com o número de casas decimais especificadas em decimals com um ponto (".") na frente, e uma vírgula (",") entre cada grupo de milhar.
Se todos os quatro parâmetros forem dados, number será formatado com o número de casas decimais em decimals, dec_point ao invés do ponto (".") antes das casas decimais e thousands_sep ao invés de uma vírgula (",") entre os grupos de milhares.
Somente o primeiro caractere de thousands_sep é usado. Por exemplo, se você usar foo como o parâmetro thousands_sep no número 1000, number_format() irá retornar 1f000.
Exemplo 1. Exemplo number_format()
Por exemplo, a notação Francesa usa duas casas decimais, vírgula (',') como separador decimal, e espaço (' ') como separador de milhar. Isto é feito com a linha :

```
<?php
// string number_format ( float number, int decimals, string dec_point, string
thousands_sep )
$number = 1234.56;
// Notação Brasileira
$numero_format_brasil = number_format($numero, 2, ',', '.');
// 1.234,56
echo "O número' $number' no formato brasileiro fica '$numero_format_brasil'<br><br>";
?>
```

Validação de Ano Bisexto

```
<?php
function ano_bisexto($ano){
   return (((($ano%4)==0 && ($ano%100) != 0) || ($ano%400)==0);
}
if(ano_bisexto(2006))
   echo "Ano bisexto";
else
   echo "Ano não bisexto";
?>
```

Algumas Funções Matemáticas

abs -- Valor absoluto
mixed **abs** (mixed número)

```
<?php
$abs = abs(-4.2); // $abs = 4.2; (double/float)
$abs2 = abs(5);  // $abs2 = 5; (inteiro)
$abs3 = abs(-5);  // $abs3 = 5; (inteiro)
?>
```

ceil -- Arredonda frações para cima
float **ceil** (float valor)

```
<?php
```

```
echo ceil(4.3);    // 5
echo ceil(9.999);  // 10
?>
```

floor -- Arredonda frações para baixo
float **floor** (float valor)

```
<?php
echo floor(4.3);  // 4
echo floor(9.999); // 9
?>
```

max -- Localiza o maior número
mixed **max** (number arg1, number arg2 [, number ...])
mixed **max** (array numbers [, array ...])

```
<?php
echo max(1, 3, 5, 6, 7);  // 7
echo max(array(2, 4, 5)); // 5

echo max(0, 'hello');    // 0
echo max('hello', 0);    // hello
echo max(-1, 'hello');    // hello

// Com arrays múltiplos, max compara da esquerda para direita,
// assim nesse exemplo: 2 == 2, mas 4 < 5
$val = max(array(2, 4, 8), array(2, 5, 7)); // array(2, 5, 7)

// Se forem informados um array e um não array, o array
// é sempre retornado como se ele fosse o maior
$val = max('string', array(2, 5, 7), 42);  // array(2, 5, 7)
?>
```

min -- Localiza o menor número
mixed **min** (number arg1, number arg2 [, number ...])
mixed **min** (array numbers [, array ...])

```
<?php
echo min(2, 3, 1, 6, 7);  // 1
echo min(array(2, 4, 5)); // 2

echo min(0, 'hello');    // 0
echo min('hello', 0);    // hello
echo min('hello', -1);    // -1

// Com arrays múltiplos, min compara da esquerda para direita,
// assim nesse exemplo: 2 == 2, mas 4 < 5
$val = min(array(2, 4, 8), array(2, 5, 1)); // array(2, 4, 8)

// Se ambos forem um array e um não array, o array
// nunca será retornado porque ele é sempre considerado o maior
$val = min('string', array(2, 5, 7), 42);  // string
?>
```

count -- Conta o número de elementos de uma variável
int **count** (mixed var [, int mode])

```
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count($a);
// $result == 3

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$result = count($b);
// $result == 3;
?>
```

Exemplo 2. Uso recursivo da função count() (PHP >= 4.2.0)

```
<?php
$food = array( 'fruits'  => array('orange', 'banana', 'apple'),
'veggie'  => array('carrot', 'collard','pea'));
// recursive count
echo count($food,COUNT_RECURSIVE);  // mostra 8
// normal count
echo count($food);                  // mostra2 2
?>
```

pow -- Potência
number **pow** (number base, number exp)

```
var_dump( pow(2,8) ); // int(256)
echo pow(-1,20); // 1
echo pow(0, 0); // 1
echo pow(-1, 5.5); // erro
```

rand -- Gera um número aleatório
int **rand** ([int min, int max])

```
echo "Exibir 20 números aleatórios entre 20 e 160<br><br>";
for($x=20;$x<40;$x++){
   echo rand($x, 4*$x)."<br>";
}
exit("Saindo...");
```

round -- Arredonda um número
float **round** (float val [, int precision])

```
<?php
echo round(3.4);         // 3
echo round(3.5);         // 4
echo round(3.6);         // 4
```

```
echo round(3.6, 0);      // 4
echo round(1.95583, 2);  // 1.96
echo round(1241757, -3); // 1242000
echo round(5.045, 2);    // 5.04
echo round(5.055, 2);    // 5.06
?>
```

sqrt -- Raiz quadrada
float **sqrt** (float arg)

```
<?php
// Precisão depende de sua diretiva precision
echo sqrt(9); // 3
echo sqrt(10); // 3.16227766 ...
?>
```

Forçando a Limpeza do Cache

```
<?php
// Este é para os servidores de proxy. Diz para baixar, bypassando o proxy
header ("Cache-Control: no-cache, must-revalidate");
// Este é para o navegador e nem sempre funciona (falta de padrão entre eles)
header ("Pragma: no-cache");
?>
```

Redirecionamento de Páginas

```
header("location: novapagina.php");
```

Exibir o conteúdo de um arquivo
readfile -- Lê e exibe o conteúdo de um arquivo

```
readfile('http://www.google.com.br');
readfile('/home/1www/pagina2.php');
```

Enviar E-mail via PHP

```
<?php
$to = "ribamar.sousa@dnocs.gov.br";
$subject="Apenas testando";
$message = "Estou testando o envio de e-mail pelo PHP.";
$email="ribafs@gmail.com";
$ret=mail($to, $subject, $message, "From: $email\r\nReply-to: $email\r\n");
echo $ret;
?>
```

Usando Favicon em sites
Aquele pequeno ícone que fica à esquerda da URL do site, na caixa Location do Browser, que inclusive personaliza a barra de links quando arrastamos o endereço do site.
Para que nosso site apareça para o visitante com um link, devemos seguir os seguintes procedimentos:
- Criar uma imagem com o logo do site, no formato png, jpg, gif, etc com 32x32 pixels
- Salvar como favicon.png ou outra extensão
- Adicionar a TAG abaixo, na TAG head, como abaixo:

```
<head>
...
```

<link href="http://www.seusite.com.br/diretorio/favicon.png" type="image/gif" rel="icon">
...
</head>
Também pode ser assim:
rel="shortcut icon"
Este ícone pode ser inclusive animado.

Referência: lista da Dicas-L (http://www.dicas-l.com.br).

Livros
Desenvolvendo Web Sites com PHP
Editora Novatec
Juliano Niederauer
PHP para quem conhece PHP
Editora Novatec
Juliano Niederauer
PHP Guia do Desenvolvedor
Ed. Berkeley
Sterling Hughes
(Do grupo de desenvolvimento do PHP)
Desvendando aplicações na Web com PHP 4.0
Ed. Ciência Moderna
Tobias Retschiller e
Till Gerken

Links
PHP
http://www.php.net
http://www.php.net/manual/pt_BR/
http://www.php.net/downloads.php
http://www.phpbrasil.com/
http://www.planet-source-code.com/vb/default.asp?lngWId=8
http://www.hotscripts.com/PHP/index.html
http://www.zend.com (empresa dos dois membros israelenses da equipe de desenvolvimento
do PHP, especial ênfase na seção Developer Zone)
http://www.phpwizard.net
http://www.phpclasses.org
http://www.weberdev.com
http://www.devshed.com
http://www.phpmania.org/
http://www.phpnet.us/ - Hospedagem free para PHP e MySQL (300MB)
http://www.superphp.com.br
http://www.scriptbrasil.com/
http://www.faqts.com/knowledge_base/index.phtml/fid/51/
http://www.alt-php-faq.org/
http://ribafs.byethost2.com/
CURSOS GRÁTIS
http://cursos.cdtc.org.br/brasil/ - Cursos Grátis para Func.Públ.
http://www.apostilando.com/download.php?cod=171&categoria=PHP

http://www.aprendaemcasa.com.br/apcasa6.htm
http://www.solocursosgratis.com/cursos_gratis_php-slctema264.htm
CLIPARTS
http://www.digitmania.holowww.com/digital.html – Dígitos (cliparts)
EDITORES
http://paginas.terra.com.br/informatica/php_editor/
POSTGRESQL
http://www.postgresql.org/docs/current/interactive/
http://pgdocptbr.sourceforge.net/pg80/index.html
http://www.designmagick.com/category/3/PostgreSQL
MYSQL
http://dev.mysql.com/doc/refman/4.1/pt/index.html
http://dev.mysql.com/doc/refman/5.0/en/index.html
SQL
http://www.firstsql.com/tutor.htm
http://sqlzoo.net/
http://www.sql-tutorial.net/SQL-tutorial.asp
http://www.programmingtutorials.com/sql.aspx
http://tutorials.findtutorials.com/
http://www.hardened-php.net/home.8.html - Hardened PHP Project
http://www.1phpscripts.com/
http://www.weberdev.com/
http://www.goldsofts.com/scriptscategory/10/0/1/0.html

A História do PHP

PHP 5
PHP 5 foi lançado em julho de 2004 depois de um longo desenvolvimento e vários pre-releases. Ele principalmente introduziu o core, a Zend Engine 2.0 com um novo modelo de orientação a objetos e várias outras características.

Além da orientação a objetos, a versão 5 também trouxe funcionalidades muito importantes para a evolução da linguagem, como SimpleXML, SOAP, MySQLi e mais diversas extensões de importância significativa.

No dia 11 de novembro de 2005, Rasmus, Zeev e outros desenvolvedores do núcleo do PHP, se reuniram em Paris para discutir sobre o futuro da linguagem. Dessa reunião surgiu o documento "Minutes PHP Developers Meeting" escrito por Derick Rethans um dos atuais desenvolvedores do núcleo do PHP.

Esse documento passou a ser o guia para o desenvolvimento do PHP 6.

Hoje (12/08/2006), a edição 5 do PHP está em sua versão 5.1.4 e a edição 4 está na versão 4.4.3. O PHP continua crescendo e evoluindo muito e a cada versão traz novas funcionalidades e facilidades para os profissionais que utilizam a linguagem. (http://www.htmlstaff.org/ver.php?id=327)

PHP 6
PHP 6 - A revolução

Neste artigo falarei sobre o PHP 6 as boas notícias e as más notícias que estão por vim, então vamos lá.

Começando

Quando agente começou a se acostumar com o PHP 5 já esta mudando, não é? Como falam a "felicidade de pobre dura pouco.". Mais não vamos desanimar, e aliás o PHP 6 já a um tempinho atrás á ser desenvolvido, e uma das promessas é o suporte ao Unicode, que vai permitir a criação e gestão mais fácil de aplicações internacionalizáveis.

O maior colaborador do PHP o Derick Rethans, falou sobre o PHP 6 á um tempinho atrás. Com o PHP 5.1 vindo aí, fica a pergunta, como vão chamar a próxima versão do PHP??? O Derick deu a sugestão de PHP 6 que já é oficial, isso porquê vai ter uma grande mudança no suporte ao Unicode.

4 dias depois de Derick ter mostrado suas sugestões e opiniões sobre a nova versão do PHP, Rasmus, o pai do PHP, deixou uma lista de características para próxima versão, nas quais acho positivas para mim:

- Será removido o register_globals;
- Será removido também o magic_quotes_;

- Adicionar a extensão do filtro da entrada que incluirá um mecanismo para que os colaboradores de aplicação o girem muito facilmente fora de qual trocaria as disposições cruas de GPC para trás caso que o local o teve girado sobre pelo defeito.
- Incluir um "esconderijo" do opcode pelo defeito. Os muitos do trabalho têm entrado no pecl/apc recentemente, mas eu não sou pendurado acima em qual um vai.
- Remover o safe_mode e focalizá-lo no open_basedir;
- Remover algum material que foi marcado como "deprecated" desde PHP 3/4;
- Fazer identificadores caixa-sensíveis;
- Remover os vários pseudônimos da função.

Se prepare para o PHP 6

Unicode

A sustentação de Unicode no presente pode ser ajustada em a por a base do pedido. Isto iguala a PHP que tem que armazenar Unicode e variants do non-Unicode de nomes da classe, do método e da função nas tabelas de símbolo. No short - usa-se acima de mais recursos. Sua decisão deve fazer o Unicode que ajusta o usuário largo, para não pedir largamente. Unicode de giro fora onde não requerido pode ajudar ao desempenho e citam alguma corda funcionam como sendo até aplicações mais lentas e inteiras de 300% 25% mais lento em conseqüência. A decisão para movê-la para o php.ini em minha mente faz exame do controle away do usuário, e põe-no nas mãos do anfitrião da correia fotorreceptora.

Se você compilar PHP você mesmo ou for responsável para este em seus usuários então você pode ser interessado saber que PHP 6 requererá os libs de ICU (de qualquer maneira se Unicode for desligado sobre ou). O sistema da configuração afiançará para fora se os libs requeridos de ICU não puderem ser encontrados. Em um nutshell, você terá uma outra coisa a instalar se você quiser compilar PHP.

Registo Globals a ir

Diga adeus povos, este está indo finalmente. Será não mais por muito tempo um ajuste da lima do ini, e se encontrado lhe levantar um E_CORE_ERROR, apontá-lo à documentação sobre porque é "bad". Isto significa que PHP 6 quebrará finalmente todos os certificados da era PHP3 (ou algum certificado usando globals do registro) com nenhum recourse em tudo com exceção de para re-code o. Aquele é um movimento bold(realce), mas needed.

Citações da mágica a ir

A característica mágica das citações de PHP estará indo, e como com globals do registo está indo levantar um E_CORE_ERROR se o ajuste for encontrado em qualquer lugar. Isto afetará magic_quotes, magic_quotes_sybase e magic_quotes_gpc.

Modalidade segura a ir

Isto pode satisfazer os colaboradores que têm os anfitriões da correia fotorreceptora que

insistem em cima da modalidade segura! Mas irá agora totalmente, outra vez levantando um E_CORE_ERROR se encontrado. A razão é que aparentemente sentiram que deu ' o sinal errado ', implicando que fêz PHP seguro, quando o infact ele não em toda vontade do open_basedir (thankfully) seja mantido.

' var ' ao pseudônimo ' público '

' var usado PHP4 ' dentro das classes. PHP5 (em seu movimento de OO) fêz com que isto levantasse um aviso sob E_STRICT. Este aviso será removido em PHP 6 e preferivelmente ' o var ' significará a mesma coisa que ' o público '. Este é um movimento agradável mas I se qualquer um atualizar seus certificados para o trabalhar sob E_STRICT em PHP5 será redundante para ele.

Retorne pelo erro da vontade de Referência

Ambos ' # = & StdClass() novo ' e ' o &foo da função ' levantarão agora um erro de E_STRICT.

Modalidade do compatbility zend.ze1 a ir

ze1 tentado sempre retê-lo o comportamento PHP4 velho, mas aparentemente "não trabalha 100%" de qualquer maneira, assim que será removido totalmente e jogar um E_CORE_ERROR se detectado.

Freetype 1 e sustentação de GD 1 a ir

A sustentação para ambos estes libs (muito muito velhos) será removida.

o dl() move-se para SAPI somente

Cada SAPI registará o uso desta função como necessário, only os CLI e encaixam SAPIs farão isto de agora sobre. Não estará disponível em outra parte.

FastCGI sempre sobre

O código de FastCGI será limpado acima e permitido sempre para o cgi SAPI, não poderá ser incapacitado.

Disposições longas do registo a ir

Recorde os globals de HTTP_*_VARS do yesteryear? Jorre se você não estão usando já $_get, $_post, etc. - comece fazer assim agora, porque a opção para permitir disposições longas está indo (e jogará um E_CORE_ERROR).

Movimentos da extensão

As extensões de XMLReader e de XMLWriter mover-se-ão na distribuição do núcleo e ser-se-ão sobre optam perto.

A extensão do ereg mover-se-á para PECL (e para ser removido assim de PHP). Isto

significa que PCRE não estará permitido ser incapacitado. Isto fará a maneira para a extensão regular nova da expressão baseada em ICU.

O exntesion extremamente útil de Fileinfo mover-se-á na distribuição do núcleo e permitido pelo defeito.

Adições ao motor do PHP

64 inteiros do bocado

Um inteiro novo de 64 bocados será adicionado (int64). Não haverá nenhum int32 (se supõe a menos que você especificar int64)

Goto

Nenhum comando ' goto ' será adicionado, mas o keyword da ruptura será estendido com uma etiqueta de estática - assim que você poderia fazer ' o foo da ruptura ' e saltará ao foo da etiqueta: em seu código.

ifsetor()

Olha como nós não estaremos vendo este, que é um shame. Mas preferivelmente?: o operador terá ' a exigência do parâmetro médio ' deixada cair, que os meios você poderiam fazer algo como este: "# = $_get['foo ']?: 42;" (isto é se o foo for verdadeiro, $foo igualará 42). Isto deve conservar algum código, mas eu pessoalmente não penso que é como ' readable ' porque o ifsetor seria.

disposições multi-não ofuscantes do foreach

Esta é uma mudança agradável - você poderá ao foreach através das listas de disposição, isto é. "foreach($a como $k = > list($a, #))".

{} contra [ ]

Você pode atualmente usar-se {} e [ ] alcançar índices da corda. Mas {} a notação levantará um E_STRICT em PHP5.1 e será ida totalmente em PHP 6. Também [ ] a versão ganhará a funcionalidade do substr e do array_slice diretamente - assim que você poderia fazer "[2,]" para alcançar os caráteres 2 à extremidade, etc.. Muito acessível.

Mudanças de OO

Emperramento De estática

Um keyword novo será criado para permitir o emperramento atrasado da estática - static::static2(), este executarão a avaliação runtime do statics.

Namespaces

Olha como este é ainda undecided - se executam namespaces que estará usando seu

estilo somente. Meu conselho? Não prenda sua respiração!

Valores Do retorno Tipo-sugeridos

Embora se decidissem de encontro a permitir propriedades tipo-sugeridas (becaue é "não a maneira de PHP") que adicionarão a sustentação para valores do retorno tipo-sugeridos, mas têm para decidir-se ainda em uma sintaxe para isto. Mesmo assim, será uma adição agradável.

Chamando funções dinâmicas como a vontade E_FATAL da estática

No momento em que você pode se chamar métodos de estática e dinâmicos, se são de estática ou não. Chamar uma função dinâmica com a sintaxe de estática da chamada levantará um E_FATAL.

Adições a PHP

Apc está na distribuição do núcleo

O esconderijo APC do opcode será incluído na distribuição do núcleo de PHP como o padrão, entretanto não será girado sobre o defeito (mas o ter conserva a compilação de contudo uma outra coisa em seu usuário, e os anfitriões da correia fotorreceptora são mais prováveis permitir que seja permitido)

Remendo endurecido de PHP

Este remendo executa um grupo da segurança extra verifica dentro PHP. Ultrapassaram ele e as seguintes mudanças ocorrerão agora dentro de PHP: A proteção de encontro a rachar da resposta do HTTP será incluída allow_url_fopen será rachada em dois: allow_url_fopen e allow_url_include. allow_url_fopen serão permitidos pelo allow_url_include do defeito serão incapacitados pelo defeito.

E_STRICT funde em E_ALL

O wow, este é completamente sério! As mensagens do nível de E_STRICT serão adicionadas a E_ALL pelo defeito. Isto mostra que um movimento marcado pela equipe de PHP educar colaboradores ' em mais melhor praticou e indicando avisos do língua-nível no "hey, você o está fazendo a maneira errada".

Farewell < %

Removerão a sustentação para os Tag do estilo do ASP, mas o Tag do curto-código de PHP remanescerá (<?) - assim àqueles no general do php que contam o curto-Tag são ' depreceated ' - hah! ; -)

Conclusão

PHP 6 está fazendo exame de um movimento interessante em minha mente - é como se os colaboradores de PHP querem educar agora colaboradores sobre a maneira direita codificar algo, e remove aquelas edições lingering com o "bom você DEVE fazê-lo esta

maneira, mas você pode imóvel fá-la a maneira velha". Este não será o caso mais longo. Removendo totalmente os gostos de globals do registo, as citações da mágica, long disposições, {} índices da corda e as cham-tempo-pass-por-referências forçarão colaboradores a limpar acima de seu código.
Quebrará também um crapload dos certificados além do reparo que não envolve algum reescrever sério. É isto um a coisa má? Eu não penso assim que eu mesmo, mas eu v o fazer o adoption de mais lento PHP 6 uniforme do que aquele de PHP5, que é um shame real. Entretanto têm que pular este obstáculo em algum ponto, e uma vez que o fizeram a progressão às versões futuras deve ser mais rápida.

Texto traduzido de:
Se prepare para o PHP 6 - CorePHP

Autor/fonte: Lion
E-mail/Url: http://www.ievolutionweb.com/tutorial/php/12558/php_6_a_revolucao.htm

Características e Recursos do PHP

Da Wikipedia:
A linguagem PHP é uma linguagem de programação de domínio específico, ou seja, seu escopo se estende a um campo de atuação que é o desenvolvimento web, embora tenha variantes como o PHP-GTK. Seu propósito principal é de implementar soluções web velozes, simples e eficientes. Características:[*carece de fontes*]

- Velocidade e robustez
- Estruturado e orientação a objetos
- Portabilidade - independência de plataforma - escreva uma vez, rode em qualquer lugar
- Tipagem dinâmica
- Sintaxe similar a C/C++ e o Perl
- Open-source

Do site oficial:
Autenticação HTTP com PHP

- Cookies
- Sessões
- Lidando com XForms
- Gerenciar o upload de arquivos
    - Upload de arquivos com o método POST
    - Explicação das mensagens de erro
    - Problemas comuns
    - Carregando múltiplos arquivos
    - Suporte ao método PUT
- Usando arquivos remotos
- Tratamento de Conexões
- Conexão Permanente com o Banco de Dados
- Safe Mode
    - Security and Safe Mode

        - Functions restricted/disabled by safe mode
    - Command line usage — Using PHP from the command line
        - Introduction
        - Differences to other SAPIs
        - Options — Command line options
        - Usage — Executing PHP files
        - I/O streams — Input/output streams
        - Interactive shell
        - Built-in web server
        - INI settings
    - Garbage Collection
        - Reference Counting Basics
        - Collecting Cycles
        - Performance Considerations

Grandes Projetos em PHP
- Joomla - http://joomla.org
- Drupal - http://www.drupal.org
- WordPress - http://www.wordpress.org
- phpBB - http://www.phpbb.com
- osCommerce - http://www.oscommerce.com
- Moodle - http://moodle.org
- phpMyAdmin - http://www.phpmyadmin.net
- phpPgAdmin - http://phppgadmin.sf.net
- OpenX - http://www.openx.org
- osTicket - http://www.osticket.com
- Elgg (rede social) - http://elgg.org
- E-commerce - http://prestashop.com
- EyeOS (nuvens) - http://eyeos.com/
- Wikipédia - http://www.wikipedia.org
- Prestashop - http://prestashop.com
- SourceForge - http://sourceforge.net/
- Eclipse - http://www.eclipse.org
- Facebook - http://www.facebook.com/
- Flickr - http://www.flickr.com/
- Digg - http://www.digg.com

Funciona em quais sistemas operacionais?

Pode ser utilizado na maioria dos sistemas operacionais,
- Linux,
- Várias variantes Unix (incluindo HP-UX, Solaris e OpenBSD),
- Microsoft Windows,
- Mac OS X,
- RISC OS,
- E provavelmente outros.

Funciona em quais servidores web?

O PHP também é suportado pela maioria dos servidores web atuais,
- Apache,
- Microsoft Internet Information Server (IIS),
- Microsoft 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.

Suporta o Paradigma de Orientação a Objetos?
Com o PHP5 o PHP ganhou um modelo de orientação a objetos completo.

Talvez **a maior força do PHP** seja **sua integração com Sistemas de Bancos de Dados**, e faz isso com grande facilidade.

Suporta a maioria dos SGBDs existentes:
* Adabas D
* dBase
* Empress
* FilePro (read-only)
* Hyperwave
* IBM DB2
* Informix
* Ingres
* InterBase
* FrontBase
* mSQL
* Direct MS-SQL
* MySQL
* ODBC
* Oracle (OCI7 and OCI8)
* Ovrimos
* PostgreSQL
* SQLite
* Solid
* Sybase
* Velocis
* Unix dbm

Também foi providenciado uma abstração de banco de dados (chamada PDO) permitindo a você utilizar qualquer banco de dados transparentemente com sua extensão. 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.

Alguns recursos do PHP

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, on the fly. 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.

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. Vários utilitários de compressão (gzip, bz2, zip), calendário e conversões de datas, etc.
Para uma boa relação dos recursos existentes:
http://www.php.net/manual/pt_BR/funcref.php
http://www.php.net/manual/pt_BR/extensions.alphabetical.php
PHP no desktop - http://gtk.php.net/
PHP Estruturado

Aqui eu quis ter pelo menos um pouco de tudo que diz respeito a criação de aplicativos em PHP.

Grandes Projetos em PHP
- Joomla - http://joomla.org
- Drupal - http://www.drupal.org
- WordPress - http://www.wordpress.org
- phpBB - http://www.phpbb.com
- osCommerce - http://www.oscommerce.com
- Moodle - http://moodle.org
- phpMyAdmin - http://www.phpmyadmin.net
- phpPgAdmin - http://phppgadmin.sf.net
- OpenX - http://www.openx.org
- osTicket - http://www.osticket.com
- Elgg (rede social) - http://elgg.org
- E-commerce - http://prestashop.com
- EyeOS (nuvens) - http://eyeos.com/
- Wikipédia - http://www.wikipedia.org
- Prestashop - http://prestashop.com
- SourceForge - http://sourceforge.net/
- Eclipse - http://www.eclipse.org
- Facebook - http://www.facebook.com/
- Flickr - http://www.flickr.com/
- Digg - http://www.digg.com

Funciona em quais sistemas operacionais?

Pode ser utilizado na maioria dos sistemas operacionais,
- Linux,
- Várias variantes Unix (incluindo HP-UX, Solaris e OpenBSD),
- Microsoft Windows,

- Mac OS X,
- RISC OS,
- E provavelmente outros.

Funciona em quais servidores web?

O PHP também é suportado pela maioria dos servidores web atuais,
- Apache,
- Microsoft Internet Information Server (IIS),
- Microsoft 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.

Suporta o Paradigma de Orientação a Objetos?
Com o PHP5 o PHP ganhou um modelo de orientação a objetos completo.

Talvez **a maior força do PHP** seja **sua integração com Sistemas de Bancos de Dados**, e faz isso com grande facilidade.

Suporta a maioria dos SGBDs existentes:

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

Também foi providenciado uma abstração de banco de dados (chamada PDO) permitindo a você utilizar qualquer banco de dados transparentemente com sua extensão. 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.

Alguns recursos do PHP

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, on the fly. 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.

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. Vários utilitários de compressão (gzip, bz2, zip), calendário e conversões de datas, etc.

Para uma boa relação dos recursos existentes:
http://www.php.net/manual/pt_BR/funcref.php
http://www.php.net/manual/pt_BR/extensions.alphabetical.php

PHP no desktop - http://gtk.php.net/

Algumas das Melhores Práticas para Iniciar em PHP

Tradução livre e adaptação do original em inglês: http://net.tutsplus.com/tutorials/php/30-php-best-practices-for-beginners/

1 – Seja amigo do manual do PHP e da busca de funções em:
Manual - http://www.php.net/manual/pt_BR/
Busca por Funções - http://www.php.net/
Muito rico: sintaxe, explicações e vários exemplos de uso.

2 – Ative a reportagem de erros.
http://www.php.net/manual/en/function.error-reporting.php
Se não estiver ativa você não receberá nenhuma indicação sobre os erros. Quando ativa você será ajudado, com mensagens de erro contendo muitos detalhes que o ajudarão a corrigir o erro.

3 – Adote uma IDE. Experimente:
NetBeans for PHP – http://www.netbeans.org
Eclipse PDT – http://www.eclipse.org/pdt

Ambas com ótimos recursos que tornarão seu trabalho de programador mais produtivo:
– destaque de sintaxe

– auto-completar o código
– avisos de erros (tratamento de erros)

E muitos outros bons recursos.

4 – Atente para não se repetir no código (DRY). Sempre que existir um código que você usa em vários locais, procure criar uma função e evite ao máximo copiar e colar código.

5 – Para grande clareza do seu código sempre use indentação. Alguns códigos sem indentação ficam quase ilegíveis, em especial em estruturas de controle.

6 – Seja organizado e sempre procure separar seu código em camadas coerentes. Seu código JavaScript deve ficar em um arquivo .js. Seu código CSS deve ficar em um arquivo separado .css.
Assim como deve ter um arquivo de funções, um arquivo com configurações, um diretório para os templates, um para as imagens, etc.

7 – Sempre usa as tags:

<?php

?>

São as mais completas, as únicas que suportam o XML. Evite as outras possíveis.

8 – Outro assunto importante é o sistema de nomeação de variáveis, funções, constantes e classes.
Use um sistema coerente. Não precisa seguir nenhum padrão existente, embora não seja também problema se o fizer, mas o importante é que sempre siga o mesmo padrão.

9 – Comentários. Estes tanto são importantes para quem ler seu código quanto para você mesmo. Após algum tempo de ter criado um código e precisarmos efetuar alguma manutenção, caso não tenha nenhum comentário, corre o risco de você ter dificuldade. Não são somente os comentários que são importantes, é toda a forma de programar: indentação, procurando ser claro, consiso e simples, seguindo padrões, etc. Mas os comentários ajudam muito. Para ajudar tem o PHP Documentor (http://www.phpdoc.org).

10 – Para iniciar use um bom pacote instalador. Existem vários, mas meu preferido é o Xampp, por varios motivos. Experimente um e o adote:

http://xampp.sf.net

11 – Existe uma função no PHP, que controla o tempo de execução dos scripts, que chama-se,
set_time_limit() - http://php.net/manual/en/function.set-time-limit.php.
Esta função evita loops infinitos e timeout da conexão com o banco de dados. Ela define a quantidade máxima de segundos em que a função deve rodar (default é 30 segundos). Após esse tempo um erro fatal será disparado.

12 – Use POO somente quando necessário e realmente se caracterizar o uso de objetos.

13 – Evite o uso de frameworks, pois geralmente engessam qualquer manutenção ou alguma alteração que precise fazer. Frameworks devem ser usados apenas quando você usa exatamente os mesmos estilos de aplicativos. Melhor é ir guardando os exemplos usados, as funções, rotinas e com o tempo ter sua biblioteca que irá facilitar a construção de aplicativos.

14 – Evite o uso de aspas duplas. Somente use quando necessário. As aspas simples são mais seguras e com melhor desempenho.

15 – Caso crie um arquivo contendo a função phpinfo() em seu servidor de produção, remova-o após a execução. É muita informação que poderá cair em mãos erradas.

16 – Sempre valide os dados recebidos dos usuários. Por mais confiáveis que eles sejam eles estão sujeitos a erros. Valide sempre antes de armazenar no banco.

17 – Armazene senhas criptografadas no banco.

18 – Use visualizadores de dados dos SGBDs. Para MySQL use o DBDesigner (http://fabforce.net/dbdesigner4/) ou o MySQL WorkBench (http://dev.mysql.com/workbench/).
Para PostgreSQL use DBVisualizer (http://www.dbvis.com).

19 – Proteja seus script contra Injeção em SQL. Para MySQL use a função mysql_real_escape_string(): http://us3.php.net/mysql_real_escape_string
Para PostgreSQL existe a função pg_escape_string() e pg-escape-bytea():
http://www.php.net/manual/pt_BR/function.pg-escape-string.php
http://www.php.net/manual/pt_BR/function.pg-escape-bytea.php
E para reforçar a segurança:
$my_data = pg_escape_string(utf8_encode($_POST['my_data']));

20 – Valide dados de Cookies. Para isso use as funções htmlspecialchars(), mysql_real_escape_string e pg_escape_string():
http://us3.php.net/manual/en/function.htmlspecialchars.php
http://us3.php.net/mysql_real_escape_string

21 – Mantenha funções fora de laços.

22 – Trabalhe sempre com as versões mais recentes dos softwares que utiliza, inclusive PHP, Apache, MySQL, PostgreSQL. As versões mais recentes corrigem bugs, trazem novas e importantes funcionalidades.

23 – Assine boas listas de discussão. Visite bons forums e bons sites, assim como também em ainda sobrando alguma dúvida procure no Google.

Veja estes recursos:
http://www.php.net/support.php

2 - PHP Básico

Tags de início e final do script PHP

Para criar um arquivo/script em PHP, que seja interpretado pelo Apache, devemos marcar o início e o final deste script com tags que o Apache conheça. Essas tags podem ser configuradas mas idealmente devemos usar as tags abaixo, pois são as mais completas e recomendadas:

```
<?php

?>
```

A criação de um script em PHP é muito simples, basta conter as tags inicial e final e as instruções em PHP, como por exemplo:

```
<?php
print "Como vai?";
?>
```

Que exibirá a frase (Como vai?) na tela.

Outro pequeno e útil exemplo:

```
<?php
echo 'Versão do PHP: ' .phpversion();
?>
```
Este mostrará a versão do PHP instalado;

Mais um útil:
```
<?php
phpinfo();
?>
```

Separador de Instruções

Observe que ao final da linha:

```
print "Como vai?";
```

adicionamos um ponto e vírgula (;). Isso diz ao Apache que ele pode executar todo o trecho que esocntra-se à esquerda do ;. No caso, o ; é usado para separar instruções.

Embora possamos escrever código assim:

```
<?php
print "Como vai?"; $c = 0;
?>
```
Isso não é recomendado, pois dificulda a leitura do código e pode ocorrer que não

percebamos o que estiver à direita do ;.

Recomendação: Portanto procure escrever código com apenas uma instrução por linha.

Comentários

Em PHP existem três formas de se escrever comentários:

```
// - este é comentário oriundo do C++ e para apenas uma linha
/*
Este é o comentário oriundo do C
e para
múltiplas linhas
*/
# - Este é o comentário oriundo do shell, para uma única linha e que deve ser evitado, pois está em processo de obsolescência.
```

Variáveis

As variáveis em PHP recebem o $ à esquerda do nome, como no exemplo abaixo:

```
$nome_variavel = valor;

$minha_variavel = 'Valor da variável';
$x = 15;
```

Dica: em PHP não se declara o tipo da variável. O tipo é interpretado em tempo de execução pelo seu conteúdo.

Nomeando Variáveis

–O nome das variáveis precisa começar com uma letra ou pelo símbolo de sublinhado
–O nome de variáveis pode conter somente caracteres alfa-numéricos e sublinhado: a-z, A-Z, 0-9 e _ .
–Não pode conter espaços. Quando várias palavras podemos usar o sublinhado para separá-las: $minha_variavel.

As variáveis podem ter seu valor alterado durante a execução do script.

Constantes

As constantes, ao contrário das variáveis, não podem ter seu valor alterado durante a execução do script. Por isso são indicadas para maior segurança em alguns casos, como dados da conexão com o banco, que não se alteram durante toda a execução do script.

Criando uma constante:
define('NOME', 'Valor');

Por convenção (não obrigatoriamente), os nomes de constantes são escritos com todas as letras em maiúsculas.

define('HOST', 'localhost');

Usando uma constante:

```
<?php
define('HOST', 'localhost');
print HOST;
?>
```

O uso de constantes não requer o $ à esquerda como nas variáveis, usamos apenas seu nome.

Strings

O PHP tem uma boa quantidade de funções para trabalhar com strings e lida bem com elas.
Uma string é um conjunto de caracteres delimitado por aspas ou apóstrofos (chamados de aspas simples).

Exemplo:
$str_curso = 'Curso de PHP';
$valor = 30;

Podemos escrever assim:

print "$str_curso com $valor";

Mas devemos preferir assim:

print $str_curso . ' com ' . $valor . ' alunos';

Concatenação de strings

A concatenação de strings é feita usando o operador ponto (.), como no exemplo acima.

Também podemos concatenar assim, com .=:

$str_curso_avançado .= ' Avançado';

Dica: só devemos usar aspas (aspas duplas) quando necessário. Devemos dar prioridade aos apóstrofos (aspas simples), por motivos de segurança e desempenho.

Textos ou Strings Grandes
Quando temos uma string grande a forma mais prática é usar o demilitador chamado heredoc:
$str = <<< END
Eu estou apenas testando este recurso.
Quando temos uma string grande a forma mais prática é usar o
demilitador chamado heredoc.
Da forma sugerida aqui.
END;

Podendo a constante receber qualquer nome:

$str = <<< CURSO
Eu estou apenas testando este recurso.
Quando temos uma string grande a forma mais prática é usar o
demilitador chamado heredoc.
Da forma sugerida aqui.
CURSO;

Operadores

Atribuição

Para atribuir o PHP usa o que a maioria das linguagens utiliza:

$x = 4;

Com isso estamos atribuindo o valor 4 para a variável $x.

Comparação

Para comparar valores o PHP usa = =, diferente de algumas linguagens, que utilizam apenas =.

if($x = = 4)

Com isso apenas comparamos se a variável $x tem valor igual a 4.

Se quizermos que seja comparado o valor e o tipo da variável devemos utilizar = = =:

if($x = = = 4)

Tipos de Dados das Variáveis e Conversão (CAST):

```
$x = 3.14159;
echo gettype($x);
echo " :: $x<br>";        // double :: 3.14
settype($x, "string");
echo gettype($x);
echo " :: $x<br>";        // string::3.14
echo gettype((integer)$x);
echo " :: $x<br>";        // integer::3.14
echo gettype($x);
echo " :: $x<br>";        // string::3.14
settype($x, "integer");
echo gettype($x);
echo " :: $x<br>";        // integer::3
settype($x, "double");
echo gettype($x);
echo " :: $x<br>";        // float::3.0
settype($x, "string");
echo gettype($x);
echo " :: $x<br>";        // string::3.0
settype($x, "boolean");
echo gettype($x);
echo " :: $x<br>";        // boolean::1
settype($x, "integer");
echo gettype($x);
echo " :: $x<br>";        // integer::TRUE (1)
$x = FALSE;
settype($x, "boolean");
echo gettype($x);
echo " :: $x<br>";        // boolean::FALSE (0) (o que faz não mostrar nada)
```

Estruturas de Controle

if, if ... else, if... elseif... else

Como o PHP é oriundo do C, então sua sintaxe é semelhante às demais linguagens que tiveram a mesma origem como C++, Java, JavaScript etc.

Sintaxe:

```
if(condicao){
        instruções;
}
```

Exemplo:

```
if($x = = 3){
        print 'x vale '.$x;
}
```

Sintaxe:
```
if(condicao){
	instruções para condição verdadeira;
}else{
	instruções para condição falsa;
}
```

Exemplo:

```
if($x = = 3){
	print 'x vale '.$x;
}else{
	print 'x não vale '.$x;
}
```

Sintaxe:
```
if(condicao){
	instruções para condição verdadeira;
}elseif(condicao2){
	instruções para condição2 verdadeira;
}else{
	instruções para condição2 falsa;
}
```

Exemplo:

```
if($x = = 3){
	print 'x vale '.$x;
}elseif($x>3){
	print 'x maior que 3 ';
}else{
	print 'x menor que 3 ';
}
```

Expressões Condiçõnais ou Operador Ternário

```
$x = 3;
print ($x > 4) ? print 'Maior' : 'Menor';
```

Imprimirá Menor.

for

O laço for tem a seguinte sintaxe em geral:

Sintaxe:
```
for(inicial, condicao, incremento){
	Código a ser executado;
}
```

Exemplo:

```
for($x=0; $x<5;$x++){
        print 'X vale '. $x.'<br>';
}
```

Imprimirá:

```
X vale 0;
X vale 1;
X vale 2;
X vale 3;
X vale 4;
```

Observe que variará de 0 até 4 (menor que 5).

while

Sintaxe:

```
while (condicao){
        Código a ser executado;
}
```

Exemplo:

```
$x = 5;
$y = 3;
while ($y < $x){
        print 'Y vale '.$y.'<br>';
        $y = $y +1;
}
```

Imprimirá:

```
Y vale 3;
Y vale 4;
```

do ... while

Sintaxe:

```
do{
        Código a serexecutado;
}while (condicao);
```

Observe que o código no do ... while será executado incondicionalmente pelo menos uma vez, ou seja, na primeira vez é executado sem se testar a condição e somente da segunra

vez em diante testa-se a condição.

Exemplo:

```
$i=1;
do{
	$i++;
	echo "O número vale " . $i . "<br />";
}while ($i<=5);
```

Que imprimirá:

```
O número vale 2
O número vale 3
O número vale 4
O número vale 5
O número vale 6
```

switch

Sintaxe:

```
switch ($variavel){
	case expressao1:
		Código a ser executado de $variavel = expressao1;
		break;
	case expressao2:
		Código a ser executado de $variavel = expressao2;
		break;
	default:
		Código a ser executado de $variavel  diferente de expressao1 e de
expressao2;
}
```

Exemplo:

```
$x=2;
switch ($x)
{
case 1:
  echo "Número 1";
  break;
case 2:
  echo "Número 2";
  break;
case 3:
  echo "Número 3";
  break;
default:
  echo "Nenhum Número entre 1 e 3";
```

```
}
```

continue e break

Os laços for, while, do...while e switch podem ser interrrompidos com os comandos:
continue - interrompe a iteração atual e volta o processamento para o início do laço.
break - interrompe o laço e passa o processamento para a linha seguinte ao final do laço.

Lembrando que a iteração é o processo que é executado a cada volta do laço. Um laço com:
for($x=0;$x<4;$x++) será executado 4 vezes: 0, 1, 2, 3, ou seja, terá 4 iterações.

Exemplo do Continue:

```
for ($x = 0; $x < 5; ++$x) {
    if ($x == 2){
        continue;
          }
          print "$x<br>";

}
```

Imprimirá:

```
0
1
3
4
```

Exemplo do Break:

```
for ($x = 0; $x < 5; ++$x) {
    if ($x == 2) {
                    break;
          }
          print "$x<br>";

}
```

Imprimirá:

```
0
1
```

Referências:

Apostila do Maurício :
http://www-pet.inf.ufsm.br/OficinaPHP_2007/apostilas/php_tutorial2.pdf
Aplicativos em PHP (644 páginas) - http://pt.wikibooks.org/wiki/Aplicativos_em_PHP
PHP Não é Coisa de Moleque -
http://www.escolaqi.com.br/professor/downloads/download7454.pdf
http://codewalkers.com/tutorials/12/1.html

3 - PHP Avançado

3.1 - Trabalhando com Arrays em PHP

Um array é uma variável, mas diferente das demais ele armazena uma coleção de valores e não somente um. E ainda por cima podem conter outras variáveis e de tipos diferentes.

Detalhe importante: Quando em uma função precisarmos rtornar mais de um valor array é a saída, basta retornar todos os valors em forma de array.

Além disso é semelhante ao que estudamos na matemática: linhas e colunas. Matriz 3x4 (3 linhas e 4 colunas).

Um array no PHP é um mapa ordenado, que relaciona valores com chaves (em linhas e colunas).

Especificando um array()

array([chave =>] valor, ...);

A chave pode ser uma string ou um inteiro. O valor pode ser qualquer coisa.

Arrays (matrizes) simples e multidimensionais são suportados e podem ser criados pelo usuário ou por outras funções. Existem diversas funções específicas para bancos de dados, que preenchem arrays com os dados retornados em consultas, e vários outros tipos de funções também retornam arrays.

Arrays no Site do PHP

...

Exemplo de array multidimensional

```
$produto[1][codigo] = "1";
$produto[1][nome] = "João Pereira Brito";
$produto[1][email] = "joao@joao.org";
$produto[1][rua] = "Vasco da Gama";
$produto[1][numero] = "1345";

$produto[2][codigo] = "2";
$produto[2][nome] = "Antônio queiroz";
```

Exemplo de Array

```
$i=0;
while($i < $numregs){
        $codigo=pg_result($consulta,$i,codigo);
        $nome=pg_result($consulta,$i,nome);
        $venc=pg_result($consulta,$i,vencimento);
        $apartamento=pg_result($consulta,$i,apartamento);
        $pessoas=pg_result($consulta,$i,pessoas);
        $cota_agua=pg_result($consulta,$i,cota_agua);
        $cota_condominio=pg_result($consulta,$i,cota_condominio);
        $cota_reserva=pg_result($consulta,$i,cota_reserva);

        $total = $cota_agua + $cota_condominio + $cota_reserva;
        $total = number_format($total,2, ',','.');
...
```

```
$i++;
}
```

Também podemos ter um array formado por outros arrays (neste caso, cada sub array é uma linha do principal)

```
$arrayvarios = array(
        array(1, 3, 5, 7),
        array(2, 4, 6, 8),
        array(1, 1, 1, 1)
);
```

Neste caso temos um array 2x4 (2 linhas por 4 colunas, que iniciam sempre com índice zero).

Então se queremos retornar o valor 8, que está na linha 2 e coluna 4, devemos retornar o índice 1,3 (linha2=índice 1, coluna4=índice3).

print $arrayvarios[1][3];

Agora veremos com detalhes os pares: chave => valor:

```
$alunos = array(
        "0732355" => "Ribamar FS",
        "0823456" => "Antônio Brito",
        "0654345" => "Roberto Queiroz"
);
```

O que isto retornaria?

```
print $alunos["0732355"];
print $alunos[0];
```

Experimente!!

Atribuindo valores às chaves de arrays

Também podemos fazer diretamente assim:

print $alunos["0732355"] = "João Brito";

Lembrando que, a chave, é exclusiva. Podemos ter

```
$alunos["0732355"] = "João Brito";
$alunos["0932355"] = "João Brito";

Mas não podemos ter:
$alunos["0732355"] = "João Brito";
$alunos["0732355"] = "Ribamar FS";
```

Anexo agora um excelente tutorial sobre Arrays do Celso Goya publicado em:
http://www.xoopstotal.com.br/modules/wfsection/article.php?articleid=51

Trabalhando com arrays

Visão geral Para facilitar o entendimento, vamos definir array como um conjunto de valores, que podem ser identificados em grupo ou então separadamente. Estes conjuntos podem ser muito úteis enquanto programamos, pois em alguns casos podem substituir uma tabela em banco de dados ou então utilizando métodos mais avançados podemos carregá-los dinâmicamente e utilizar quase como um banco de dados em memória.

A linguagem PHP oferece uma incrível gama de recursos para se trabalhar com arrays. Com destaque para as funções auxiliares que permitem fazer desde uma simples contagem de elementos até a conversão automática de um array em string.

Neste artigo desenvolveremos como exemplo uma função para gerar combo boxes com os estados do Brasil. Muitas vezes criamos uma tabela no banco de dados para armazenar a lista de estados do Brasil sendo que neste caso existe um número finito de registros e menor que 100, então as operações de banco de dados não são tão ágeis quanto o uso de um array.

Criando o primeiro array

Para utilizar um array, antes de mais nada é preciso criar uma variável do tipo array.

```
<?php
$estados = array();
?>
```

O próximo passo é montar nossa lista de estados.

```
<?php
$estados = array();
$estados[0] = "Acre";
$estados[1] = "Alagoas";
$estados[2] = "Amapá";
$estados[3] = "Amazonas";
?>
```

Os colchetes servem para identificar qual elemento do nosso conjunto estamos nos referindo e o número entre colchetes é o código identificador do elemento.

Podemos fazer o seguinte teste:

```
<?php
$estados = array();
$estados[0] = "Acre";
$estados[1] = "Alagoas";
$estados[2] = "Amapá";
$estados[3] = "Amazonas";
echo($estados[0]);
?>
```

Neste caso será exibida a palavra Acre, pois indicamos o ítem [0] da variável $estados, que é um array.

Você deve estar se perguntando "O que há de tão fantástico em um array?". Agora vamos mostrar alguns recursos.

Criando o array de estados

Nosso array não será de grande valia se não permitir que as siglas dos estados sejam armazenadas também, pois desta forma podemos guardar no banco de dados apenas os dois caractéres correspondentes à sigla do estado, ou seja, utilizaremos apenas dois bytes no banco de dados.

Então vamos criar um array com duas colunas, sendo a primeira a sigla do estado e a segunda seu nome por extenso.

```
<?php
$estados = array();
$estado[0][0] = "AC";
$estado[0][1] = "Acre";
$estado[1][0] = "AL";
$estado[1][1] = "Alagoas";
$estado[2][0] = "AP";
$estado[2][1] = "Amapá";
$estado[3][0] = "AM";
$estado[3][1] = "Amazonas";
$estado[4][0] = "BA";
$estado[4][1] = "Bahia";
$estado[5][0] = "CE";
$estado[5][1] = "Ceará";
$estado[6][0] = "DF";
$estado[6][1] = "Distrito Federal";
$estado[7][0] = "ES";
$estado[7][1] = "Espírito Santo";
$estado[8][0] = "GO";
$estado[8][1] = "Goiás";
$estado[9][0] = "MA";
$estado[9][1] = "Maranhão";
$estado[10][0] = "MG";
$estado[10][1] = "Minas Gerais";
$estado[11][0] = "MT";
$estado[11][1] = "Mato Grosso";
$estado[12][0] = "MS";
$estado[12][1] = "Mato Grosso do Sul";
$estado[13][0] = "PA";
$estado[13][1] = "Pará";
$estado[14][0] = "PR";
$estado[14][1] = "Paraná";
$estado[15][0] = "PE";
$estado[15][1] = "Pernambuco";
$estado[16][0] = "PI";
$estado[16][1] = "Piauí";
$estado[17][0] = "RJ";
$estado[17][1] = "Rio de Janeiro";
$estado[18][0] = "RN";
$estado[18][1] = "Rio Grande do Norte";
$estado[19][0] = "RS";
$estado[19][1] = "Rio Grande do Sul";
$estado[20][0] = "RO";
$estado[20][1] = "Rondônia";
$estado[21][0] = "RR";
$estado[21][1] = "Roraima";
$estado[22][0] = "SC";
```

```
$estado[22][1] = "Santa Catarina";
$estado[23][0] = "SP";
$estado[23][1] = "São Paulo";
$estado[24][0] = "SE";
$estado[24][1] = "Sergipe";
$estado[25][0] = "TO";
$estado[25][1] = "Tocantins";
?>
```

A diferença neste exemplo é que utilizamos dois identificadores de elemento, ou seja, agora para cada elemento do array possuimos mais outros dois dependentes. Da mesma forma que criamos dois elementos o 0 e 1 para cada item de estado poderiamos criar n novos sub-elementos, por exemplo:

```
<?php
$estado = array();

$estado[0][0] = "SP";
$estado[0][1] = "São Paulo";
$estado[0][2] = "Sudeste";
?>
```

Vamos considerar à partir de agora que um array possui linhas e colunas, onde as linhas são equivalentes ao primeiro conjunto de colchetes e as colunas são equivalentes ao segundo conjunto de colchetes.

A função de exibição do combo box de estados

Agora vamos exibir todos os elementos de nosso array em uma função:

```
<?php
/*Nossa função reebe 3 parâmetros
$pNome :: Corresponde ao nome do SELECT
$pSelected :: Corresponde ao elemento que deverá possuir o status de selecionado
automaticamente
$extra :: Caso precise adicionar um style, ou então opção de multiple
*/
function renderCombo($pNome = "", $pSelected = "SP", $extra = ""){
        echo("<SELECT NAME='".$pNome."' ".$extra.">");

/*
Para exibir todos os itens do nosso combo utilizamos o comando for ,
lembre-se que como usamos números para identificar nosso array,
então podemos substituí-lo automaticamente com o for
*/
        //Realiza o loop em todos os elementos do array
        for( $i = 0; $i < 26;$i++ ){
                //Imprime a TAG OPTION usando a primeira coluna do array
                echo("<OPTION VALUE='".$estado[$i][0]."'");
                //Efetua a comparação para verificar se este é o ítem
selecionado
                if( $estado[$i][0] == $pSelected ){
                        //Caso a comparação seja verdadeira seleciona o ítem
                        echo(" SELECTED");
                }
                //Imprime o nome por extenso do estado, equilavente a segunda
coluna do array
                echo(">".$estado[$i][1]."</option>\n");
```

```
        }
        //Finaliza a tag SELECT
        echo("</SELECT>\n");
}
?>
```

Eureka! Esta feita uma função para exibir um combo de estados.

Identificadores alternativos

Na linguagem PHP podemos utilizar palavras para identificar um elemento de um array, este recurso é muito bom, pois facilita muito a depuração e o entendimento de programas que utilizam arrays.

Vamos utilizar nosso array de estados como exemplo:

```
<?php
$estado[0]["sigla"] = "SP";
$estado[0]["nome"] = "São Paulo";
$estado[0]["regiao"] = "Sudeste";

echo($estado[0]["sigla"]);
?>
```

Desta forma podemos deixar o código de nossos programas mais fáceis de se compreender. Repare que utilizamos uma string simples para identificar um elemento do array, sendo assim, podemos utilizar variáveis para identificar um item do array, por exemplo:

```
<?php
$estado[0]["sigla"] = "SP";
$estado[0]["nome"] = "São Paulo";
$estado[0]["regiao"] = "Sudeste";

$variavel = "sigla";

echo($estado[0][$variavel]);
?>
```

É importante lembrar que mesmo existindo uma string para identificar um elemento do array ainda podemos utilizar números se quisermos, por exemplo:

```
<?php
$estado[0]["sigla"] = "SP";
$estado[0]["nome"] = "São Paulo";
$estado[0]["regiao"] = "Sudeste";

echo($estado[0][0]);
?>
```

Nos três casos o resultado é o mesmo, diferindo apenas no método como chamamos o array.

O que você viu neste artigo é o básico sobre arrays, caso você se interessar pelo assunto e queira dar uma pesquisada rápida na web, vai encontrar outras formas de declarar arrays bem como usos diferenciados. O XOOPS utiliza muito este recurso. É só dar uma

olhada em algum arquivo xoops_version.php, que você vai encontrar um exemplo prático do uso de arrays.

Final do tutorial do Celso Goya.

Convertendo objetos para um array
http://www.revistaphp.com.br/print.php?id=147

3.2 - Trabalhando com Formulários em PHP

Manipulando dados de formulários com PHP – Parte 1

Uma das dúvidas mais freqüentes entre programadores PHP iniciantes é como manipular os dados de formulário enviados para os scripts PHP, principalmente dados de "checkbox" e upload de arquivos. Nessa primeira parte desse artigo, estarei mostrando como receber e manipular dados de campos comuns de formulários.

Na próxima semana estarei mostrando como manipular o upload de arquivos através dos formulários.

Para facilitar, esta primeira parte está divida nos seguintes tópicos:

1 - Introdução
2 - Campos Hidden
3 - Campos Text e Textarea
4 - Campos Radio
5 - Campos Checkbox
6 - Campos Select

1 - Introdução

Um formulário HTML é apenas um "rosto bonito" para onde os usuários poderão inserir informações que serão interpretados de alguma maneira por algum script do lado do servidor. E no nosso caso, esse script é um script PHP.

Primeiro: antes para poder enviar as informações, seu formulário deve conter um botão "submit", isso se consegue através do comando:

```
<input type=submit value="Texto do Botão">
```

Segundo: todos os campos que serão tratados no script PHP devem conter o parâmetro "NAME", caso contrário, os dados não serão passados para o script PHP. Ex: <input type=text name=nome_do_campo>

Como as informações do formulário são passadas para esse script PHP e como as informações do formulário enviado são tratadas, dependem de você.

Existem 2 métodos como as informações podem ser passadas: GET e POST. O recomendável sempre, para todos os formulários é usar o método POST, onde os dados enviados não são visíveis nas URLs, ocultando possíveis importantes informações e

permitindo o envio de longas informações. O GET é totalmente o contrário disso.

Como as informações chegam para o script PHP?

Assuma o seguinte formulário:

```
<form action="script.php" method="post">
Campo 1: <input type=text name=campo1><br>
Campo 2: <input type=text name=campo2><br>
<input type=submit value="OK">
</form>
```

Esse formulário usa o método POST para envio das informações, então em "script.php":

```
<?php
echo "O valor de CAMPO 1 é: " . $_POST["campo1"];
echo "<br>O valor de CAMPO 2 é: " . $_POST["campo2"];
?>
```

Se o formulário tivesse sido enviado usando o método GET, você simplesmente usaria $_GET no lugar de $_POST.

Observações:

Em vez de usar $_GET ou $_POST você pode escrever a variável com o mesmo nome do campo do formulário (no exemplo, $campo1 e $campo2). Mas, esse uso não é recomendado, pois se a diretiva "register_globals" na configuração do seu PHP estiver desativada, as variáveis com nome dos campos dos formulários, terão um valor vazio.

Uma solução para isso é usar a função **import_request_variables** no começo dos seus scripts que interpretam formulários. Essa função aceita 3 letras como argumentos: P, G e C, referentes a $_POST, $_GET e $_COOKIE respectivamente. Exemplo de uso:

```
<?php
import_request_variables("gP");
?>
```

O que acontece?

Exemplo: Você possui formulário com os campos "nome", "endereço" e "idade". Assuma que a diretiva "register_globals" do seu PHP esteja desligada, mas, você já havia programado o script usando as variáveis no escopo global, no lugar de $_POST.

Adicionando aquela função no começo do script, as variáveis do seu formulário postado: $_POST["nome"], $_POST["endereco"] e $_POST["idade"] serão extraídas cada para uma variável diferente: $nome, $endereco e $idade.

2 - Campos Hidden

Os campos hidden são usados para passar informações que não podem ser alteradas pelo usuário que estará inserindo informações no formulário. Por exemplo: você tem um site com sistema de login e o usuário quer alterar as informações de login dele. O script que irá manipular esse formulário, precisa saber o ID do usuário para poder alterar as informações no banco de dados, então esse ID é um campo hidden.

Códigos Exemplos:

hidden.html

```
<form action="hidden.php" method="post">
<input type=hidden name=escondido value="valor do escondido">
<input type=hidden name=id value="111">
<input type=submit>
</form>
```

hidden.php

```
<?php
echo "Campo Hidden: " . $_POST["escondido"];
echo "<br>Oi, seu ID é: " . $_POST["id"];
?>
```

3 - Campos Text e Textarea

Os campos text e textarea são os tipos mais simples, onde há somente um possível valor por campo. Dispensam maiores explicações.

Códigos Exemplos:

texts.html

```
<form action="texts.php" method="post">
Nome: <input type=text name=nome><br>
Email: <input type=text name=email><br><br>
Mensagem: <textarea name=mensagem cols=8 rows=3></textarea><br>
<input type=submit>
</form>
```

texts.php

```
<?php
echo "Olá " . $_POST["nome"] . " (email: " . $_POST["email"] . ")<br><br>";
echo "Sua mensagem: " . $_POST["mensagem"];
?>
```

4 - Campos Radio

Campos Radio permitem um relacionamento de um para muitos entre identificador e valor, ou seja, eles têm múltiplos possíveis valores, mas somente um pode ser pré-exibido ou selecionado. Por exemplo: você tem um sistema de "quiz". Cada pergunta possui 5 possíveis respostas. Cada resposta é um radio, onde os 5 radios dessa pergunta possuem o mesmo identificador, mas cada com valores diferentes.

Códigos Exemplos:

radio.html

```
<form action="radio.php" method="post">
<B>Qual seu sistema operacional?</B><br>
<input type=radio name=sistema value="Windows 98"> Win 98
<input type=radio name=sistema value="Windows XP"> Win XP
<input type=radio name=sistema value="Linux"> Linux
<input type=radio name=sistema value="Mac"> Mac
<br><br>
<B>Qual a marca de seu monitor?</B><br>
<input type=radio name=monitor value="Samsung"> Samsung
<input type=radio name=monitor value="LG"> LG
<input type=radio name=monitor value="Desconhecido"> Desconhecido
<br><br>
<input type=submit>
</form>
```

radio.php

```
<?php
echo "Seu sistema operacional é: " . $_POST["sistema"];
echo "<br>Seu monitor é: " . $_POST["monitor"];
?>
```

5 - Campos Checkbox

O tipo Checkbox tem somente um possível valor por entrada: on value (marcado) ou no value (desmarcado). No script você deve fazer a verificação para saber se o campo foi marcado ou não.

Se é possível também utilizar grupos de checkbox com o mesmo nome. Para você deve adicionar "[]" no final do nome, para o PHP interpretar como array, veja o código exemplo.

Códigos Exemplos:

checkbox.html

```
<form action="checkbox.php" method="post">
<B>Escolha os numeros de sua preferência:</B><br>
<input type=checkbox name="numeros[]" value=10> 10<br>
<input type=checkbox name="numeros[]" value=100> 100<br>
<input type=checkbox name="numeros[]" value=1000> 1000<br>
<input type=checkbox name="numeros[]" value=10000> 10000<br>
```

```
<input type=checkbox name="numeros[]" value=90> 90<br>
<input type=checkbox name="numeros[]" value=50> 50<br>
<input type=checkbox name="numeros[]" value=30> 30<br>
<input type=checkbox name="numeros[]" value=15> 15<br><BR>
<input type=checkbox name="news" value=1> <B>Receber
Newsletter?</B><br><BR>
<input type=submit>
</form>
```

checkbox.php

```
<?php
// Verifica se usuário escolheu algum número
if(isset($_POST["numeros"]))
{
     echo "Os números de sua preferência são:<BR>";

     // Faz loop pelo array dos numeros
     foreach($_POST["numeros"] as $numero)
     {
          echo "- " . $numero . "<BR>";
     }
}
else
{
     echo "Você não escolheu número preferido!<br>";
}

// Verifica se usuário quer receber newsletter
if(isset($_POST["news"]))
{
     echo "Você deseja receber as novidades por email!";
}
else
{
     echo "Você não quer receber novidades por email...";
}
?>
```

6 - Campos Select

Os campos select permitem tratar uma variedade de opções, onde o usuário pode selecionar apenas uma opção ou múltiplas opções. Quando você permite múltiplas seleções, deve adicionar "[]" no final do nome, para o PHP interpretar como array.

Nos exemplos, mostro o funcionamento e tratamento de ambas.

Códigos Exemplos:

select.html

```
<form action="select.php" method="post">
<B>Qual seu processador?</B><br>
<select name=processador>
<option value=Pentium>Pentium</option>
```

```
<option value=AMD>AMD</option>
<option value=Celeron>Celeron</option>
</select><BR><BR>
<B>Livros que deseja comprar?</B><br>
Obs: segure "CTRL" para selecionar mais de um.<BR>
<select name="livros[]" multiple>
<option value="Biblia do PHP 4">Biblia do PHP 4</option>
<option value="PHP Professional">PHP Professional</option>
<option value="Iniciando em PHP">Iniciando em PHP</option>
<option value="Novidades do PHP 5">Novidades do PHP 5</option>
<option value="Biblia do MySQL">Biblia do MySQL</option>
</select><BR><BR>
<input type=submit>
</form>
```

select.php

```
<?php
echo "Seu processador é: " . $_POST["processador"] . "<BR>";

// Verifica se usuário escolheu algum livro
if(isset($_POST["livros"]))
{
	echo "O(s) livro(s) que você deseja comprar:<br>";
	// Faz loop para os livros
	foreach($_POST["livros"] as $livro)
	{
		echo "- " . $livro . "<br>";
	}
}
else
{
	echo "Você não escolheu nenhum livro!";
}
?>
```

Clique aqui para baixar os códigos desse artigo

Quaisquer dúvidas que tiver, não hesite em contatar-me!

Até a próxima semana, onde estarei mostrando como manipular o upload de arquivos de formulários e algumas boas técnicas para com formulários => PHP.

Alfred Reinold Baudisch
Desenvolvedor de Sistemas Web
alfred@auriumsoft.com.br
www.estacaonet.com

Excluir Marcados

```
<?php
/*
Banco - excluir_varios
Tabela
CREATE TABLE `produtos` (
  `id` int(11) NOT NULL,
  `produto` char(45) default NULL,
  `categoria` char(45) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
*/

// Form com arquivo chamando a si mesmo
$conexao = mysql_connect('localhost','root','ribafs');
mysql_select_db('excluir_varios',$conexao);
$consultar = "SELECT * FROM produtos ORDER BY id";
$resultado = mysql_query($consultar, $conexao);

if(mysql_num_rows($resultado) != 0){
	echo "<form name='frmExcluir' method='post' action='teste.php'>";
	echo "<table
border=1><tr><th> </th><th>Produto:</th><th>Categoria:</th></tr>";

	while($linha = mysql_fetch_row($resultado)){
		echo "<td><input type='checkbox' name='id[]' value='$linha[0]'></td>
			<td>$linha[1]</td>
			<td>$linha[2]</td></tr>";
	}
	echo "<tr><td colspan='3'><input type='submit' name='excluir'
value='Excluir!'></td></tr>";
	echo "</table></form>";
}else{
	echo "Nenhum registro foi encontrado!";
}
if(isset($_POST['id'])){
	$opcoes = $_POST['id'];
	$opcoes_text = implode(", ", $opcoes);
	$strexcluir = "DELETE FROM produtos WHERE id in (" . $opcoes_text . ")";
	mysql_query($strexcluir, $conexao) or die("Ocorreu algum erro");
}else{
	echo "É necessário escolher quem será excluído<br>";
	echo "<a href='javascript: history.back();'>Voltar</a>";
}
?>
```

Básico de GET e POST

Usar GET em situações seguras e POST em situações que precisem de segurança.

GET suporta apenas 1024 caracteres na string.
POST é sem limite.

Exemplo de uso de formulário com PHP chamando outro script

```
<body onLoad="document.form1.nome.focus()">
<form name="form1" method="POST" action="inserir.php">
Nome.<input type="text" name="nome"><br>
E-mail<input type="text" name="email"><br>
<input type="hidden" name="oculto" value="OK">
Senha<input type="password" name="senha"><br>
SenhaCript<input type="password" name="senhacript"><br>
<input type="submit" value="Enviar"><br>
</form>

<?php
//$nome=$_POST["nome"];
//$email=$_POST["email"];
if (isset($nome)){
        echo "O nome digitado foi " . $nome . "<br>";
        echo "O e-mail digitado foi " . $email . "<br>";
        echo "O campo oculto contém " . $oculto . "<br>";
        echo "A senha digitada foi " . $senha . "<br>";
        echo "A senha md5 digitada foi " . md5($senhacript) . "<br>";
}
?>
```

Exemplo de uso de formulário com PHP chamando o mesmo script

```
<body onLoad="document.form1.nome.focus()">
<form name="form1" method="POST" action="<?php $_PHPSELF ?>">
Nome.<input type="text" name="nome"><br>
E-mail<input type="text" name="email"><br>
<input type="submit" name="enviar" value="Enviar"><br>
</form>

<?php
if(isset($_POST['enviar'])){
$nome = $_POST["nome"];
$email = $_POST["email"];

        if(isset($nome)){
                echo "O nome digitado foi " . $nome . "<br>";
                echo "O e-mail digitado foi " . $email . "<br>";
        }
```

```
}
?>
```

import_request_variables("gP");
import_request_variables -- Import GET/POST/Cookie variables into the global scope

Muito indicado para quem tem o register_globals desativado e não quer digitar $_POST, $_GET ou $_COOKIE.

Sempre deve vir na primeira linha do script ou antes do uso da variável.

```
import_request_variables("gP", "rvar_");

echo $rvar_foo;
```

Alerta: evite usar o recurso acima.

As informações digitadas em um formulário podem ser capturadas por um script PHP.
Veja como:

form.html

```
<form action="recebe.php" method="POST">
Nome <input type="text" name="nome"><br>
Idade <input type="text" name="idade"><br>
<input type="submit" value="Enviar">
</form>
```

recebe.php

```
<?php
echo $_POST["nome"];
echo "<br>";
echo $_POST["idade"];
?>
```

Recebendo dados via URL

Se temos a seguinte situação:
Temos um arquivo c:\www\teste.php, com o seguinte código:

```
<?php
   echo $_GET['codigo'];
?>
```

Queremos passar uma variável código com valor 5, fazemos:
http://127.0.0.1/teste.php?codigo=5

Para receber mais de uma variável ou campo usar:
http://127.0.0.1/teste.php?codigo=5&nome="Antônio"&email="ribafs@yahoo.com"

echo "Código = ".$_GET['codigo']." Nome = ".$_GET['nome']." E-mail = ".$_GET['email'];

Também podemos passar variáveis pela URL, assim teste.php?codigo=$cod

3.3 - Trabalhando com Arquivos em PHP

Quando abrimos um arquivo para leitura (r+) o ponteiro situa-se no início do arquivo.
Quando abrimos um arquivo para anexar (a+) o ponteiro situa-se no final do arquivo.

Sempre usar fclose() quando finalizar de trabalhar com o arquivo para economizar memória.

O PHP suporta duas funções básicas para escrever em arquivos:
file_put_contents() - que escreve ou anexa uma string para um arquivo.
fwrite() - função que incrementalmente escreve dados para um arquivo texto.

Exemplos:

file_put_contents($string, $arquivo);

A função file($arquivo) lê arquivos texto.

Ler todo um arquivo em uma string:
```
$file = file_get_contents("/windows/system32/drivers/etc/services");
print("Size of the file: ".strlen($file)."\n");
```

Abrir um arquivo para leitura
```
$file = fopen("/windows/system32/drivers/etc/hosts", "r");
print("Tipo de manipulador de arquivo: " . gettype($file) . "\n");
print("Primeira linha do manipulador do arquivo: " . fgets($file));
fclose($file);
```

Ler um único caractere de um arquivo:
```
<?php
$file = fopen("/windows/system32/drivers/etc/services", "r");
$count = 0;
while ( ($char=fgetc($file)) !== false ) {
  if ($char=="/") $count++;
}
fclose($file);
print("Number of /: $count\n");
?>
```

Conteúdo do meuarquivo.txt

1|Meu nome|Meu site|Olá pessoal
2|Meu nome1|Meu site|Olá pessoal
3|Meu nome2|Meu site|Olá pessoal
4|Meu nome3|Meu site|Olá pessoal
5|Meu nome4|Meu site|Olá pessoal

Primeiro, vamos colocar todo o arquivo num array. Use o seguinte código:
$meuArray = file("nomedoarquivo.txt");

Vamos excluir a linha que tenha o primeiro valor do array igual a 3.

Para isto, temos que ler cada linha do array e fazer um "explode" delas, separando os "|" como se fosse em colunas.

Depois, faremos um [cfIF[/cf] perguntando se a coluna desejada é igual ao valor que queremos: 3

```
for($n=0; $n < count($meuArray); $n++) {
         $cadaLinha = explode("|", $meuArray[$n]);
        if($cadaLinha[0] == 3) {
                echo "Meu nome: $cadalinha[1] - Site: $cadalinha[2]";
        }
}
```

Excluindo linhas do array e salvando nos arquivos:
$meuArray = file("nomedoarquivo.txt"); // coloco todo o arquivo num array
$arrayModificado = array(); // crio um array vazio.

```
for($n=0; $n < count($meuArray); $n++) {
        $cadaLinha = explode("|", $meuArray[$n]);

        if($cadaLinha[0] <> 3) {
                $arrayModificado[] = $meuArray[$n];
        }
}
```

Depois é só salvar o array no arquivo, incluindo cada linha do array já modificado:

```
$bufferArquivo = fopen('nomedoarquivo.txt',"w");

for($n=0; $n < count($arrayModificado); $n++) {
    fwrite($bufferArquivo, $arrayModificado[$n]);
}

fclose($bufferArquivo);
```

Autor: Emannuel em http://www.codigofonte.net/dicas/php/130_trabalhando-com-arquivos
Veja também:
http://onlamp.com/pub/a/php/2002/10/03/php_foundations.html
http://onjava.com/pub/a/php/2002/12/12/php_foundations.html
http://onlamp.com/pub/a/php/2003/01/09/php_foundations.html

3.4 - Trabalhando com Diretórios no PHP

```
$Dir = “C:\\PHP”;
$DirOpen = opendir($Dir);
while ($CurFile = readdir($DirOpen)) {
                echo $CurFile . “<br />”;
}
closedir($DirOpen);

$Dir = “C:\\PHP”;
$DirEntries = scandir($Dir);
foreach ($DirEntries as $Entry) {
                echo $Entry . “<br />”;
}

mkdir($diretorio);

is_writable($dir);

file_exists($dir);

is_dir($dir);

copy($arq_origem, $arq_destino);

unlink($arquivo);

rmdir($dir);



<?php
// Listar conteúdo de diretório
function listar_dir($arquivo){
        $dn = opendir ($arquivo);
        while ($file = readdir ($dn)) {
      return "$file<br>";
        }
        closedir($dn);
}

// Espaço total em disco em KB
function total_space($dir){
        return round(disk_total_space($dir)/(1024*1024),2) . ' KB';
}
//print total_space('/home/ribafs');

// Espaço livre em disco em KB
function free_space($dir){
        return round(disk_free_space($dir)/(1024*1024),2) . ' KB';
```

```
}
//print free_space('/home/ribafs');

// Receber tamanho de diretório recursivamente
function getdirsize($dir) {
	// Initialize the size value
	$size = 0;
	// Get the files and directories
	// inside the given directory
	$files = scandir($dir);
	// Loop through the files
	foreach ($files as $file) {
		// Filter-out .. and .
		if (strlen(str_replace('.', '', $file))>0) {
			if (is_dir($dir.'/'.$file)) {
				// If the item is a directory
				// run this function again and
				// get the size of the files inside
				$size += getdirsize($dir.'/'.$file);
			} else {
				// If it's a file, calculate the size
				// and add it to the size variable
				$size += filesize($dir.'/'.$file);
			}
		}
	}
	// Finally, return the calculated size
	return $size;
}
//print getdirsize('/home/ribafs/uteis');

function dir_size($directory, $type, $round) {
  $dir = scandir($directory);
  $size = 0;
  foreach($dir as $value) {
    if(is_dir($directory.'/'.$value)) {
      $init = scandir($directory.'/'.$value);
        foreach($init as $subfile) {
          $filesize = filesize($directory.'/'.$value.'/'.$subfile);
          $size = $size + $filesize;
        }
      if(is_file($directory.'/'.$value)) {
        $filesize = filesize($directory.'/'.$value);
        $size = $size = $filesize;
      }
    }
  }
  if($type==1) {
    // returns KB
    $size = $size / 1024;
```

```
    if(isset($round))
      $size = round($size, $round);
    echo $size.'KB';
  }
  elseif($type==2) {
    $size = $size / 1048576;
    if(isset($round))
      $size = round($size, $round);
    echo $size.'MB';
  }
  elseif($type==3) {
    $size = $size / 1073741824;
    if(isset($round))
      $size = round($size, $round);
    echo $size.'GB';
  }
  else {
    if(isset($round))
      $size = round($size, $round);
    echo $size.'Bytes';
  }
}
/*
 * #1 with how you would like the data to be shown (1 = KB, 2 = MB, 3 = GB, 0 or nothing is Bytes), and #2 is optional, if you want it to make the number perttier enter how many numbers AFTER the . to have, so if you put in 2, and the number before it rounded it was 3.4325214 it would be something like, 3.44 or so after it rounds it.
 */
dir_size("/home/ribafs", 1, 2);


function dir_size2( $dir )
{
    if( !$dir or !is_dir( $dir ) )
    {
        return 0;
    }

    $ret = 0;
    $sub = opendir( $dir );
    while( $file = readdir( $sub ) )
    {
        if( is_dir( $dir . '/' . $file ) && $file !== ".." && $file !== "." )
        {
            $ret += dir_size( $dir . '/' . $file );
            unset( $file );
        }
        elseif( !is_dir( $dir . '/' . $file ) )
        {
            $stats = stat( $dir . '/' . $file );
```

```
            $ret += $stats['size'];
            unset( $file );
        }
    }
    closedir( $sub );
    unset( $sub );
    return $ret;
}

//print dir_size2('/home/ribafs/htdocs');

/**
 * Removes the directory and all its contents.
 *
 * @param string the directory name to remove
 * @param boolean whether to just empty the given directory, without deleting the given
directory.
 * @return boolean True/False whether the directory was deleted.
 */
function delete_dir($dirname,$only_empty=false) {
   if (!is_dir($dirname))
      return false;
   $dscan = array(realpath($dirname));
   $darr = array();
   while (!empty($dscan)) {
      $dcur = array_pop($dscan);
      $darr[] = $dcur;
      if ($d=opendir($dcur)) {
         while ($f=readdir($d)) {
            if ($f=='.' || $f=='..')
               continue;
            $f=$dcur.'/'.$f;
            if (is_dir($f))
               $dscan[] = $f;
            else
               unlink($f);
         }
         closedir($d);
      }
   }
   $i_until = ($only_empty)? 1 : 0;
   for ($i=count($darr)-1; $i>=$i_until; $i--) {
      echo "\nDeleting '".$darr[$i]."' ... ";
      if (rmdir($darr[$i]))
         echo "ok";
      else
         echo "FAIL";
   }
   return (($only_empty)? (count(scandir)<=2) : (!is_dir($dirname)));
}
```

```
/* Function to remove directories, even if they contain files or
subdirectories.  Returns array of removed/deleted items, or false if nothing
was removed/deleted.

by Justin Frim.  2007-01-18

Feel free to use this in your own code.
*/

function rmdirtree($dirname) {
  if (is_dir($dirname)) {    //Operate on dirs only
     $result=array();
     if (substr($dirname,-1)!='/') {$dirname.='/';}    //Append slash if necessary
     $handle = opendir($dirname);
     while (false !== ($file = readdir($handle))) {
        if ($file!='.' && $file!= '..') {    //Ignore . and ..
           $path = $dirname.$file;
           if (is_dir($path)) {    //Recurse if subdir, Delete if file
              $result=array_merge($result,rmdirtree($path));
           }else{
              unlink($path);
              $result[].=$path;
           }
        }
     }
     closedir($handle);
     rmdir($dirname);    //Remove dir
     $result[].=$dirname;
     return $result;    //Return array of deleted items
  }else{
     return false;    //Return false if attempting to operate on a file
  }
}

?>
```

3.5 - Trabalhando com Path no PHP

Exemplos simples de uso de funções para path do PHP

```
<?php
   $path=dirname(realpath($_SERVER['SCRIPT_FILENAME']));
   $path=substr($path,0,strlen($path) - 5);
echo "<br>Path deste script sem 5 finais caracteres - " . $path;
```

```
echo "<br><br>Diretório atual - ".dirname(__FILE__);
echo "<br>Caminho completo do script atual - ".__FILE__;

echo "<br>URL do script atual - " . "http://" . $_SERVER['HTTP_HOST'] .
$HTTP_SERVER_VARS["SCRIPT_NAME"];
?>

<?php
$path = "/etc/passwd";
$file = dirname($path); // $file is set to "/etc"
?>

<?php
$path = "/home/httpd/html/index.php";
$file = basename($path);        // $file is set to "index.php"
$file = basename($path, ".php"); // $file is set to "index"
?>

<?php
echo dirname($_SERVER["REQUEST_URI"]);
?>
```

Recebendo o Path Absoluto do Script Atual

dirname(__FILE__)

Recebendo o path relativo do webserver do script atual

```
function GetRelativePath($path){
   $npath = str_replace('\\', '/', $path);
   return str_replace(GetVar('DOCUMENT_ROOT'), '', $npath);
}
```

GetRelativePath(dirname(__FILE__));

<?php

if (DIRECTORY_SEPARATOR=='/')

```
 $absolute_path = dirname(__FILE__).'/';
```

else

```
 $absolute_path = str_replace('\\\\', '/', dirname(__FILE__)).'/';
```

?>

Resultará em um path absoluto no estilo UNIX que funciona também em PHP5 sob Windows.

Em algumas instalações (< 4.4.1) $_SERVER['REQUEST_URI'] não está configurado, usado o código para corrigir:

```
<?php

if (!isset($_SERVER['REQUEST_URI'])) {
   $_SERVER['REQUEST_URI'] = substr($_SERVER['PHP_SELF'],1);
   if (isset($_SERVER['QUERY_STRING'])) $_SERVER['REQUEST_URI'].='?'.
$_SERVER['QUERY_STRING'];
}
```

```
?>

$my_uri = "http://" . $_SERVER['HTTP_HOST'] . $HTTP_SERVER_VARS["SCRIPT_NAME"];

// então

<?php echo ("$my_uri");?>

<?php
include("{$_SERVER['DOCUMENT_ROOT']}/includes/my_include.php");
?>
```

Você pode usar isso para receber o diretório pai:

dirname(dirname(__FILE__))

...include a file relative to file path:

include(dirname(__FILE__) . '/path/relative/file_to_include.php');

Isso colocará ambos os paths "www" e "file" de forma fácil para transportar o array.

```
<?php
// build the www path:
$me = $_SERVER['PHP_SELF'];
$Apathweb = explode("/", $me);
$myFileName = array_pop($Apathweb);
$pathweb = implode("/", $Apathweb);
$myURL = "http://".$_SERVER['HTTP_HOST'].$pathweb."/".$myFileName;
$PAGE_BASE['www'] = $myURL;

// build the file path:
strstr( PHP_OS, "WIN") ? $strPathSeparator = "\\" : $strPathSeparator = "/";
$pathfile = getcwd ();
$PAGE_BASE['physical'] = $pathfile.$strPathSeparator.$myFileName;

// this is so you can verify the results:
$www = $PAGE_BASE['www'];
$physical = $PAGE_BASE['physical'];

echo "$physical<p>";
echo "$www<p>";
?>
```

retornará algo como:

Windows:

F:\dev\Inetpub\wwwroot\somedirectory\index.php

http://devserver/somedirectory/index.php

Unix:

/home/somepathto/gieson.com/webroot/index.php

http://www.gieson.com/index.php

Path absoluto do script em execução

$path=dirname(realpath($_SERVER['SCRIPT_FILENAME']));

```
print 'Diretório atual=> '.dirname($_SERVER["SCRIPT_FILENAME"]);
print '<br>Diretório atual=> '.dirname(__FILE__);
print '<br>Diretório anterior=> '.dirname(dirname(__FILE__));
print '<br>Diretório anterior 2 níveis=> '.dirname(dirname(dirname(__FILE__)));
print '<br>Diretório raiz geral => '.dirname($_SERVER['PHP_SELF']);
print '<br>Diretório atual isolado => '.basename(dirname(__FILE__));
print '<br>Arquivo atual isolado => '.$_SERVER['REQUEST_URI'];
print '<br>Arquivo atual path completo => '.__FILE__;
print '<br>Arquivo raiz geral => '.dirname($_SERVER["SCRIPT_NAME"]);


 $site_path = realpath(dirname(__FILE__));
 define ('__SITE_PATH', $site_path);


function fixPath ($path)
(
      return dirname($path . '/.');
)

fixPath('/one');                // '/one'
fixPath('/one/');               // '/one'
fixPath('/one//two');           // '/one/two'
fixPath('/one////two////');     // '/one/two'
```

Checar permissão de escrita:

```
error_reporting(E_ALL);
$dir_name = '/var/www/virtual/phpintra/htdocs/php/';
do {
    $b_is_writable = is_writable($dir_name);
    echo sprintf("Dir[%s]Writable[%s]\n", $dir_name, $b_is_writable? 'YES':'NO');
}while (($dir_name = dirname($dir_name)) !='/');

//Restringindo acesso direto
if (!defined("BASE_PATH")) define('BASE_PATH',
dirname($_SERVER['SCRIPT_NAME'])=='/' ? './' : str_repeat("../",
substr_count(dirname($_SERVER["SCRIPT_NAME"]), "/")));


function myRealPath($path) {
```

```
    // Check if path begins with "/". => use === with strpos
    if (strpos($path,"/") === 0) {
        $path = $_SERVER['DOCUMENT_ROOT'].$path;
    }
    else {
        // Strip slashes and convert to arrays.
        $currentDir = preg_split("/\//",dirname($_SERVER['PATH_TRANSLATED']));
        $newDir = preg_split('/\//',$path);
        // Drop one directory from the array for each ".."; add one otherwise.
        foreach ($newDir as $dir) {
            if ($dir == "..")
              array_pop($currentDir);
            elseif ($dir != ".") //test for "." which represents current dir (do nothing in that case)
              array_push($currentDir,$dir);
        }
        // Return the slashes.
        $path = implode($currentDir,"/");
    }
    return $path;
}
//print myRealPath("./");

$p = getcwd(); // recebe o diretório atual
//echo $p;

$dir_path = str_replace( $_SERVER['DOCUMENT_ROOT'], "",
dirname(realpath(__FILE__)) ) . DIRECTORY_SEPARATOR;
//print $dir_path;

/*
echo realpath(dirname(__FILE__)); //Similar ao getcwd()

$inc = $_SERVER["DOCUMENT_ROOT"]."/inc";
include $inc."/config.php";
include $about."/setup.php";
include $inc."/header.php";
include $inc."/topnav.php";
include $inc."/left.php";
include $inc."/right.php";


echo __FILE__AbsolutePath(__FILE__)."\n";

function __FILE__AbsolutePath($Filename)
 {
       switch (PHP_OS)
        {
              case "WINNT": $needle = "\\"; break;
              case "Linux": $needle = "/";  break;
              default:      $needle = "/";  break; // TODO : Mac check
```

```
        }
        $AbsPath = substr($Filename, 0, strrpos($Filename, $needle));
        return $AbsPath;
 }
*/
function __FILE__AbsolutePath()
 {
        $d = debug_backtrace();
        $Filename = $d[0]['file'];

        switch (PHP_OS)
        {
                case "WINNT": $needle = "\\"; break;
                case "Linux": $needle = "/";  break;
                default:      $needle = "/";  break; // TODO : Mac check
        }
        $AbsPath = substr($Filename, 0, strrpos($Filename, $needle));
        return $AbsPath;
 }
//print __FILE__AbsolutePath() ;
```

3.6 - Trabalhando com Includes em PHP

INCLUDE - inclue e avalia o conteúdo do arquivo.
REQUIRE - também inclue e avalia o conteúdo do arquivo incluído.

A diferença entre ambos é que o include ao encontrar um erro, lança um warning apenas, enquanto que o require lança um Fatal Error, que pára o processamento.

Uso do require: para códigos que requerem maior segurança.

INCLUDE_ONCE e REQUIRE_ONCE - são semelhantes ao include e require, sendo que estes incluem um arquivo somente uma vez.

Mostrando uso do require_once:

echo.php

```
<?php
echo "Hello";
?>
```

teste.php

```
<?php
require('echo.php');
require_once('echo.php');
?>
```

Executar teste.php
saída: "Hello".

Agora teste2.php:
```
<?php
require('echo.php');
require('echo.php');
?>
```

Executar teste2.php
saída: "HelloHello".

Agora teste3.php:
```
<?php
require_once('echo.php');
require('echo.php');
?>
```

Executar teste3.php
saída: "HelloHello".

Ou seja, ao encontrar require_once, ele verifica se o arquivo já foi incluído, e somente o incluirá novamente se ele ainda não tiver sido incluído.
```
<?php
// Isto está errado e não funcionará como desejado
if ($condition)
   include $arquivo;
else
   include $outro;


// E este está correto
if ($condition) {
   include $arquivo;
} else {
   include $outro;
}

?>

<?php
$path="/full/path/to/script/";
if (getdomain($path) == 'yourdomain'){
    include($path.'somefile.php');
}
?>

"variables.php"
<?php
$includer = basename($_SERVER['SCRIPT_NAME']);

switch ($includer) {
```

```
   case 'a.php':
   $this_variable = 'included by script a.php';
   break;

   case 'b.php':
   $this_variable = 'included by script b.php';
   break;

   default:
   $this_variable = 'included by unkown script';
}
echo $this_variable;
?>
```

Test with 3 different files "a.php", "b.php", "c.php", all with the same content:

```
<?php
include 'variables.php';
?>
```

3.7 - Trabalhando com Funções no PHP

O PHP conta com uma grande quantidade de boas bibliotecas de funções internar sobre vários assuntos, como strings, arrays, arquivos, diretórios, data e hora, etc.

Além das funções embutidas também podemos criar nossas próprias funções quando não encontrarmos uma função que atenda as nossas necessidades.

Funções são blocos de código, conjuntos de instruções, que ficam quietos, sem vida, até que sejam chamadas em algum lugar do código. Funções reduzem o trabalho de digitação, o tamanho dos scripts em geral, os erros, reaproveitando código, facilitando e organizando o trabalho de desenvolvimento.

Exemplos de funções definidas pelo usuário

```
function quadrado($numero) {
        print $numero*$numero;
}
```

Executando:

quadrado(6); // Saída: 36

Variáveis globais

```
$var1 = 5;
function testeGlobal1(){
        $var1 = 1;
        print "<br>Valor de \$var1: $var1";
}
echo testeGlobal1();
```

Variável externa acessível na função

```
$var2 = 10;
function testeGlobal2(){
        global $var2;
        print "<br>Valor de \$var2 $var2";
}
echo testeGlobal2();

$var5 = 15;
function testeGlobal5(){
        $var5 = 5;
    print "<br><br>A variável global vale $GLOBALS[var5], ";
    print "Já a variável local vale $var5<br><br>";
}

testeGlobal5();

function cliente($codigo, $idade = 18){
        print "Código = $codigo, Idade = $idade";
}

cliente(1);  //Exibirá: Código = 1, Idade = 18

function cubo($num){
        return ($num*$num*$num);
}

$var1 = 2 * cubo(5);echo "<br>".$var1;
```

Acessando variáveis Externas

Normalmente de dentro de uma função não temos acesso às variáveis que estão fora dela. Mas veja como contornar isso:

Variável externa inacessível

```
$var1 = 5;
function testeGlobal1(){
        $var1 = 1;
        print ”Valor de \$var1: $var1”;
}
echo testeGlobal1();
Variável externa acessível na função

$var2 = 10;
function testeGlobal2(){
        global $var2;
        print ”Valor de \$var2 $var2”;
}
echo testeGlobal2();
```

Outra alternativa é usar o array $GLOBALS[], que contém todas as variáveis globais. Veja um exemplo:

```
$var5 = 15;
function testeGlobal5(){
        $var5 = 5;
        print "<br><br>A variável global vale $GLOBALS[var5], ";
        print "Já a variável local vale $var5<br><br>";
}
```

```
testeGlobal5();
```

Variáveis estáticas

```
function contador(){
        static $x = 0;
        return $x++;
}
echo "<br>";
echo contador();echo contador();echo contador();
//A saída será: 012

function contador2(){
        $x = 0;
        return $x++;
}
echo "<br>";
echo contador2();echo contador2();echo contador2();
//A saída será: 000.

  function staticfunction() {
   static $count = 0;
   $count++;
        if ($count==1){
       echo "A Função foi executada $count vez<br>";
        }else{
           echo "A Função foi executada $count vezes<br>";
        }
  }

  for($i = 0; $i < 5; $i++) {
   staticfunction();
  }

function Teste1()
{
    static $a = 0;
    echo $a;
    $a++;
}

for($x=0;$x<=10;$x++){
        echo Teste1()." ";
}
```

Recursiva

```
function Teste()
{
    static $count = 0;

    $count++;
    echo $count." ";
    if ($count < 10) {
        Teste ();
    }
    $count--;
}

Teste();
```

Declarando variáveis static

```
function foo(){
    static $int = 0;          // correro
   // static $int = 1+2;        // errado (é uma expressão)
   // static $int = sqrt(121);  // errado  (é uma expressão também)

    $int++;
    echo $int;
}


function aumentoSalario($sal, $perc=5){
        $salario = $sal * $perc/100;
        echo $salario;
}
echo "<br>Aumento: " . aumentoSalario(1956);

function redirecionar($url){
        header("Location: $url");
}
echo "<br>";
redirecionar("http://ribafs.phpnet.us/");
echo "<br>";
```

Retornar mais de um valor de uma função

usa-se arrays e list()

array() retorna e list() exibe

//Exemplo:

```
function recebe_hoje(){
        $data_abreviada=date("d/m/Y");
        $data_extensa=date("l, d F \d\e Y");

        return array($data_abreviada, $data_extensa);
}

list($data_abreviada, $data_extensa)=recebe_hoje();
        print $data_extensa;
echo "<br>";
        print $data_abreviada;


<h2>Declaração dinâmica de função</h2>

<pre>
if ($f == 1){
        function f1(){
                echo "funcao1";
        }
}else{
        function f2(){
                echo "funcao2";
        }
}

<h2>Retornando o número de argumentos de uma função</h2>

<pre>
```

```
function ret_args_funcao() {
    $numargs = func_num_args();
    echo "Número de argumentos: $numargs<br>\n";
    if ($numargs >= 2) {
    echo "Segundo argumento vale : " . func_get_arg (1) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
    echo "Argumento $i vale: " . $arg_list[$i] . "<br>\n";
    }
}

ret_args_funcao (1, 2, 3);
```

Passando Argumentos através de Funções
O default é 'por valor', que passa uma cópia do valor da variável.

Também podemos passar 'por referência', onde se passa o endereço da própria variável. Quando atribuímos uma variável a outra passando como referência, não fazemos cópia alguma, mas sim passamos o endereço da variável original, portanto qualquer alteração nesta referência refletirá na variável original.

ByRef é mais eficiente em termos de memória ao lidar com variáveis e arrays grandes e também permite alterar o valor da variável original, o que não acontece com o ByVal, mas a vantagem de desempnho somente é percebida em grandes arrays em grandes loops.

Para maior segurança setar:

allow_call_time_pass_reference no php.ini

Impede a passagem de valores por referência nas chamadas, mas permite somente na definição das funções.

$var1 = & $var2;

Agora ambas apontam para o mesmo endereço e valor.

Reter valor de variáveis entre chamadas (static)

(Guarda o valor da última chamada até o final da execução do script, tantas vezes quantas a função for chamada).

Exemplo:

```
$valor = 4;
$ref = &$valor;

$ref = 3;

$valor = 4;
$ref = &$valor; // Aqui tornamos ambas as variáveis com o mesmo endereço
                // O que alterarmos em $ref alteramos em $valor

$ref = 3;       // Com isso também alteramos $valor para 3, veja abaixo.
echo $valor . "<br>";

$valor=0;          // Com isso também alteramos $ref para 0, veja abaixo.

echo $ref;
```

Por Valor

```
function val_subtracao($num1, $num2){
        if($num1 < $num2){
                die("Números negativos");
        }else{
                $return_result=0;
                while($num1 > $num2){
                        $num1 = $num1 - 1;
                        $return_result = $return_result + 1;
                }
        }
        return($return_result);
}

$primeiro_op=493;
$segundo_op=355;
$resultado1 = val_subtracao($primeiro_op, $segundo_op);
print ("Resultado1 é $resultado1<br>");
$resultado2 = val_subtracao($primeiro_op, $segundo_op);
print("Resultado2 é $resultado2<br>");
```

Por Referência

```
function subtracao_ref(&$num1, &$num2){
        if($num1 < $num2){
                die("Números negativos");
        }else{
                $return_result=0;
                while($num1 > $num2){
                        $num1 = $num1 - 1;
                        $return_result = $return_result + 1;
                }
        }
        return($return_result);
}

$primeiro_op=493;
$segundo_op=355;
$resultado1 = subtracao_ref($primeiro_op, $segundo_op);
print ("<br><br>Resultado1 é $resultado1<br>");
$resultado2 = subtracao_ref($primeiro_op, $segundo_op);
print("Resultado2 é $resultado2<br>");

echo "Agora, se se nós executarmos exatamente a mesma chamada da subtração como
fizemos a primeira vez, receberemos a saída:
resultado1 é 138 e resultado2 é 0";


/*
Sugestão de chamada de função:

if (nome_funcao($argumento){
        ....
        ....
}else{
        ....
}
```

```
*/
?>

<?php
// Retorna o tipo e o valor de variável
function ss_as_string (&$thing, $column = 0) {
    if (is_object($thing)) {
        return ss_object_as_string($thing, $column);
    }
    elseif (is_array($thing)) {
        return ss_array_as_string($thing, $column);
    }
    elseif (is_double($thing)) {
        return "Double(".$thing.")";
    }
    elseif (is_long($thing)) {
        return "Long(".$thing.")";
    }
    elseif (is_string($thing)) {
        return "String(".$thing.")";
    }
    else {
        return "Unknown(".$thing.")";
    }
}

// Retorna o tipo e o valor de array
function ss_array_as_string (&$array, $column = 0) {
    $str = "Array(<BR>\n";
    while(list($var, $val) = each($array)){
        for ($i = 0; $i < $column+1; $i++){
            $str .= "    ";
        }
        $str .= $var.' ==> ';
        $str .= ss_as_string($val, $column+1)."<BR>\n";
    }
    for ($i = 0; $i < $column; $i++){
        $str .= "    ";
    }
    return $str.')';
}

// Retorna o tipo e o valor de objeto
function ss_object_as_string (&$object, $column = 0) {
    if (empty($object->classname)) {
        return "$object";
    }
    else {
        $str = $object->classname."(<BR>\n";
        while (list(,$var) = each($object->persistent_slots)) {
            for ($i = 0; $i < $column; $i++){
                $str .= "    ";
            }
            global $$var;
            $str .= $var.' ==> ';
            $str .= ss_as_string($$var, column+1)."<BR>\n";
        }
        for ($i = 0; $i < $column; $i++){
            $str .= "    ";
        }
```

```
        return $str.')';
    }
}

$var="Riba";
echo ss_as_string($var);
//echo ss_as_string($GLOBALS);
```

Otimização do tempo de execução

```
function ss_timing_start ($name = 'default') {
    global $ss_timing_start_times;
    $ss_timing_start_times[$name] = explode(' ', microtime());
}

function ss_timing_stop ($name = 'default') {
    global $ss_timing_stop_times;
    $ss_timing_stop_times[$name] = explode(' ', microtime());
}

function ss_timing_current ($name = 'default') {
    global $ss_timing_start_times, $ss_timing_stop_times;
    if (!isset($ss_timing_start_times[$name])) {
        return 0;
    }
    if (!isset($ss_timing_stop_times[$name])) {
        $stop_time = explode(' ', microtime());
    }
    else {
        $stop_time = $ss_timing_stop_times[$name];
    }
    // do the big numbers first so the small ones aren't lost
    $current = $stop_time[1] - $ss_timing_start_times[$name][1];
    $current += $stop_time[0] - $ss_timing_start_times[$name][0];
    return $current;
}

?>
```

Função com parâmetro default

Parâmetro default é aquele cujo valor já faz parte da função, sendo opcional.

```
function cliente($codigo, $idade = 18){
        print "Código = $codigo, Idade = $idade";
}

cliente(1);  //Exibirá: Código = 1, Idade = 18
```

Trabalhando com Funções OB

[- Artigo do Adriano Oliveira Gonçalves na Revista do PHP]

3.8 - Trabalhando com Session no PHP

Sessões são usadas para guardar dados enquanto a janela do browser estiver aberta. São geralmente usadas para manter dados como nome do usuário, tipo do usuário (se é o administrador ou se é um visitante não cadastrado, por exemplo), entre outros dados

importantes. Dica: NUNCA COLOQUE A SENHA NA SESSÃO. VOCÊ SÓ PRECISA DELA PARA AUTENTICAÇÃO!
Para iniciar uma sessão no PHP usamos a função:

```
session_start();
```

Se quisermos destruir a sessão (no caso de o usuário ter feito logoff), usamos a função:

```
session_destroy();
```

Agora precisamos apenas setar os valores que ficarão na sessão. No PHP os valores são armazenados em um vetor associativo chamado $_SESSION. As associações são feitas com pares chave e valor. Por exemplo, vamos setar o nome do usuário e a permissão dele, e colocar esses dados na seção. Vamos buscar esses dados a partir de um formulário fictício, via POST.

```
< ?php
    session_start(); //iniciamos a sessão
    $_SESSION['usuario'] = $_POST['usuario']; //colocamos na sessão o valor do
campo usuário vindo do formulário
    $_SESSION['permissao'] = $_POST['admin']; //da mesma forma setamos suas
permissões
    header("location:pagina_principal.php"); //Redirecionamos para a pagina
principal
? >
```

Com isso feito, podemos acessar as variáveis da sessão de qualquer lugar da nossa aplicação através do vetor $_SESSION.

Fonte: http://maozinhadaweb.blogspot.com/2007/05/tutorial-de-php-parte-3-sesses-e.html

Sessions em PHP
Session ou sessão é o método ou via pela qual podemos gravar informação de variáveis, funcionando assim como uma variável global. A grande vantagem das sessions é podermos navegar por várias páginas sem que esta informação se perca evitando assim passar o valor das variáveis indivídualmente de página em página.

Ao contrário dos cookies que ficam gravadas no computador do utilizador, as sessions são gravadas no servidor e automaticamente eliminadas quando o utilizador fecha o browser.

Exemplificando:
Vamos criar a página pagina1.php

```
<?php
//Iniciar sessão
session_start();
//Gravar variáveis em sessão
```

```
$_SESSION[‘id’]=1;
$_SESSION[‘nome’]=’Paulo’;
//Imprimir o valor das variáveis guardadas
print(“$id”);
print(“$nome”);
?>
```

Podemos verificar que são imprimidos os valores das variáveis atribuídas à session: 1 e Paulo, ou seja, esses valores ficam gravados numa session e podem ser utilizados em qualquer página onde essa sessão esteja aberta.

Modificar e remover sessions

Quando utilizamos sessions podemos ter a necessidade de modificar ou remover variáveis de sessão.
Para alterar o valor de uma variável basta fazer nova atribuição:

```
$_SESSION[‘nome’]=’Pedro’;
```

A variável $nome passaria a ter o valor de Pedro a partir desta linha.

Para remover todas as variáveis da session basta colocar o comando session_unset() ou apenas unset() para remover variáveis indivídualmente.
Para eliminar ou destruir completamente toda a sessão utilizamos o comando session_destroy().

Como referido anteriormente, a sessão acaba sempre que o utilizador fechar o browser, mas a duração das sessões também podem ser alteradas para um espaço de tempo pre-definido. Para isso deverá ser alterado o php.ini na linha session.cookie_lifetime = 0 colocando o número de segundos desejados no lugar do 0 (zero).

Gravar sessions num directório

As sessões também podem ser gravadas num directório no servidor. Para que esta operação seja realizada devera ser dada a instrução através da função session_save_path().
Nesta função deverá ser dado o caminho para a pasta que guardará as sessões: session_save_path(“sessions”). Neste caso a pasta sessions que guardará as sessões fica localizada na raís.

Para que esta função funcione deverá ser colocada sempre antes do comando session_start().

Vamos experimentar:

```
 <?php
//Pasta onde vai gravar  sessão
session_save_path(“sessions”);
//Iniciar sessão
session_start();
//Gravar variáveis em sessão
```

```
$_SESSION[‘id’]=1;
$_SESSION[‘nome’]=’Paulo’;
//Imprimir o valor das variáveis guardadas
print(“$id”);
print(“$nome”);
?>
```

Pode confirmar que é gravado um ficheiro na pasta sessions com um nome do tipo “sess_t4k884rd8uunp8ojt83picc9j1”. Se abrir este ficheiro com o Notepad pode verificar que se encontram lá gravadas as variáveis dadas à sessão.

Convém salientar que, por razões óbvias, a pasta onde são gravadas as sessions deverá estar protegida.

Fonte: http://www.techcaffe.net/site/tutoriais.php?recID=4

Exemplo prático de Session em PHP

index.php

```
<?php
session_start();
?>
<h1 align="center">Usando SESSION em PHP</h1>
Podemos preservar valores de variáveis enquanto durar uma sessão do browser através do uso de SESSION.<br>
Para isso devemos startar a sessão em cada página que desejamos usar esta variável com<br><br>
sesssion_start();<br><br>
Lembrando que esta função deve vir antes de qualquer comando que mande algo para a tela, caso<br>
o session esteja configurado para usar cookie.<br>
Na primeira página deve ter um formulário com algum campo que devemos usar no session.<br>
Experimente gravar a URL de uma das páginas internas e acessar diretamente (http://localhost/session3)
<br>Primeiro feche todas as seções do browser e depois abra o browser com essa URL.<br>
<br>
Veja que SESSION é muito bom para preservar o valor de variáveis entre páginas de um site numa seção.<br>
Portanto seu uso é muito útil quando pretendemos autenticas os visitantes de todas as as páginas de um site.<BR>
```

Como também para outros usos em que pretendemos reaproveitar o valor de variáveis (algo como global).
Acompanhe este exemplo para ver detalhes.


```
<form method=post action=session2.php>
Login<input type=text size=8 name=login><br>
<input type=submit value=Enviar>
</form></center>
```

session2.php

```
<?php
session_start();
$_SESSION['login']=$_POST['login'];
if (isset($_SESSION['login'])){
echo "Entre. Session2. <a href=session3.php>Session3</a>";
} else {
echo "Acesso não autenticado!";
}
?>
```

session3.php

```
<?php
session_start();
if (isset($_SESSION['login'])){
  echo "Entre. Session3. <a href=session4.php>Session4</a><br>";
  echo "<a href=destruir.php>Drestruir Sessão</a>";
} else {
  echo "Acesso não autenticado!";
}
?>
```

session4.php

```
<?php
session_start();
if (isset($_SESSION['login'])){
```

```
echo "Entre. Session4. <a href=session5.php>Session5</a>";
} else {
echo "Acesso não autenticado!";
}
?>
```

session5.php

```
<?php
session_start();
if (isset($_SESSION['login'])){
echo "Entre. Session5. <a href=index.php>Index</a><br><br>";
echo "Informações: <br>ID da Sessão: <b>" . session_id() . "</b><br>Variável mantida pela SuperGlobal \$_SESSION: <b>" . $_SESSION['login'];
} else {
echo "Acesso não autenticado!";
}
?>
```

Dicas sobre Session:

```
print ini_get('max_execution_time');
```

```
ini_set("session.gc_maxlifetime","3600");
session_save_path("/my/own/path/without/url/sessions");
```

Bom tutorial em: http://www.phpro.org/tutorials/Introduction-To-PHP-Sessions.html

3.9 - Trabalhando com Cookies no PHP

Setando o Cookie

Para gravar um cookie no navegador do visitante usa-se a função setcookie:

setcookie*(nome, valor, data_expiração)*:

Exemplo:

```
//Calculate 60 dias no futuro
//segundos * minutos * horas * dias + tempo atual
$emDoisMeses = 60 * 60 * 24 * 60 + time();
setcookie('UltimaVisita', date("G:i - d/m/Y"), $emDoisMeses);
```

Recebendo o Cookie

```
if(isset($_COOKIE['UltimaVisita']))
      $visita = $_COOKIE['UltimaVisita'];
else
      print "You've got some stale cookies!";

echo "Sua última visita foi - ". $visita;
```

Fonte: http://www.tizag.com/phpT/phpcookies.php

3.10 - Tratamento de Erros no PHP

Os 25 erros de programação mais perigosos segundo a SANS
By coredump

Saiu no site da SANS a lista criada com o consenso entre varios profissionais e empresas do ramo de segurança e desenvolvimento descrevendo os 25 erros de programação mais perigosos para o desenvolvimento seguro. Eu vou traduzir os nomes e informação básicos mas o melhor é ler o artigo na íntegra, em inglês.

Os erros estão separados em três categorias: Interação insegura entre componentes, Gerenciamento arriscado de recursos, Defensas porosas.

Categoria: Interação insegura entre componentes

1. **Validação Imprópria de Entradas**: Entradas que recebem dados e os aceitam mesmo sem certificar que eles são do tipo/formato esperado.
2. **Codificação ou Escape Impróprios de Saída**: Saídas que não são codificadas ou escapadas corretamente são a maior fonte de ataques de injeção de código.
3. **Falha ao Preservar a Estrutura da Busca SQL (conhecido como Injeção de SQL)**: Se os atacantes podem influenciar as procuras SQL do seu programa, então eles podem controlar o seu banco de dados.
4. **Falha ao Preservar a Estrutura do Código da Página (conhecido como "Cross-site Scripting")**: Assim como o anterior, se os atacantes podem injetar código ou scripts em sua página, eles podem controlar a página.
5. **Falha ao Preservar a Estrutura de Comandos do Sistema Operacional**: Se você permitir que entradas ilegais sejam passadas para aplicativos do sistema operacional, o atacante pode controlar o servidor.
6. **Transmissão de Dados Sensíveis em Texto Puro**: Senhas, dados de cartão e qualquer informação considerada sensível deve ser criptografada.
7. **Falsificação de Requisição Entre Sites**: Um atacante pode criar uma requisição que é enviada a outro site forjando a origem e fazendo o mesmo partir de um usuário inocente, aproveitando credenciais de autenticação e acessos.
8. **Condição de Corrida**: Atacantes vão sempre procurar por condições de corrida no software para conferir se alguma informação importante não é obtida no processo.
9. **Vazamento de Informações em Mensagens de Erro**: Atacantes vão procurar por mensagens de erro que descrevam mais que o necessário, como nomes de campos SQL, objetos e bibliotecas sendo utilizadas.

Categoria: Gerenciamento arriscado de recursos:

1. **Falha ao Limitar Operações aos Limites de um Buffer de Memória**: O conhecido buffer overflow.
2. **Controle Externo de Dados Sensíveis**: Informações críticas que são mantidas fora de um banco de dados por questões de performance não deviam ser facilmente acessíveis por atacantes.
3. **Controle Externo de de Caminho ou Nome de Arquivo**: Quando você usa dados externos para montar um nome de arquivo ou caminho de gravação, você está se arriscando a ser atacado.
4. **Caminho de Procura Inseguro**: Se o caminho de procura de recursos estiver em algum lugar sob controle de um atacante, bibliotecas ou código pode ser inserido a revelia.
5. **Falha ao Controlar a Geração de Código**: Caso o atacante consiga influenciar a geração de código dinâmico (se geração de código dinâmico for utilizada no programa) ele poderá controlar todo seu código.
6. **Download de Código sem Verificação de Integridade**: Se você executa código obtido por download, você confia na fonte. Atacantes podem aproveitar esta confiança.
7. **Desligamento ou Liberação Impróprias de Recursos**: Arquivos, conexões e classes precisam ser corretamente encerradas.
8. **Inicialização Imprópria**: Dados, bibliotecas e sistemas inicializados incorretamente podem abrir margens para problemas.
9. **Cálculos Incorretos**: Quando o atacante tem algum controle sobre as entradas usadas em operações matemáticas, isso pode gerar vulnerabilidades.

Categoria: Defensas porosas:

1. **Controle de Acesso Impróprio**: Se você não garante que seus usuários estão fazendo apenas o que deviam, os atacantes irão se aproveitar de sua autenticação.
2. **Uso de um Algoritmo Criptográfico Quebrado ou Vulnerável**: Utilização de algoritmos fracos ou comprometidos levam a falhas de criptografia e vulnerabilidades.
3. **Senha no Código**: deixar um usuário e uma senha no próprio código traz inúmeros problemas.
4. **Permissão de Acesso Insegura para Recurso Crítico**: Configurações, arquivos de dados e bancos de dados devem ter suas permissões de acesso protegidas.
5. **Uso de Valores Insuficientemente Aleatórios**: Se você usa tipos de segurança que dependem de aleatoriedade, usar um gerador aleatório insuficiente só vai causar problemas.
6. **Execução com Privilégios Desnecessários**: Se seu programa precisa de privilégios elevados para executar suas funções, ele deve abrir mão destes direitos assim que ele termina de executar as ações que precisavam dos privilégios.
7. **Aplicação de Segurança do Lado do Servidor pelo Cliente**: Atacantes podem usar engenharia reversa em um cliente de software e escrever seus próprios clientes removendo testes e aplicações de segurança.

Algumas coisas foram realmente chatas de traduzir, sinta-se livre para sugerir correções.

intel

PS: Lembre-se sempre “os 25 mais” não quer dizer “os 25 únicos”. *Grain of Salt* faz bem.

Fonte: http://core.eti.br/2009/01/22/os-25-erros-de-programacao-mais-perigosos-segundo-a-sans/

Exemplo simples de uso de Exception

```
<?php
/*
http://www.sourcecode4you.com/article/articleview/956dd1bb-a5a2-42ad-ae31-
ad836eec24f3/try-catch-block-in-php.aspx

Try Catch Block In Php

Try catch block handle runtime exception which occur in function.
*/

function dividebyZero()
{
	try
	{
		$k= 10/0;
	}
	catch(Exception $ex)
	{
		echo $ex;
	}
}

dividebyZero();

function throwException()
{
	try
	{
		throw new Exception("My Custom Exception");
	}
	catch(Exception $ex)
	{
		echo $ex;
	}
}
throwException();
?>
```

Introdução a manipulação de erros em PHP
Autor: Lorran Luiz <lorranluiz at click21.com.br>
Data: 22/01/2009

Introdução

Na história humana muitos conquistaram grandes coisas. Tiveram problemas é claro, mas em suas decisões não erraram ou erraram pouco (pois é, com os erros que aprendemos). Os erros enfim tiveram um papel relativamente importante para o sucesso dessas pessoas. Foi com eles (os erros) que as soluções apareceram, pois não existe solução se não há um erro para ser consertado.

É claro que erros nos decepcionam, mas eles só refletem a realidade - de que ainda não

estamos completamente qualificados a fazer tal ato com êxito completo. Os erros mostram que não somos perfeitos.

A fato de não sermos perfeitos esbarra no que fazemos. Enfim, os programadores erram, os usuários podem errar ou até mesmo alguma parte do programa que está a ser executado pode falhar de alguma maneira.

São nesses erros que envolvem o aplicativo e/ou os atores envolvidos no sistema que grandes desastres começam a se formar. Verdadeiras bolas de neve vão crescendo a medida que um programador digita seu código sem preocupar-se com a menor possibilidade de algo dar errado.

Depois que um sistema está completamente construído, sem base alguma para o tratamento correto de erros, o "problema quase que irresolvível" estará montado! A cada passo na vida desse sistema a possibilidade da ocorrência de erros "não identificáveis" aumentará, e a cada erro, mais e mais do seu tempo se consumirá para se chegar a resolução do mesmo. O sistema enfim se torna insustentável e todas aquelas horas de trabalho foram desperdiçadas!

Situações como estas poderiam ser evitadas se medidas simples como a construção de um código "supervisionado" com cláusulas e blocos que garantem a manipulação correta e segura de erros fossem implementadas logo nas primeiras linhas.

Situações de possíveis erros

Veremos a partir de então o básico quando o assunto é manipulação de erros. Com tal conhecimento o programador se capacitará a avançar no estudo de tratamento de erros e posteriormente aprender técnicas para tal.

Algumas situações

Suspeite de qualquer situação em que seu código fará algo decisivo para o funcionamento do seu sistema. Coisas como conectar a um banco de dados, abrir algum arquivo que contenha parâmetros importantes, carregar uma classe (ou uma biblioteca) essencial na aplicação ou armazenar alguma informação numa tabela de banco de dados, entre muitas outras situações requerem uma execução sem falhas e por isso deve haver uma preocupação maior com os erros que acontecerem.

Podemos dizer que em certas partes de seu código necessitaremos de "vigilantes" atentos a possíveis situações anormais. Veremos então quem são os vigilantes da programação no PHP.

Os "vigilantes"

Podemos rodear determinadas áreas de seu script com "vigilantes", que esperam que algo dê errado para passar o controle da situação aos seus "superiores".

Usamos a palavra-chave *try* para representar esses vigias virtuais no seu código fonte.

O try é seguido de um bloco de códigos no qual ele estará responsável por capturar eventuais erros. Os erros são denominados exceções, ou seja, situações que saem do esperado.

Sintaxe:

```
try {
//Código suspeito
}
```

Exceções
Em orientação a objeto, no PHP, exceções representam um objeto. Esse objeto necessita de um "molde" para ser construído, esse molde é denominado *Classe* e a classe usada para "modelar" uma exceção é a classe *Exception*.

Abaixo vemos a classe Exception:

```
<?php
class Exception {

  protected $message = 'Unknown exception'; // Mensagem da exceção
  protected $code = 0;                      // Código da exceção definido pelo usuário
  protected $file;                          // Arquivo gerador da exceção
  protected $line;                          // Linha geradora da exceção

  function __construct(string $message=NULL, int code=0);

  final function getMessage();              // Mensagem da exceção
  final function getCode();                 // Código da exceção
  final function getFile();                 // Arquivo gerador
  final function getTrace();                // um array com o backtrace()
  final function getTraceAsString();        // String formatada do trace

  /* Sobrecarregável */
  function _toString();                     // String formatada para ser mostrada

}
?>
```

A classe Exception não precisa ser inserida em seu código, ela é uma classe nativa do PHP introduzida no Zend 2.

Os "superiores"
Os "superiores" são aqueles que tem competência para fazer algo, devem resolver um problema ou passar essa responsabilidade para quem resolva. No PHP usamos a palavra-chave *catch* para representar esse superior.

O "catch" é responsável pela identificação do erro, ou seja, ele "sabe" qual o tipo de exceção ocorreu, de acordo com a classe, derivada de Exception, que "modelou" o objeto de exceção passado para ele.

O "catch" necessita de um parâmetro, o nome da classe Exception ou derivada seguida da variável que conterá o objeto da exceção e um bloco de código.

Sintaxe:

```
catch (Exception $variavel) {
//Código que manipulará tal exceção
}
```

A palavra-chave catch deve ser posicionada logo após o bloco de código rodeado pela cláusula try. Ficando então assim:

Sintaxe:

```
try { //Código suspeito } catch (Exception $variavel) { //Código que manipulará tal exceção }
```

Preparando seu código para as exceções

Já vimos como cercar uma parte do código para manipular os erros que nela ocorrerem. Veremos a partir de então como "delatar" os erros e alertar o sistema quanto a eles.

Construindo uma exceção

O método construtor da classe Exception requer 2 parâmetros, sendo 1 opcional:

```
public Exception::__construct ([ string $message=NULL [, int $code ]] )
```

O primeiro parâmetro é a mensagem do erro e o segundo o código do erro (opcional).

Disparando o alarme

Em PHP usamos a palavra-chave *throw* para alertar o sistema da ocorrência de um erro, uma exceção. Throw é seguido de um código que solicita a construção de um novo objeto de exceção.

Para instanciar uma classe no PHP usamos a palavra-chave *new* seguida do nome da classe. Então a sintaxe ficará assim:

```
throw new Exception("Mensagem", 1);
```

Como usar "throw"

Podemos usar a condicional *if* para decidirmos quando disparar uma exceção.

Veja o exemplo abaixo:

```
if (!@mysql_connect ("localhost", "usuário", "senha")) throw new Exception("Não foi possível conectar ao banco de dados");
```

Obs.: Podemos usar o operador de supressão (@) de erros para evitar que sejam exibidas as mensagens de erro padrão do PHP.

Estendendo e especificando exceções

Podemos tornar nossas exceções mais específicas e por consequência saber com mais precisão que tipo de erro ocorreu, manipulando-o assim da melhor maneira possível.

Usamos a palavra-chave *extends* para estender uma classe, ou seja, criar uma classe derivada (ou classe filha). Quando uma classe estende outra, ela herda da última atributos e métodos dependendo do nível de acesso de cada elemento que constitui o corpo da classe. A classe filha pode sobrecarregar métodos (redefiní-los).

Exception é uma classe que pode ser herdada, sendo assim podemos definir classes que derivem e aumentem sua funcionalidade, passando de uma simples exceção genérica para exceções mais precisas.

Exceções derivadas
Definamos algumas classes derivadas de Exception:

```
class DBException extends Exception {
    protected $variavel; //Uma variavel qualquer

    public function __construct($exceptionMessage, $errorCode = 0) {
        parent::__construct($exceptionMessage, $errorCode);
    }

    //Aqui vemos alguns métodos característicos da classe derivada de Exception
    public function getVariavel() {
        return $this->variavel_qualquer;
    }

    public function setVariavel($valor = NULL) {
        $this->variavel = $valor;
    }
}

class FopenException extends Exception {
    protected $arquivo; //Variável contendo o nome de um arquivo

    public function __construct($nomeDoArquivo, $exceptionMessage, $errorCode = 0) {
        parent::__construct($exceptionMessage, $errorCode);
        $this->arquivo = $nomeDoArquivo;
    }

    //Métodos característicos dessa classe
    public function getNomeDoArquivo() {
        return $this->arquivo;
    }

    public function exibirNomeDoArquivo() {
        echo "Nome do arquivo: {$this->arquivo}";
    }

}
```

Um código básico com tratamento de erros

Veremos e entenderemos como funcionaria então um código básico completo que trataria

de uma maneira certa os possíveis erros que ocorressem durante sua execução.

O código
O código que se seguirá é meramente explicativo e por isso talvez não possua utilidade numa situação real.

- Criaremos um código onde será criado um manipulador de arquivos;
- O conteúdo do arquivo será lido;
- Efetuaremos uma conexão com o servidor de banco de dados;
- Esse conteúdo será inserido numa célula de uma tabela de um banco de dados;
- E se tudo ocorrer bem, será exibida a mensagem: Operação efetuada com sucesso;
- Caso ocorra um erro será exibida uma mensagem de erro que será reportada ao programador.

Veja abaixo:

```
//A CLASSE:

class ArmazenadorDeArquivoEmDB {
   protected $db_obj;
   protected $db_usuario;
   protected $db_senha;
   protected $db_dsn;
   protected $nomeDoArquivo;
   protected $manipuladorDoArquivo;
   protected $conteudoDoArquivo;
   protected $db_tabela;
   protected $db_dbName;

   public function __construct($usuario, $senha, $tabela, $db, $servidor) { //Construir
objeto
      $this->db_usuario = $usuario;
      $this->db_senha = $senha;
      $this->db_tabela = $tabela;
      $this->db_dbName = $db;
      $this->db_dsn = $servidor;
   }

   public function conectar() {
      if(!($this->db_obj = @mysql_connect($this->db_dsn, $this->db_usuario, $this-
>db_senha))) {
         throw new DBException("Não foi possível conectar ao banco de dados", 281);
//Lembrando que o código de erro não é um parâmetro obrigatório
      }
   }

   public function armazenarArquivoNoDB($nomeDoArquivo) { //O parâmetro requer
endereço completo do arquivo
      $this->nomeDoArquivo = $nomeDoArquivo;
```

```
        try {
            if (empty($this->db_obj)) { //Se não existir uma conexão aberta com o DB
                $this->conectar();
            }
          } catch  ( DBException $e ) { //Caso haja algum erro durante a tentativa de
conexão...
            throw $e;                    // ...o "alarme" irá disparar
          }

        //Iremos abrir o arquivo:
        if (!file_exists($nomeDoArquivo)) throw new FopenException($nomeDoArquivo, "O
arquivo especificado não existe.", 282);
        if (!($this->manipuladorDoArquivo = @fopen($nomeDoArquivo, "r"))) throw new
FopenException($nomeDoArquivo, "Não foi possível abrir um manipulador para este
arquivo.", 283);
        if (!($this->conteudoDoArquivo = @fread($this->manipuladorDoArquivo,
intval(sprintf("%u", filesize($nomeDoArquivo)))))) throw new
FopenException($nomeDoArquivo, "Não foi possível abrir o conteúdo deste arquivo.",
284); //O 2º parâmetro de fread() possibilita a leitura de arquivos maiores que 2GB

        //Armazenaremos agora o conteúdo do arquivo no DB
        $sql = 'INSERT INTO `' . $this->db_dbName . '`.`' . $this->db_tabela . '` (`nome`,
`conteudo`) VALUES (\'' . $this->nomeDoArquivo . '\', \'". $this->conteudoDoArquivo . '\');';
        if(!($insert = @mysql_query($sql))) throw new DBException("Não foi possível inserir
o arquivo no banco de dados.", 285);
        else echo '<html><head></head><body onload="alert(\'O arquivo foi armazenado
com sucesso no banco de dados\')"></body></html>';
    }
}

//O OBJETO EM AÇÃO:

$armazenadorDeArquivoEmDB = new ArmazenadorDeArquivoEmDB("usuario", "senha",
"tabela", "banco_de_dados", "sevidor"); //Criamos um objeto que armazena um arquivo
num banco de dados
try { //Tentiva de conexão
    $armazenadorDeArquivoEmDB->armazenarArquivoNoDB("Ergue-te Marcos.txt"); //Aqui
entra o nome do arquivo a ser armazenado no DB
} catch ( DBException $e ) {
    //Houve um erro relativo ao banco de dados
    echo nl2br("<b>{$e->getMessage()}</b>\n<br />Detalhes:\n{$e->__toString()}"); //Exibir
string contendo informações sobre a exceção
    $gossipy->reportar($e); //A nível exemplificativo, temos este objeto fictício, imaginário,
que envia por email informações sobre o erro ao programador
} catch ( FopenException $e ) {
    //Houve um erro referente ao arquivo, e podemos dar um tratamento específico a este
tipo de exceção!
    echo nl2br("<b>{$e->getMessage()}</b>\n<br />Detalhes:\n{$e->__toString()}"); //Exibir
string contendo informações sobre a exceção
    $gossipy->reportar($e); //A nível exemplificativo, temos este objeto fictício, imaginário,
```

que envia por email informações sobre o erro ao programador
}
Poderíamos ter explorado mais desta fantástica linguagem que é o PHP e ter feito um tratamento de erros mais proveitoso no código acima, mas acho que para fins meramente explicativos e didáticos não é necessário tanta complexidade.

Vocês podem ver que poderíamos ter criado no código acima subclasses de Exception bem mais específicas, a idéia é a mesma de quando criamos as classes "FopenException" e "DBException". Classes mais específicas para a manipulação de exceções relativas ao banco de dados poderiam ser definidas apenas estendendo a classe "DBException", veja:

SQLException extends DBException { }

E cada vez podemos ir tornando as exceções mais específicas (e a manipulação mais precisa).

Resumindo

Faremos então uma compilação de tudo o que aprendemos:

- Neste artigo revisamos conceitos básicos do poderoso recurso que é a orientação a objetos do PHP 5;
- Vimos algumas situações de risco que podem surgir durante a execução de seu código;
- Vimos também que podemos aprender com as situações frustradoras se tratarmos melhor as informações relativas a aquele problema;
- Vimos que o PHP 5 conta com excelentes ferramentas para tratarmos da maneira certa as exceções que ocorrerem;
- Estudamos as principais palavras-chave quando o assunto é tratamento de erros em PHP 5: try, catch e throw;
- Estudamos superficialmente a classe Exception;
- Aprendemos a disparar o "sinal de alerta" aproveitando a condicional "if" e usando-a em conjunto com throw.

Aprendemos o funcionamento básico do "sistema" de manipulação dos erros no PHP 5:

Quando um erro (exceção) ocorre, este é "percebido" (com por exemplo a condicional "if"), instancia-se então a classe Exception (com "throw new Exception('msg', 01)") ou uma derivada, o controle é passado para o bloco "catch" correspondente, que por vez pode usar as informações que recebeu a respeito do erro na sua manipulação, como melhor convir.

Depois que todo o código do bloco catch que manipulou o erro é executado, o controle do código volta ao escopo mais geral (exceto se uma função exit() ou similar for executada dentro do bloco catch, o que terminaria a execução do código neste instante).

Encerrando

Espero sinceramente ter sido claro, e mesmo sem aprofundar muito neste estudo, ter passado o máximo de conhecimento a respeito dos fundamentos do tratamento de erro

com exceções em PHP.

Continuemos a estudar e talvez um dia melhorar nosso país com o conhecimento!

Um abraço para todos da comunidade VOL!

L. Luiz

---

http://www.vivaolinux.com.br/artigo/Introducao-a-manipulacao-de-erros-em-PHP

3.11 - Trabalhando com Validação de Dados no PHP

Ótimo artigo em 5 partes na Revista do PHP de autoria do Er Abbott
- Validação de Formulários - 5º e última parte: Validando no servidor

- Validação de Formulários - Parte 04 (A fonteira cliente/servidor)

- Validação de Formulários - Parte 3: O baile de máscaras

- Validação de Formulários - Parte 2: Os Campos Especiais

- Validação de formulários - Parte 1 (O Planejamento)

Turbinando a Validação de Formulários

Validação de e-mails
Check valid e-mail

```
function esEmailValido($email)
{
    if (ereg("^[_a-zA-Z0-9-]+(\.[_a-zA-Z0-9-]+)*@([_a-zA-Z0-9-]+\.)*[a-zA-Z0-9-]
{2,200}\.[a-zA-Z]{2,6}$", $email ) )
        {
       return true;
    }
        else
        {
       return false;
    }
}
```

http://snippets.dzone.com/posts/show/4346

Tipos de Variáveis

```
<?php

function ss_array_as_string (&$array, $column = 0) {
    $str = "Array(<BR>\n";
    while(list($var, $val) = each($array)){
        for ($i = 0; $i < $column+1; $i++){
            $str .= "    ";
```

```
        }
        $str .= $var.' ==> ';
        $str .= ss_as_string($val, $column+1)."<BR>\n";
    }
    for ($i = 0; $i < $column; $i++){
        $str .= "    ";
    }
    return $str.')';
}

function ss_object_as_string (&$object, $column = 0) {
    if (empty($object->classname)) {
        return "$object";
    }
    else {
        $str = $object->classname."(<BR>\n";
        while (list(,$var) = each($object->persistent_slots)) {
            for ($i = 0; $i < $column; $i++){
                $str .= "    ";
            }
            global $$var;
            $str .= $var.' ==> ';
            $str .= ss_as_string($$var, column+1)."<BR>\n";
        }
        for ($i = 0; $i < $column; $i++){
            $str .= "    ";
        }
        return $str.')';
    }
}

function ss_as_string (&$thing, $column = 0) {
    if (is_object($thing)) {
        return ss_object_as_string($thing, $column);
    }
    elseif (is_array($thing)) {
        return ss_array_as_string($thing, $column);
    }
    elseif (is_double($thing)) {
        return "Double(".$thing.")";
    }
    elseif (is_long($thing)) {
        return "Long(".$thing.")";
    }
    elseif (is_string($thing)) {
        return "String(".$thing.")";
    }
    else {
        return "Unknown(".$thing.")";
    }
}

$my_variable=3;
//echo ss_as_string($my_variable);
echo ss_as_string($GLOBALS);
?>
```

Validação Usando Filtros do PHP

Suportados pelo PHP com versão 5.2 ou superior.

Constantes pré-definidas

As contantes abaixo são definidas por esta extensão e somente estarão disponíveis quando a extensão foi compilada com o PHP ou carregada dinamicamente durante a execução.

INPUT_POST (integer)
Variáveis POST.
INPUT_GET (integer)
Variáveis GET.
INPUT_COOKIE (integer)
Variáveis COOKIE.
INPUT_ENV (integer)
Variáveis ENV.
INPUT_SERVER (integer)
Variáveis SERVER.
INPUT_SESSION (integer)
Variáveis SESSION. (ainda não implementada)
INPUT_REQUEST (integer)
Variáveis REQUEST. (ainda não implementada)

FILTER_FLAG_NONE (integer)
Sem flags.
FILTER_REQUIRE_SCALAR (integer)
Flag usada para requerir escalar como entrada
FILTER_REQUIRE_ARRAY (integer)
Requer um array como entrada.
FILTER_FORCE_ARRAY (integer)
Sempre retorna um array.
FILTER_NULL_ON_FAILURE (integer)
Usa NULL ao invés de FALSE em falha.
FILTER_VALIDATE_INT (integer)
ID do filtro "int".
FILTER_VALIDATE_BOOLEAN (integer)
ID do filtro "boolean".
FILTER_VALIDATE_FLOAT (integer)
ID do filtro "float".
FILTER_VALIDATE_REGEXP (integer)
ID do filtro "validate_regexp".
FILTER_VALIDATE_URL (integer)
ID do filtro "validate_url".
FILTER_VALIDATE_EMAIL (integer)
ID do filtro "validate_email".
FILTER_VALIDATE_IP (integer)
ID do filtro "validate_ip".
FILTER_DEFAULT (integer)
ID do filtro padrão ("string").

FILTER_UNSAFE_RAW (integer)
ID do filtro "unsafe_raw".
FILTER_SANITIZE_STRING (integer)
ID do filtro "string".
FILTER_SANITIZE_STRIPPED (integer)
ID do filtro "stripped".
FILTER_SANITIZE_ENCODED (integer)
ID do filtro "encoded".
FILTER_SANITIZE_SPECIAL_CHARS (integer)
ID do filtro "special_chars".
FILTER_SANITIZE_EMAIL (integer)
ID do filtro "email".
FILTER_SANITIZE_URL (integer)
ID do filtro "url".
FILTER_SANITIZE_NUMBER_INT (integer)
ID do filtro "number_int".
FILTER_SANITIZE_NUMBER_FLOAT (integer)
ID of "number_float" filter.
FILTER_SANITIZE_MAGIC_QUOTES (integer)
ID do filtro "magic_quotes".
FILTER_CALLBACK (integer)
ID do filtro "callback".
FILTER_FLAG_ALLOW_OCTAL (integer)
Permite notação octal (*0[0-7]+*) no filtro "int".
FILTER_FLAG_ALLOW_HEX (integer)
Permite notação hexadecimal (*0x[0-9a-fA-F]+*) no filtro "int".
FILTER_FLAG_STRIP_LOW (integer)
Remove caracteres com valor ASCII menor que 32.
FILTER_FLAG_STRIP_HIGH (integer)
Remove caracteres com valor ASCII maior que 127.
FILTER_FLAG_ENCODE_LOW (integer)
Codifica caracteres com valor ASCII menor que 32.
FILTER_FLAG_ENCODE_HIGH (integer)
Codifica caracteres com valor ASCII maior que 127.
FILTER_FLAG_ENCODE_AMP (integer)
Codifica *&*.
FILTER_FLAG_NO_ENCODE_QUOTES (integer)
Não codifica *'* e *"*.
FILTER_FLAG_EMPTY_STRING_NULL (integer)
(Nenhum uso no momento.)
FILTER_FLAG_ALLOW_FRACTION (integer)
Permite parte fracional no filtro "number_float".
FILTER_FLAG_ALLOW_THOUSAND (integer)
Permite separador de milhar (*,*) no filtro "number_float".
FILTER_FLAG_ALLOW_SCIENTIFIC (integer)
Permite notação científica (*e*, *E*) no filtro "number_float".
FILTER_FLAG_SCHEME_REQUIRED (integer)
Requer scheme no filtro "validate_url".
FILTER_FLAG_HOST_REQUIRED (integer)
Requer host no filtro "validate_url".

`FILTER_FLAG_PATH_REQUIRED` (integer)
Requer path no filtro "validate_url".
`FILTER_FLAG_QUERY_REQUIRED` (integer)
Requer query no filtro "validate_url".
`FILTER_FLAG_IPV4` (integer)
Permite somente endereço IPv4 no filtro "validate_ip".
`FILTER_FLAG_IPV6` (integer)
Permite somente endereço IPv6 no filtro "validate_ip".
`FILTER_FLAG_NO_RES_RANGE` (integer)
Não permite endereços reservados no filtro "validate_ip".
`FILTER_FLAG_NO_PRIV_RANGE` (integer)
Não permite endereços privados no filtro "validate_ip".

Funções de Filtragem

filter_has_var — Verifica se a variável é de um especificado tipo existente

- filter_id — Retorna o ID de um dado nome de filtro
- filter_input_array — Obtem variáveis externas e opcionalmente as filtra
- filter_input — Obtem a específica variável externa pelo nome e opcionalmente a filtra
- filter_list — Retorna a lista de todos filtros suportados
- filter_var_array — Obtêm múltiplas variáveis e opcionalmente as filtra
- filter_var — Filtra a variável com um especificado filtro

Detectando o tipo de variável:

```
$_GET['test'] = 1;
echo filter_has_var(INPUT_GET, 'test') ? 'Sim' : 'Não';

Retornando uma lista com os tipos de filtros suportados
```

```
<?php
print_r(filter_list());
?>
```

Filter_input:

```
$search_html = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_SPECIAL_CHARS);
$search_url = filter_input(INPUT_GET, 'search', FILTER_SANITIZE_ENCODED);
echo "You have searched for $search_html.\n";
echo "<a href='?search=$search_url'>Search again.</a>";
```

filter_var

```
var_dump(filter_var('bob@example.com', FILTER_VALIDATE_EMAIL));
var_dump(filter_var('example.com', FILTER_VALIDATE_URL, FILTER_FLAG_SCHEME_
REQUIRED));
```

```php
<?php
error_reporting(E_ALL | E_STRICT);
$data = array(
    'product_id'    => 'libgd<script>',
    'component'     => '10',
    'versions'      => '2.0.33',
    'testscalar'    => array('2', '23', '10', '12'),
    'testarray'     => '2',
);

$args = array(
    'product_id'   => FILTER_SANITIZE_ENCODED,
    'component'    => array('filter'    => FILTER_VALIDATE_INT,
                            'flags'     => FILTER_FORCE_ARRAY,
                            'options'   => array('min_range' => 1,
'max_range' => 10)
                           ),
    'versions'     => FILTER_SANITIZE_ENCODED,
    'doesnotexist' => FILTER_VALIDATE_INT,
    'testscalar'   => array(
                            'filter' => FILTER_VALIDATE_INT,
                            'flags'  => FILTER_REQUIRE_SCALAR,
                           ),
    'testarray'    => array(
                            'filter' => FILTER_VALIDATE_INT,
                            'flags'  => FILTER_FORCE_ARRAY,
                           )

);

$myinputs = filter_var_array($data, $args);

var_dump($myinputs);
echo "\n";
?>


Just a little filter to validate IP v4 & v6
This little script display result in function of the query
<?php
$ipv6="2a01:e35:aaa4:6860:a5e7:5ba9:965e:cc93";
$ipv4="82.237.3.3";
$fake = "3342423423";
$ipv4priv = "255.255.255.255";
$ipv6priv = "::1";
echo "<pre>";
echo $ipv4;
echo "<br />";
var_dump(filter_var($ipv4,FILTER_VALIDATE_IP));
echo "<br />";
```

```
echo $ipv6;
echo "<br />";
var_dump(filter_var($ipv6,FILTER_VALIDATE_IP));
echo "<br />";
echo $fake;
echo "<br />";
var_dump(filter_var($fake,FILTER_VALIDATE_IP));
echo "<br />";
echo $ipv4priv;
echo "<br/>";
echo "FILTER_VALIDATE_IP, FILTER_FLAG_NO_RES_RANGE";
echo "<br />";
var_dump(filter_var($ipv4priv,FILTER_VALIDATE_IP,
FILTER_FLAG_NO_RES_RANGE));
echo "<br />";
echo $ipv4priv;
echo "<br/>";
echo "FILTER_FLAG_NO_PRIV_RANGE";
echo "<br />";
var_dump(filter_var($ipv4priv,FILTER_FLAG_NO_PRIV_RANGE));
echo "<br />";
echo $ipv6priv;
echo "<br/>";
echo "FILTER_FLAG_NO_PRIV_RANGE";
echo "<br />";
var_dump(filter_var($ipv6priv,FILTER_FLAG_NO_PRIV_RANGE));
echo "<br />";
echo $ipv6priv;
echo "<br />";
var_dump(filter_var($ipv6priv,FILTER_VALIDATE_IP));

echo "</pre>";
```

3.12 - Trabalhando com XML em PHP

XML no Manual do PHP - http://www.php.net/manual/pt_BR/book.xml.php
Introdução ao XML - http://www.criarweb.com/manuais/24

Lendo extrutura completa de arquivo XML:

data.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Titulo</TITLE>
 <para>
```

```
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <section id="about">
  <title>Sobre este documento</title>
  <para>
   <!-- this is a comment -->
   <?php echo 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </section>
</chapter>
```

ler_xml.php

```
<?php
//Exemplo de Entity externa

$file = "data.xml";

function trustedFile($file) {
   // only trust local files owned by ourselves
   if (!eregi("^([a-z]+)://", $file)
       && fileowner($file) == getmyuid()) {
           return true;
   }
   return false;
}

function startElement($parser, $name, $attribs) {
   echo "&lt;<font color=\"#0000cc\">$name</font>";
   if (sizeof($attribs)) {
       while (list($k, $v) = each($attribs)) {
           echo " <font color=\"#009900\">$k</font>=\"<font
                   color=\"#990000\">$v</font>\"";
       }
   }
   echo "&gt;";
}

function endElement($parser, $name) {
   echo "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}
```

```
function characterData($parser, $data) {
  echo "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {
  switch (strtolower($target)) {
    case "php":
      global $parser_file;
      // If the parsed document is "trusted", we say it is safe
      // to execute PHP code inside it.  If not, display the code
      // instead.
      if (trustedFile($parser_file[$parser])) {
        eval($data);
      } else {
        printf("Untrusted PHP code: <i>%s</i>",
            htmlspecialchars($data));
      }
      break;
  }
}

function defaultHandler($parser, $data) {
  if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
    printf('<font color="#aa00aa">%s</font>',
        htmlspecialchars($data));
  } else {
    printf('<font size="-1">%s</font>',
        htmlspecialchars($data));
  }
}
function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                          $publicId) {
  if ($systemId) {
    if (!list($parser, $fp) = new_xml_parser($systemId)) {
      printf("Could not open entity %s at %s\n", $openEntityNames,
          $systemId);
      return false;
    }
    while ($data = fread($fp, 4096)) {
      if (!xml_parse($parser, $data, feof($fp))) {
        printf("XML error: %s at line %d while parsing entity %s\n",
            xml_error_string(xml_get_error_code($parser)),
            xml_get_current_line_number($parser), $openEntityNames);
        xml_parser_free($parser);
        return false;
      }
    }
    xml_parser_free($parser);
    return true;
  }
```

```
   return false;
}

function new_xml_parser($file) {
   global $parser_file;

   $xml_parser = xml_parser_create();
   xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
   xml_set_element_handler($xml_parser, "startElement", "endElement");
   xml_set_character_data_handler($xml_parser, "characterData");
   xml_set_processing_instruction_handler($xml_parser, "PIHandler");
   xml_set_default_handler($xml_parser, "defaultHandler");
   xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

   if (!($fp = @fopen($file, "r"))) {
      return false;
   }
   if (!is_array($parser_file)) {
      settype($parser_file, "array");
   }
   $parser_file[$xml_parser] = $file;
   return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
   die("could not open XML input");
}

echo "<pre>";
while ($data = fread($fp, 4096)) {
   if (!xml_parse($xml_parser, $data, feof($fp))) {
      die(sprintf("XML error: %s at line %d\n",
                  xml_error_string(xml_get_error_code($xml_parser)),
                  xml_get_current_line_number($xml_parser)));
   }
}
echo "</pre>";
echo "parse complete\n";
xml_parser_free($xml_parser);
?>
```

Ler XML usando simplexml:

xml_lido.php

```
<?php
$xmlstr = <<<XML
<?xml version='1.0' encoding='ISO-8859-1' ?>
<filmes>
 <filme>
  <titulo>PHP: Iniciando o Parser</titulo>
  <personagens>
```

```
  <personagem>
  <nome>João de Brito</nome>
  <actor>Brito</actor>
  </personagem>
  <personagem>
  <nome>Manoel Cunha</nome>
  <actor>Manoel</actor>
  </personagem>
 </personagens>
 <comentario>
  O XML é uma linguagem. Ela é como uma linguagem de programação. Ou uma
  linguagem de script? Tudo será revelado após ler bem toda a
  documentação.
 </comentario>
 <votos type="thumbs">7</votos>
 <votos type="stars">5</votos>
 </filme>
</filmes>
XML;
?>
```

xml_ler.php

```
<?php
include 'xml_lido.php';
$xml = simplexml_load_string($xmlstr);

echo $xml->filme[0]->comentario; // "So this language. It's like..."
print '<br>';
echo $xml->filme[0]->titulo;
print '<br>';
echo $xml->filme[0]->personagens[0]->personagem[0]->nome;
print '<br>';
echo $xml->filme[0]->votos[0];
print '<br>';
echo $xml->filme[0]->votos[1];
?>
```

3.12 - Trabalhando com Constantes Mágicas e Superglonais em PHP

Variáveis do servidor

$_SERVER

Este é um array (vetor) 'superglobal', ou automaticamente global. Isto significa que ele é disponível em todos os escopos (níveis) de um script. Você não precisa fazer um: ... global $_SERVER; ... para poder acessá-lo dentro de funções ou métodos, como era necessário com $HTTP_SERVER_VARS. O array superglobal $_SERVER existe em qualquer sessão PHP e já contém um conjunto de chaves (índices) pré definidos e valorados. Os índices mais importantes são:

'REQUEST_URI'

O URI fornecido para acessar a página atual, por exemplo, '/index.html'.

'SCRIPT_NAME'

Contém o caminho completo do script atual. Útil para páginas que precisam apontar para elas mesmas (dinamicamente). A constante __FILE__ contém o caminho completo e nome do arquivo (mesmo incluído) atual.

'PHP_SELF'

O nome do arquivo do script atualmente em uso, relativo ao document root. Por exemplo, $_SERVER['PHP_SELF'] em um script com o endereço http://example.com/test.php/foo.bar pode ser /test.php/foo.bar. A constante __FILE__ contém o caminho completo e nome do arquivo (mesmo incluído) atual.

Se estiver rodando o PHP em linha de comando, esta variável não está disponível.

'SERVER_NAME'

O nome host do servidor onde o script atual é executado. Se o script está rodando em um host virtual, este será o valor definido para aquele host virtual.

'REQUEST_METHOD'

Contém o método de request utilizando para acessar a página. Geralmente 'GET', 'HEAD', 'POST' ou 'PUT'.

'QUERY_STRING'

A query string (string de solicitação), se houver, pela qual a página foi acessada.

'DOCUMENT_ROOT'

O diretório raiz sob onde o script atual é executado, como definido no arquivos de configuração do servidor.

'SCRIPT_FILENAME'

O caminho absoluto o script atualmente em execução.

Nota: Se o script for executado pela CLI com um caminho relativo, como file.php ou ../file.php, $_SERVER['SCRIPT_FILENAME'] irá conter o caminho relativo especificado pelo usuário.

Exemplos

```
$current_script = dirname($_SERVER['SCRIPT_NAME']);

$current_path  = dirname($_SERVER['SCRIPT_FILENAME']);

$request_uri = $_SERVER['REQUEST_URI'];

// Pick the predefined variable that works on your server
     return $_ENV['SCRIPT_URL'];

$_SERVER['QUERY_STRING'])

      $sPathPS = $_SERVER[PHP_SELF];
       $sPathFS = __FILE__;

echo 'http';
if($_SERVER['HTTPS']=='on'){echo 's';}
```

```
echo '://'.$_SERVER['SERVER_PORT'].$_SERVER['SCRIPT_NAME'];
if($_SERVER['QUERY_STRING']>' '){echo '?'.$_SERVER['QUERY_STRING'];}
```

Constantes Mágicas

```
__LINE__         A linha atual do script.

__FILE__         O caminho completo e nome do arquivo. Se utilizado dentro de um
include, o nome do arquivo incluído será retornado.

__FUNCTION__    O nome da função (Acrescentado no PHP 4.3.0). A partir do PHP 5
esta constante retorna o nome da função como ela foi
   declarada (sensível a maiúsculas e minúsculas). No PHP 4 sempre retorna o
nome em minúsculas.

__CLASS__       O nome da classe (Acrescentado no PHP 4.3.0). A partir do PHP 5
esta constante retorna o nome da função como ela foi
   declarada (sensível a maiúsculas e minúsculas). No PHP 4 sempre retorna o
nome em minúsculas.

__METHOD__      O nome do método de classe. (Acrescentado no PHP 5.0.0). O nome
do método é retornado como foi declarado (sensível a
   maiúsculas e minúsculas).

Exemplo:

if (realpath(__FILE__) == realpath($_SERVER['SCRIPT_FILENAME'])) {
  exit;
}
```

3.13 - Trabalhando com Formatação de Saída em PHP

Temos as três funções - printf, sprintf e vprintf

printf -- Mostra uma string formatada
void printf (string format [, mixed args])

sscanf -- Interpreta a entrada de uma string de acordo com um formato
mixed sscanf (string str, string formato [, string var1])

```
<?php
// Pegando o número serial
$serial = sscanf("SN/2350001","SN/%d");
// e a data de criação
$mandate = "January 01 2000";
list($month, $day, $year) = sscanf($mandate,"%s %d %d");
echo "O Item $serial foi criado em: $year-".substr($month,0,3)."-$day\n";
?>
```

Se parâetros opcionais são passados, a função retornará o número de valores assumidos. Os parâmetros opcionais devem ser passados por referência.

Exemplo 2. sscanf() - usando parâmetros opcionais

```
<?php
```

```
// pega informação do autor e gera uma entrada de DocBook
$auth = "24\tLewis Carroll";
$n = sscanf($auth,"%d\t%s %s", &$id, &$first, &$last);
echo "<author id='$id'>
  <firstname>$first</firstname>
  <surname>$last</surname>
</author>\n";
?>
```

fscanf -- Interpreta a leitura de um arquivo de acordo com um formato
mixed fscanf (resource handle, string formato [, string var1])

```
$handle = fopen ("users.txt","r");
while ($userinfo = fscanf ($handle, "%s\t%s\t%s\n")) {
  list ($name, $profission, $countrycode) = $userinfo;
  //... fazer algo com os valores
}
fclose($handle);

$goodevil = array ('There is a difference between %s and %s', 'good', 'evil');
echo call_user_func_array('sprintf',$goodevil);

<?php
$heading1 = "Label 1";
$heading2 = "Label 2";
$value1 = "31298";
$value2 = "98";

print "<pre>\n";
printf ("%'.-15.15s%'.6.6s\n", $heading1, $value1);
printf ("%'.-15.15s%'.6.6s\n", $heading2, $value2);
print "</pre>\n";
?>

<?php $f='<?php $f=%c%s%c; printf($f,39,$f,39); ?>'; printf($f,39,$f,39); ?>
```

sprintf -- Retorna uma string formatada
string sprintf (string format [, mixed args])

Um especificador de tipo que diz que o argumento deve ser tratado como do tipo. Os tipos possivéis são:

% - Um caractere porcento. Não é requerido neenhum argumento.
b - O argumento é tratado com um inteiro, e mostrado como um binário.
c - O argumento é tratado como um inteiro, e mostrado como o caractere ASCII correspondente.
d - O argumento é tratado como um inteiro, e mostrado como um número decimal com sinal.
u - O argumento é tratado com um inteiro, e mostrado como um número decimal sem

sinal.
f - O argumento é tratado como um float, e mostrado como um número de ponto flutuante.
o - O argumento é tratado com um inteiro, e mostrado como un número octal.
s - O argumento é tratado e mostrado como uma string.
x - O argumento é tratado como um inteiro, e mostrado como um número hexadecimal (com as letras minúsculas).
X - O argumento é tratado como um inteiro, e mostrado como um número hexadecimal (com as letras maiúsculas).

```
<?php
$format = "There are %d monkeys in the %s";
printf($format,$num,$location);
?>
```

Este deve mostrar, "There are 5 monkeys in the tree". Mas imagine que nós estejamos criando a string de formatação em um arquivo separado, normalmente para internacionalizar e rescrevemos como:

Exemplo 2. Troca de argumentos

```
<?php
$format = "The %s contains %d monkeys";
printf($format,$num,$location);
?>
```

Agora nós temos um problema. A ordem dos argumentos na string de formatação não combina com os argumentos no código. Nós gostariamos de deixar o código como esta e simplesmente indicar na string de formatação quais argumentos pertencem aonde. Podemos escrever a string de formatação assim:

Exemplo 3. Troca de argumento

```
<?php
$format = "The %2\$s contains %1\$d monkeys";
printf($format,$num,$location);
?>
```

Um beneficio adicional disto é ue você pode repetir os especificadores de conversão sem adicionar mais argumentos em seu código. Por exemplo:

Exemplo 4. Troca de argumento

```
<?php
$format = "The %2\$s contains %1\$d monkeys.
           That's a nice %2\$s full of %1\$d monkeys.";
printf($format, $num, $location);
?>
```

Veja também printf(), sscanf(), fscanf(), vsprintf() e number_format().

Examplos

Exemplo 5. sprintf(): inteiros prenchidos com zero

```
<?php
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
?>
```

Exemplo 6. sprintf(): formatando dinheiro

```
<?php
```

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money irá mostrar "123.1";
$formatted = sprintf("%01.2f", $money);
// echo $formatted irá mostrar "123.10"
?>
```

vprintf -- Mostra uma string formatada
void vprintf (string formato, array args)

Mostra uma string formatada de acordo com o formato (o qual é descrito na documentação para a função sprintf()).

Funciona como printf() mas aceita uma matriz de argumentos, ao invés de um número variavel de argumentos.

```
<?php
$fruits = array(1, 'banana',1, 'apples', 3, 'oranges', 2, 'peaches');

vprintf("I have %d %s, %d %s, %d %s and %d %s.", $fruits);
?>
```

3.14 - Trabalhando com Imagens e Gráficos em PHP

Trabalhando com a biblioteca gráfica GD
(Se no Windows remover o ponto-e-vírgula ";" da linha "extension=php_gd.dll" do php.ini)

Gerando Thumbnails com GD
Artigo do BOZO no PHPBrasil:

http://phpbrasil.com/articles/article.php/id/1350

Gerando Imagens Dinamicamente
por Luiz Ribeiro

O PHP oferece uma interface ao módulo GD de Thomas Boutell. Usando tal módulo, você pode criar e editar imagens nos formatos JPEG e PNG. O formato GIF já foi aceito, mas como o algoritmo de compressão do GIF (LZW) contém uma patente de posse da Unisys, os desenvolvedores do módulo foram obrigados à retirar o suporte a esse formato nas versões mais recentes.

Bom, para iniciar vou explicar o procedimento para criar uma imagem usando o módulo GD em PHP. Se você não tem esse módulo, você pode fazer o download dele em http://www.boutell.com/gd/. Normalmente a GD acompanha uma instalação completa do PHP.

Para se criar a imagem, será usada a função ImageCreate(), então serão realizadas as alterações na imagem, então será finalizada a imagem usando ImageJpeg(), ImagePng() ou até ImageGif() se a versão do módulo GD for inferior à 1.4.

Bom, vamos ao que interessa. Primeiramente vamos criar uma pequena imagem com o seguinte texto: PHPBrasil. O código ficará da seguinte forma:

```
<?php
header("Content-type: image/gif"); //Informa ao browser que o arquivo é uma
imagem no formato GIF

$imagem = ImageCreate(150,40); //Cria uma imagem com as dimensões 100x20
```

```
$vermelho = ImageColorAllocate($imagem, 255, 0, 0); //Cria o segundo plano da
imagem e o configura para vermelho
$branco = ImageColorAllocate($imagem, 255, 255, 255); //Cria a cor de primeiro
plano da imagem e configura-a para branco

ImageString($imagem, 3, 3, 3, "PHPBrasil", $branco);
//Imprime na imagem o texto PHPBrasil na cor branca que está na variável $branco

ImageGif($imagem); //Converte a imagem para um GIF e a envia para o browser

ImageDestroy($imagem); //Destrói a memória alocada para a construção da imagem
GIF.
?>
```

Bom, o script está todo comentado e acho que você entendeu. Se alguma dúvida ficar martelando aí, manda um comentário. =D

Bom, neste exemplo usamos a função ImageGif() para converter a imagem, $imagem, e depoisa enviamos ao navegador. Mas poderíamos ter salvo esta imagem em um arquivo, ao invés de mostrar ela no navegador. Veja o exemplo:

```
<?php
$arquivo = "imagem1.gif";

$imagem = ImageCreate(150,40);

$vermelho = ImageColorAllocate($imagem, 255, 0, 0);
$branco = ImageColorAllocate($imagem, 255, 255, 255);

ImageString($imagem, 3, 3, 3, "PHPBrasil", $branco);
ImageGif($imagem, $arquivo);

ImageDestroy($imagem);

echo "A imagem foi salva no arquivo $arquivo.";
?>
```

Como você deve ter notado, apenas retiramos aquele header() (que informava ao browser que o arquivo era uma imagem), afinal este exemplo não irá mostrar a imagem no navegador e sim gravar ela em $arquivo, e também mudamos os parâmetros da função ImageGif() para salvar a imagem no arquivo.

Nesta parte do artigo, irei explicar como desenhar retângulos, polígonos e arcos.

Desenhando retângulos

Vamos ao primeiro exemplo, que irá desenhar um simples retângulo preenchido usando GD (o formato da imagem a seguir é PNG).

```
<?php
header("Content-type: image/png");
$imagem = ImageCreate(100, 20);
$branco = ImageColorAllocate($imagem, 255, 255, 255);
$azul = ImageColorAllocate($imagem, 20, 93, 233);
ImageFilledRectangle($imagem, 5, 10, 60, 14, $azul);
ImagePng($imagem);
ImageDestroy($imagem);
?>
```

Bom, neste exemplo só há uma função nova, a função ImageFilledRectangle() que como seu próprio nome diz é uma função que cria um retângulo com as dimensões e posição informadas, e na cor azul, que foi definida na variável $azul.

Já para criar um retângulo sem preenchimento você simplesmente irá trocar a função ImageFilledRectangle() por ImageRectangle(). O exemplo ficará da seguinte forma:

```
<?php
header("Content-type: image/png");
$imagem = ImageCreate(100, 20);
$branco = ImageColorAllocate($imagem, 255, 255, 255);
$azul = ImageColorAllocate($imagem, 20, 93, 233);
ImageRectangle($imagem, 5, 10, 60, 14, $azul);
ImagePng($imagem);
ImageDestroy($imagem);
?>
```

Como foi dito, este exemplo irá criar uma imagem com um retângulo sem preenchimento, mas sua borda terá a cor $azul.

Desenhando polígonos
Para desenhar polígonos, vamos usar a função ImagePolygon(), que irá criar um polígono sem preenchimento, e a função ImageFilledPolygon() que irá desenhar um polígono com preenchimento.

Em nosso primeiro exemplo, vamos desenhar um polígono com vértices de (12, 10), (15, 20), (50, 17) e (70, 10) com uma borda de azul-claro:

```
<?php
header("Content-type: image/png");
$imagem = ImageCreate(100, 20);
$branco = ImageColorAllocate($imagem, 255, 255, 255);
$azul = ImageColorAllocate($imagem, 20, 93, 233);
$pontos = array(12, 10, 15, 20, 50, 17, 70, 10);
ImagePolygon($imagem, $pontos, 4, $azul);
ImagePng($imagem);
ImageDestroy($imagem);
?>
```

Bom, agora vamos criar um polígono preenchido, você já deve ter pensado que o código será o mesmo, mas ao invés de ImagePolygon() usaremos ImageFilledPolygon(), se você énsou isso, acertou em cheio. Vamos ver como ficaria nossa imagem com um retângulo preenchido:

```
<?php
header("Content-type: image/png");
$imagem = ImageCreate(100, 20);
$branco = ImageColorAllocate($imagem, 255, 255, 255);
$azul = ImageColorAllocate($imagem, 20, 93, 233);
$pontos = array(12, 10, 15, 20, 50, 17, 70, 10);
ImageFilledPolygon($imagem, $pontos, 4, $azul);
ImagePng($imagem);
ImageDestroy($imagem);
?>
```

Desenhando arcos
Bom, agora vamos desenhar alguns arcos em nossas imagens, para isso vamos usar a função ImageArc(). Antes de começarmos, vou passar a sintaxe da função:

int ImageArc(int im, int cx, int cy, int w, int h, int s, int e, int co1);

Esta função desenha um arco em uma imagem, im, com uma posição inicial de X de cx e uma posição inicial Y de cy. O arco é de largura w e altura h, com um ângulo inicial de s e um ângulo final de e, tudo na cor co1.

Agora que já entendemos a função ImageArc() vamos ao nosso primeiro exemplo que irá desenhar uma elipse:

```
<?php
header("Content-type: image/gif");
$imagem = ImageCreate(500, 100);
$branco = ImageColorAllocate($imagem, 255, 255, 255);
ImageColorTransparent($imagem, $branco);
$vermelho = ImageColorAllocate($imagem, 20, 93, 233);
ImageArc($imagem, 40, 50, 50, 40, 0, 360, $vermelho);
ImageGif($imagem);
ImageDestroy($imagem);
?>
```

O código acima funciona, pois para ter uma elipse, você precisa de uma diferença de 360 graus entre a posição inicial e a posição final. Aplicando esse conhecimento, também podemos desenhar um círculo preenchido usando a função ImageFillToBorder(). (Note que isso é um círculo, não uma elipse, porque os parâmetros de largura e altura têm o mesmo valor.)

```
<?php
header("Content-type: image/gif");
$imagem = ImageCreate(500, 100);
$branco = ImageColorAllocate($imagem, 255, 255, 255);
ImageColorTransparent($imagem, $branco);
$vermelho = ImageColorAllocate($imagem, 20, 93, 233);
ImageArc($imagem, 40, 50, 50, 50, 0, 360, $vermelho);
ImageFillToBorder($imagem, 50, 40, $vermelho);
ImageGif($imagem);
ImageDestroy($imagem);
?>
```

Observação: os exemplos acima foram retirados do livro PHP: Guia do Desenvolvedor que está na lista de livros recomendados da PHPBrasil. Nos exemplos só foram alterados os nomes de algumas variáveis para facilitar a compreensão.

Bom, esse é o básico do módulo GD. Você com certeza tem muito a explorar ainda, em breve vou trazer mais alguns artigos sobre o assunto, para os que se interessaram, ou não entenderam alguma função podem ver no manual do PHP todas as funções de imagem: http://br.php.net/manual/pt_BR/ref.image.php

Gerando Gráficos em PHP com a Biblioteca JPGraph

O que é a JpGrapf?

A JpGraph é uma biblioteca orientada a objetos de criação de gráficos, inteiramente escrita em PHP, que tem como base a extensão GD2 ou GD1 que acompanha o PHP.

Pode ser utilizada para criar diversos tipos de gráficos, de maneira fácil e escrevendo um mínimo de código. Quando aliada a bancos de dados torna os gráficos ainda mais interessantes.

Requisitos
Apache (http://httpd.apache.org)

PHP (www.php.net)

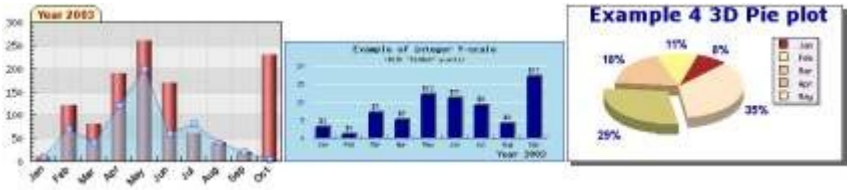
JpGraph (http://www.aditus.nu/jpgraph/)

Parâmetros de Compilação
- Compilar o PHP com suporte a GD e às fontes TTF (Linux):

Sugestão da documentação oficial da JpGrapf:

```
./configure --prefix=/usr/share \
--datadir=/usr/share/php \
--with-apxs=/usr/sbin/apxs \
--libdir=/usr/share \
--includedir=/usr/include \
--bindir=/usr/bin \
--with-config-file-path=/etc \
--enable-mbstring --enable-mbregex \
--with-pdflib=/usr \
--with-mysql  \
--with-ttf-dir=/usr/lib \
--with-freetype-dir=/usr/lib \
--with-gd --enable-gd-imgstrttf --enable-gd-native-ttf \
--with-zlib-dir=/usr/lib \
--with-png-dir=/usr/lib --with-jpeg-dir=/usr/lib --with-xpm-dir=/usr/X11R6 \
--with-tiff-dir=/usr/lib \
--enable-ftp \
--enable-memory-limit --enable-safe-mode \
--bindir=/usr/bin \
--enable-bcmath -enable-calendar \
--enable-ctype --with-ftp \
--enable-magic-quotes \
--enable-inline-optimization \
--with-bz2 \
--with-iconv
```

- No Windows basta descomentar no php.ini o suporte à GD2.
Obs.: No código de um gráfico não pode haver nenhuma saída em HTML ou texto.



Captura do site oficial.

Versões

Para PHP4 é indicada a versão 1.19 da JpGrapf e para a versão 5 do PHP é indicada a versão 2.0 beta.

Download das Fontes TTF (Linux)

http://corefonts.sourceforge.net/

http://www.gnome.org/fonts/

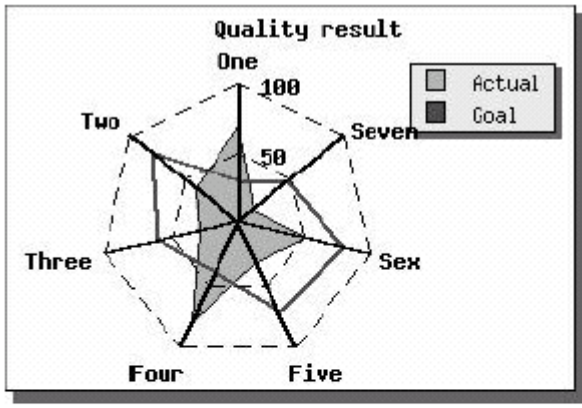
Como saber se o PHP já tem o suporte à JpGrapf?

Executar a função phpinfo(), que deve retornar:

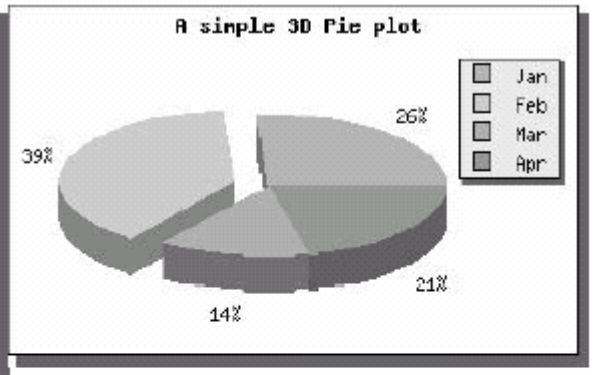
- GD support - enabled
- FreeTypeSupport – enabled
- JPG support – enabled
- PNG support – enabled
- WBMP support – enabled

Exemplos que podem ser encontrados no site official em

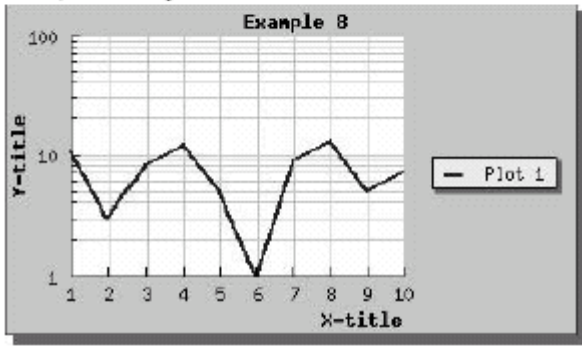
http://www.aditus.nu/jpgraph/pdf/jpgraphddda.pdf



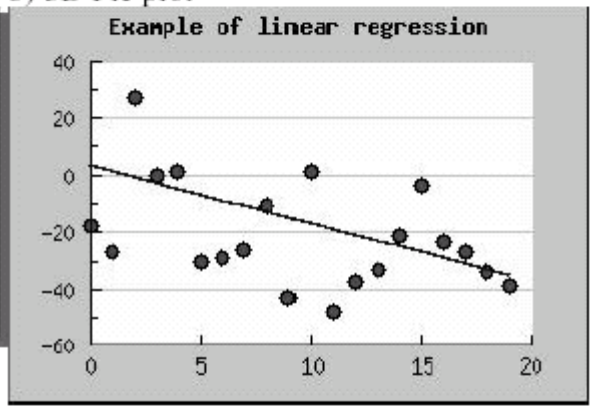
a) Radar plot



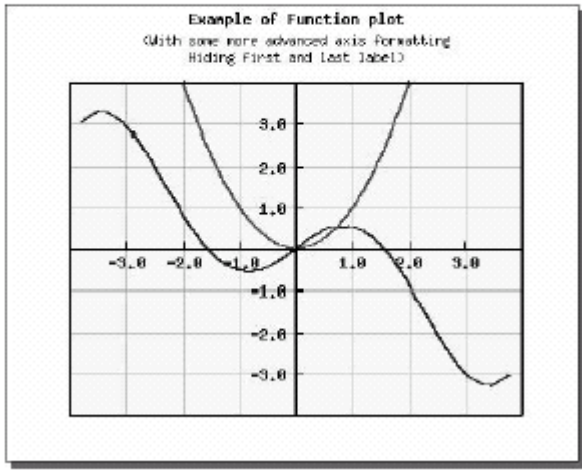
b) 3D Pie plot



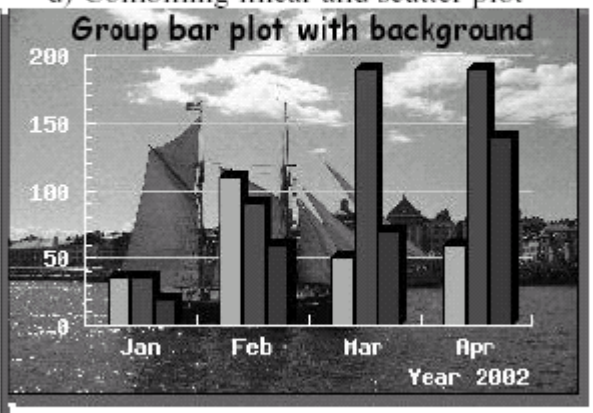
c) Text–logarithmic linear plot



d) Combining linear and scatter plot



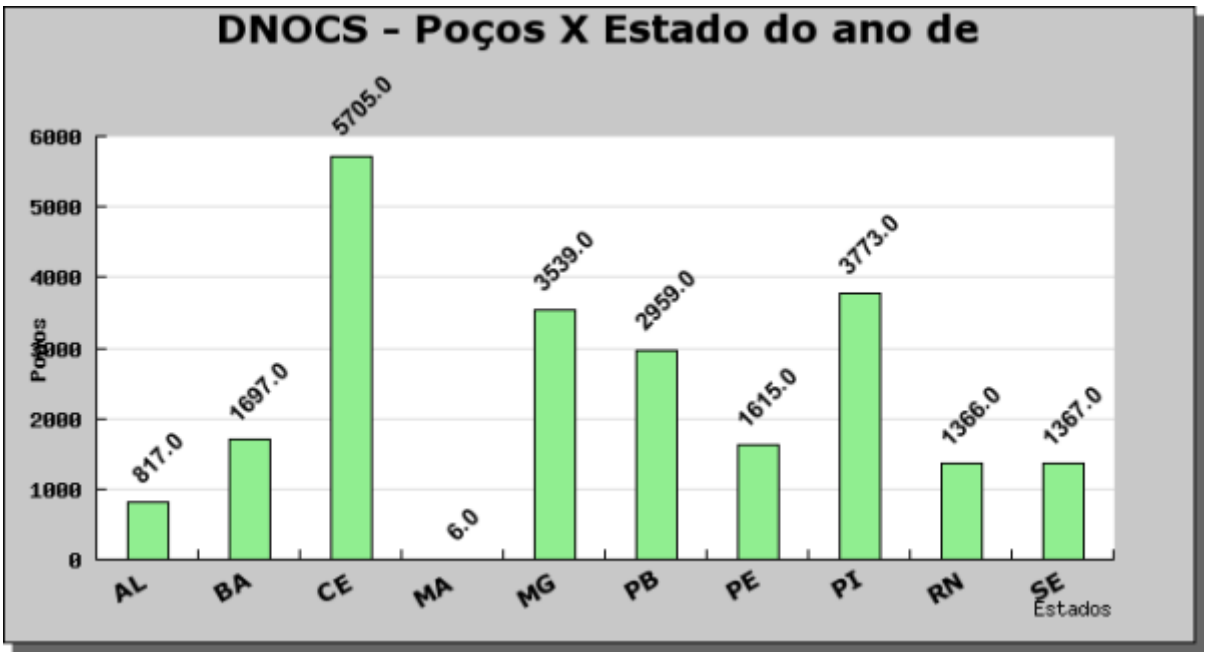
e) Scientific style plot



f) Using a background image

Exemplos de Gráficos

Gráfico dos Poços perfurados pelo DNOCS de 1900 a 1999, por estado:



Este gráfico acessa um banco PostgreSQL. Num pequeno form alguém informa um ano entre 1900 e 1999 e recebe o gráfico correspondente. Caso não informe o ano retornará o gráfico de todos os anos.

Código abaixo:

```
//Pequneo form de consulta
<h2 align=center>Consulta de Poços - DNOCS</h2>
<form name=frmPocos action="barras_pocos.php" method="post">
<center>Que ano que deseja Consultar? (Todos = vazio)<input name="ano" size=10>
<input type=submit value=Consultar></center>
</form>

//Arquivo barras_pocos.php
<?php
// Inclusão da biblioteca
include ("jpgraph.php");
include ("jpgraph_bar.php");
// Conexão ao banco PostgreSQL e consulta
$db = pg_connect("host=10.0.0.100 dbname=banco port=5432 user=user
password=pass") or die(pg_last_error());
$ano=$_POST['ano'];
// Se não for informado o ano da pesquisa no form anterior, exibirá todos os
poços, caso contrário mostra
// somente os poços do ano solicitado
if ($ano == "")
$sql=pg_query($db,"SELECT estado, count(poco) as quant from
recursos_hidricos.pocos group by estado");
else
$sql=pg_query($db,"SELECT estado, count(poco) as quant from
recursos_hidricos.pocos where
recursos_hidricos.pocos.ano = $ano group by estado");
```

```
while($row = pg_fetch_array($sql)) {
$quant[] = $row[1];     //Este array ($quant[]) sera usado em um dos eixos
$estado[] = $row[0];   // Este em outro eixo
}
// Construção da base do gráfico
$graph = new Graph(650,350,"auto");
$graph->SetScale("textint"); //Exibir as escalas
$graph->img->SetMargin(50,50,70,50);  //Margens dos 4 lados
$graph->title->Set('DNOCS - Poços X Estado do ano de '.$ano); // Título do
gráfico
$graph->title->SetFont(FF_VERDANA, FS_BOLD, 16);//Fonte do título
$graph->AdjBackgroundImage(0.4,0.7,-1); //Tipo de background
$graph->xaxis->title->Set('Estados'); //Título do eixo X
$graph->xaxis->SetLabelAngle(30); //Ângulo dos labels do eixo X
$graph->xaxis->SetTickLabels('Estados');
$graph->xaxis->SetFont(FF_VERDANA, FS_BOLD);//Fonte para o título do eixo X
$graph->xaxis->SetTickLabels($estado);  // Recebe o array dos estados do banco
$graph->yaxis->title->Set('Poços');
$graph->yaxis->SetFont(FF_FONT1, FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1, FS_BOLD);
$graph->SetShadow(); //Adicionar sombra ao gráfico

//Adicionar um tipo de gráfico (barras)
$bplot = new BarPlot($quant);  //Recebe o outro array do banco
$bplot->SetFillColor("lightgreen"); // Cor do gráfico
$bplot->value->Show();
$bplot->value->SetFont(FF_ARIAL,FS_BOLD); //Fonte
$bplot->value->SetAngle(45); //Ângulo
$bplot->value->SetColor("black","navy"); //Cores

$graph->Add($bplot); //Adicionar o gráfico à base
$graph->Stroke();
?>
```

Ao baixar a JpGraph e descompactar no diretório web, veja a documentação, que exibe inúmeros tipos de gráficos com seus respectivos códigos ao lado, como também o subdiretório samples que tem 337 exemplos de gráficos.

Referência

http://www.phpbrasil.com/articles/print.php/id/164

http://www.zend.com/zend/tut/tutsweat3.php (ótimo tutorial, em inglês)

http://phpbrasil.com/articles/print.php/id/315 (outro muito bom e em português)

http://www.phpfreaks.com/print.php?cmd=tutorial&tut_id=115 (este abordando uso do MySQL)

3.15 - Trabalhando com Números em PHP

Muito Cuidado ao Lidar com Números em Ponto Flutuante
Teste em PHP

```
<?php
echo (int) ((0.1 + 0.7 ) * 10);
?>
```

Agora teste isso:

echo (int) ((0.2 + 0.7) * 10);

Não conclua muito apressadamente que é deficiência do PHP.

Neste momento devemos ter conhecimento de como se comportam os números, especialmente os floats, que são normalizados pelo IEEE.

Teste em Java

```
class teste {
    public static void main(String[] args) {
        System.out.println((int) ((0.1 + 0.7 ) * 10)); //Display the string.
    }
}
```

Em Java também dá o mesmo resultado do PHP, o que leva a crer que a coisa não depende da linguagem mas das normas de como foram construídos os números pelo IEEE.

O Effective Java sugere que se use int, long ou BigDecimal para representar os valores monetários. A classe BigDecimal foi desenvolvida para resolver dois tipos de problemas associados a números de ponto flutuante (floats e doubles): primeiro, resolve o problema da inexatidão da representação de números decimais; segundo, pode ser usado para trabalhar com números com mais de 16 dígitos significativos. Em compensação, utilizar BigDecimal pode tornar o programa menos legível por não haver sobrecarga dos operadores matemáticos para ela, sendo necessário usar métodos da classe. Veja, por exemplo, como você faria o programa da listagem 1 com BigDecimal:

BigDecimal d1 = new BigDecimal("1.95");

BigDecimal d2 = new BigDecimal("1.03");

System.out.println(d1.subtract(d2));

Utilizar os primitivos normalmente é mais rápido e mais prático, mas o problema fica por conta da definição das casas decimais. Você pode controlar diretamente as casas decimais, por exemplo, utilizando como unidade para os valores o centavo ao invés de real. Um int ou um long passariam a representar a quantidade de centavos presentes no valor, e não a quatidade de reais. Por exemplo:

long l1 = 195;

long l2 = 103;

System.out.println(l1 ? l2);

Listagem 6: Programa da listagem 1 com long

As variáveis acima dizem que você tem 195 centavos (e não R$ 1,95) e vai gastar 103 centavos, e não R$ 1,03. No final você ficará com 92 centavos (e não R$ 0,92).

Agora veja as recomendações do manual do PHP
O tamanho de um float depende também da plataforma e é de 64bits no formato IEEE(*). Nunca compare números em ponto flutuante em igualdades, sob pena de cometer erros.

Teste com PostgreSQL

SELECT CAST((0.1 + 0.7)*10 AS INTEGER);

Este sim, retorna o valor esperado.

Em Java:

System.out.println(1.95 - 1.03); // Retorna errado e em PHP retorna OK.

Em Ruby

(1.8+0.1)==(1.9) retorna false

O mesmo ocorre em Phyton.

```
<?php
/*
   extenso.php
   Copyright (C) 2002 Lyma

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

   Lyma (lyma@lymas.com)
   http://lymas.com

   Esta função recebe um valor numérico e retorna uma string contendo o
   valor de entrada por extenso.
   entrada: $valor (use ponto para centavos.)
   Ex.:

   echo extenso("12428.12"); //retorna: doze mil, quatrocentos e vinte e oito reais e doze
centavos

   ou use:
   echo extenso("12428.12", true); //esta linha retorna: Doze Mil, Quatrocentos E Vinte E
Oito Reais E Doze Centavos

   saída..: string com $valor por extenso em reais e pode ser com iniciais em maiúsculas
(true) ou não.
```

```
    ------------------------------------------------------------------------------------------------------------
    Modificado por Claudio Monteoliva - claudio@simplesefacil.com
    também é possível passar o valor para a funcao com a vírgula decimal.
    exemplo: echo extenso("12428,12"); // o retorna é o mesmo que a passagem com
ponto decimal
*/


function extenso($valor=0, $maiusculas=false)
{
    // verifica se tem virgula decimal
    if (strpos($valor,",") > 0)
    {
      // retira o ponto de milhar, se tiver
      $valor = str_replace(".","",$valor);

      // troca a virgula decimal por ponto decimal
      $valor = str_replace(",",".",$valor);
    }

    $singular = array("centavo", "real", "mil", "milhão", "bilhão", "trilhão", "quatrilhão");
    $plural = array("centavos", "reais", "mil", "milhões", "bilhões", "trilhões", "quatrilhões");

    $c = array("", "cem", "duzentos", "trezentos", "quatrocentos", "quinhentos", "seiscentos",
"setecentos", "oitocentos", "novecentos");
    $d = array("", "dez", "vinte", "trinta", "quarenta", "cinquenta", "sessenta", "setenta",
"oitenta", "noventa");
    $d10 = array("dez", "onze", "doze", "treze", "quatorze", "quinze","dezesseis",
"dezesete", "dezoito", "dezenove");
    $u = array("", "um", "dois", "três", "quatro", "cinco", "seis", "sete", "oito", "nove");

    $z=0;

    $valor = number_format($valor, 2, ".", ".");
    $inteiro = explode(".", $valor);
    for($i=0;$i<count($inteiro);$i++)
        for($ii=strlen($inteiro[$i]);$ii<3;$ii++)
            $inteiro[$i] = "0".$inteiro[$i];

    $fim = count($inteiro) - ($inteiro[count($inteiro)-1] > 0 ? 1 : 2);
    for ($i=0;$i<count($inteiro);$i++) {
        $valor = $inteiro[$i];
        $rc = (($valor > 100) && ($valor < 200)) ? "cento" : $c[$valor[0]];
        $rd = ($valor[1] < 2) ? "" : $d[$valor[1]];
        $ru = ($valor > 0) ? (($valor[1] == 1) ? $d10[$valor[2]] : $u[$valor[2]]) : "";

        $r = $rc.(($rc && ($rd || $ru)) ? " e " : "").$rd.(($rd &&
$ru) ? " e " : "").$ru;
        $t = count($inteiro)-1-$i;
```

```
        $r .= $r ? " ".($valor > 1 ? $plural[$t] : $singular[$t]) : "";
        if ($valor == "000")$z++; elseif ($z > 0) $z--;
        if (($t==1) && ($z>0) && ($inteiro[0] > 0)) $r .= (($z>1) ? " de " : "").$plural[$t];
        if ($r) $rt = $rt . ((($i > 0) && ($i <= $fim) &&
($inteiro[0] > 0) && ($z < 1)) ? ( ($i < $fim) ? ", " : " e ") : " ") . $r;
    }

        if(!$maiusculas){
                return($rt ? $rt : "zero");
        } else {
                return (ucwords($rt) ? ucwords($rt) : "Zero");
        }

}

//echo extenso("12428,12");

//Completar com zeros à esquerda de um numero até que fique com o numero de
caracteres $X,
// retornando como Caráter

function ZeroEsquerda($kNumero, $X)
{
 $NumStrZero = trim((string)$kNumero);
 $QuantosZeros = $X - strlen($NumStrZero);
 for ( $i = 1; $i <= $QuantosZeros; $i++ )
  {
    $NumStrZero = '0'.$NumStrZero;
  }
 return $NumStrZero;
}

?>
```

3.16 - Trabalhando com Permissões de Arquivos e Diretórios

chmod - altera permissões de arquivos e diretórios

```
<?php
chmod ("/arquivo/diretorio", 755);  // decimal; provavelmente incorreto
chmod ("/arquivo/diretorio", "u+rwx,go+rx"); // string; incorreto
chmod ("/arquivo/diretorio", 0755);  // octal; representaÃ§Ã£o correta do modo
?>

function permissoes($arquivo,$perms,$acao){
        print "<form name=frm method=post action=acoes.php>";
        print "<input name=pm value=$perms>";
        print "<input type=hidden name=perms value=$perms>";
        print "<input type=hidden name=ar value=$arquivo>";
        print "<input type=hidden name=acao value=$acao>";
        print "<input name=ar value=$arquivo readonly style='background-
color:#FAEBD7'>";
        print "<input type=submit name=prm value=Alterar>";
        print "</form>";
```

```
        if (isset($_POST['prm'])){
                $ar=$_POST['ar'];
                $perms=octdec($_POST['pm']);
                $ch = chmod($ar, $perms);
                if(!$ch) {
                        die ("Erro ao alterar as permissÃÂµes!");
                }else{
                        print "<script>location='index.php'</script>";
                }
        }
}

<?php
// Escrita e leitura para o proprietario, nada ninguem mais
chmod ("/somedir/somefile", 0600);

// Escrita e leitura para o proprietario, leitura para todos os outros
chmod ("/somedir/somefile", 0644);

// Tudo para o proprietario, leitura e execucao para os outros
chmod ("/somedir/somefile", 0755);

// Tudo para o proprietario, leitura e execucao para o grupo do prop
chmod ("/somedir/somefile", 0750);
?>

Value    Permission Level
400    Owner Read
200    Owner Write
100    Owner Execute
40    Group Read
20    Group Write
10    Group Execute
4    Global Read
2    Global Write
1    Global Execute


<?php
function chmodnum($mode) {
   $mode2=$mode;
   $realmode = "";
   $legal =  array("","w","r","x","-");
   $attarray = preg_split("//",$mode);
   for($i=0;$i<count($attarray);$i++){
       if($key = array_search($attarray[$i],$legal)){
           $realmode .= $legal[$key];
       }
   }
   $mode = str_pad($realmode,9,'-');
   $trans = array('-'=>'0','r'=>'4','w'=>'2','x'=>'1');
   $mode = strtr($mode,$trans);
   $newmode = '';
   $newmode .= $mode[0]+$mode[1]+$mode[2];
   $newmode .= $mode[3]+$mode[4]+$mode[5];
   $newmode .= $mode[6]+$mode[7]+$mode[8];
   return $mode2.' = '.$newmode;
}
```

```
echo chmodnum('drwxr-xr-x');
?>

alguns exemplos:

drwxr-xr-x => 755
drwxr-xr-x => 755
dr-xr-xr-x => 555
drwxr-xr-x => 755
drwxr-xr-x => 755
drwxr-xr-x => 755
drwxr-xr-x => 755
drwxrwxrwt => 776
drwxr-xr-x => 755
drwxr-xr-x => 755
lrwxrwxrwx => 777
```

chown

Esta função não trabalha com arquivos remotos

```
<?php

$file_name= "test";
$path = "/var/www/html/test/" . $file_name ;

$user_name = "root";

chown($path, $user_name);

?>

<?php
function recurse_chown_chgrp($mypath, $uid, $gid)
{
   $d = opendir ($mypath) ;
   while(($file = readdir($d)) !== false) {
       if ($file != "." && $file != "..") {

           $typepath = $mypath . "/" . $file ;

           //print $typepath. " : " . filetype ($typepath). "<BR>" ;
           if (filetype ($typepath) == 'dir') {
               recurse_chown_chgrp ($typepath, $uid, $gid);
           }

           chown($typepath, $uid);
           chgrp($typepath, $gid);

       }
   }

 }

recurse_chown_chgrp ("uploads", "ribafs", "meugrupo") ;
?>

<?php
function recurse_chown_chgrp($path2dir, $uid, $gid){
```

```
   $dir = new dir($path2dir);
   while(($file = $dir->read()) !== false) {
       if(is_dir($dir->path.$file)) {
           recurse_chown_chgrp($dir->path.$file, $uid, $gid);
       } else {
           chown($file, $uid);
           chgrp($file, $gid);
       }
   }
   $dir->close();
}
?>
```

chgrp -- Modifica o grupo do arquivo

filegroup -- Lê o grupo do arquivo

fileperms -- Lê as permissões do arquivo

fileowner -- Lê o dono (owner) do arquivo

is_readable -- Diz se o arquivo/diretório é legivel (readable)

```
<?php
if (is_readable('my_link')) {
  header('Location: /my_link');
}
?>
```

is_writable -- Diz se pode-se escrever para o arquivo (writable)

```
<?php

$file = '/home/vincent/arquivo.sh';

if(is_executable($file)) {
   echo $file.' Ã© executÃ¡vel';
} else {
   echo $file.' nÃ£o Ã© executÃ¡vel';
}

?>
```

umask -- Modificar a umask atual

```
<?php
umask(0670);                     //- set umask
$handle = fopen('file', 'w');  //- 0006
mkdir("/path/dir");            //- 0107
?>

calculate the result:
<?php
$umask = 0670;
umask($umask);
//- if you are creating a new directory, $permission = 0777;
//- if you are creating a new file, $permission = 0666.
printf( "result: %04o", $permission & ( 0777 - $umask) );
?>
```

3.17 - Trabalhando com Strings em PHP

Retorna uma parte de uma string
string substr (string string, int start [, int length])

Exemplo 1. Uso básico de substr()

```
<?php
$rest = substr("abcdef", 1);    // retorna "bcdef"
$rest = substr("abcdef", 1, 3); // retorna "bcd"
$rest = substr("abcdef", 0, 4); // retorna "abcd"
$rest = substr("abcdef", 0, 8); // retorna "abcdef"

// Outra opção é acessar atravéz de chaves
$string = 'abcdef';
echo $string{0};                // retorna a
echo $string{3};                // retorna d
?>
```

Se start for negativo, a string retornada irá começar no caractere start a partir do fim de string.

Exemplo 2. Usando um inicio negativo

```
<?php
$rest = substr("abcdef", -1);    // retorna "f"
$rest = substr("abcdef", -2);    // retorna "ef"
$rest = substr("abcdef", -3, 1); // retorna "d"
?>
```

Exemplo 3. Usando um length negativo

```
<?php
```

```
$rest = substr("abcdef", 0, -1);  // retorna "abcde"
$rest = substr("abcdef", 2, -1);  // retorna "cde"
$rest = substr("abcdef", 4, -4);  // retorna ""
$rest = substr("abcdef", -3, -1); // retorna "de"
?>

<h2>Sobrescrevendo Strings</h2>

str_replace

str_replace --  Substitui todas as ocorrências da string de procura com a string
de substituição

mixed str_replace ( mixed pesquisa, mixed substitui, mixed assunto [, int
&count] )

<pre>
<?php
// Fornece: <body text='black'>
$bodytag = str_replace("%body%", "black", "<body text='%body%'>");

// Fornece: Hll Wrld f PHP
$vowels = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
$onlyconsonants = str_replace($vowels, "", "Hello World of PHP");

// Fornece: você comeria pizza, cerveja e sorvete todos os dias
$frase  = "você comeria frutas, vegetais, e fibra todos os dias.";
$saudavel = array("frutas", "vegetais", "fibra");
$saboroso  = array("pizza", "cerveja", "sorvete");

$novafrase = str_replace($saudavel, $saboroso, $frase);

// Uso do parâmetro count está disponível no PHP 5.0.0
$str = str_replace("ll", "", "good golly miss molly!", $count);
echo $count; // 2
?>
```

substr_replace

substr_replace -- Substitui o texto dentro de uma parte de uma string

string substr_replace (string string, string replacement, int start [, int length])

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* Estes dois exemplos substituem tudo de $var com 'bob'. */
echo substr_replace($var, 'bob', 0) . "<br>\n";
echo substr_replace($var, 'bob', 0, strlen($var)) . "<br>\n";

/* Insere 'bob' direto no começo de $var. */
echo substr_replace($var, 'bob', 0, 0) . "<br>\n";

/* Estes dois exemplos substituem 'MNRPQR' em $var com 'bob'. */
echo substr_replace($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace($var, 'bob', -7, -1) . "<br>\n";

/* Deleta 'MNRPQR' de $var. */
echo substr_replace($var, '', 10, -1) . "<br>\n";
?>
```

Encontrar Posição de caractere em String
strpos

strpos -- Encontra a posição da primeira ocorrência de uma string

int strpos (string str, string procurar [, int offset])

Exemplos strpos()

```
<?php
//$str = 'abc';
$str = 'cba';
$procurar  = 'a';
$posicao = strpos($str, $procurar);

// Note o uso de ===.  Simples == não funcionaria como esperado
// por causa da posição de 'a' é 0 (primeiro) caractere.
if ($pos === false) {
   echo "A string '$procurar' não foi encontrada na string '$str'";
} else {
   echo "A string '$procurar' foi encontrada na string '$str'";
   echo " e está na posição $posicao";
}

?>

<?php

//$email = 'ribafs@gmail.com.br';
$email = 'ribafs@gmail.com';
$usuario = substr ($email, 0, strpos ($email, '@'));
// Lembrando: substr ( string string, int start [, int length] )
$dominio = substr ($email, strpos ($email, '@')+1);
echo "Usuário '$usuario' e Domínio '$dominio'"; // o comprimento default é até o
final
?>
```

Contando Ocorrências de Substring em String
substr_count -- Conta o número de ocorrências de uma substring

int substr_count (string str, string conte_me)

substr_count() retorna o número de vezes que a substring conte_me ocorre na string str.

```
<?php
  $str = "Olá mundo do PHP";

  if (substr_count($str, "do") == 0)
   echo "nenhum";

  // same as:

  if (strpos($str, "do") === false)
   echo "nenhum";
?>
```

Exemplo 1. Exemplo substr_count()

```
<?php
print substr_count("This is a test", "is"); // mostra 2
?>
```

Trocando Ponto por Vírgula e vice-versa

Se temos campos tipo moeda, devemos exibir com vírgula e gravar no banco com ponto.

Para isso uma boa saída é usar a dupla de funções implode e explode.

Antes de exibir na tela (em consultas):

$f_custo_produtivo=explode(".",$f_custo_produtivo);

$f_custo_produtivo=implode(",",$f_custo_produtivo);

Antes de gravar no banco (inclusão e atualização):

$f_custo_produtivo=explode(",",$f_custo_produtivo);

$f_custo_produtivo=implode(".",$f_custo_produtivo);

Conversão de Strings

```
$foo = 1 + "10.5";echo $foo."<br>";              // $foo é float (11.5)
$foo = 1 + "-1.3e3";echo $foo."<br>";               // $foo é float (-1299)
$foo = 1 + "bob-1.3e3";echo $foo."<br>";          // $foo é integer (1)
$foo = 1 + "bob3";echo $foo."<br>";                 // $foo é integer (1)
$foo = 1 + "10 Small Pigs";echo $foo."<br>";      // $foo é integer (11)
$foo = 4 + "10.2 Little Piggies";echo $foo."<br>"; // $foo é float (14.2)
$foo = "10.0 pigs " + 1;echo $foo."<br>";           // $foo é float (11)
$foo = "10.0 pigs " + 1.0;echo $foo."<br>";         // $foo é float (11)
```

Trabalhando com os Caracteres de Strings

```
// Pega o primeiro caracter da string
$str = 'Isto é um teste.';
$first = $str{0};
echo $first."<br>";
// Pega o terceiro caracter da string
$third = $str{2};
echo $third."<br>";
// Pega o último caracter da string
$str = 'Isto ainda é um teste.';
$last = $str{strlen($str)-1};
echo $last."<br>";
// Modifica o ultimo caracter da string
$str = 'Olhe o mal';
echo $str{strlen($str)-1} = 'r';
```

Validação de Caracteres

```
ctype_alnum
ctype_alpha
ctype_cntrl
ctype_digit
ctype_graph
ctype_lower
```

```
ctype_print
ctype_punct
ctype_space
ctype_upper
ctype_xdigit
```

ctype_alnum - Checa por caracteres alfanuméricos

```
$strings = array('AbCd1zyZ9', 'foo!#$bar');

foreach ($strings as $testcase) {
   if (ctype_alnum($testcase)) {
       echo "The string $testcase consists of all letters or digits.\n";
   } else {
       echo "The string $testcase does not consist of all letters or digits.\n";
   }
}
```

ctype_alpha - Checa por caracteres alfabéticos

```
$strings = array('KjgWZC', 'arf12');
foreach ($strings as $testcase) {
   if (ctype_alpha($testcase)) {
       echo "The string $testcase consists of all letters.\n";
   } else {
       echo "The string $testcase does not consist of all letters.\n";
   }
}
```

ctype_digit - Checa por caracteres numéricos

```
$strings = array('1820.20', '10002', 'wsl!12');
foreach ($strings as $testcase) {
   if (ctype_digit($testcase)) {
       echo "The string $testcase consists of all digits.\n";
   } else {
       echo "The string $testcase does not consist of all digits.\n";
   }
}
// Alerta: Ao executar veja que somente é válido quando todos são dígitos
// Não é indicado para testar valores decimais, com ponto ou vírgula
```

ctype_lower - Checa por caracteres minúsculos

```
$strings = array('aac123', 'qiutoas', 'QASsdks');
foreach ($strings as $testcase) {
   if (ctype_lower($testcase)) {
       echo "The string $testcase consists of all lowercase letters.\n";
   } else {
       echo "The string $testcase does not consist of all lowercase letters.\n";
   }
}
```

ctype_punct - Checa por Caracteres que não sejam espaço em branco nem alfanuméricos

```
$strings = array('ABasdk!@!$#', '!@ # $', '*&$()');
foreach ($strings as $testcase) {
   if (ctype_punct($testcase)) {
       echo "The string $testcase consists of all punctuation.\n";
   } else {
```

```
        echo "The string $testcase does not consist of all punctuation.\n";
    }
}
```

ctype_space - Checa por espaços em branco

Validação de Tipos

```
 intval
 is_array
 is_bool
 is_callable
 is_double
 is_float
 is_int
 is_integer
 is_long
 is_null
 is_numeric
 is_object
 is_real
 is_resource
 is_scalar
 is_string
 isset
 print_r
 serialize
 settype
 strval
 unserialize
 unset
```

Cases

```
strtoupper($str) - tudo maiúsculo
strtolower($str) - tudo minúsculo
ucfirst($str) - Converte para maiúscula o primeiro caractere de uma STRING
ucwords($STR) - Converte para maiúsculas o primeiro caractere de cada PALAVRA
```

Índices com Str_Pad

str_pad -- Preenche uma string para um certo tamanho com outra string

string str_pad (string input, int pad_length [, string pad_string [, int pad_type]])

Exemplo:

$players =

```
  array("DUNCAN, king of Scotland"=>"Larry",
        "MALCOLM, son of the king"=>"Curly",
        "MACBETH"=>"Moe",
        "MACDUFF"=>"Rafael");
```

echo "

";

```
// Print a heading
echo str_pad("Dramatis Personae", 50, " ", STR_PAD_BOTH) . "\n";

// Print an index line for each entry
foreach($players as $role=>$actor)
  echo str_pad($role, 30, ".")
      . str_pad($actor, 20, ".", STR_PAD_LEFT) . "\n";

echo "

";
```

Resultado:

```
                    Dramatis Personae
DUNCAN, king of Scotland.....................Larry
MALCOLM, son of the king.....................Curly
MACBETH.....................................Moe
MACDUFF.....................................Rafael
```

String para TimeStamp

```
// Absolute dates and times
$var = strtotime("25 December 2002");
$var = strtotime("14/5/1955");
$var = strtotime("Fr1, 7 Sep 2001 10:28:07 -1000");

// The current time: equivalent to time(  )
$var = strtotime("now");

// Relative times
echo strtotime("+1 day");
echo strtotime("-2 weeks");
echo strtotime("+2 hours 2 seconds");

//Care should be taken when using strtotime( ) with user-supplied dates. It's
better to limit the use of strtotime( ) to cases when
//the string to be parsed is under the control of the script, for example,
checking a minimum age using a relative date:
// date of birth: timestamp for 16 August, 1983
$dob = mktime(0, 0, 0, 16, 8, 1982);

// Now check that the individual is over 18
if ((float)$dob < (float)strtotime("-18 years"))
  echo "Legal to drive in the state of Victoria";
```

3.18 - Trabalhando com URLs no PHP

Passando Parâmetros pela URL
Primeiro (âncora)

<a href="arquivo.php?
parametro1=valor1&parametro2=valor2&parametro3=valor3">Link

```
arquivo.php
$par1=$_GET['parametro1'];
$par2=$_GET['parametro2'];
```

```
$par3=$_GET['parametro3'];
```

Segundo (action de form)

```
<form name=frm method=post action="arquivo2.php?
parametro1=valor1&parametro2=valor2">
...
```

arquivo2.php

```
$par1=$_POST['parametro1'];
$par2=$_POST['parametro1'];
```

Terceiro (URL)

http://localhost/teste.php?parametro1=valor1

teste.php

$par1=$_GET['parametro1'];

Quarto (location no javascript)

```
<?php
// Já vindo de outro script, chamado via POST
$a = $_POST['a'];

?>
<script>
if(confirm("Confirma?")){
        location="vai.php?a='<?=$a?>' ";
}else{
        location='volta.php';
}
</script>
```

Reconstruct URL string in PHP

```
 // find out the domain:
 $domain = $_SERVER['HTTP_HOST'];
 // find out the path to the current file:
 $path = $_SERVER['SCRIPT_NAME'];
 // find out the QueryString:
 $queryString = $_SERVER['QUERY_STRING'];
 // put it all together:
 $url = "http://" . $domain . $path . "?" . $queryString;
 echo "The current URL is: " . $url . "
";

 // An alternative way is to use REQUEST_URI instead of both
 // SCRIPT_NAME and QUERY_STRING, if you don't need them seperate:
 $url2 = "http://" . $domain . $_SERVER['REQUEST_URI'];
 echo "The alternative way: " . $url2;
```

Do site - http://snippets.dzone.com/posts/show/4054

3.19 - Trabalhando com Criptografia no PHP

Classe usando a função crypt do PHP para criptografar e descriptografar uma script.
Fonte: http://phpro.org

```
<?php

// make it of break it
error_reporting(E_ALL);

/**
* Class to provide 2 way encryption of data
*
* @author    Kevin Waterson
* @copyright    2009    PHPRO.ORG
*
*/
class proCrypt
{
    /**
    *
    * This is called when we wish to set a variable
    *
    * @access    public
    * @param    string    $name
    * @param    string    $value
    *
    */
    public function __set( $name, $value )
    {
        switch( $name)
        {
            case 'key':
            case 'ivs':
            case 'iv':
            $this->$name = $value;
            break;

            default:
            throw new Exception( "$name cannot be set" );
        }
    }

    /**
    *
    * Gettor - This is called when an non existant variable is cal
led
    *
    * @access    public
```

```
    * @param    string    $name
    *
    */
    public function __get( $name )
    {
        switch( $name )
        {
            case 'key':
            return 'keee';

            case 'ivs':
            return mcrypt_get_iv_size( MCRYPT_RIJNDAEL_128, MCRYPT
_MODE_ECB );

            case 'iv':
            return mcrypt_create_iv( $this->ivs );

            default:
            throw new Exception( "$name cannot be called" );
        }
    }

    /**
    *
    * Encrypt a string
    *
    * @access    public
    * @param    string    $text
    * @return    string    The encrypted string
    *
    */
    public function encrypt( $text )
    {
        // add end of text delimiter
        $data = mcrypt_encrypt( MCRYPT_RIJNDAEL_128, $this->key, $
text, MCRYPT_MODE_ECB, $this->iv );
        return base64_encode( $data );
    }

    /**
    *
    * Decrypt a string
    *
    * @access    public
    * @param    string    $text
    * @return    string    The decrypted string
    *
    */
    public function decrypt( $text )
    {
```

```
        $text = base64_decode( $text );
        return mcrypt_decrypt( MCRYPT_RIJNDAEL_128, $this->key, $t
ext, MCRYPT_MODE_ECB, $this->iv );
    }
} // end of class
```

Exemplo de Uso

```
<?php

// a new proCrypt instance
$crypt = new proCrypt;

// encrypt the string
$encoded = $crypt->encrypt( 'my message');
echo $encoded."\n";

// decrypt the string
echo $crypt->decrypt( $encoded ) . "\n";

?>
```

Encode e Decode

PHP/MySQL - Função ()encode e ()decode

Fala galera! Nesse artigo estarei dando continuação ao assunto criptografia em PHP. Como vocês puderam observar na semana passada, eu expliquei a função password(), que tem por finalidade fazer a criptografia de senhas.

Uma desvantagem do uso dessa função, é que não podemos descriptografar o dado depois. A seguir, vou apresentar um outro método de criptografia que é o encode (codifica o dado) e o decode (decodifica o dado).

Exemplo prático

1. Vamos criar uma tabela (mensagem.php):

```
CREATE TABLE tb_mensagem (
id int(3) NOT NULL auto_increment,
de varchar(80) NOT NULL DEFAULT '' ,
para varchar(80) NOT NULL DEFAULT '' ,
mensagem text NOT NULL DEFAULT '' ,
PRIMARY KEY (id)
);
```

2. "Popular" a nossa base de dados:

```
INSERT INTO tb_mensagem(de,para,mensagem)
```

VALUES('Júlio César Martini','iMasters',
encode('Mensagem sigilosa: Artigo sobre criptografia','teste'))

Resultado:

Id | De | Para | Mensagem
1 | Júlio César Martini | iMasters |$P²Ý5¾î_¯¹“ÁøJ®nzOAHG²³¼8¿ ê6êEòYÉ%O,{˜

Depois de ter efetuado este comando SQL, o resultado que você tem na sua tabela (tb_mensagem) é o apresentado acima.

Como você pode observar, o campo mensagem é o que recebeu a função encode(). Veja o monte de caracteres esquisitos que apareceram no lugar da mensagem original.

Agora vocês vão me perguntar, como eu faço para ele mostrar a mensagem certa? Para isso, vamos fazer uso da função decode().

3. Arquivo que conecta com a nossa base de dados MySQL:

```
<?
/* Conecta com um banco de dados MySQL conforme parâmetros enviados (servidor = localhost)
Banco de Dados: $dbname
Porta: $usuario
Senha: $senha*/

$dbname="nomedobd";
$usuario="";
$password="";

//1º passo - Conecta ao servidor MySQL
if(!($id = mysql_connect("localhost",$usuario,$password))) {
echo "<p align="center"><big><strong>Não foi possível estabelecer
uma conexão com o gerenciador MySQL. Favor Contactar o Administrador.
</strong></big></p>";
exit;
}

//2º passo - Seleciona o Banco de Dados
if(!($con=mysql_select_db($dbname,$id))) {
echo " <p align="center"><big><strong>Não foi possível estabelecer
uma conexão com o gerenciador MySQL. Favor Contactar o Administrador.
</strong></big></p>";
exit;
}
?>
```

4. Vamos criar uma página index.php que vai exibir a mensagem na tela:

```
<?
```

```
include "conecta.php"; //Arquivo que conecta com a nossa base de dados MySQL

$sql = mysql_query("SELECT de,para,decode(mensagem,'teste') FROM tb_mensagem")
or die("ERRO no SQL : ".mysql_error());
$array = mysql_fetch_array($sql);

echo $array['de']; echo "<br>";
echo $array['para']; echo "<br><br>";
echo $array['mensagem'];

?>
```

Na linha do SELECT, fazemos uso da função decode(), que vai decodificar aquela mensagem para que ela possa ser exibida corretamente na tela para o usuário.

Como vocês podem observar nessa linha: decode(mensagem,'teste'), passamos a chave para decodificar a mensagem. Nesse caso é teste.

Lembre-se que na hora de fazer o encode(), definimos teste como sendo a chave. Então, para decodificar, precisamos informar ela novamente.

Convido o leitor a tirar o decode() do SQL e ver o resultado.

A linha vai ficar assim:

```
<?
...
$sql = mysql_query("SELECT de,para,mensagem) FROM tb_mensagem") or die("ERRO
no SQL : ".mysql_error());
...
?>
```

Não deixe de nos enviar críticas ou sugestões para o próximo assunto, afinal a coluna é de vocês.

Autor/fonte: Júlio César Martini
E-mail/Url: n/a
Site: http://www.htmlstaff.org

Encode e Decode usando base64

```
$str = 'Ribamar Ferreira de Sousa';
//echo base64_encode($str);


$str2 = 'UmliYW1hciBGZXJyZWlyYSBkZSBTb3VzYQ== ';
echo base64_decode($str2);
```

Encode-Decode in PHP

```php
<?php

/* Tutorial by AwesomePHP.com -> www.AwesomePHP.com */
/* Function: Encode or Decode anything based on a string */

$secretPass = 'kljhflk73#OO#*U$O(*YO';
$encodeThis = 'Please meet me at 05:44 time.';

/* Regular Encoding */
$encoded = Encode($encodeThis,$secretPass);
/* Another pass to decode */
$decoded = Encode($encoded,$secretPass);

echo 'Encoded String: '.$encoded;
echo '<br />Decoded String: '.$decoded;

/* Important: If passing this value via URL you might want to make it
explorer friendler */
$encoded = bin2hex(Encode($encodeThis,$secretPass));
/* Another pass to decode */
$decoded = Encode(hex2bin($encoded),$secretPass);

echo '<br /><br />Encoded String: '.$encoded;
echo '<br />Decoded String: '.$decoded;

function Encode($data,$pwd)
{
    $pwd_length = strlen($pwd);
    for ($i = 0; $i < 255; $i++) {
        $key[$i] = ord(substr($pwd, ($i % $pwd_length)+1, 1));
        $counter[$i] = $i;
    }
    for ($i = 0; $i < 255; $i++) {
        $x = ($x + $counter[$i] + $key[$i]) % 256;
        $temp_swap = $counter[$i];
        $counter[$i] = $counter[$x];
        $counter[$x] = $temp_swap;
    }
    for ($i = 0; $i < strlen($data); $i++) {
        $a = ($a + 1) % 256;
        $j = ($j + $counter[$a]) % 256;
        $temp = $counter[$a];
        $counter[$a] = $counter[$j];
        $counter[$j] = $temp;
        $k = $counter[(($counter[$a] + $counter[$j]) % 256)];
        $Zcipher = ord(substr($data, $i, 1)) ^ $k;
        $Zcrypt .= chr($Zcipher);
    }
    return $Zcrypt;
}
function hex2bin($hexdata) {
    for ($i=0;$i<strlen($hexdata);$i+=2) {
        $bindata.=chr(hexdec(substr($hexdata,$i,2)));
    }
    return $bindata;
} ?>
```

Site de origem: http://www.awesomephp.com/?Tutorials*3/Encode-Decode-in-PHP.html

```
<?php
Encriptar
$cod = base64_encode("senha");

print $cod;

Decriptar
echo "<br>".base64_decode($cod);

        /**
         * Decodes a text with many algorithms many times
         *
         * @param    string  $text   the text to decode
         * @param    string  $mode         the sequence of decryption methods separated by
commas [,]
         *                               G      : gEncryption method using the class gEncrypter
         *                               base64 : base64 decoding algorithm
         *                               uu     : decryption using the function uudecode()
         *                       [default: G]
         * @param    string  $key    the key to use separated by commas [,]
         *                               one for each "G" you put in the sequence
         *                               [gEncrypter only] [default: ilarvet]
         * @return   string          the text finally decrypted
         */
        function decode($text, $mode = "G", $key = "ilarvet")
        {
            $exmode = explode(",", $mode);
            $exkey = explode(",", $key);

            $kcount = 0;
            $dec = $text;

            for ($i=0; $i<count($exmode); $i++)
            {
                $exmode[$i] = trim($exmode[$i]);

                switch (strtolower($exmode[$i]))
                {
                    case "g":
                        include_once($this->include_path . "gEncrypter.php");
                        $e = new gEncrypter;
                        $dec = $e->gED($dec, $exkey[$kcount]);
                        $kcount ++;
                        break;

                    case "base64":
                        $dec = base64_decode($dec);
```

```
                break;

            case "uu":
                $dec = convert_uudecode($dec);
                break;

            default:
                break;
        }
    }

    return $dec;
}

function encode($text, $mode = "G", $key = "ilarvet")
{
    $exmode = explode(",", $mode);
    $exkey = explode(",", $key);

    $kcount = 0;
    $enc = $text;

    for ($i=0; $i<count($exmode); $i++)
    {
        $exmode[$i] = trim($exmode[$i]);

        switch (strtolower($exmode[$i]))
        {
            case "g":
                include_once($this->include_path . "gEncrypter.php");
                $e = new gEncrypter;
                $enc = $e->gED($enc, $exkey[$kcount]);
                $kcount ++;
                break;

            case "base64":
                $enc = base64_encode($enc);
                break;

            case "uu":
                $enc = convert_uuencode($enc);
                break;

            default:
                break;
        }
    }
}
?>
```

3.20 - Trabalhando com e-mails em PHP

Enviar e-mail simples:
mail("cursos@ribafs.org", "Aqui fica o assunto", "Aqui o corpo do e-mail");

Envio de e-mail com tratamento de erro:
$to = cursos@ribafs.org';
$subject = 'Assunto';
$from = 'elias@ribafs.org';
$message = 'Como vai?';

if(mail($to, $subject, $message, "From: $from"))
  echo "E-mail enviado com sucesso!";
else
  echo "Falha no envio do e-mail – mensagem não enviada"; ?>

Mais um:
$subject = "Assunto";
$content = "Text type email does not need br tags for line breaks;

A simple line break in your code will do the trick, as the Content-type is set to text in the header.";

$headers = 'MIME-Version: 1.0' . "\n";
$headers .= 'Content-type: text; charset=iso-8859-1' . "\n";

$headers .= 'From: info@domain.com';
mail($to,$subject,$content,$headers);

Mais um exemplo de envio de e-mail com PHP:
$headers = "From: Testando Envio de Email <$assunto>";

// pegando data
$date = date("d/m/Y h:i");

// teoricamente iremos enviar para esse email fictício, mas podemos trazer de um BD o email sem algum problema.
$seuemail = "joao@testando.com.br";

// assunto do email, como vai ficar em sua caixa de entrada, cuidado para não colocar nomes inseridas em blacklists de email.
$assunto = "Sistema de envio de Email";

// Corpo do texto, fiz um exemplo básico de cadastro trazendo em variavel.
$mensagem = "
Nome: $variavel1
Endereco: $variavel2
Bairro: $variavel3
Telefone: $variavel4

```
Email: $variavel5

Enviado em: $date";


// eis aqui o envio própriamente dito.... essa linha é onde se dispara o email.
mail($seuemail, $assunto, $mensagem, $headers);
```

Envio de E-mail no formato HTML

```
<?php
//define the receiver of the email
$to = 'youraddress@example.com';
//define the subject of the email
$subject = 'Test HTML email';
//create a boundary string. It must be unique
//so we use the MD5 algorithm to generate a random hash
$random_hash = md5(date('r', time()));
//define the headers we want passed. Note that they are separated with \r\n
$headers = "From: webmaster@example.com\r\nReply-To: webmaster@example.com";
//add boundary string and mime type specification
$headers .= "\r\nContent-Type: multipart/alternative; boundary=\"PHP-alt-".
$random_hash."\"";
//define the body of the message.
ob_start(); //Turn on output buffering
?>
--PHP-alt-<?php echo $random_hash; ?>
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

Hello World!!!
This is simple text email message.

--PHP-alt-<?php echo $random_hash; ?>
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

<h2>Hello World!</h2>
<p>This is something with <b>HTML</b> formatting.</p>

--PHP-alt-<?php echo $random_hash; ?>--
<?php
//copy current buffer contents into $message variable and delete current output buffer
$message = ob_get_clean();
//send the email
$mail_sent = @mail( $to, $subject, $message, $headers );
//if the message is sent successfully print "Mail sent". Otherwise print "Mail failed"
echo $mail_sent ? "Mail sent" : "Mail failed";
?>
```

Envio de e-mail com anexo:

```
<?php
//define the receiver of the email
$to = 'youraddress@example.com';
//define the subject of the email
$subject = 'Test email with attachment';
//create a boundary string. It must be unique
//so we use the MD5 algorithm to generate a random hash
$random_hash = md5(date('r', time()));
//define the headers we want passed. Note that they are separated with \r\n
$headers = "From: webmaster@example.com\r\nReply-To: webmaster@example.com";
//add boundary string and mime type specification
$headers .= "\r\nContent-Type: multipart/mixed; boundary=\"PHP-mixed-".
$random_hash."\"";
//read the atachment file contents into a string,
//encode it with MIME base64,
//and split it into smaller chunks
$attachment = chunk_split(base64_encode(file_get_contents('attachment.zip')));
//define the body of the message.
ob_start(); //Turn on output buffering
?>
--PHP-mixed-<?php echo $random_hash; ?>
Content-Type: multipart/alternative; boundary="PHP-alt-<?php echo $random_hash; ?>"

--PHP-alt-<?php echo $random_hash; ?>
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

Hello World!!!
This is simple text email message.

--PHP-alt-<?php echo $random_hash; ?>
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

<h2>Hello World!</h2>
<p>This is something with <b>HTML</b> formatting.</p>

--PHP-alt-<?php echo $random_hash; ?>--

--PHP-mixed-<?php echo $random_hash; ?>
Content-Type: application/zip; name="attachment.zip"
Content-Transfer-Encoding: base64
Content-Disposition: attachment

<?php echo $attachment; ?>
--PHP-mixed-<?php echo $random_hash; ?>--

<?php
//copy current buffer contents into $message variable and delete current output buffer
```

```
$message = ob_get_clean();
//send the email
$mail_sent = @mail( $to, $subject, $message, $headers );
//if the message is sent successfully print "Mail sent". Otherwise print "Mail failed"
echo $mail_sent ? "Mail sent" : "Mail failed";
?>
```

3.21 - Trabalhando com Data e Hora em PHP

```
<?php
// Funções diversas

// Definir o horário brasileiro:
//date_default_timezone_set('Brazil/East');


// Somar dias a data atual
function add_dias($dias)
{
        return date('d/m/Y',mktime(0,0,0,date('m'),date('d')+$dias,date('Y')));
}

   /*
      Função SomarDias()
      Usada para calcular a data de feriados móveis.
      Esta função adiciona 'n_dias' à data 'data', passado como argumento, a qual deve estar no formato dd/mm/yyyy ou yyyy-mm-dd.
      O argumento 'forma' serve para especificar o formato da data retornada. Ele pode conter os seguintes valores:

      "pt" => Retornará a data no formato DD/MM/YYYY
      "en" => Retornará a data no formato YYYY-MM-DD

      Se 'forma' não for especificada, adotar-se-á a forma brasileira (pt).
   */
   public function SomarDias ($data, $n_dias, $forma = "pt")
   {
      if (!is_int ($n_dias))
      {
         echo "<p>Função <strong>". __FUNCTION__ ."</strong>: o argumento \"n_dias\" deve ser um número inteiro.</p>";
         return false;
      }

      $forma = strtolower ($forma);
      if ($forma != "en" AND $forma != "pt")
         $forma = "pt";

      if (preg_match ("/^[0-9]{2}\/[0-9]{2}\/[0-9]{4}$/", $data))
```

```
            list ($dia, $mês, $ano) = explode ("/", $data);
        elseif (preg_match ("/^[0-9]{4}-[0-9]{2}-[0-9]{2}$/", $data))
            list ($ano, $mês, $dia) = explode ("-", $data);
        else
        {
            echo "<p>Função <strong>". __FUNCTION__ ."</strong>: Formato de data
inválido (". $data .").</p>";
            return false;
        }

        //transforma $n_dias em segundos
        //86400 = 60 * 60 * 24
        $segs_n_dias = $n_dias * 86400;

        // tranforma $data em timestamp
        $segs_data = strtotime ($ano . "-" . $mês . "-" . $dia);
        $segs_nova_data = $segs_data + $segs_n_dias;
        $nova_data = ($forma == "pt") ? date("d/m/Y", $segs_nova_data) : date("Y-m-d",
$segs_nova_data);

        return $nova_data;
    }



    /*
       Função SubtrairDias()
       Usada para calcular a data de feriados móveis.
       Esta função subtrai 'n_dias' da data 'data', passado como argumento, a qual deve
estar no formato dd/mm/yyyy ou yyyy-mm-dd.
       O argumento 'forma' serve para especificar o formato da data retornada. Ele pode
conter os seguintes valores:

       "pt" => Retornará a data no formato DD/MM/YYYY
       "en" => Retornará a data no formato YYYY-MM-DD

       Se 'forma' não for especificada, adotar-se-á a forma brasileira (pt).
    */
    public function SubtrairDias ($data, $n_dias, $forma = "pt")
    {
        if (!is_int ($n_dias))
        {
            echo "<p>Função <strong>". __FUNCTION__ ."</strong>: O argumento \"n_dias\"
deve ser um número inteiro.</p>";
            return false;
        }

        $forma = strtolower ($forma);
        if ($forma != "en" AND $forma != "pt")
            $forma = "pt";
```

```
        if (preg_match ("/^[0-9]{2}\/[0-9]{2}\/[0-9]{4}$/", $data))
            list ($dia, $mês, $ano) = explode ("/", $data);
        elseif (preg_match ("/^[0-9]{4}-[0-9]{2}-[0-9]{2}$/", $data))
            list ($ano, $mês, $dia) = explode ("-", $data);
        else
        {
            echo "<p>Função <strong>". __FUNCTION__ ."</strong>: Formato de data
inválido (". $data .").</p>";
            return false;
        }

        //transforma $n_dias em segundos
        //86400 = 60 * 60 * 24
        $segs_n_dias = $n_dias * 86400;

        // tranforma $data em timestamp
        $segs_data = strtotime ($ano . "-" . $mês . "-" . $dia);
        $segs_nova_data = $segs_data - $segs_n_dias;
        $nova_data = ($forma == "pt") ? date("d/m/Y", $segs_nova_data) : date("Y-m-d",
$segs_nova_data);

        return $nova_data;
    }

// Somar meses
function add_meses($meses)
{
        return date('d/m/Y',mktime(0,0,0,date('m')+$meses,date('d'),date('Y')));
}

// Somar anos
function add_anos($anos)
{
        return date('d/m/Y',mktime(0,0,0,date('m'),date('d'),date('Y')+$anos));
}

function somar_dias_uteis($str_data,$int_qtd_dias_somar = 7) {
    // Caso seja informado uma data do MySQL ou PostgreSQL do tipo DATETIME - aaaa-
mm-dd 00:00:00
    // Transforma para DATE - aaaa-mm-dd

    $str_data = substr($str_data,0,10);
    // Se a data estiver no formato brasileiro: dd/mm/aaaa
    // Converte-a para o padrão americano: aaaa-mm-dd
    if ( preg_match("@/@",$str_data) == 1 ) {
        $str_data = implode("-", array_reverse(explode("/",$str_data)));
    }
    $array_data = explode('-', $str_data);
    $count_days = 0;
```

```
    $int_qtd_dias_uteis = 0;
    while ( $int_qtd_dias_uteis < $int_qtd_dias_somar ) {
        $count_days++;
            if ( ( $dias_da_semana = gmdate('w', strtotime('+'.$count_days.' day', mktime(0,
0, 0, $array_data[1], $array_data[2], $array_data[0]))) ) != '0' && $dias_da_semana != '6' )
{
            $int_qtd_dias_uteis++;
        }
    }
    return gmdate('d/m/Y',strtotime('+'.$count_days.' day',strtotime($str_data)));
}

//Exemplo de uso:
//echo somar_dias_uteis('05/12/2006');
//echo somar_dias_uteis('2006-12-01',15);
// http://leandrovieira.com/archive/somando-dias-uteis-a-uma-data-especifica-com-php


function SomarData($data, $dias, $meses, $ano)
{
  //passe a data no formato dd/mm/yyyy
    $data = explode('/', $data);
    $newData = date('d/m/Y', mktime(0, 0, 0, $data[1] + $meses,
    $data[0] + $dias, $data[2] + $ano) );
    return $newData;
}

//Exemplo de como usar:
//echo SomarData('04/04/2007', 1, 2, 1);

//Este exemplo acima estamos adicionando 1 dia, 2 meses e 1 ano na data informada. O
resultado então seria '05/06/2008'
//http://www.codigofonte.com.br/codigo/php/data-hora/funcao-para-somar-datas-em-php


/*
Funções sub_data() e som_data()
Desenvolvidas por
InFog (Evaldo Junior Bento)
em Junho de 2007
junior_pd_bento@yahoo.com.br
Este script é disponibilizado utilizando
a licença GPL em sua versão mais atual.
Distribua, aprenda, ensine
mas mantenha os créditos do autor
Viva ao Software Livre e à livre informação
*/

/*
Função sub_data()
```

```
Esta função recebe a data no formato brasileiro dd/mm/AAAA
e o número de dias que serão subtraídos dela.
Certifique-se de checar se a data é válida antes de chamar a função
*/

function sub_data($data, $dias) {
	$data_e = explode("/",$data);
	$data2 = date("m/d/Y", mktime(0,0,0,$data_e[1],$data_e[0] - $dias,$data_e[2]));
	$data2_e = explode("/",$data2);
	$data_final = $data2_e[1] . "/". $data2_e[0] . "/" . $data2_e[2];
	return $data_final;
}

/*
Função soma_data()
Esta função recebe a data no formato brasileiro dd/mm/AAAA
e o número de dias que serão adicionados à dela.
Certifique-se de checar se a data é válida antes de chamar a função
*/

function soma_data($data, $dias) {
	$data_e = explode("/",$data);
	$data2 = date("m/d/Y", mktime(0,0,0,$data_e[1],$data_e[0] + $dias,$data_e[2]));
	$data2_e = explode("/",$data2);
	$data_final = $data2_e[1] . "/". $data2_e[0] . "/" . $data2_e[2];
	return $data_final;
}


//Autor/fonte: Evaldo Junior
//E-mail/Url: http://tuxmasters.blogspot.com/2007/06/somando-datas-no-php.html

function hoje(){
	return date("d/m/Y");
}

function agora(){
	return date("H:i:s");
}

function databr_atual(){
	return date("d/m/Y");
}

function hora_atual(){
	return date("H:i:s");
}


//Verifica se Data1 é menor que Data2, além de validá-las
```

```
//Entrada no formato Brasileiro
function ChecaVarData($data1,$data2)
{
 if (DataValida($data1) and DataValida($data1))
 {
  $data1=DataBrasilMySQL($data1);
  $data2=DataBrasilMySQL($data2);
  Return($data1<=$data2);
 }
 else
 {
  Return(false);
 }

}

function data_valida($data){
	$d = explode("/",$data);
	$d2 = checkdate ($d[1],$d[0],$d[2]);
	if (!$d2){
		?><script>alert("A data "+<?=$d[0].$d[1].$d[2] ?>+" não é
válida!")</script><?php
		exit;
	}
}

function idade($dia,$mes,$ano) {
	if($mes >= date(m)) {
		if($dia >=date(d)) {
			$idade = date(Y) - $ano + 1;
		}else{
			$idade = date(Y) - $ano;
		}
	}else{
		$idade = date(Y) - $ano;
	}
	return $idade;
}
//echo idade(03,08,1956);

/*
 *
 *
$arrDia=array ('Domingo','Segunda','Terça','Quarta','Quinta','Sexta','Sábado ');
$arrMes=array (1=>'Janeiro','Fevereiro','Março','Abril','Maio','Junho',
'Julho','Agosto','Setembro','Outubro','Novembro','Dezembro');
?>
<form>
Selecione a data completa<br>
<select name="semana">
```

```
<?php
for($i = 0; $i < 7; $i++){
echo"<option> $arrDia[$i]";
}
echo'</select><select name="dia">';
for ($i=1;$i<=31; $i++){
echo"<option>$i";
}
echo '</select> de <select name="mes">';
for ($i=1;$i<=12;$i++){
echo "<option value=\"$i\"> $arrMes[$i]";
}
echo '</select> de <select name="ano">';        //Faltava ;
$start_year = date('Y') - 10;
$end_year = $start_year + 20;
for($i = $start_year; $i <= $end_year; $i++){
echo "<option> $i";
}

 */

$arrDia=array
('Domingo','Segunda','Terça','Quarta','Quinta','Sexta','Sábado
');
$arrMes=array
(1=>'Janeiro','Fevereiro','Março','Abril','Maio','Junho',
'Julho','Agosto','Setembro','Outubro','Novembro','Dezembro');
?>
<form>
Selecione a data completa<br>
<select name="semana">
<?php
for($i = 0; $i < 7; $i++){
echo"<option> $arrDia[$i]";
}
echo'</select><select name="dia">';
for ($i=1;$i<=31; $i++){
echo"<option>$i";
}
echo '</select> de <select name="mes">';
for ($i=1;$i<=12;$i++){
echo "<option value=\"$i\"> $arrMes[$i]";
}
echo '</select> de <select name="ano">';        //Faltava ;
$start_year = date('Y') - 10;
$end_year = $start_year + 20;
for($i = $start_year; $i <= $end_year; $i++){
echo "<option> $i";
}
```

```
/* Código Original/ Original Code by Woodys
* http://www.zend.com/codex.php?id=176
*
* adaptação/tradução para o português e formato brasileiro das datas
*
* Alguns exclarecimentos
* - O que é TIMESTAMP do Unix?
* - É a contagem, em segundos, desde o dia 1 de janeiro de 1970 00:00:00 GMT
*     ,i.e., os segundos que se passaram até momento desde as ZERO horas do dia 1 de
janeiro de 1970
*     exemplo:
*     timestamp = 1042752929  => passaram-se 1042752929 segundos desde 1/jan/1970
00horas 00min 00 seg
*
*  Tradução e Adaptação by Calvin
*/

?>

<?php

/**
* Calcula a quantidade de dias úteis entre duas datas (sem contar feriados)
* @author Marcos Regis
* @param String $datainicial
* @param String $datafinal=null
*/
function dias_uteis($datainicial,$datafinal=null){
  if (!isset($datainicial)) return false;
  if (!isset($datafinal)) $datafinal=time();

$segundos_datainicial = strtotime(preg_replace("#(\d{2})/(\d{2})/(\d{4})#","$3/$2/$1",
$datainicial));
$segundos_datafinal = strtotime(preg_replace("#(\d{2})/(\d{2})/(\d{4})#","$3/$2/$1",
$datafinal));
$dias = abs(floor(floor(($segundos_datafinal-$segundos_datainicial)/3600)/24 ) );
$uteis=0;
for($i=1;$i<=$dias;$i++){
$diai = $segundos_datainicial+($i*3600*24);
$w = date('w',$diai);
if ($w==0){
//echo date('d/m/Y',$diai)." é Domingo<br />";
}elseif($w==6){
//echo date('d/m/Y',$diai)." é Sábado<br />";
}else{
//echo date('d/m/Y',$diai)." é dia útil<br />";
$uteis++;
}
}
```

```
return $uteis;
}
?>
ex. de uso
<?php
$data='28/02/2007';
echo "Existem ".dias_uteis($data)." dias úteis entre $data e hoje";
?>

<?php

//Impresso através do site http://www.htmlstaff.org

//Função para pegar todos os feriados do ano

//Uma função para pegar todos os feriados do ano (fixos e móveis). Coloquei dois
estaduais, mas não será problema em editá-los conforme a cidade/estado.

//Função:

/**
* @param $ano ano em que se quer calcular os feriados
* @return array com os feriados do ano (fixo e moveis)
*/
function getFeriados($ano){
	$dia = 86400;
	$datas = array();
	$datas['pascoa'] = easter_date($ano);
	$datas['sexta_santa'] = $datas['pascoa'] - (2 * $dia);
	$datas['carnaval'] = $datas['pascoa'] - (47 * $dia);
	$datas['corpus_cristi'] = $datas['pascoa'] + (60 * $dia);
	$feriados = array (
			'01/01', // Confraternização universal
			'02/02', // Navegantes
			date('d/m',$datas['carnaval']),
			date('d/m',$datas['sexta_santa']),
			date('d/m',$datas['pascoa']),
			'21/04', //Tiradentes
			'01/05',
			date('d/m',$datas['corpus_cristi']),
			'20/09', // Revolução Farroupilha m/
			'12/10',
			'02/11',
			'15/11',
			'25/12',
			);
	return $feriados;
}
$feriados = getFeriados('2008');
```

```
for($x=0;$x<12;$x++){
        print $feriados[$x].'<br>';
}

//Autor/fonte: Maiquel Leonel
//E-mail/Url: http://www.vivaolinux.com.br/scripts/verScript.php?codigo=3430
?>
```

Feriados Brasileiros

Feriados nacionais

Os feriados nacionais são definidos pelas seguintes leis: nº 662 (de 1949)[1], nº 6.802 (de 1980)[2], nº 9.093 (de 1995)[3], e nº 10.607 (de 2002)[4]. Os feriados nacionais brasileiros são:

Data	Feriado	Motivação
1º de janeiro	Confraternização Universal	social
21 de abril	Tiradentes	cívica
1º de maio	Dia do Trabalho	social
7 de setembro	Independência do Brasil	cívica
12 de outubro	Nossa Senhora Aparecida	religiosa (católica)
2 de novembro	Finados	religiosa (católica)
15 de novembro	Proclamação da República	cívica
25 de dezembro	Natal	religiosa (cristã)

Feriados móveis

São feriados que dependem da Páscoa. Os judeus celebram a Páscoa segundo o que prescreve o livro do Êxodo, no capítulo 12, no dia 14 do mês de Nissan. É a celebração da libertação da escravidão do Egito para a liberdade da Terra Prometida por Deus a Abraão. O cristianismo celebra a Páscoa cristã, Ressurreição de Cristo, acompanhando de certa forma a data Páscoa judaica. Mas o calendário judeu é baseado na Lua, então a data da Páscoa cristã passou a ser móvel no calendário cristão, assim como as demais datas referentes à Páscoa, tanto na Igreja Católica como nas Igrejas Protestantes e Igrejas Ortodoxas. O primeiro Concilio geral da Igreja, o de Nicéia, no ano 325, determinou que a Páscoa cristã seria celebrada no domingo seguinte à primeira Lua cheia após o equinócio da primavera do hemisfério Norte (21 de março); podendo ocorrer entre 22 de março e 25 de abril.

Outros feriados que dependem da Páscoa:

Carnaval (terça-feira) - quarenta e sete dias antes da Páscoa.

Quaresma - inicia na quarta-feira de cinzas e termina no Domingo de Ramos (uma semana antes da Páscoa).

Sexta-feira Santa - a sexta-feira imediatamente anterior Sábado da Solene Vigília Pascal - o sábado de véspera

Pentecostes - o oitavo domingo após a Páscoa.
Corpus Christi - a quinta-feira imediatamente após o Pentecostes.

Feriados estaduais

A Lei nº 9.093, de 12 de setembro de 1995, incluiu entre os feriados civis, antes apenas os declarados em lei federal, a "data magna do Estado fixada em lei estadual".

Acre

Data	Feriado
23 de janeiro	Dia do evangélico
15 de junho	Aniversário do estado
6 de agosto	Início da Revolução Acreana
5 de setembro	Dia da Amazônia
17 de novembro	Assinatura do Tratado de Petrópolis

Alagoas

Data	Feriado
24 de junho	São João (decreto estadual n.4.087, de 18 de dezembro de 2008)
29 de junho	São Pedro (decreto estadual 4.087, de 18 de dezembro de 2008)
16 de setembro	Emancipação política
20 de novembro	Dia da Consciência Negra

Amapá

Data	Feriado
19 de março	Dia de São José
5 de outubro	Criação do estado
20 de novembro	Dia da Consciência Negra

Amazonas

Data	Feriado
5 de setembro	Elevação do Amazonas à categoria de província
20 de novembro	Dia da Consciência negra
8 de dezembro	Dia de Nossa Senhora da Conceição

Bahia

Data	Feriado
2 de julho	Independência da Bahia
20 de novembro	Dia da Consciência Negra

Ceará

O Ceará não tem data magna. Em 2007, o deputado estadual Lula Morais apresentou o Projeto de Lei nº 53, para definir o dia 19 de março como Data Magna do Estado do Ceará. O relator do projeto na Comissão de Constituição e Justiça da Assembleia Legislativa, deputado João Jaime, devolveu o projeto ao autor, com a sugestão de transformá-lo em projeto de indicação. Em junho de 2007, o projeto de lei foi retirado pelo

autor e arquivado. Desde então, ninguém mais apresentou projeto que fixasse a data magna estadual cearense.

Distrito Federal

Data	Feriado
21 de abril	Fundação de Brasília
30 de novembro	Dia do Evangélico

Espírito Santo

Data	Feriado
23 de maio	Colonização do solo espírito-santense
28 de outubro	Dia do Servidor Público

Goiás

Data	Feriado
28 de outubro	Dia do Servidor Público

Maranhão

Data	Feriado
28 de julho	Adesão do Maranhão à independência do Brasil
8 de dezembro	Dia de Nossa Senhora da Conceição

Mato Grosso

Data	Feriado
20 de novembro	Dia da Consciência Negra

Mato Grosso do Sul

Data	Feriado
11 de outubro	Criação do estado

Minas Gerais

Data	Feriado
21 de abril	Tiradentes

Pará

Data	Feriado
15 de agosto	Adesão do Grão-Pará à independência do Brasil
8 de dezembro	Nossa Senhora da Conceição

Paraíba

Data	Feriado
5 de agosto	Emancipação política do estado

Paraná

Data	Feriado
19 de dezembro	Emancipação política (emancipação do Paraná de São Paulo)

Pernambuco

Data	Feriado	Legislação	Observações
6 de março	Revolução Pernambucana de 1817	Lei nº 13.386[1], de 24 de dezembro	

de 2007 ponto facultativo

Piauí
Data Feriado
13 de março Dia da Batalha do Jenipapo
19 de outubro Dia de Piauí

Rio de Janeiro
Data Feriado
23 de abril Dia de São Jorge
15 de outubro Dia do Comércio
20 de novembro Dia da Consciência Negra

Rio Grande do Norte
Data Feriado
29 de junho Dia de São Pedro
3 de outubro Mártires de Cunhaú e Uruaçu

Rio Grande do Sul
Data Feriado
20 de setembro Revolução Farroupilha

Rondônia
Data Feriado
4 de janeiro Criação do estado
18 de junho Dia do Evangélico

Roraima
Data Feriado
5 de outubro Criação do estado

Santa Catarina
Data Feriado
11 de agosto Criação da capitania, separando-se de São Paulo

São Paulo
Data Feriado
25 de janeiro Dia de São Paulo
9 de julho Revolução Constitucionalista de 1932
20 de novembro Dia da Consciência Negra
12 de outubro Nossa Senhora da Conceição Aparecida

Sergipe
Data Feriado
8 de julho Autonomia política de Sergipe

Tocantins
Data Feriado
5 de outubro Criação do estado

Feriados municipais

Os municípios podem declarar, em lei municipal, até quatro feriados religiosos, de acordo com a tradição local, entre eles a Sexta-Feira da Paixão. A Lei nº 9.335, de 10 de dezembro de 1996[7], acrescentou, ainda, como feriado civil, os dias do início e do término do ano do centenário de fundação do município, desde que fixado em lei municipal.
Fonte: http://pt.wikipedia.org/wiki/Feriados_no_Brasil

Referências

http://php.net
http://pt.wikibooks.org/wiki/Aplicativos_em_PHP

3-PHPOO

Conceitos

Objeto
Representa uma coisa física, tangível, uma idéia ou conceito. Possui um estado (o que ele sabe) e um comportamento (o que ele é capaz de fazer, como ele reage a estímulos externos). São criados quando instanciamos uma classe. É com estes que trabalhamos, os objetos e não com as classes.

Classe
É um "molde" para a criação de objetos, fornecendo o seu comportamento padrão e a definição de todos os seus estados possíveis.
Ex: Classe Correntista

Instância
É uma ocorrência particular, identificada, de um objeto de uma determinada classe, com seu estado particular, independente de outras instâncias da mesma classe.
Ex: o objeto Correntista "Fernando Lozano"

Atributos ou Propriedades
São as variáveis da classe. O atributo de uma classe, pode ter valor diferente em cada objeto instanciado.

Métodos ou Funções
Juntamente com as variáveis formam os membros da classe. São também chamados de operações.

Mensagens
São trocadas entre os objetos, formam a interação entre os objetos.

Encapsulamento
Um objeto contém todas as informações (variáveis) e toda a inteligência (código) de que necessita para realizar suas atribuições.

Ele deve ser tanto quanto possível autocontido, independente de informações ou código que não façam parte dele mesmo.

Ocultamento de Informações
Deve ser possível utilizar um objeto apenas pelo conhecimento da sua estrutura externa (isto é, sua interface). Mudanças na estrutura interna de um objeto (isto é, sua implementação) não devem afetar aos usuários do objeto.

Polimorfismo
A mesma mensagem, quando enviada para objetos de classes diferentes, executa código particular da classe, mesmo que quem enviou a mensagem não tenha conhecimento do tipo específico de
objeto sendo referenciado.

Herança ou Especialização
Uma nova classe pode ser definida em termos de uma classe pai, herdando o seu comportamento. A nova classe especializa a classe pai, defindo apenas onde o seu comportamento deve ser diferente. Temos então classes super e sub, pai e filha.

Agregação e Composição
Objetos podem conter outros objetos como partes constituintes, imitando o mundo real onde objetos são construídos em função de outros objetos. Podemos ou não expor as partes constituintes como
parte da interface de um objeto.

Construtor
Função que é chamada automaticamente sempre que se instância uma classe, através do operador NEW. Tem o mesmo nome da classe e pode conter parâmetros. Até a versão 4 do PHP o construtor tinha o mesmo nome da classe. A partir da versão 5 existe também uma palavra reservada "__construct", ou seja, function __construct.

Herança
Capacidade de criar uma classe tendo outra como base, da qual herda todos os métodos e propriedades e ainda pode adicionar as suas próprias

Instância
Objeto criado a partir de uma classe com o operador “new”

Definições

New
Operador usado para criar novos objetos

->
Operador utilizado para atribuir valores de propriedades ou métodos à variáveis. Semelhante ao operador de atribuição “=”.

$this – variável que representa o objeto atual.

Porque Usar Classes em PHP

Utilizar uma sintaxe mais clara e limpa para operações complexas, que requeiram uma grande quantidade de a r g u m e n to s.

Tornar programas complexos mais facilmente gerenciáveis e manuteníveis pelo agrupamento de variáveis e código inter-relacionados.

Fomentar a reutilização de código, criando componentes genéricos.

Permitir a substituição de componentes de um sistema, em estilo plug-ins.

Definindo Classes em PHP

No PHP 5 há um novo Modelo de Objeto. O tratamento de objetos do PHP foi completamente re-escrito, permitindo uma performance melhor e mais vantagens.

A palavrachave class indica uma declaração de classe, delimitada por chaves.

Dentro da classe podemos definir atributos (variáveis) e métodos (funções) que formam o estado e o comportamento do objeto.

Um método com o mesmo nome da classe é o construtor do objeto, sendo executado sempre que uma instância for criada. A partir da versão 5 do PHP um construtor é definido pela função
function __construct()

A classe deve utilizar a variável $this para referenciar seus próprios métodos e atributos

Utilizando uma Classe

Após definirmos uma Classe, sua definição deve estar disponível no script ou página PHP que utilizará a classe (comandos include ou require, ou no próprio script onde foi definida). O mais usual é em outro script.

Um objeto da classe deve ser instanciado pelo operador new.

O operador **->** permite referenciarmos atributos e métodos do objeto

Definindo uma Classe

```
// Tradicional classe inicial: olaMundo
class olaMundo {

	function olaMundo(){
		return "Olá Mundo!";
	}
};
```

Usando a Classe

```
$ola = new olaMundo();
print 'Classe olaMundo:<br>';
print $ola->olaMundo();
```

Especificadores de Acesso dos métodos e variáveis

Descrição: Específica o acesso de controle na classe evitando erros lógicos, existem 3 acessos possíveis(public, protected, private) caso não coloque nenhum entende-se como "public" (default).

Public = Pode ser acessado por qualquer classe.
Protected = Poder ser acessado somente por quem extende sua classe.
Private = Poder ser acessado somente por sua classe.

Exemplos:

```php
<?php
class EspecificaAcesso
{
    public $varPublic = "Variável pública<br>";
    protected $varProtected = "Variável Protegida<br>";
    private $varPrivate = "Variável Privada<br>";

    public function getPublic()
    {
        return $this->varPublic;
    }

    protected function getProtected()
    {
        return $this->varProtected;
    }

    private function getPrivate()
    {
        return $this->varPrivate;
    }

    public function getMetodoPrivate()
    {
        return EspecificaAcesso::getPrivate();
    }
}

$especificaAcesso = new EspecificaAcesso();
echo $especificaAcesso->getPublic();
echo $especificaAcesso->getMetodoPrivate();

// Descomentando a linha abaixo dará um Fatal Error porq o método tem acesso
Privado(private).
// echo $especificaAcesso->getPrivate();
// Descomentando a linha abaixo dará um Fatal Error porq o método tem acesso
Protegido(protected).
// echo $especificaAcesso->getProtected();

class UsandoProtegido extends EspecificaAcesso
{
    function getProtegido()
    {
        return parent::getProtected(); // Se você chamar um método privado"getPrivate()"
dará um Fatal Error.
    }
}

$usandoProtegido = new UsandoProtegido();
```

```
echo $usandoProtegido->getProtegido();
?>
```

2) Métodos e variáveis estáticas
Descrição: Quando declarar static no método ou variável poderá usá-lo sem precisar instânciar sua classe.
Exemplo:

```
<?php
class Estatica
{
   static $varStatic = "Variável Estática<br>";

   static function getStatic()
   {
      return Estatica::$varStatic;
   }
}

// Ou chamando a variável diretamente "Estatica::$varStatic".
echo Estatica::getStatic();
?>
```

3) Métodos finais
Descrição: Quando declarar final no método seu valor inicial não poderá ser alterado, o engraçado é que se declarar nas variáveis da Fatal Error, que coisa não :).
Exemplo:

```
<?php
class ClasseFinal
{
   final function getFinal()
   {
      echo "Método Final";
   }
}

$classeFinal = new ClasseFinal();
$classeFinal->getFinal();
?>
```

4) Métodos construtor/destrutor
Descrição: Método __construct() é executado quando instância a classe e o __destruct() é executado na destruição do objeto, caso sua classe herda outra automaticamente irá sobrepor os 2 métodos, para usalos precisa chamar diretamente "parent::__construct()".
Exemplo:

```php
<?php
class ContrutorDestrutor
{
   private $varMethod;

   function __construct()
   {
      $this->varMethod = "Construtor()";
      echo "Método {$this->varMethod}<br>";
   }

   function __destruct()
   {
      $this->varMethod = "Destrutor()";
      echo "Método {$this->varMethod}<br>";
   }
}

$contrutorDestrutor = new ContrutorDestrutor();
unset($contrutorDestrutor);

class ContrutorDestrutorFilho extends ContrutorDestrutor
{
   function __construct()
   {
      parent::__construct();
      echo "Método Filho Construtor<br>";
   }

   function __destruct()
   {
      parent::__destruct();
      echo "Método Filho Destrutor<br>";
   }
}

echo "<br>";
$contrutorDestrutorFilho = new ContrutorDestrutorFilho();
?>
```

5) Constantes da Classe
Descrição: Constante é alguma informação da classe que pode ser acessada diretamente, logicamente é o mesmo que static.
Exemplo:

```php
<?php
class Constante
{
```

```
    const constante = "Minha Constante";
}
echo Constante::constante;
?>
```

6) Clonando objetos
Descrição: O método __clone() define o comportamento do objeto clonado.
Exemplo:

```
<?php
class ClasseClonando
{
    public $varClone;

    function __construct()
    {
        $this->varClone = "Php5";
    }

    function __clone()
    {
        $this->varClone = "Php5 Clone";
    }
}

$classeClonando = new ClasseClonando();
$cloneClasseClonando = clone $classeClonando;
echo $classeClonando->varClone . "<br>" . $cloneClasseClonando->varClone;
?>
```

7) Verificando se a instância é da Classe específica
Descrição: Comando instanceof verifica se a instância passada é da Classe específica retornando um boolean(true ou false).
Exemplo:

```
<?php
class TesteInstanceOf
{
}

class ClasseInstanceOf
{
    function __construct($obj)
    {
        if ($obj instanceof TesteInstanceOf) {
            echo "Objeto da classe(TesteInstanceOf)";
        } else {
            echo "Não é um objeto da classe(TesteInstanceOf)";
```

```
        }
    }
}

$testeInstanceOf = new TesteInstanceOf();
$classeInstanceOf = new ClasseInstanceOf($testeInstanceOf);
?>
```

8) Classes Abstratas
Descrição: Declaramos uma Classe abstrata quando um conjunto de métodos será utilizado em diferentes classes.
A classe abstrata não pode ser instânciada diretamente. Para usá-la precisa extendâ-la e criar todos os métodos abstratos com seus acessos.
Qualquer classe com pelo menos um método abstrato, deve ser ela também abstrata.
Outras classes somente podem implementar uma única classe abstrata.
Exemplo:

```
<?php
abstract class Abstrata
{
    public abstract function setNome($nome);
    public abstract function getNome();
}

class ClasseAbstrata extends Abstrata
{
    private $nome;

    public function setNome($newNome)
    {
        $this->nome = $newNome;
    }

    public function getNome()
    {
        return $this->nome;
    }
}

$classeAbstrata = new ClasseAbstrata();
$classeAbstrata->setNome("Php5");
echo $classeAbstrata->getNome();
?>
```

9) Implementando Interfaces
Descrição: Interfaces tem quase o mesmo conceito da abstração, com a diferênça que você pode implementar "n" interfaces.
Especifica quais métodos e variáveis outras classes devem implementar, sem definir

como serão tratados.
Não possuem implementação interna, apenas a declaração de assinatura dos métodos.
Exemplo:

```
<?php
interface IPessoa
{
    public function setNome($nome);
    public function getNome();
}

interface IPessoaFisica
{
    public function setCpf($cpf);
    public function getCpf();
}

interface IPessoaJuridica
{
    public function setCnpj($cnpj);
    public function getCnpj();
}

class ClassePessoa implements IPessoa, IPessoaFisica, IPessoaJuridica
{
    function __construct($nome, $cpf, $cnpj)
    {
        ClassePessoa::setNome($nome);
        ClassePessoa::setCpf($cpf);
        ClassePessoa::setCnpj($cnpj);
    }

    /* Métodos Set */
    public function setNome($nome)
    {
        $this->nome = $nome;
    }

    public function setCpf($cpf)
    {
        $this->cpf = $cpf;
    }

    public function setCnpj($cnpj)
    {
        $this->cnpj = $cnpj;
    }

    /* Métodos Get */
    public function getNome()
```

```
    {
        return $this->nome;
    }

    public function getCpf()
    {
        return $this->cpf;
    }

    public function getCnpj()
    {
        return $this->cnpj;
    }

    function __destruct()
    {
        echo
ClassePessoa::getNome()."<br>".ClassePessoa::getCpf()."<br>".ClassePessoa::getCnpj(
);
    }
}

$classePessoa = new ClassePessoa("Rodrigo", "324.541.588-98", "6545.2101/0001");
?>
```

10) Tratamento de erros(lógicos)
Descrição: Caso aconteça algum erro lógico em RunTime(tempo de execução) da para tratá-lo de outras maneiras. O código a ser executado fica dentro do "try" e acontecendo algum erro vai para o "catch".
Exemplo:

```
<?php
class BusinessException extends Exception
{
    function __construct($msg)
    {
        // Vai para a função construtora do Exception.
        parent::__construct($msg);
    }
}

class Excecao
{
    function __construct($nome)
    {
        try {
            if ($nome == "") {
                throw new BusinessException("Nome não pode ser em branco.");
            } elseif(strlen($nome) < 5) {
```

```
            throw new BusinessException("Nome precisa ter no mínimo 5 letras.");
        } elseif(strtolower($nome) == "corinthians") {
            throw new BusinessException("Corinthians campeão ou não tu moras no meu
coração.");
        } else {
            throw new BusinessException("Paramêtro inválido.");
        }
    } catch (BusinessException $businessException) {
        echo $businessException->getMessage();
    }
  }
}

$excecao = new Excecao("Corinthians");
?>
```

11) Pattern Singleton
Descrição: Garante que terá somente um único objeto de uma classe, um exemplo bom para usá-lo seria na comunicação com o banco de dados.
Exemplo:

```
<?php
class Singleton
{
    private static $instance = null;

    public static function getInstance()
    {
        if (Singleton::$instance == null) {
            Singleton::$instance = new Singleton();
        }
        return Singleton::$instance;
    }
}

$objA = Singleton::getInstance();
$objB = Singleton::getInstance();
if ($objA == $objB) {
    echo "Instância única";
} else {
    echo "Instâncias diferentes";
}
?>
```

12) Pattern Factory
Descrição: É uma classe onde centraliza a criação de objetos sendo assim não precisa instânciá-los diretamente nas classes.
Exemplo:

```php
<?php
abstract class AbstractFactory
{
    private $nome;
    private $rendaMensal;

    function __construct($nome, $rendaMensal)
    {
        $this->setNome($nome);
        $this->setRendaMensal($rendaMensal);
    }

    public function setNome($newNome)
    {
        $this->nome = $newNome;
    }

    public function setRendaMensal($newRendaMensal)
    {
        $this->rendaMensal = $newRendaMensal;
    }

    public function getNome()
    {
        return $this->nome;
    }

    public function getRendaMensal()
    {
        return $this->rendaMensal;
    }

    public abstract function analisarCredito(); // Boolean
    public abstract function getCategoria(); // String
}

class ClientePadrao extends AbstractFactory
{
    function __construct($nome, $rendaMensal)
    {
        parent::__construct($nome, $rendaMensal);
    }

    // Foi declarada no AbstractFactory
    public function analisarCredito()
    {
        return true;
    }
```

```
    // Foi declarada no AbstractFactory
    public function getCategoria()
    {
        return "Cliente Padrão";
    }
}

class ClienteRisco extends AbstractFactory
{
    function __construct($nome, $rendaMensal)
    {
        parent::__construct($nome, $rendaMensal);
    }

    // Foi declarada no AbstractFactory
    public function analisarCredito()
    {
        return false;
    }

    // Foi declarada no AbstractFactory
    public function getCategoria()
    {
        return "Cliente Risco";
    }
}

class ClienteSeguro extends AbstractFactory
{
    function __construct($nome, $rendaMensal)
    {
        parent::__construct($nome, $rendaMensal);
    }

    // Foi declarada no AbstractFactory
    public function analisarCredito()
    {
        return true;
    }

    // Foi declarada no AbstractFactory
    public function getCategoria()
    {
        return "Cliente com alta credibilidade";
    }
}

class SingletonFactory
{
    private static $rendaMedia = 500;
```

```
    private static $rendaBaixa = 240;
    private static $instance = null;

    public static function getCliente($nome, $rendaMensal)
    {
        if ($rendaMensal <= SingletonFactory::$rendaBaixa) {
            SingletonFactory::$instance = new ClienteRisco($nome, $rendaMensal);
        } elseif ($rendaMensal > SingletonFactory::$rendaBaixa and $rendaMensal <=
SingletonFactory::$rendaMedia) {
            SingletonFactory::$instance = new ClientePadrao($nome, $rendaMensal);
        } else {
            SingletonFactory::$instance = new ClienteSeguro($nome, $rendaMensal);
        }

        $clienteAprovacao = "reprovado";
        if (SingletonFactory::$instance->analisarCredito()) {
            $clienteAprovacao = "aprovado";
        }

        echo "Cliente = ".SingletonFactory::$instance->getNome()."<br>";
        echo "Categoria = ".SingletonFactory::$instance->getCategoria()."<br>";
        echo "Crédito = ".$clienteAprovacao;
        echo "<hr>";
    }
}

SingletonFactory::getCliente("Rodrigo", 1977);
SingletonFactory::getCliente("Corinthians", 350);
SingletonFactory::getCliente("John", 220);
?>
```

Caso algo esteja errado favor me avise.

Referências:
http://www.phpbrasil.com – Rodrigo Rodrigues (Novos Features no PHP5)
http://www.lozano.eti.br – Lozano (oo-php)
http://www.tarcisiolopes.com
http://www.zend.com

Introdução ao PHPOO

1) Introdução

A orientação a objetos é atualmente o paradigma de programação mais aceito e considerado moderno, pois apareceu juntamente com os padrões de projeto para resolver problemas da programação procedural.

Atualmente as empresas que trabalham com programação, ao contratar programadores para trabalhar com PHP estão exigindo uma formatura ou estudando na área, o conhecimento de programação orientada a objetos e algumas exigem o conhecimento de um dos bons frameworks.

Conhecer programação orientada a objetos, padrões de projeto e bons frameworks amplia os horizontes em termos de opções e conhecimento, assim como valoriza profissionais de programação.

A programação orientada a objetos tem uma grande preocupação em esconder o que não é importante para o usuário e em realçar o que é importante (encapsulamento).

Atualmente o PHP5 oferece um suporte muito rico em termos de OO.
Com o PHP5 o PHP ganhou uma completa infraestrutura de orientação a objetos.

A OO é um paradigma voltado para modularização do código.

Vale lembrar que orientação a objetos não é uma linguagem, mas sim uma forma de programar, um paradigma, que é implementado por várias das linguagens modernas atuais.

Programação Orientada a Objetos - é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.

Em alguns contextos, prefere-se usar modelagem orientada ao objeto, em vez de programação estruturada.
A análise e projeto orientados a objetos têm como meta identificar o melhor conjunto de objetos para descrever um sistema de software. O funcionamento deste sistema se dá através do relacionamento e troca de mensagens entre estes objetos.

Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema de software. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos.
(Wikipédia).

2) Programação Procedural x Orientada a Objetos

PHP é hoje uma das mais populares linguagens de programação web.
Mais de 60% dos servidores web rodam Apache com PHP.

Uma das razões da popularidade do PHP é sua baixa curva de aprendizado.

Qualquer um pode escrever código PHP sem seguir nenhuma convenção e misturar as camadas de apresentação com a de negócios. Isso acontecendo em um grande projeto por vários anos pode se transformar em inimaginável monstro.

A Programação Orientada a Objetos (POO) é uma boa prática de programação para a criação e gerenciamento de projetos mais facilmente.

A POO estimula qualquer linguagem para uma melhor codificação, para melhor performance e para escrever projetos muito grandes sem se preocupar muito sobre o gerenciamento.
POO elimina os aborrecimentos e dificuldades do gerenciamento de grandes aplicações.

Identificando Classes, Objetos, Propriedades e Métodos

Vale lembrar que uma classe contém atributos e os atributos são propriedades e métodos.
Tomemos como exemplo um carro:
Classe - carro
Propriedades – número de portas, cor, preço, marca, modelo, ano, etc
Métodos – buzinar, dirigir, virar à esquerda, partir, parar, etc

Lembrar também que classes são a matriz e objetos nascem das classes. Um bom exemplo:

Planta de uma casa – classe
Cada uma das casas construídas através da planta – objetos

3) História do PHP

No Início

Em 1995, Rasmus Lerdorf começou a desenvolver o PHP/FI.
Ele não imaginou que sua criação acabaria por levar ao desenvolvimento do PHP que nós conhecemos hoje, que está sendo usado por milhões de pessoas. A primeira versão do "PHP/FI", chamado Personal Homepage Tools/ Form Interpreter, foi uma coleção de scripts Perl em 1995(1). Um dos básica recursos foi algo parecido com Perl, para manipulação de envios de formulários, mas faltou muitas características úteis comun em uma linguagem, como laços.

(1)
http://groups.google.com/group/comp.infosystems.www.authoring.cgi/msg/cc7d43454d64d133?pli=1

Onde ele anunciou o PHP Tools (mensagem histórica transcrita abaixo):

Mensagem sobre o tópico Announce: Personal Home Page Tools (PHP Tools)
O grupo no qual você está postando é um grupo da Usenet. As mensagens postadas neste grupo farão com que o seu e-mail fique visível para qualquer pessoa na internet.

Com o objetivo de verificação, digite os caracteres que você vê na figura abaixo ou os

números que ouvir ao clicar no ícone de acessibilidade. Ouça e digite os números que ouvir
For a complete demonstration of these tools: http://www.io.org/~rasmus
História da Orientação a Objetos no PHP

Quando o PHP foi criado ele não implementava a OO em si.
Após o PHP/FI, quando Zeev, Rasmus e Andy reescreveram o core e lançaram o PHP3, foi introduzido uma muito básica orientação a objetos.

Quando o PHP4 foi lançado as características do OO amadureceram com algumas características introduzidas.

Mas a equipe reescreveu o core engine e introduziu um modelo de objetos completamente novo no lançamento do PHP5.

O PHP é uma linguagem que nos permite escrever código em dois sabores: procedural e orientado a objetos.

Quando escrevemos uma grande aplicação no estilo procedural ele deverá ficar quase impossível de gerenciar após algumas versões.

A maioria das grandes aplicações é escrita usando o estilo orientado a objetos.

4) Benefícios da POO

A POO foi criada para tornar a vida dos programadores mais fácil.
Com POO podemos quebrar problemas em pedaços menores que serão comparativamente mais fáceis de entender.
O principal objetivo da POO é: tudo que você desejar fazer, faça com objetos. Objetos são basicamente pequenas e discretas peças de código que podem incorporar dados e comportamento neles. Em uma plicação todos os objetos são conectados entre si. Eles compartilham dados para resolver problemas.

- Reusabilidade - DRY 'Don't Repeat Yourself' - Não repita seu código, mas ao contrário reutilize-o.

- Refactoring - Na hora de refazer fica mais simples, visto que temos pedados menores e mais simples.

- Extensível - Podemos criar novos objetos a partir dos existentes, que herdam todas as suas características e adicionar novas.

- Manutenibilidade - São fáceis de manter devido a que seus objetos/partes são pequenos e inclusive permitem facilidade para juntar.

- Eficiência - O conceito de orientação a objetos é atualmente associado a uma melhor eficiência devido as suas características.
Vários padrões de projeto são criados para melhorar a eficiência.

5) Classe

É um pedaço de código que contém propriedades e métodos.
É semelhante a um array, que armazena dados chamados de chaves e valores.
Classes são mais que arrays, por que contém métodos. Classes também podem ocultar e exibir informações, o que não é possível para os arrays.
Classes também parecem com estruturas de dados e podem incorporar vários outros objetos em si.

A orientação a objetos no PHP5 tem grandes diferenças da POO no PHP4.

Vejamos um exemplo de classe útil e como usar:

```
<?php
//class.emailer.php
class emailer
{
	private $sender;
	private $recipients;
	private $subject;
	private $body;

	function __construct($sender)
	{
		$this->sender = $sender;
		$this->recipients = array();
	}

	public function addRecipients($recipient)
	{
		array_push($this->recipients, $recipient);
	}

	public function setSubject($subject)
	{
		$this->subject = $subject;
	}

	public function setBody($body)
	{
		$this->body = $body;
	}
	public function sendEmail()
	{
		foreach ($this->recipients as $recipient)
		{
			$result = mail($recipient, $this->subject, $this->body, "From: {$this-
>sender}\r\n");
			if ($result) echo "Mail successfully sent to {$recipient}<br/>";
		}
```

```
        }
}
?>

<?php
// Exemplo de uso:
$emailer = new emailer("ribafs@gmail.com"); // Consrutor
$emailer->addRecipients("tiago@ribafs.org"); // Acessando o método e passando dados
$emailer->setSubject("Apenas um teste do Curso PHP5OO");
$emailer->SetBody("Olá Tiago, como vai meu amigo?");
$emailer->sendEmail();
?>
```

6) Diferenças entre a POO no PHP4 e no PHP5

Orientação a objetos em PHP5 tem grande diferença em relação ao PHP4.

A POO em PHP4 é pobre e inteiramente aberta, sem qualquer restrição de uso das propriedades e métodos. Não podemos usar os modificadores public, protected e private para os métodos e propriedades.

Em PHP podemos encontrar interfaces mas não recursos como abstract e final.

Uma **interface** é uma peça de código que qualquer classe pode implementar e significa que a classe precisa ter todos os métodos declarados na interface. Precisamos implementar todos os métodos que existem na interface. Na interface podemos declarar apenas o nome e o tipo de acesso dos métodos.
Uma interface só pode extender outra interface.

Uma classe **abstrata** (abstract) é onde alguns métodos podem ter algum corpo também. Então qualquer classe pode extender esta classe abstrata e extender também todos os seus métodos na classe abstrata.

Uma classe **final** é uma classe que não podemos extender.
Em PHP5 podemos usar todas essas.

Em PHP4 não existe herança múltipla por interfaces.
No PHP5 múltiplas heranças são suportadas através da implementação de múltiplas interfaces.

Em PHP4 qualquer coisa é static. Isso significa que se você declara qualquer método na classe, você pode chamar este diretamente sem criar uma instância da classe.
Por exemplo:

```
<?php
class Abc
{
        var $ab;

        function abc()
```

```
	{
		$this->ab = 7;
	}

	function mostrealgo()
	{
		echo $this->ab;
	}
}
abc::mostrealgo();
?>
```

Rodando este código no PHP4 ele funciona, mas no PHP5 acusa erro, pois o this não vale numa chamada static.

No PHP4 não existe:

– Constante de classe;
– Propriedade static;
– Destrutor.
– Exceptions

Existe sobrecarga de métodos via métodos mágicos como __get() e __set() no PHP5.

7) Iniciando com a Programação Orientada a Objetos no PHP5

Vamos analisar a classe Emailer:

```
<?php
//class.emailer.php
class emailer
{
	private $sender;
	private $recipients;
	private $subject;
	private $body;

	function __construct($sender)
	{
		$this->sender = $sender;
		$this->recipients = array();
	}

	public function addRecipients($recipient)
	{
		array_push($this->recipients, $recipient);
	}

	public function setSubject($subject)
```

```
	{
		$this->subject = $subject;
	}

	public function setBody($body)
	{
		$this->body = $body;
	}

	public function sendEmail()
	{
		foreach ($this->recipients as $recipient)
		{
			$result = mail($recipient, $this->subject, $this->body, "From: {$this-
>sender}\r\n");
			if ($result) echo "Mail successfully sent to {$recipient}<br/>";
		}
	}
}
?>
```

Temos aí quatro propriedades, todas com visibilidade tipo private. Isso é uma das recomendações em termos de segurança, que as propriedades sejam desse tipo para que ninguém tenha acesso a elas diretamente.
Para ter acesso devemos criar os métodos getter e setter.

Depois temos um método construtor (__construct($sender)), que recebe um parâmetro e retorna o remetente e os destinatários (recipients). Cada vez que essa classe for instanciada deveremos passar o remetente como parâmetro e receberemos o mesmo e os destinatários.

Observe que o método construtor não tem modificador de visibilidade, portanto assume o padrão, que é public.
Depois temos mais quatro métodos, todos publics:

addRecipients($recipient)
setSubject($subject)
setBody($body)
sendEmail()

Observe seus nomes, camelCase e iniciando com minúsculas.
Se todos os métodos são public, significa que ao instanciar essa classe teremos acesso a todos os métodos, mas não às propriedades da classe, que são private.

Antes de usar uma classe precisamos instanciá-la. Após instanciar podemos acessar suas propriedades e métodos usando o operador -> após o nome da instância.

Veja o exemplo de uso abaixo da nossa classe Emailer:

```
<?php
```

```
include_once('class.emailer.php');
// Exemplo de uso:
$emailer = new emailer("ribafs@gmail.com"); // Consrutor
$emailer->addRecipients("tiago@ribafs.org"); // Acessando o método e passando dados
$emailer->setSubject("Apenas um teste do Curso PHP5OO");
$emailer->SetBody("Olá Tiago, como vai meu amigo?");
$emailer->sendEmail();
?>
```

Veja que incluímos a classe que vamos usar na primeira linha.
Primeiro criamos uma instância chamada $emailer da classe emailer() e passamos um e-mail como parâmetro:
$emailer = new emailer("ribafs@gmail.com"); // Construtor

Nós passamos o e-mail como parâmetro porque o construtor da classe recebe um e-mail como parâmetro.

Caso não passemos parâmetro ou passemos um número diferente de parâmetros para a classe ao instanciar acontecerá um erro fatal.

8) Modificadores de Acesso

Os modificadores foram introduzidos no PHP5. São palavras-chaves que ajudam a definir como será o acesso das propriedades e métodos quando alguém instanciar a classe.

private – este modificador não permite ser chamado fora da classe, somente de dentro da classe pode ser chamado qualquer método.
public – este é o modificador default, o que significa que quando um método não tiver modificador ele será public. Public significa que pode ser chamado de fora da classe.
protected – somente pode ser acessado de uma subclasse, ou seja, de uma classe que extendeu a classe que criou a propriedade ou método.

Vamos a um exemplo do modificador protected:

Abrir a classe Emailer e mudar a propriedade $sender para protected:

```
<?php
//class.emailer.php
class Emailer
{
	protected $sender;  // Mudar aqui
	private $recipients;
	private $subject;
	private $body;

	function __construct($sender)
	{
		$this->sender = $sender;
		$this->recipients = array();
	}
```

```
	public function addRecipients($recipient)
	{
		array_push($this->recipients, $recipient);
	}

	public function setSubject($subject)
	{
		$this->subject = $subject;
	}

	public function setBody($body)
	{
		$this->body = $body;
	}

	public function sendEmail()
	{
		foreach ($this->recipients as $recipient)
		{
			$result = mail($recipient, $this->subject, $this->body,"From: {$this-
>sender}\r\n");
			if ($result) echo "Mail successfully sent to {$recipient}<br/>";
		}
	}
}
?>
```

Agora criar o arquivo class.extendedemailer.php com o código:

```
<?php
class ExtendedEmailer extends emailer
{
	function __construct(){}

	pubic function setSender($sender)
	{
		$this->sender = $sender;
	}
}
?>
```

Agora testar assim:

```
<?php
include_once("class.emailer.php");
include_once("class.extendedemailer.php");

$xemailer = new ExtendexEmailer();
$xemailer->setSender("joaobrito@joao.com");
$xemailer->setSubject("Teste extendido");
```

```
$xemailer->setBody("Olá João, espero que esteja tudo bem com você!");
$xemailer->sendEmail();
?>
```

Observe que a propriedade sender não será acessível de fora dessas duas classes.

9) Construtor

O PHP5 aceita dois tipos de construtor, o que usa a palavra reservada __construct() e o compatível com a versão 4 do PHP, que tem o mesmo nome da classe.
Quando acontecer de uma classe tiver dois construtores, um com __construct e outro com o mesmo nome da classe, o __construct será usado e o outro ignorado.

Veja o exemplo a seguir e teste instanciando a classe Fatorial:

```
<?php
//class.factorial.php
class Factorial
{
	private $result = 1;
	private $number;

	function __construct($number)
	{
		$this->number = $number;
		for($i=2; $i<=$number; $i++)
		{
			$this->result*=$i;
		}
		echo "__construct() executed. ";
	}

	function factorial($number)
	{
		$this->number = $number;
		for($i=2; $i<=$number; $i++)
		{
			$this->result*=$i;
		}
		echo "factorial() executed. ";
	}
	public function showResult()
	{
		echo "Factorial of {$this->number} is {$this->result}. ";
	}
}
?>
```

Destrutor

É o método que quando executado destrói o objeto, invocado pela palavra-chave __destruct(). Será invocado automaticamente sempre ao final da execução do script.

Exemplo de uso:

```
function __destruct()
{
        print " O objeto foi destruído.";
}
```

Exemplo com __construct e __destruct

```
<?php

class teste{
        public function __construct(){
                echo 'A classe '.__CLASS__.' foi inicializada!<br>';
        }

        public function __destruct(){
                echo 'A classe '.__CLASS__.' foi destruída!<br>';
        }

}

$c = new teste();
```

10) Constantes de Classe

No PHP5 para criar uma constante de classe usamos a palavra reservada **const.** Atualmente esta deve funcionar como uma variável static, a diferença é que ela é somente leitura.

Veja este exemplo:

```
<?php
class WordCounter
{
        const ASC=1; //you need not use $ sign before Constants
        const DESC=2;
        private $words;

        function __construct($filename)
        {
                $file_content = file_get_contents($filename);
                $this->words = (array_count_values(str_word_count(strtolower
($file_content),1)));
        }

        public function count($order)
```

```
	{
		if ($order==self::ASC)
		asort($this->words);
		else if($order==self::DESC)
		arsort($this->words);
		foreach ($this->words as $key=>$val)
		echo $key ." = ". $val."<br/>";
	}
}
?>
```

Exemplo de uso, com o arquivo words.txt:

```
<?php
include_once("class.wordcounter.php");
$wc = new WordCounter("words.txt");
$wc->count(WordCounter::DESC);
?>
```

Observe que estamos acessando uma propriedade da classe WordCounter (DESC) de fora sem passar pela instância mas diretamente com o operador ::.
Execute e veja que é um bom utilitário.

11) Herança – Extendendo uma Classe

Uma característica forte do POO e que é uma das mais utilizadas, onde podemos criar uma classe inteiramente nova partindo de uma existente. A nova classe pode preservar todas as características da classe pai ou pode sobrescrever uma ou todas. A nova classe (subclasse) também pode adicionar novas funcionalidades.
Nós temos uma classe para envio de e-mails e agora precisamos de uma classe que envie e-mails do tipo HTML. Não precisamos criar uma classe inteiramente nova, podemos partir da existente e adicionar apenas uma funcionalidade que dará suporte ao HTML.

```
<?php
class HtmlEmailer extends emailer
{
	public function sendHTMLEmail()
	{
		foreach ($this->recipients as $recipient)
		{
			$headers = 'MIME-Version: 1.0' . "\r\n";
			$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
			$headers .= 'From: {$this->sender}' . "\r\n";
			$result = mail($recipient, $this->subject, $this->body, $headers);
			if ($result) echo "HTML Mail successfully sent to {$recipient}<br/>";
		}
	}
}
?>
```

A classe HtmlEmailer é uma **subclasse** da classe Emailer
e a classe Emailer é uma **superclasse** da classe HtmlEmailer.

Veja que a classe HtmlEmailer extendeu a classe Emailer e adicionou um método sendHTMLEmail() e pode continuar usando todos os métodos da classe pai.

Alerta: caso a subclasse não tenha construtor então será invocado o construtor da superclasse.
Não podemos extender mais que uma classe de cada vez. Herança múltipla somente é suportada com interfaces.

Exemplo de uso da classe HtmlEmailer:

```
<?php
include_once("class.htmlemailer.php");
$hm = new HtmlEmailer();
// fazer outras cosias ...
$hm->sendEmail();
$hm->sendHTMLEmail();
?>
```

12) Sobrescrevendo Métodos

Quando se estende uma classe, a subclasse pode sobrescrever todos os métodos da superclasse, desde que eles tenham modificadores protected ou public. Para isso basta criar um método com o mesmo nome do que desejamos sobrescrever. Por exemplo, se criarmos um método com o nome sendMail na subclasse HTMLMailer ele irá sobrescrever o método da superclasse. Assim também acontecerá com as propriedades. Se criarmos uma propriedade na subclasse com o mesmo nome de uma na superclasse ela ofuscará a da superclasse.

Prevenindo a Sobrescrição

Se acontecer que ao criar um método não quisermos que ela seja sobrescrito nunca então devemos adicionar o modificador **final** a ele, como no exemplo:

```
public final function nomeMetodo()
```

Assim ao ser estendida esta classe o método não poderá ser sobrescrito.

Prevenindo para não Extender uma Classe

As classes também tem o mesmo recurso para que não sejam estendidas. Para isso também usamos o modificador **final**:

```
final class NomeClasse
```

Esta classe não poderá ser extendida.

13) Polimorfismo

Polimorfismo é um termo grego que significa muitas formas (poli: muitas, morphos: formas). Na programação é o mesmo que dizer que várias classes podem possuir a mesma estrutura e comportamentos diferentes.

Ao utilizar herança não apenas podemos reutilizar métodos da classe pai, como também podemos sobrescrever os métodos da classe pai, fazendo assim com que algumas características sejam modificadas.

Vamos supor que um banco possui dois tipos de cheques: o Cheque Comum e o Cheque Especial. Ambos os cheques possuem a mesma estrutura, o que diferencia um do outro é a forma de cobrança de juros. O Cheque Comum cobra 25% em cima do valor, então a classe será assim:

```
<?php
class ChequeComum
{
        private $valor;

        // ... outros métodos da classe

        public function setValor( $valor )
        {
                $this->valor = $valor;
        }

        public function getValor()
        {
                return $this->valor;
        }

        public function calculaJuros()
        {
                return $this->valor * 1.25; // soma 25% em cima do valor
        }
}
?>
```

O método calculaJuros() irá naturalmente retornar o valor que deverá ser cobrado do Cheque Comum com juros. O problema é que no Cheque Especial o valor dos juros cai para 10%. A classe seria algo como:

```
<?php
class ChequeEspecial extends ChequeComum
{
        public function calculaJuros()
        {
                return $this->valor * 1.10; // soma 10% em cima do valor
        }
}
?>
```

Assim reaproveitamos a estrutura dos objetos mudando apenas o que for preciso. Mas às

vezes é preciso que mesmo sobrescrevendo um método, o código na classe pai seja executado. Para isso basta você chamar o nome do método, mudando onde tem $this para parent.

Checar se classe é instância de outra:
operador **instanceof**

14) Interface

Interface é uma classe vazia que contém somente as declarações dos métodos (corpo em branco). Qualquer classe que implemente uma interface precisa conter todas as declarações dos métodos.
Uma classe usa uma interface passando a palavra reservada "implements", assim como uma classe usa uma classe pai passando a palavra reservada "extends". Lembrando que nas interfaces podemos apenas declarar métodos mas não podemos escrever o corpo dos métodos, que obrigatoriamente precisam permanecer vazios.

Usada para criar regras para a criação de novas classes. Bom para definir e amarrar certas características.

Vamos tentar mostrar a necessidade das interfaces: supondo que trabalhamos numa empresa coordenando uma equipe de três programadores e queremos criar uma classe de driver para os bancos de dados, onde usamos três SGBDs, MySQL, PostgreSQL e SQLite e queremos deixar a cargo de cada programador um SGBD, cada um criando uma classe para o seu.
Nós queremos que os programadores sempre trabalhem, obrigatoriamente, com dois métodos, connect e execute. Aí é onde entra nossa interface, que conterá a assinatura dos dois métodos, connect e execute, mas os corpos dependerão de cada SGBD, no caso a cargo de um dos programadores.

Veja nossa interface:

```
<?php
//interface.dbdriver.php
interface DBDriver
{
	public function connect();
	public function execute($sql);
}
?>
```

Cada programador criará uma classe para seu SGBD que deve implementar essa interface e agora eles usarão os dois métodos da interface e adicionarão corpo aos mesmos.

Vejamos um exemplo do primeiro programador criando uma classe que implementará a interface DBDriver:

```
<?php
//class.postgresqldriver.php
class PostgreSQLDriver implements DBDriver
{

}
?>
```

Obrigatoriamente devemos definir os dois métodos da interface e atente para o parâmetro do método execute.

15) Classes Abstratas

Semelhante às interfaces, mas agora os métodos podem conter corpo e não se implementa classes abstratas, mas ao contrário se estende.

Praticamente uma classe abstrata existe para ser estendida.

Abstrair (simplificar, considerar isoladamente). Um sistema OO deve ser um sistema separado em módulos, focando nas peças mais importantes e ignorando as menos importantes (na primeira etapa) para a construção de sistemas robustos e reusáveis.

Exemplo simples:

```
<?php
//abstract.reportgenerator.php
abstract class ReportGenerator
{
	public function generateReport($resultArray)
	{
		// Código do gerator de relatórios
	}
}
```

Porque colocamos um método abstrato nesta classe? Porque a geração de relatórios sempre envolve bancos de dados.

```
<?php
include_once("interface.dbdriver.php");
include_once("abstract.reportgenerator.php");

class MySQLDriver extends ReportGenerator implements DBDriver
{
	public function connect()
	{
		// conectar ao SGBD
	}

	public function execute($query)
	{
		// Executar consulta e mostrar resultado
```

```
	}

	// Não precisamos declarar ou escrever novamente o método reportGenerator aqui, pois foi
	// extendido da classe abstrata
}
?>
```

Observe que podemos implementar uma interface e ao mesmo tempo extender uma classe, como no exemplo acima.

Alerta
Não podemos declarar uma classe abstrata como final, pois a final não pode ser extendida e a abstrata foi criada para ser extendida.

Quando um método foi declarado como abstrato isso significa que a subclasse precisa sobrescrever o método. Um método abstrato pode não conter nenhum corpo onde é definido.

Declaração de um método abstrato:

```
abstract public function connectDB();
```

16) Propriedades e Métodos Static

Para acessar qualquer método ou propriedade de uma classe temos que criar uma instância, ou seja usar: $objeto = new Classe(); De outra forma não podemos acessar. Mas existe uma exceção para métodos e propriedades static. Estes podem ser acessados diretamente sem a criação de instância.
Um membro static é como um membro global.

Onde utilizamos um método static?
A criação de novos objetos com instância é algo que pode consumir recursos do computador e um método estático evita isso.

Exemplo:

```
<?php
// class.dbmanager.php

class DBManager
{
	public static function getMySQLDriver()
	{
		// Instanciar o novo objeto do Driver do MySQL e retornar
	}

	public static function getPostgreSQLDriver()
	{
```

```
                // Instanciar o novo objeto do Driver do PostgreSQL e retornar
        }

        public static function getSQLiteDriver()
        {
                // Instanciar o novo objeto do Driver do SQLite e retornar
        }
}
?>
```

Podemos acessar qualquer método static usando o operador :: ao invés do ->.
Veja:

```
<?php
// test.dbmanager.php
include_once("class.dbmanager.php");
$dbdriver = DBManager::getMySQL();
// agora processamos oeprações do banco com o objeto $dbdriver
?>
```

Veja que usamos o operador :: e não precisamos criar nenhuma instância.

Métodos static geralmente executam uma tarefa e finalizam a mesma.

Alerta: não podemos usar ***$this*** com métodos static. Como a classe não é instanciada então $this não existe. Ao contrário podemos usar a palavra reservada ***self***.

Um exemplo de como a propriedade static funciona:

```
<?php
//class.statictester.php
class StaticTester
{
        private static $id=0;

        function __construct()
        {
                self::$id +=1;
        }

        public static function checkIdFromStaticMehod()
        {
                echo "Current Id From Static Method is ".self::$id."\n";
        }

        public function checkIdFromNonStaticMethod()
        {
                echo "Current Id From Non Static Method is ".self::$id."\n";
        }
}
```

```
$st1 = new StaticTester();
StaticTester::checkIdFromStaticMehod();
$st2 = new StaticTester();
$st1->checkIdFromNonStaticMethod(); //returns the val of $id as 2
$st1->checkIdFromStaticMehod();
$st2->checkIdFromNonStaticMethod();
$st3 = new StaticTester();
StaticTester::checkIdFromStaticMehod();
?>
```

Sempre que criamos uma nova instância ela afeta todas as instância pois a propriedade é declarada como static.

Membros static tornam a orientação a objetos no PHP como a antiga programação procedural. Use métodos static com cuidado.

17) Métodos Acessores

Métodos acessores são simplesmente métodos que são devotados somente a receber e setar o valor de qualquer das propriedades de classe. É uma boa prática acessar o valor das propriedades indiretamente através dos métodos acessores.
Existem dois tipos de métodos acessores: os getters (retornam valor de uma propriedade) e os setters (setam o vlaor de uma propriedade).

Exemplo:

```
<?php
//class.student.php
class Student
{
	private $properties = array();

	function __get($property)
	{
		return $this->properties[$property];
	}

	function __set($property, $value)
	{
		$this->properties[$property]="AutoSet {$property} as: ".$value;
	}
}
?>
```

Convenções:

setter:
setNome()

getNome()

Estes métodos podem ajudar a filtrar a entrada de dados antes de configurar no trabalho.

Existem métodos mágicos que fazem esse trabalho de forma automática e reduzem o trabalho em 90%.

18) Métodos Mágicos para as Propriedades da Classe

O processo é chamado de sobrecarga de propriedade.

Os dois métodos mágicos são o __get() e o __set().

Exemplo:

```
<?php
//class.student.php
class Student
{
	private $properties = array();

	function __get($property)
	{
		return $this->properties($property);
	}

	function __set($property, $value)
	{
		return $this->properties($property)="AutoSet {$property} as: ".$value;
	}
}
?>
```

Agora testando:

```
<?php
$st = new Estudante();
$st->name = "Tiago";
$st->email = "tiago@ribafs.org";
echo $st->name."<br>";
echo $st->email;
?>
```

Quando executamos o código anterior o PHP percebe que não existem propriedades com nome name nem email na classe. Desde que as propriedades não existam então o método __set() será chamado atribuindo valor para a propriedade.

19) Método Mágico para Sobrecarregar Método de Classe

O método mágico __cal() ajuda a sobrecarregar qualquer método chamado no contexto do PHP5.
Permite prover ações ou retornar valores quando métodos indefinidos são chamados em um objeto.
Têm dois argumentos, o nome do método e um array de argumentos passado para o método indefinido.

Exemplo:

```
<?php
class Overloader
{
        function __call($method, $arguments)
        {
                echo "You called a method named {$method} with the following arguments
<br/>";
                print_r($arguments);
                echo "<br/>";
        }
}

$ol = new Overloader();
$ol->access(2,3,4);
$ol->notAnyMethod("boo");
?>
```

20) Funções de Informações sobre Classes

get_class_methods

Retorna um vetor com os nomes dos métodos de uma determinada classe.
array get_class_methods (string nome_classe)
Exemplo:

```
<?php
class Funcionario
{
   function SetSalario()
   {
   }
   function GetSalario()
   {
   }
   function SetNome()
   {
   }
   function GetNome()
   {
   }
}
```

```
  print_r(get_class_methods('Funcionario'));
?>
```

get_class_vars

Retorna um vetor com os nomes das propriedades e conteúdos de uma determinada classe. Note que são valores estáticos definidos na criação da classe.

array get_class_vars (string nome_classe)

```
<?php
class Funcionario
{
    var $Codigo;
    var $Nome;
    var $Salario = 586;
    var $Departamento = 'Contabilidade';
    function SetSalario()
    {
    }
    function GetSalario()
    {
    }
}
print_r(get_class_vars('Funcionario'));
?>
```

get_object_vars

Retorna um vetor com os nomes e conteúdos das propriedades de um objeto. São valores dinâmicos que se alteram de acordo com o ciclo de vida do objeto.

array get_object_vars (object nome_objeto)

Exemplo:

```
<?php
class Funcionario
{
    var $Codigo;
    var $Nome;
    var $Salario = 586;
    var $Departamento = 'Contabilidade';
    function SetSalario()
    {
        }
   function GetSalario()
   {
   }
}
$jose = new Funcionario;
$jose->Codigo = 44;
$jose->Nome = 'José da Silva';
$jose->Salario += 100;
```

```
$jose->Departamento = 'Financeiro';
print_r(get_object_vars($jose));
?>
```

get_class

Retorna o nome da classe a qual um objeto pertence.

string get_class (object nome_objeto)

Exemplo:

```
<?php
class Funcionario
{
   var $Codigo;
   var $Nome;
   function SetSalario()
   {
   }
   function GetSalario()
   {
   }
}
$jose = new Funcionario;
echo get_class($jose);
?>
```

get_parent_class

Retorna o nome da classe ancestral (classe-pai). Se o parâmetro for um objeto, retorna o nome da classe ancestral da classe à qual o objeto pertence. Se o parâmetro for uma string, retorna o nome da classe ancestral da classe passada como parâmetro.

string get_parent_class (mixed objeto)

Parâmetros	Descrição
objeto	Objeto ou nome de uma classe.

Exemplo:

```
<?php
class Funcionario
{
   var $Codigo;
   var $Nome;
}
class Estagiario extends Funcionario
{
   var $Salario;
   var $Bolsa;
}
```

```
$jose = new Estagiario;
echo get_parent_class($jose);
echo "\n";      // quebra de linha
echo get_parent_class('estagiario');
?>
```

is_subclass_of

Indica se um determinado objeto ou classe é derivado de uma determinada classe.

boolean is_subclass_of (mixed objeto, string classe)

Parâmetros	Descrição
objeto	Objeto ou nome de uma classe.
classe	Nome de uma classe ancestral.

Exemplo:

```
<?php
class Funcionario
{
   var $Codigo;
   var $Nome;
}
class Estagiario extends Funcionario
{
   var $Salario;
   var $Bolsa;
}
$jose = new Estagiario;
if (is_subclass_of($jose, 'Funcionario'))
{
   echo "Classe do objeto Jose é derivada de Funcionario";
}
echo "\n";      // quebra de linha
if (is_subclass_of('Estagiario', 'Funcionario'))
{
   echo "Classe Estagiario é derivada de Funcionario";
}
?>
```

method_exists

Verifica se um determinado objeto possui o método descrito. Podemos verificar a existência de um método antes de executar por engano um método inexistente.

boolean method_exists (object objeto, string método)

Parâmetros	Descrição
objeto	Objeto qualquer.
método	Nome de um método do objeto.

Exemplo:

```
<?php
class Funcionario
{
   var $Codigo;
   var $Nome;
   function GetSalario()
   {
   }
   function SetSalario()
   {
   }
}
$jose = new Funcionario;
if (method_exists($jose, 'SetNome'))
{
   echo 'Objeto Jose possui método SetNome()';
}
if (method_exists($jose, 'SetSalario'))
{
   echo 'Objeto Jose possui método SetSalario()';
}
?>
```

call_user_func

Executa uma função ou um método de uma classe passado como parâmetro. Para executar uma função, basta passar seu nome como uma string, e, para executar um método de um objeto, basta passar o parâmetro como um array contendo na posição 0 o objeto e na posição 1 o método a ser executado. Para executar métodos estáticos, basta passar o nome da classe em vez do objeto na posição 0 do array.
mixed call_user_func (callback função [, mixed parâmetro [, mixed ...]])

Parâmetros	Descrição
função	Função a ser executada.
parâmetro	Parâmetro(s) da função.

Exemplo:

```
<?php
// exemplo chamada simples
function minhafuncao()
{
   echo "minha função! \n";
}
call_user_func('minhafuncao');
// declaração de classe
class MinhaClasse
{
   function MeuMetodo()
   {
```

```
        echo "Meu método! \n";
    }
}
// chamada de método estático
call_user_func(array('MinhaClasse', 'MeuMetodo'));
// chamada de método
$obj = new MinhaClasse();
call_user_func(array($obj, 'MeuMetodo'));
?>
```

class_exists
Checa se uma classe existe.

```
<?php
include_once("class.emailer.php");
echo class_exists("Emailer");
// Retorna true se existir e false se o contrário
?>
```

A melhor maneira de usar a função class_exists() é sempre antes de instanciar uma classe, checando primeiro se ela existe. Então se existir instanciamos, caso contrário disparamos um erro. Assim fica mais estável.

```
<?php
include_once("class.emailer.php");
if( class_exists("Emailer"))
{
        $emailer = new Emailer("joaobrito@joao.com");
}else{
        die ("Classe Emailer não encontrada!");
}
?>
```

get_declared_classes

Procurar classes carregadas atualmente. Retorna um array com as classes carregadas.

```
<?php
include_once("class.emailer.php");
print_r(get_declared_classes());
?>
```

is_a

Checa o tipo de uma classe.

```
<?php
class ClassePai
{
```

```
}

class Filha extends ClassPai
{

}

$cc = new Filha();
if (is_a($cc, "Filha")) echo "Esta classe é um objeto do tipo Filha";
echo "<br>";
if (is_a($cc, "ClassePai")) echo "Esta classe é um objeto do tipo ClassePai";
?>
```

21) Tratamento de Exceções

Um recurso muito importante adicionado no PHP5. Com isso ganhamos um tratamento de erros mais eficiente e com mais recursos.

```
<?php
//class.db.php
error_reporting(E_ALL - E_WARNING);
class db
{
	function connect()
	{
		if (!pg_connect("host=localhost password=pass user=username
dbname=db"))
			throw new Exception("Cannot connect to the database");
	}
}

	$db = new db();
	try {
		$db->connect();
	}
	catch (Exception $e)
	{
		print_r($e);
	}
?>
```

Usando um bloco try ... catch podemos capturar todos os error. Podemos usar blocos dentro de blocos. Veja um exemplo:

```
<?php
include_once("PGSQLConnectionException.class.php");
include_once("PGSQLQueryException.class.php");
error_reporting(0);
class DAL
{
	public $connection;
```

```
	public $result;

	public function connect($ConnectionString)
	{
		$this->connection = pg_connect($ConnectionString);
		if ($this->connection==false)
		{
			throw new PGSQLConnectionException($this->connection);
		}
	}
	public function execute($query)
	{
		$this->result = pg_query($this->connection,$query);
		if (!is_resource($this->result))
		{
			throw new PGSQLQueryException($this->connection);
		}
		//else do the necessary works
	}
}

$db = new DAL();
try{
		$db->connect("dbname=golpo user=postgres2");
	try{
		$db->execute("select * from abc");
	}

	catch (Exception $queryexception)
	{
		echo $queryexception->getMessage();
	}
}
catch(Exception $connectionexception)
{
	echo $connectionexception->getMessage();
}
?>
```

Se o código não conseguir conectar ao SGBD ele captura o erro e exibe a mensagem: Sorry, couldn't connect to PostgreSQL server.

Se a conexão for bem sucedida mas existir problema na consulta, então exibirá uma mensagem adequada.

Veja que para uma falha na consão usamos o objeto PGSQLConnectionException e para a falha na consulta usamos o objeto PGSQLQueryException.

Podemos personalizar o desenvolvimento desses objetos extendendo a classe core Exception do PHP5. Veja os exemplos:

```
<?php
Class PGSQLConnectionException extends Exception
{
	public function __construct()
	{
		$message = "Desculpe, impossível conectar ao servidor do PostgreSQL!";
		parent::__construct($message, 0000);
	}
}
?>
```

Agora a classe PGSQLQueryException:

```
<?php
Class PGSQLQueryException extends Exception
{
	public function __construct($connection)
	{
		parent::__construct(pg_last_error($connection), 0);
	}
}
?>
```

Podemos capturar todos os erros como exceptions, exceto os erros fatais. Veja o código:

```
<?php
function exceptions_error_handler($severity, $message, $filename, $lineno) {
	throw new ErrorException ($message, 0, $severity, $filename, $lineno);
}

set_error_handler('exception_error_handler');
?>
```

Autor: fjoggen@gmail.com no manual do PHP.

22) Convenções para Nomes

- Cada classe deve ficar em um script php e somente uma classe
- Classe de nome Emailer fica no arquivo class.emailer.php.
- Todos os arquivos em minúsculas e nomes de classes em CamelCase. Nomes de classes e arquivos não podem ininiar com algarismos.
- Sugestão para os vários nomes:
  - class.emailer.php
  - interface.emailer.php
  - abstract.emailer.php
  - final.emailer.php

- Nomes de propriedades e métodos em camelCase, mas iniciando com minúsculas: sendEmail().

23) Modelando algumas Classes

Pessoa
    Propriedades: nome, cpf, altura, endereco, telefone, uf, cidade
    Métodos (funcionalidades): nascer, crescer, trabalhar, sorrir, passear

Carros
    Propriedades: cor, ano, placas, marca, modelo
    Métodos: buzinar, funcionar, frear, partir

Computadores
    Propriedades: processador, memoria, teclado, gabinete, monitor, ano, fabricante, marca, modelo
    Métodos: iniciar, funcionar, processar, acessar

Contas
    Propriedades: cpf_conta, titular, banco, agencia, data_abertura, saldo, cancelado, senha
    Métodos: abrir, encerrar, depositar, debitar, obter_saldo

Observe que algumas classes se relacionam com outras, como é o caso de pessoa com contas,
através da propriedade cpf e cpf_titular.

Dica: Observe também que cada classe deve lidar com apenas um assunto.

24) Conceitos OO

Classe – uma classe é um template para a criação de objetos. A classe contém o código que define como os objetos devem se comportar e interagir com os demais objetos ou consigo mesmo. Cada vez que criamos um objeto de uma classe e herdando tudo que foi planejado na classe.

Objeto – um objeto é criado quando instanciamos uma classe. A classe é apenas o modelo, o que usamos pra valer são os objetos.

Instanciar - é o ato de criar um objeto de uma classe. Neste momento chama-se o construtor da classe instanciada. É executado assim: $objeto = new NomeClasse();

Propriedade – são similares às variáveis, mas pertencem a uma classe. O PHP não checa o tipo das propriedades.

Método – são semelhantes às funções mas pertencem a uma classe.

Membros – Membros de uma classe são suas propriedades e métodos.

Construtor – é o método da classe que é automaticamente executado sempre que se instancia uma classe.

Destrutor – é o método que quando executado destrói o objeto, invocado pela palavra-

chave __destruct(). Será invocado automaticamente sempre ao final da execução do script.

Herança – O processo chave de derivar um objeto de uma classe é chamado de herança. A classe que extende a outra é chamada de subclasse e deriva todas as propriedades e métodos da superclasse (classe extendida). A subclasse então poderá processar cada método da superclasse.

extends – palavra reservada usada quando uma classe herda de outra, diz-se que ela extendeu a outra, ou seja ClasseSub extends ClasseSuper.
Interface -
Abstrata

Encapsulamento - Encapsulamento é o mecanismo que liga o código aos dados que ele manipula e mantém tanto a salvo de interferência externa e uso indevido. O fechamento de dados e métodos em uma única unidade (chamada classe) é conhecido como encapsulamento. O benefício do encapsulamento é que executa tarefas no interior sem preocupações.

Acoplamento – É a dependência e interdependência entre classes. Quanto menos acoplamento melhor.

Polimorfismo – É o princípio que permite que classes derivadas de uma mesma superclasse tenham métodos iguais (mesma assinatura e parâmetros) mas com comportamentos diferentes definidos em cada uma das classes . Este processo é chamado de polimorfismo. É o processo de criar vários objetos de uma classe básica específica. .

this – this significa uma referência para a atual instância deste objeto. O this tanto dá acesso às propriedades quanto aos métodos. $this->body e $this->setSubject(). O this somente é válido no escopo de métodos que não sejam static.

self - usado para acessar membros static e const de dentro da classe. Para as demais usa-se this.

:: - operador usado para acessar membros static (inclusive const) no lugar de ->.

-> - operador utilizado para acessar membros de uma classe, exceto static.

Sobrescrever/Overwriting - Em um objeto derivado podemos sobrescrever qualquer um dos métodos herdados, desde que sejam declarados como protected ou public e executem alguma coisa como desejamos. Simplesmente crie um método com o mesmo nome do que deseja sobrescrever.
Exemplo:
A classe SendMail tem um método enviarEmail(). O objeto $objmail criado desta classe herdará este método. Queremos que o método tenha um comportamento diferente então criamos um novo método no objeto com o mesmo nome do objeto da classe pai, o que significa sobrescrever o método da classe pai.

Sobrecarregar/Overloading - Tanto chamada de métodos e acesso a membros podem

ser sobrecarregados pelos métodos __call, __get e __set. Esses métodos só serão disparados quando seu objeto ou o objeto herdado não contiver o membro public ou método que você está tentando acessar. Todos os métodos sobrecarregados devem ser definidos estáticos. Todos os métodos sobrecarregados devem ser definidos public.
A partir do PHP 5.1.0 também é possível sobrecarregar as funções isset() and unset() através dos métodos __isset e __unset respectivamente. O método __isset também é chamado com a função empty().

Sobrecarga de membros
void __set (string $name , mixed $value)
mixed __get (string $name)
bool __isset (string $name)
void __unset (string $name)

Membros de classes podem ser sobrecarregados para executar código específico definido na sua classe definindo esses métodos especialmente nomeados. O parâmetro $name usado é o nome da variável que deve ser configurada ou recuperada. O parâmetro $value do método __set() especifica o valor que o objeto deve atribuir à variável $name.
Nota: O método __set() não pode obter argumentos por referência.

implements – palavra-chave usada quando uma nova classe vai implementar uma interface.

new - É utilizada para criar novos objetos de uma classe. $obj = new Pessoa():
Os parêntesis ao final do nome da classe "Pessoa()" são opcionais, mas recomendados para tornar o código mais claro.

25) Padrões de Projeto (Design Patterns)

Inventados pela Gangue dos Quatro (Gang of Four – GoF). Foram criados para solucionar conjuntos de problemas similares de forma inteligente. Os padrões de projeto podem incrementar a performance dos aplicativos com código mínimo. Algumas vezes não é possível aplicar os design patterns. Desnecessário e não planejado uso dos DP pode também degradar a performance de aplicativos.

Padrões GoF

Os padrões "GoF" são organizados em famílias de padrões: de criação, estruturais e comportamentais. Os padrões de criação são relacionados à criação de objetos, os estruturais tratam das associações entre classes e objetos e os comportamentais das interações e divisões de responsabilidades entre as classes ou objetos.

Um padrão "GoF" também é classificado segundo o seu escopo: de classe ou de objeto. Nos padrões com escopo de classe os relacionamentos que definem este padrão são definidos através de herança e em tempo de compilação. Nos padrões com escopo de objeto o padrão é encontrado no relacionamento entre os objetos definidos em tempo de execução.

Padrões "GoF" organizados nas suas famílias:

Padrões de criação

- *Abstract Factory*
- *Builder*
- *Factory Method*
- *Prototype*
- *Singleton*

Padrões estruturais

- *Adapter*
- *Bridge*
- *Composite*
- *Decorator*
- *Façade*
- *Flyweight*
- *Proxy*

Padrões comportamentais

- *Chain of Responsibility*
- *Command*
- *Interpreter*
- *Iterator*
- *Mediator*
- *Memento*
- *Observer*
- *State*
- *Strategy*

- *Template Method*
- *Visitor*

Wikipédia - http://pt.wikipedia.org/wiki/Padr%C3%B5es_de_projeto_de_software

Apenas para citar alguns dos padrões de projeto.
O padrão de projeto mais comum hoje é o MVC.

26) Ferramentas para Trabalhar com PHP Orientado a Objetos

IDEs para programação com PHP

Eclipse PDT
http://www.eclipse.org/pdt/
http://www.eclipse.org/pdt/articles/debugger/os-php-eclipse-pdt-debug-pdf.pdf

NetBeans for PHP
http://www.netbeans.org/downloads/index.html

Ferramentas para trabalhar com UML em PHP

Umbrello - gera código PHP - for Linux (KDE) - http://uml.sourceforge.net/

ArgoUML - http://argouml.tigris.org/

DIA - http://projects.gnome.org/dia/

StartUML - http://staruml.sourceforge.net/en/

UML na Wikipedia - http://en.wikipedia.org/wiki/List_of_UML_tools

Tutorial Creating Use Case Diagrams -
http://www.developer.com/design/article.php/2109801

Poseidon - http://www.gentleware.com/products.html
Versão para a comunidade:
* Download the software. - http://www.gentleware.com/downloadcenter.html

Documentação das classes com o PHP Documentor - http://www.phpdoc.org/
O PHP Documentor ou phpdoc é uma ferramenta de geração automática de documentação para código PHP documentado de forma adequada. Ele é similar ao JavaDoc.

Veja no diretório Documentacao como usar o PHP Documentor para documentar as classes em PHP.

27) Referências

Livro Object-Oriented Programming with PHP5

Autor - Hasin Hayder
Editora – Packt Publishing - http://www.packtpub.com
Código de Exemplo - http://www.packtpub.com/support/

E-book PHP 5 Power Programming
Autores - Andi Gutmans, Stig Sæther Bakken, and Derick Rethans
Editora - PRENTICE HALL – http://www.phptr.com

Livro Programando com Orientação a Objetos
Capítulo 1 de desmonstração
Autor - Pablo Dall'Oglio
Editora – Novatec – http://www.novatec.com.br

Aplicativo em PHPOO para a criação de um Blog
Abaixo um ótimo exemplo de aplicação simples criada com PHPOO e bem comentada, inclusive com código fonte para download:
http://net.tutsplus.com/news/how-to-create-an-object-oriented-blog-using-php/
Fontes: http://nettuts.s3.amazonaws.com/097_PHPBlog/SimpleBlogFiles.zip

Bom Tutorial sobre PHP Orientado a Objetos
http://www.phpro.org/tutorials/Object-Oriented-Programming-with-PHP.html

Exemplo:
A classe SendMail tem um método enviarEmail(). O objeto $objmail criado desta classe herdará este método. Queremos que o método tenha um comportamento diferente então criamos um novo método no objeto com o mesmo nome do objeto da classe pai, o que significa sobrescrever o método da classe pai.

A partir do PHP 5.1.0 também é possível sobrecarregar as funções isset() and unset() através dos métodos __isset e __unset respectivamente. O método __isset também é chamado com a função empty().

Sobrecarga de membros
void __set (string $name , mixed $value)
mixed __get (string $name)
bool __isset (string $name)
void __unset (string $name)

Membros de classes podem ser sobrecarregados para executar código específico definido na sua classe definindo esses métodos especialmente nomeados. O parâmetro $name usado é o nome da variável que deve ser configurada ou recuperada. O parâmetro $value do método __set() especifica o valor que o objeto deve atribuir à variável $name.

Nota: O método __set() não pode obter argumentos por referência.

implements – palavra-chave usada quando uma nova classe vai implementar uma interface.

Apenas para citar alguns dos padrões de projeto.
O padrão de projeto mais comum hoje é o MVC.

Ferramentas para Trabalhar com PHP Orientado a Objetos

IDEs para programação com PHP

Eclipse PDT
http://www.eclipse.org/pdt/
http://www.eclipse.org/pdt/articles/debugger/os-php-eclipse-pdt-debug-pdf.pdf

NetBeans for PHP
http://www.netbeans.org/downloads/index.html

Ferramentas para trabalhar com UML em PHP

Umbrello - gera código PHP - for Linux (KDE) - http://uml.sourceforge.net/

ArgoUML - http://argouml.tigris.org/

DIA - http://projects.gnome.org/dia/

StartUML - http://staruml.sourceforge.net/en/

UML na Wikipedia - http://en.wikipedia.org/wiki/List_of_UML_tools

Tutorial Creating Use Case Diagrams -
http://www.developer.com/design/article.php/2109801

Poseidon - http://www.gentleware.com/products.html
Versão para a comunidade:
* Download the software. - http://www.gentleware.com/downloadcenter.html

Documentação das classes com o PHP Documentor - http://www.phpdoc.org/
O PHP Documentor ou phpdoc é uma ferramenta de geração automática de documentação para código PHP documentado de forma adequada. Ele é similar ao JavaDoc.

Veja no diretório Documentacao como usar o PHP Documentor para documentar as classes em PHP.

Referências

Livro Object-Oriented Programming with PHP5
Autor - Hasin Hayder
Editora – Packt Publishing - http://www.packtpub.com
Código de Exemplo - http://www.packtpub.com/support/

E-book PHP 5 Power Programming
Autores - Andi Gutmans, Stig Sæther Bakken, and Derick Rethans
Editora - PRENTICE HALL – http://www.phptr.com

Livro Programando com Orientação a Objetos
Capítulo 1 de desmonstração
Autor - Pablo Dall'Oglio
Editora – Novatec – http://www.novatec.com.br

Aplicativo em PHPOO para a criação de um Blog
Abaixo um ótimo exemplo de aplicação simples criada com PHPOO e bem comentada, inclusive com código fonte para download:
http://net.tutsplus.com/news/how-to-create-an-object-oriented-blog-using-php/
Fontes: http://nettuts.s3.amazonaws.com/097_PHPBlog/SimpleBlogFiles.zip

Bom Tutorial sobre PHP Orientado a Objetos
http://www.phpro.org/tutorials/Object-Oriented-Programming-with-PHP.html

Livro Pro PHP e jQuery
Editora Apress - http://www.apress.com/9781430228479
Código fonte disponível gratuitamente -
http://www.apress.com/downloadable/download/sample/sample_id/565/

Pro PHP Programming
http://www.apress.com/9781430235606

Código fonte:
http://www.apress.com/downloadable/download/sample/sample_id/1199/

Pro PHP - http://www.apress.com/9781590598191
Código fonte
http://www.apress.com/downloadable/download/sample/sample_id/369/

Dicas

Orientação a Objetos com PHP

Uma classe é como uma planta de casa, que é usada para construir várias casas idênticas.

A classe existe somente como modelo, digital.

O objeto é como uma casa, algo real, construída atraves da planta

PHPOO permite agrupar tarefas semelhantes como classes. Isso facilita a manutenção do código e DRY.

Então se algo mudar no programa então mudamos apenas uma alteração será necessária.

Para adicionar dados a uma classe usamos propriedades ou variáveis específicas de classes. Elas funcionam do mesmo jeito de variáveis comuns, esceto que estão ligadas ao objeto e portanto somente podem ser acessadas usando este objeto.

Propriedades são como as variáveis, mas sendo que as propriedades são ligadas ao objeto e só funciona nele.

Método - similar às funções mas somente funciona neste objeto.

Exemlo:

```
<?php

class MyClass{

        public $prop1 = "I'm a class property!";

}

$obj = new MyClass;

var_dump($obj);

?>
```

A flecha (->) para acessar os membros da classe: propriedades e métodos.

Exemplos:

```
class MyClass

{
```

```
        public $prop1 = "I'm a class property!";
}
$obj = new MyClass;
echo $obj->prop1;
?>
<?php
class MyClass
{
public $prop1 = "I'm a class property!";
        public function setProperty($newval)
        {
                $this->prop1 = $newval;
        }
        public function getProperty()
        {
                return $this->prop1 . "<br />";
        }
}
$obj = new MyClass;
echo $obj->getProperty(); // Get the property value
$obj->setProperty("I'm a new property value!"); // Set a new one
echo $obj->getProperty(); // Read it out again to show the change
?>


<?php
```

```
class MyClass
{
public $prop1 = "I'm a class property!";
public function setProperty($newval)
{
$this->prop1 = $newval;
}
public function getProperty()
{
return $this->prop1 . "<br />";
}
}
// Create two objects
$obj = new MyClass;
$obj2 = new MyClass;
// Get the value of $prop1 from both objects
echo $obj->getProperty();
echo $obj2->getProperty();
// Set new values for both objects
$obj->setProperty("I'm a new property value!");
$obj2->setProperty("I belong to the second instance!");
// Output both objects' $prop1 value
echo $obj->getProperty();
echo $obj2->getProperty();
?>
```

Usando Construtores e Destrutores

Construtores

```
<?php

class MyClass

{

        public $prop1 = "I'm a class property!";

        public function __construct()

        {

                echo 'The class '", __CLASS__, '" was initiated!<br />'; Construtores
```

Construtores

```
        public function setProperty($newval)

        {

                $this->prop1 = $newval;

        }

        public function getProperty()

        {

                return $this->prop1 . "<br />";

        }

}

// Create a new object

$obj = new MyClass;

// Get the value of $prop1

echo $obj->getProperty();

// Output a message at the end of the file
echo "End of file.<br />";

?>
```

Destrutores

```
<?php
class MyClass
{
	public $prop1 = "I'm a class property!";
	public function __construct()
	{
		echo 'The class "', __CLASS__, '" was initiated!<br />';
	}
	public function __destruct()
	{
		echo 'The class "', __CLASS__, '" was destroyed.<br />';
	}
	public function setProperty($newval)
	{
		$this->prop1 = $newval;
	}
	public function getProperty()
	{
		return $this->prop1 . "<br />";
	}
}
// Create a new object
$obj = new MyClass;
// Get the value of $prop1
```

```
echo $obj->getProperty();

// Output a message at the end of the file

echo "End of file.<br />";

?>
```

A medida que suas aplicações crescerem, a POO reduzirá significamente sua carga de trabalho se implementada corretamente.

Usando Herança

Classes podem herdar os métodos e as propriedades de outra classe usando a palavra-chave "extends".

Exemplo:

```
<?php

class MyClass

{

	public $prop1 = "I'm a class property!";

	public function __construct()

	{

		echo 'The class "', __CLASS__, '" was initiated!<br />';

	}

	public function __destruct()

	{

		echo 'The class "', __CLASS__, '" was destroyed.<br />';

	}

	public function __toString()

	{

		echo "Using the toString method: ";

		return $this->getProperty();
```

```
        }
        public function setProperty($newval)
        {
                $this->prop1 = $newval;
        }
        public function getProperty()
        {
                return $this->prop1 . "<br />";
        }
}

class MyOtherClass extends MyClass
{
        public function newMethod()
        {
                echo "From a new method in " . __CLASS__ . ".<br />";
        }
}

// Create a new object
$newobj = new MyOtherClass;

// Output the object as a string
echo $newobj->newMethod();
```

```
// Use a method from the parent class

echo $newobj->getProperty();

?>
```

Sobrescrevendo/Overwriting uma propriedade ou método herdado de uma nova classe. Você pode simplesmente sobrescrevê-lo declarando-o novamente na nova classe, como a seguir.

Exemplo:

```
<?php

class MyClass

{

	public $prop1 = "I'm a class property!";


	public function __construct()

	{

		echo 'The class "', __CLASS__, '" was initiated!<br />';

	}


	public function __destruct()

	{

		echo 'The class "', __CLASS__, '" was destroyed.<br />';

	}


	public function __toString()

	{

		echo "Using the toString method: ";
```

```
        return $this->getProperty();
    }

    public function setProperty($newval)
    {
        $this->prop1 = $newval;
    }

    public function getProperty()
    {
        return $this->prop1 . "<br />";
    }
}

class MyOtherClass extends MyClass
{
    public function __construct()
    {
        echo "A new constructor in " . __CLASS__ . ".<br />";
    }

    public function newMethod()
    {
        echo "From a new method in " . __CLASS__ . ".<br />";
    }
```

```
}

// Create a new object
$newobj = new MyOtherClass;

// Output the object as a string
echo $newobj->newMethod();

// Use a method from the parent class
echo $newobj->getProperty();
?>
```

Preservando e Funcionando o Método Original ao Sobrescrever Métodos

Para isso isamos a palavra chave parent

como o operador de resolução de escopo dois pontos duplos (::)

Exemplo:

```
<?php
class MyClass
{
	public $prop1 = "I'm a class property!";

	public function __construct()
	{
		echo 'The class "', __CLASS__, '" was initiated!<br />';
	}
	public function __destruct()
```

```
        {
                echo 'The class '", __CLASS__, "' was destroyed.<br />';
        }

        public function __toString()
        {
                echo "Using the toString method: ";
                return $this->getProperty();
        }
        public function setProperty($newval)
        {
                $this->prop1 = $newval;
        }

        public function getProperty()
        {
                return $this->prop1 . "<br />";
        }
}

class MyOtherClass extends MyClass
{
        public function __construct()
        {
                parent::__construct(); // Call the parent class's constructor
```

```
            echo "A new constructor in " . __CLASS__ . ".<br />";
        }

        public function newMethod()
        {
        echo "From a new method in " . __CLASS__ . ".<br />";
        }
}

// Create a new object
$newobj = new MyOtherClass;
// Output the object as a string
echo $newobj->newMethod();
// Use a method from the parent class
echo $newobj->getProperty();
?>
```

Saída

```
The class "MyClass" was initiated!
A new constructor in MyOtherClass.
From a new method in MyOtherClass.
I'm a class property!
The class "MyClass" was destroyed.
```

Padrões de Projeto

http://www.cic.unb.br/~jhcf/MyBooks/iess/Patterns/PatternsIntro-25slides.pdf

Padrões de Projeto (Design Patterns)

Inventados pela Gangue dos Quatro (Gang of Four – GoF). Foram criados para solucionar conjuntos de problemas similares de forma inteligente. Os padrões de projeto podem incrementar a performance dos aplicativos com código mínimo. Algumas vezes não é possível aplicar os design patterns. Desnecessário e não planejado uso dos DP pode também degradar a performance de aplicativos.

Padrões GoF

Os padrões "GoF" são organizados em famílias de padrões: de criação, estruturais e comportamentais. Os padrões de criação são relacionados à criação de objetos, os estruturais tratam das associações entre classes e objetos e os comportamentais das interações e divisões de responsabilidades entre as classes ou objetos.

Um padrão "GoF" também é classificado segundo o seu escopo: de classe ou de objeto. Nos padrões com escopo de classe os relacionamentos que definem este padrão são definidos através de herança e em tempo de compilação. Nos padrões com escopo de objeto o padrão é encontrado no relacionamento entre os objetos definidos em tempo de execução.

O padrão de projeto mais comum hoje é o MVC.

MVC com PHP em Exemplos Práticos

1) Introdução

Wikipédia - http://pt.wikipedia.org/wiki/Mvc
Model-view-controller (MVC) é um padrão de arquitetura de software. Com o aumento da complexidade das aplicações desenvolvidas torna-se fundamental a separação entre os dados (Model) e o layout (View). Desta forma, alterações feitas no layout não afetam a manipulação de dados, e estes poderão ser reorganizados sem alterar o layout.

O model-view-controller resolve este problema através da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o usuário, introduzindo um componente entre os dois: o Controller. MVC é usado em padrões de projeto de software,
mas MVC abrange mais da arquitetura de uma aplicação do que é típico para um padrão de projeto.

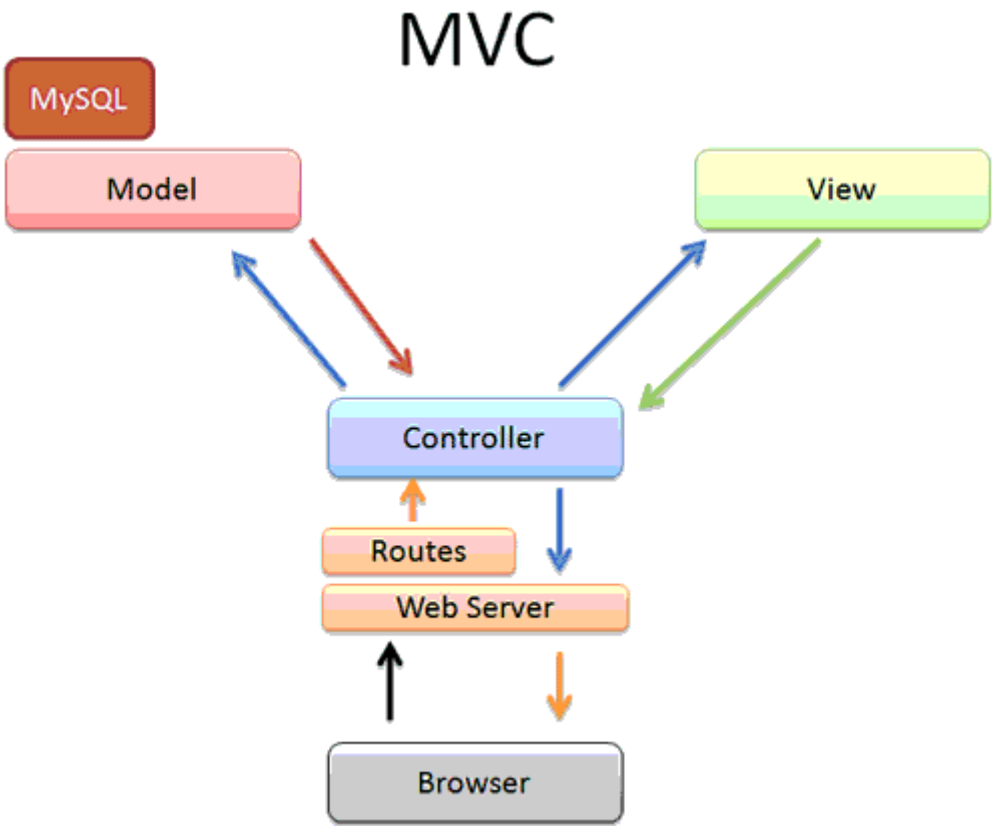
Muitos Frameworks podem parecer muito atraentes à primeira vista, porque eles parecem reduzir o desenvolvimento de aplicações web para um par de passos triviais que levam a alguma geração de código e muitas vezes a detecção automática do esquema do banco, mas estes mesmos atalhos são susceptíveis de serem seus gargalos, bem como, uma vez que alcançar esta simplicidade pode sacrificar a flexibilidade e o desempenho.
Rasmus Lerdorf (Criador do PHP).

Portanto não adianta apenas facilitar a criação, mas um framework também deve continuar flexível e permitir uma fácil manutenção do aplicativo gerado. Caso contrário o framework estará dando com uma mão e retirando com a outra.

Aqui é até redundante dizer que para os que estão querendo aprender sobre MVC, a experimentação prática dos exemplos é imprescindível, portanto experimente, altere, personalize e teste bastante até entender e ficar satisfeito.

Em caso de dúvidas, preferencialmente troque idéias com os colegas através do forum do site: efetue seu login e vá em Forum, para que assim as dúvidas e dicas, juntamente com as respostas fiquem à disposição de todos e com isso também estou tentando estimular uma maior integração de todos nós que estamos aprendendo sobre MVC.

MVC



Esquema simplificado do fluxo de informações no MVC.

Fluxo de Informações Simplificado no MVC

Alguém usa o Browser para acessar uma página.
O Servidor web recebe a solicitação e procura pela página.
Se encontra a página envia para o endereço no servidor.
O router envia para o devido controller.
O Controller através do respectivo método/action envia para o Model se for o caso.
O Modem processa e manda de volta para o Controller.
O controller manda para a view.
A view manda de volta para o Browser do usuário o resultado.
Veja que o controller pode enviar diretamente para a view ao invés do model, se for o caso.

Vale lembrar que cada software implementa esse fluxo com algumas diferenças, por isso nem sempre será assim.

Model - São ativas representações das tabelas do banco de dados: podem conectar para o banco de dados, efetuar consultas nele (se instruído para fazer isso por um controller) e samvam dados no banco de dados. Na arquitetura MVC não deve haver interação entre modelos e views. Toda a lógica é manipulada pelos controllers.
É onde toda a lógica de negócios é guardada. Se a aplicação pode acessar informações do banco de dados, o código para fazer isso deve ficar no model. Se necessário, por exemplo, para buscar dados do estoque ou informar sobre um novo produto, então o código também seria mantido no modelo.

View - é onde todos os elementos da interface de usuário da nossa aplicação são mantidos. Isso pode incluir o nosso HTML, CSS e arquivos JavaScript e algum PHP. Qualquer coisa que um usuário vê ou interage com ela pode ser mantido em uma View, e às vezes o que o usuário vê é na verdade uma combinação de muitos views diferentes no mesmo pedido.

Podem ser descritos como arquivos de templates que apresentam seu conteúdo para o usuário: variáveis, arrays e objetos que são usados em views são registrados através de um controller. Views não deveriam conter complexas lógicas de negócios, somente a estrutura de controle elementar necessária para executar operações particulares como com a iteração de coleções de dados através de uma contrução foreach, deve ser contina na view.

Controller - é o componente que liga models e views. Controladores isolam a regra de negócios de um model a partir dos elementos de interface de usuário de uma view, e lida com a forma como a aplicação irá responder ao usuário na exibição. Os controladores são o primeiro ponto de entrada para este trio de componentes (MVC), porque o pedido é primeiro passado para um controlador, o qual irá, em seguida, instanciar os modelos e as views necessárias para satisfazer ao pedido da aplicação.
Contém a lógica da aplicação. Cada controller pode oferecer diferentes funcionalidades/métodos/actions. Controles recebem e modificam os dados acessando as tabelas do banco de dados através do modelos. Registram variáveis e objetos que podem ser usados nas views.

Boas Práticas
Como um projeto grande tem muito código é coerente criar uma estrutura de código clara e bem documentada.
Separando cada classe em um arquivo, cada classe cuidando de uma única tarefa.

Namespaces
Uma adição particularmente útil ao novo PHP 5.3.0 é o namespaces. Namespaces permitem que os desenvolvedores enjaulem seu código, fora do espaço de nome global, para evitar conflitos de nome de classe e ajudar a organizar seu código melhor. Namespaces ajudam a remover as classes do namespace global. O namespace permanece dentro do espaço global, por isso deve permanecer único, no entanto, ele pode conter qualquer número de classes, que pode reutilizar nomes de classe que estão no namespace global, ou dentro de outros espaços.
http://php.net/manual/en/language.namespaces.php

2) Teoria

Abstração de Objetos

PHP 5 introduz métodos e classes abstratos. Não é permitido criar uma instância de uma class que foi definida como abstrata. Qualquer classe que contém pelo menos um método abstrato deve também ser abstrata. Métodos definidos como abstratos simplesmente declaram a assinatura do método, eles não podem definir a implementação.

Quando uma classe herda uma classe abstrata, todos os métodos marcados como abstratos na declaração da classe-pai devem ser definidos na classe filha; além disso, esses métodos devem ser definidos com a mesma (ou menos restrita) visibilidade. Por exemplo, se um método abstrado é definido como protected, a implementação da função deve ser definida ou como protected ou public, mas não private.
http://www.php.net/manual/pt_BR/language.oop5.abstract.php

O PHP tem métodos internos que ajudam quando temos vários campos:

__set() é executado ao se escrever dados para membros inacessíveis.
__get() é utilizados para ler dados de membros inacessíveis.
http://www.php.net/manual/pt_BR/language.oop5.overloading.php#language.oop5.overloading.members

Autoloading Objects

Muitos desenvolvedores ao desenvolver aplicações orientadas a objeto criam um arquivo PHP para cada definição de classe. Um dos maiores contratempos é ter de escrever uma longa lista de includes no início de cada script(um include para cada classe necessária).

Com PHP 5 isso não é mais necessário. Você pode definir uma função __autoload que é automaticamente chamada no caso de você tentar usar uma classe/interface que ainda não foi definida. Ao chamar essa função o 'scripting engine' tem uma última chance para carregar a classe antes que o PHP falhe com erro.
http://www.php.net/manual/pt_BR/language.oop5.autoload.php

Interfaces de Objetos

Interfaces de Objetos permitem a criação de código que especifica quais métodos e variáveis uma classe deve implementar, sem ter que definir como esses métodos serão tratados.

Interfaces são definidas usando a palavra-chave 'interface', da mesma maneira que uma classe comum, mas sem nenhum dos métodos ter seu conteúdo definido.

Todos os métodos declarados em uma interface devem ser public, essa é a natureza de uma interface.

implements

Para implementar uma interface, o operador implements é usado. Todos os métodos na interface devem ser implementados na classe; não fazer isso resultará em um erro fatal. Classes podem implementar mais de uma interface se assim for desejado, separando cada interface com uma vírgula.
http://www.php.net/manual/pt_BR/language.oop5.interfaces.php

3) Exemplos Práticos e Simples de MVC com PHP5

A idéia para este curso partiu de um exemplo realmente mínimo que encontrei sobre MVC no PHP5.

Após executar com sucesso comecei a mexer no exemplo, fazendo alterações e verificando o fluso das informações. Percebi que o fluxo não anda no sentido da sigla MVC, nesse sentido, pois geralmente não se admite a comunicação entre as camadas M (model) e V (view). No caso as informações andam nesse sentido:

Fluxo das Informações no MVC

- Geralmente Nascem na View quando um usuário faz uma solicitação, clicando num botão submit ou num link
- Daí são enviadas para o Controller, que a filtra (se for o caso) e a envia para o Model
- O Model analisa de acordo com a solicitação (uma consulta ao banco) e a devolve ao Controller
- O Controler por sua vez devolve o resultado para a View
- E a View renderiza o resultado e o mostra para o usuário

Dizer o que faz cada uma das camadas praticamente todas as definições dizem, mas algo que é de muita importância para o programador, que é o fluxo das informações, onde elas começam e onde terminam, isso já não é comum.

Primeiro eu executei o exemplo e funcionou bem. Depois então, com mi nhas alterações eu saí rastreando as informações para, na prática, perceber como caminhavam.

O exemplo mínimo a que me referi encontrei aqui:

http://www.sourcecode4you.com/article/articleview/5dfe0fd3-5808-4fb2-818e-51c807cf8c6a/mvc-architecture-in-php.aspx

Dando uma olhada vi que usa o banco test e com apenas uma pequena tabela:

```
create table employee
(
        deptid int auto_increment primary key,
        emp_name char(45) not null
);
```

Fiz uma alteração: mudei o nome do banco para mvc.

Inserir alguns rgistros:

```
insert into employee values (default, 'Ribamar');
insert into employee values (default,'Pedro');
insert into employee values (default,'Tiago');
insert into employee values (default,'João');
insert into employee values (default,'Elias');
```

Também dividi em dois arquivos como sugere o artigo: Employee.php e View.php

Employer.php

```
<?php
//////////////My Property Class////////////
// it contain getter setter function

abstract class EmployeeProperty
{
        private $deptid;

        public function getDeptId() //Getter
        {
            return $this->deptid;
        }

        public function setDeptId($id) //setter
        {
                $this->deptid=$id;
        }
}

//////////My Interface /////////
interface iEmployee
{
        function getEmployeeName(EmployeeProperty $objEmployeeProperty);
}

/////////My Data access layer//////
// it fetch data from database server
//it model part of mvc

class DALEmployee implements iEmployee
{
       public function getEmployeeName(EmployeeProperty $objEmployeeProperty)
   {
        $con = mysql_connect("localhost","root","ribafs"); //open connection

        if (!$con){
                                die('Could not connect to mysql ' . mysql_error()); // error
```

```
message
        } else {
            mysql_select_db("mvc",$con); // select database
            $result= mysql_query("select emp_name from employee where deptid=".
$objEmployeeProperty-> getDeptId()); // fire query
                        mysql_close($con); // close connection
        }
        return $result;
    }
}

//////////My business logic layer////////
// it is controller part of mvc
class BALEmployee extends EmployeeProperty
{
        public function getEmployeeName(EmployeeProperty $objEmployeeProperty)
    {
            $objiEmployee=new DALEmployee();
       return $objiEmployee->getEmployeeName($objEmployeeProperty);
    }
}
?>

View.php
<!--////////////////////My View Part/////////////////////////////////////-->

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> New Document </TITLE>
 </HEAD>
 <BODY>
  <?php
  include("Employee.php");
  ///////////////My View Part//////////////////

  $objBALEmployee=new BALEmployee();  //Create object of business logic layer
  $objBALEmployee->setDeptId(1);      // Set Property
  $result= $objBALEmployee->getEmployeeName($objBALEmployee); // excess bll
function
  while($row = mysql_fetch_row($result)) // fetch result
  {
          echo $row[0]."<br>"; // display result
  }
  ?>
 </BODY>
</HTML>
```

Veja que esta querendo trazer do banco o emp_name de um empregado com um certo deptid.

Portanto precisamos, antes de testar, adicionar pelo menosum registro e com deptid = 1.

Agora eu comecei a mexer, primeiro dividi em 3 arquivos e fiz o include apenas do que o respectivo iria precisar.

Na primeira experiência eu separei o arquivo Employee.php em dois: Model.php e Controller.php.
Além disso criei um script para a conexão do banco de dados, chamado connection.inc.php.

Vejamos como ficaram os includes, que são importantes para entender o fluxo das imformações. Se eu não soubesse como elas caminham ou não estivesse preocupado com elas eu simplesmente faria o include de todos na View, mas me interessa incluir apenas na hora certa o arquivo certo.

No Model.php fiz o include apenas do connection.inc.php pois ele precisa conectar ao banco.

No Controller.php fiz o include apenas do Model.php

No View.php apenas o include do Controller.php

Assim conseguimos perceber quem se comunica com quem.

Agora já criei um outro banco mais brasileiro/português para sentir que de fato estava alterando.
Criei o **banco funcionario**, com a tabela funcionarios assim:

Tabela
```
create table funcionarios
(
        codigo int auto_increment primary key,
        nome char(45) not null
);

insert into funcionarios (nome) values ('João Brito Cunha');
insert into funcionarios (nome) values ('Pedro Barbosa Abreu');
insert into funcionarios (nome) values ('Gilberto Braga');
insert into funcionarios (nome) values ('Cassimiro Abreu');
insert into funcionarios (nome) values ('João dos Anzóis Pereira');
```

Veja como ficou o código:

connection.inc.php
```
<?php

        $con = mysql_connect("localhost","root","ribafs");

    if (!$con){
```

```
        die('Could not connect to mysql ' . mysql_error());
    }else{
                mysql_select_db("funcionario",$con);
        }
?>
```

Model.php

```
<?php
// Exemplo prático e mínimo de MVC em PHP
// http://www.sourcecode4you.com/article/articleview/5dfe0fd3-5808-4fb2-818e-
51c807cf8c6a/mvc-architecture-in-php.aspx

include_once('./connection.inc.php');

/**
  * Classe Abstrata - Propriedades para getters e setters
  */

abstract class FuncionarioProperty{
        private $codigo;
        private $nome;

    public function getCodigo(){
            return $this->codigo;
    }

    public function setCodigo($id){
            $this->codigo=$id;
    }

    public function getNome(){
            return $this->nome;
    }

    public function setNome($id){
            $this->nome=$id;
    }
}

/**
  * Interface
  */

interface iFuncionario{
        function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty);
}

/**
  * Modelo - Camada de Acesso a Dados
  * Acessa o banco de dados e efetua as operações necessárias com
  */
```

```
class DALFuncionario implements iFuncionario{
        public function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty){
           $result= mysql_query("select codigo, nome from funcionarios where codigo=".
$objFuncionarioProperty-> getCodigo());
                print "<script>alert('Model');</script>";
        return $result;
        }
}
```

Controller.php

```
<?php
/**
  * Controller - Camada de Lógica de Negócios
  * Processa dados, efetua operações diversas
  * Recebe requisições do usuário através da View, efetua processamentos como
validação do usuário e outras
  * e envia para o Model a solicitação de dados
  * O Model devolve os dados ao Controller e o Controller devolve para a View
  */

include_once('./Model.php');

class BALFuncionario extends FuncionarioProperty{
        public function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty){
                $objiFuncionario=new DALFuncionario();
                print "<script>alert('Controller');</script>";
                return $objiFuncionario->getFuncionarioNome($objFuncionarioProperty);
        }
}
```

View.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Exemplo Mínimo</TITLE>
 </HEAD>
 <BODY>
 <H2> MVC - Exemplo Mínimo</H2>

  <?php
        /**
          * View - Envia solicitação de dados ao Controller
          * Ao receber os dados do Controller os apresenta formatados para o usuário
          */

  print "<script>alert('View');</script>";
  include("./Controller.php");
```

```
 $objBALFuncionario=new BALFuncionario();
 $objBALFuncionario->setCodigo(1); // Passa o código do funcionário a retornar

 $result= $objBALFuncionario->getFuncionarioNome($objBALFuncionario);
 print "<script>alert('View');</script>";

 while($row = mysql_fetch_row($result)){
         echo $row[0]."-";
         echo $row[1]."<br>";
 }
 ?>
</BODY>
</HTML>
```

Depois adicionei apenas um index.php chamando o View.php para ficar bonitinho e não precisar abrir diretamente o View.php.

Na view fiz uma pequena modificação, além de trazer o nome, trouxe também o código.

Veja que se o código requerido não estiver no banco precisará alterar para um existente para que a view traga o registro.

Então chegou o momento de ir em frente e agora alterar os arquivos para poder fazer algo de útil, podedo alterar, inserir e excluir registros.

Personalizando o Exemplo de MVC

Mo exemplo alterado que chamei de mvc_minimo3, eu implementei as operações básicas de uso do banco de dados, o CRUD (muito simples), sem nenhum acabamento, apenas com a intenção de mostrar como essas operações funcionam dentro de um aplicativo usando o famoso padrão de projeto MVC e ainda por cima de forma bem simples.

Espero ter conseguido deixar o exemplo ainda simples após as alterações.

Aqui criei uma View para Insert, uma para Update, uma para Delete, uma para Selet de um registro e uma para Select de todos os registros. Acredito que o próximo passo seja mostrar tudo em um único script.

Mas veja que cada uma dessas Views prcisa encontrar métodos respectivos no Controller e no Model.

connection.inc.php

```
<?php

        $con = mysql_connect("localhost","root","ribafs");

    if (!$con){
        die('Could not connect to mysql ' . mysql_error());
    }else{
```

```
                mysql_select_db("funcionario",$con);
        }
?>
```

Model.php

```
<?php
// Pequeno Exemplo de MVC - Requer PHP5
// http://www.sourcecode4you.com/article/articleview/5dfe0fd3-5808-4fb2-818e-
51c807cf8c6a/mvc-architecture-in-php.aspx

//////////////Classe de Propriedades///////////
// contém os métodos: getter e setter

include_once('./connection.inc.php');

abstract class FuncionarioProperty{
        private $codigo;
        private $nome;

//Código
   public function getCodigo() //Getter
   {
           return $this->codigo;
   }

   public function setCodigo($id) //Setter
   {
           $this->codigo=$id;
   }

//Nome
   public function getNome() //Getter
   {
           return $this->nome;
   }

   public function setNome($id) //Setter
   {
           $this->nome=$id;
   }

}

/////////Interface ////////
interface iFuncionario{
        function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty);
        function getFuncionario(FuncionarioProperty $objFuncionarioProperty);
        function insertFuncionario(FuncionarioProperty $objFuncionarioProperty);
        function editFuncionario(FuncionarioProperty $objFuncionarioProperty);
        function delFuncionario(FuncionarioProperty $objFuncionarioProperty);
```

```
}


///////// Modelo - Camada de Acesso a Dados//////
// Puxa dados do banco de dados

class DALFuncionario implements iFuncionario{
  public function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty) {
    $result= mysql_query("select codigo, nome from funcionarios where codigo=".
$objFuncionarioProperty->getCodigo());
    return $result;
  }

  // Comente todo este método abaixo e execute para analisar a mensagem de erro
  public function getFuncionario(FuncionarioProperty $objFuncionarioProperty) {
    $result= mysql_query("select codigo,nome from funcionarios");
    return $result;
  }

  public function insertFuncionario(FuncionarioProperty $objFuncionarioProperty) {
        $sql="insert into funcionarios (codigo, nome) values (".$objFuncionarioProperty-
>getCodigo().",'". $objFuncionarioProperty->getNome()."')";
    $result= mysql_query($sql);
    if(!$result) {
                die('Erro na inclusão<br>'.mysql_error());
        }else{
                print "Registro inserido com sucesso!";
        }
  }

  public function editFuncionario(FuncionarioProperty $objFuncionarioProperty) {
         $sql="update funcionarios set nome='". $objFuncionarioProperty->getNome(). "'
where codigo=".$objFuncionarioProperty->getCodigo();
//print $sql;exit;
    $result= mysql_query($sql);
    if(!$result) {
                die('Erro na atualização<br>'.mysql_error());
        }else{
                print "Registro atualizado com sucesso!";
        }
  }

  public function delFuncionario(FuncionarioProperty $objFuncionarioProperty) {
         $sql="delete from funcionarios where codigo=".$objFuncionarioProperty-
>getCodigo().";";
//print $sql;exit;
    $result= mysql_query($sql);
    if(!$result) {
                die('Erro na exclusão<br>'.mysql_error());
        }else{
```

```
                print "Registro excluído com sucesso!";
        }
  }

}
```

Controller.php

```
<?php
//////////Controller - Camada de Lógica de Negócios///////

include_once('./Model.php');

class BALFuncionario extends FuncionarioProperty{
  public function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty){
        $objiFuncionario=new DALFuncionario();
        return $objiFuncionario->getFuncionarioNome($objFuncionarioProperty);
  }

  public function getFuncionario(FuncionarioProperty $objFuncionarioProperty){
        $objiFuncionario=new DALFuncionario();
        return $objiFuncionario->getFuncionario($objFuncionarioProperty);
  }

  public function insertFuncionario(FuncionarioProperty $objFuncionarioProperty){
        $objiFuncionario=new DALFuncionario();
        return $objiFuncionario->insertFuncionario($objFuncionarioProperty);
  }

  public function editFuncionario(FuncionarioProperty $objFuncionarioProperty){
        $objiFuncionario=new DALFuncionario();
        return $objiFuncionario->editFuncionario($objFuncionarioProperty);
  }

  public function delFuncionario(FuncionarioProperty $objFuncionarioProperty){
        $objiFuncionario=new DALFuncionario();
        return $objiFuncionario->delFuncionario($objFuncionarioProperty);
  }

}
```

ViewDelete.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
```

```
  include("./Controller.php");

  ///////////////View - Apresentação dos dados recebidos do Controller/////////////////
  $objBALFuncionario=new BALFuncionario();
  $result= $objBALFuncionario->getFuncionario($objBALFuncionario);

  print "<h1>Excluir registros</h1>";
?>

<br>
<br>
<form name="form" method="post" action="">
Código<input type="text" name="codigo"><br>
<input type="submit" name="enviar" value="Enviar"><br>
</form>

<?php

if(isset($_POST['enviar'])){
        $objBALFuncionario->setCodigo($_POST['codigo']); // Passa o código do
funcionário a retornar
        $result = $objBALFuncionario->delFuncionario($objBALFuncionario);
}
?>

 </BODY>
</HTML>
```

ViewEdit.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
  include("./Controller.php");

  ///////////////View - Apresentação dos dados recebidos do Controller/////////////////
  $objBALFuncionario=new BALFuncionario();
  $result= $objBALFuncionario->getFuncionario($objBALFuncionario);

  print "<h1>Editar registros</h1>";
  while($row = mysql_fetch_row($result)){
          echo "<a href='View4.php?codigo=$row[0]&nome=$row[1]'>$row[0]</a>-";
          echo $row[1]."<br>";
  }

if(isset($_GET['codigo'])){
```

```
        $codigo=$_GET['codigo'];
        $nome=$_GET['nome'];
}
?>

<br>
<br>
<form name="form" method="post" action="">
Código<input type="text" name="codigo" value="<?php print $codigo;?>"><br>
Nome<input type="text" name="nome" value="<?php print $nome;?>"><br>
<input type="submit" name="enviar" value="Enviar"><br>
</form>

<?php

if(isset($_POST['enviar'])){
        $objBALFuncionario->setCodigo($_POST['codigo']); // Passa o código do
funcionário a retornar
        $objBALFuncionario->setNome($_POST['nome']);
        $result = $objBALFuncionario->editFuncionario($objBALFuncionario);
}
?>

 </BODY>
</HTML>
```

ViewInsert.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
  include("./Controller.php");

  ///////////////View - Apresentação dos dados recebidos do Controller//////////////////
  $objBALFuncionario=new BALFuncionario();
?>
<h2>Inserir Registros</h2>

<form name="form" method="post" action="">
Código<input type="text" name="codigo"><br>
Nome<input type="text" name="nome"><br>
<input type="submit" name="enviar" value="Enviar"><br>
</form>

<?php
```

```
if(isset($_POST['enviar'])){
$codigo=$_POST['codigo'];
$nome=$_POST['nome'];

  $objBALFuncionario->setCodigo($codigo); // Passa o código do funcionário a retornar
  $objBALFuncionario->setNome($nome); // Passa o código do funcionário a retornar

  $result= $objBALFuncionario->insertFuncionario($objBALFuncionario);
}
  ?>
 </BODY>
</HTML>
```

ViewSelectOne.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
  include("./Controller.php");

  ////////////////View - Apresentação dos dados recebidos do Controller//////////////////
  $objBALFuncionario=new BALFuncionario();
  $objBALFuncionario->setCodigo(3); // Passa o código do funcionário a retornar

  $result= $objBALFuncionario->getFuncionarioNome($objBALFuncionario);

  print "<h1>Retornando apenas um registro</h1>";
  while($row = mysql_fetch_row($result)){
          echo $row[0]."-";
          echo $row[1]."<br>";
  }
  ?>
 </BODY>
</HTML>
```

ViewSelectAll.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>
```

```
 <?php
 include("./Controller.php");

 ///////////////View - Apresentação dos dados recebidos do Controller/////////////////
 $objBALFuncionario=new BALFuncionario();
 $result= $objBALFuncionario->getFuncionario($objBALFuncionario);

 print "<h1>Retornando todos os registros</h1>";
 while($row = mysql_fetch_row($result)){
          echo $row[0]."-";
          echo $row[1]."<br>";
 }

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

index.php (mas que não contempla todas as views)

```
<h1>Exemplo prático e mínimo do uso do padrão MVC com PHP</h1>

<?php
if(!isset($_GET['conteudo'])){
        $conteudo='ViewSelectAll.php';
        print "<a href=\"index.php?conteudo=$conteudo\"><h3>Retornar todos os
registros</h3></a>";
}else{
        $conteudo2='ViewSelectOne.php';
        print "<a href=\"index.php?conteudo2=$conteudo2\"><h3>Retornar um
registro</h3></a>";
}

print "<a href=\"./View3.php\"><h3>Inserir Registro</h3></a>";

if(isset($conteudo)){
        include_once($conteudo);
}else{
        include_once($conteudo2);
}
?>
```

4) Referências

Livros:

1) PHP 5 Power Programming
Autores - Andi Gutmans, Stig Sæther Bakken, and Derick Rethans
Editora - PRENTICE HALL - http://www.phptr.com
Download -
http://ptgmedia.pearsoncmg.com/images/013147149X/downloads/013147149X_book.pdf

2) Object-Oriented Programming with PHP5
Autor - Hasin Hayder - http://hasin.wordpress.com/
Editora - Packt Publishing - http://www.packtpub.com

3) Open Source Content Management System Book Sampler
(Drupal, Alfresco, Moodle, Wordpress, Joomla 1.5, Plone, e107, ezPublish)
Autor - David Mercer
Editora - Packt Publishing - http://www.packtpub.com

4) Programando com Orientação a Objetos
Capítulo 1 de desmonstração
Autor - Pablo Dall'Oglio
Editora – Novatec – http://www.novatec.com.br

http://book.cakephp.org/pt/compare/10/Understanding-Model-View-Controller

http://anantgarg.com/2009/03/13/write-your-own-php-mvc-framework-part-1/
http://slimphp.sourceforge.net/

http://oreilly.com/php/archive/mvc-intro.html
http://www.onlamp.com/pub/a/php/2006/03/02/mvc_model.html
http://www.onlamp.com/php/2005/11/03/mvc_controller.html
http://www.onlamp.com/php/2006/01/26/mvc_view.html

http://www.terminally-incoherent.com/blog/2008/10/22/writing-a-minimalistic-mvc-framework-in-php/

http://filesocial.eu.s3.amazonaws.com/i8tupv6/7e595a6a5f33bef4785ce41db891de33c2d0d59d/ProgramacaoOOPHP5.pdf – Elton

http://www.henriquebarroso.com/how-to-create-a-simple-mvc-framework-in-php/#more-1

http://www.phpro.org/tutorials/Model-View-Controller-MVC.html
http://www.phpro.org/downloads/mvc-0.0.4.tar.gz

http://www.revistaphp.com.br/print.php?id=50
http://www.phpit.net/article/simple-mvc-php5/1/?pdf=yes
http://robrohan.com/projects/download.php?file=examples/MvCphp_1.0.zip
http://www.onlamp.com/lpt/a/6438
http://www.fragmental.com.br/wiki/index.php?title=MVC_e_Camadas
http://pt.wikipedia.org/wiki/MVC
http://www.joomla.com.br/-artigos-mainmenu-43/234-como-sorganizados-os-arquivos-no-joomla-15.html?tmpl=component&print=1&page=
http://imasters.uol.com.br/artigo/5795/php/mvc_em_php_com_smarty_-_parte_02/imprimir/

Pequeno Framework
http://skinnymvc.com/

5) Dicas Úteis

```
define('DS', DIRECTORY_SEPARATOR);
define('ROOT', dirname(dirname(__FILE__)));

$url = $_GET['url'];

require_once (ROOT . DS . 'library' . DS . 'bootstrap.php');

function setReporting() {
        if (DEVELOPMENT_ENVIRONMENT == true) {
      error_reporting(E_ALL);
      ini_set('display_errors','On');
        } else {
      error_reporting(E_ALL);
      ini_set('display_errors','Off');
      ini_set('log_errors', 'On');
      ini_set('error_log', ROOT.DS.'tmp'.DS.'logs'.DS.'error.log');
        }
}

/** Check for Magic Quotes and remove them **/

function stripSlashesDeep($value) {
      $value = is_array($value) ? array_map('stripSlashesDeep', $value) :
stripslashes($value);
      return $value;
}

function removeMagicQuotes() {
if ( get_magic_quotes_gpc() ) {
      $_GET    = stripSlashesDeep($_GET   );
      $_POST   = stripSlashesDeep($_POST  );
      $_COOKIE = stripSlashesDeep($_COOKIE);
}
}

/** Check register globals and remove them **/

function unregisterGlobals() {
    if (ini_get('register_globals')) {
      $array = array('_SESSION', '_POST', '_GET', '_COOKIE', '_REQUEST', '_SERVER',
'_ENV', '_FILES');
      foreach ($array as $value) {
          foreach ($GLOBALS[$value] as $key => $var) {
              if ($var === $GLOBALS[$key]) {
                  unset($GLOBALS[$key]);
              }
          }
      }
```

```
    }
}

/** Autoload any classes that are required **/

function __autoload($className) {
        if (file_exists(ROOT . DS . 'library' . DS . strtolower($className) . '.class.php')) {
                require_once(ROOT . DS . 'library' . DS . strtolower($className) . '.class.php');
        } else if (file_exists(ROOT . DS . 'application' . DS . 'controllers' . DS .
strtolower($className) . '.php')) {
                require_once(ROOT . DS . 'application' . DS . 'controllers' . DS .
strtolower($className) . '.php');
        } else if (file_exists(ROOT . DS . 'application' . DS . 'models' . DS .
strtolower($className) . '.php')) {
                require_once(ROOT . DS . 'application' . DS . 'models' . DS .
strtolower($className) . '.php');
        } else {
                /* Error Generation Code Here */
        }
}
```

Referências

Grandes Frameworks:
http://cakephp.org
http://framework.zend.com
http://codeigniter.com

Grandes CMS:
Joomla - Portal
Drupal - Portal
WordPress - Blog
Moodle - Elearning
MediaWiki - Wiki
PrestaShop - Loja virtual
Magento - Loja virtual

MVC em PHP:
http://jream.com/lab/open-source - PHP MVC Tutorial - código fonte e vídeo-aulas
http://www.videoaulasbrasil.com.br/tag/criando-um-mini-framework-PHP-5-com-MVC/

Pequenos Frameworks em PHP com MVC
PIP - http://gilbitron.github.com/PIP
Slim - http://travis-ci.org/codeguy/Slim
Fuelphp - http://fuelphp.com/
Kissmvc - http://kissmvc.com
Simplefw - http://github.com/tylerhall/OpenFeedback/tree/master
Simple php framework - http://github.com/tylerhall/simple-php-framework/
SofiaPHP - http://code.google.com/p/sofia-php/
Tiny MVC - http://www.tinymvc.com/download

Outras Referências

http://www.phpro.org/
http://pt.scribd.com/doc/77044385/Model-View-Controller-MVC
http://sam-tallerdesarrollo.googlecode.com/svn-history/r163/trunk/modelos/
http://www.sunilb.com/category/php/php5-oops-tutorials
http://pt.wikipedia.org/wiki/Mvc
http://www.videoaulasbrasil.com.br/tag/criando-um-mini-framework-PHP-5-com-MVC/
http://anantgarg.com/2009/03/13/write-your-own-php-mvc-framework-part-1/
http://stackoverflow.com/questions/6022231/simple-php-mvc-example
http://r.je/mvc.phps
http://r.je/mvc-in-php.html
http://r.je/views-are-not-templates.html
http://johnsquibb.com/tutorials/mvc-framework-in-1-hour-part-one
http://ribafs.org/portal/programacao-web/58-phpoo/727-mvc-em-php
http://ribafs.org/portal/programacao-web/58-phpoo
http://abda.la/51
http://www.tocadigital.com.br/2012/07/construcao-de-sites-em-php-com-estrutura-mvc/

MVC no Joomla 1.5
http://www.numaboa.com/informatica/tutos/joomla/885-mvc

Origem: The CakePHP Framework: Your First Bite - No SitePoint.com
http://www.sitepoint.com/article/application-development-cakephp

A gentle introduction to MVC (em 3 partes)
http://nemetral.net/2008/07/31/a-gentle-introduction-to-mvc-part-1/

PHP: Uma introdução suave ao MVC
http://pequenotux.blogspot.com/2008/12/php-uma-introduo-suave-ao-mvc-parte-1.html
http://pequenotux.blogspot.com/2008/12/php-uma-introduo-suave-ao-mvc-parte-2.html

Desenvolvimento em três camadas utilizando MVC e PHP 5
inf.unisul.br/~ines/workcomp/cd/pdfs/2905.pdf

http://www.revistaphp.com.br/print.php?id=50
http://www.phpit.net/article/simple-mvc-php5/1/?pdf=yes
http://robrohan.com/projects/download.php?file=examples/MvCphp_1.0.zip
http://www.onlamp.com/lpt/a/6438
http://www.fragmental.com.br/wiki/index.php?title=MVC_e_Camadas
http://pt.wikipedia.org/wiki/MVC
http://www.joomla.com.br/-artigos-mainmenu-43/234-como-sorganizados-os-arquivos-no-joomla-15.html?tmpl=component&print=1&page=
http://imasters.uol.com.br/artigo/5795/php/mvc_em_php_com_smarty_-_parte_02/imprimir/

Livros
http://www.novatec.com.br/livros/phpobjetos2/ - PHP Programando com Orientação a Objetos
http://www.apress.com/9781430241645 - Pro PHP MVC
http://www.packtpub.com/object-oriented-programming-php5/book

http://www.packtpub.com/support/1104

Documentação do Active Record no Frameork Yii

Apesar do DAO do Yii ser capaz de cuidar, praticamente, de qualquer tarefa relacionada a banco de dados, há uma grande chance de que, ainda, gastaríamos 90% do nosso tempo escrevendo instruções SQL para efetuar as operações de CRUD (create (criar), read (ler), update (atualizar) e delete (excluir)). Além disso, nosso código é mais difícil de manter quando temos instruções SQL misturadas com ele. Para resolver esses problemas, podemos utilizar Active Record (Registro Ativo).

Active Record (AR) é uma popular técnica de Mapeamento Objeto-Relacional (Object-Relational Mapping, ORM). Cada classe AR representa uma tabela (ou uma view) do banco de dados, cujos campos são representados por propriedades na classe AR. Uma instância de uma AR representa um único registro de uma tabela. As operações de CRUD são implementadas como métodos na classe AR. Como resultado, podemos acessar nossos dados de uma maneira orientada a objetos. Por exemplo, podemos fazer como no código a seguir para inserir um novo registro na tabela `Post`:

```
$post=new Post;
$post->title='post de exemplo';
$post->content='conteúdo do post';
$post->save();
```

A seguir, descreveremos como configurar AR e como utiliza-lo para executar operações de CRUD. Na próxima seção, iremos mostrar como utilizar AR para trabalhar com relacionamentos. Para simplificar, utilizaremos a seguinte tabela para os exemplos desta seção. Note que, se você estiver utilizando um banco de dados MySQL, você deve substituir o `AUTOINCREMENT` por `AUTO_INCREMENT` na instrução abaixo:

```
CREATE TABLE Post (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    title VARCHAR(128) NOT NULL,
    content TEXT NOT NULL,
    createTime INTEGER NOT NULL
);
```

> **Nota:** A intenção do AR não é resolver todas tarefas relacionadas a banco de dados. Ele é melhor utilizado para modelar tabelas do banco para estruturas no PHP e executar consultas que não envolvem instruções SQL complexas. O DAO do Yii é o recomendado para esses cenários mais complexos.

1. Estabelecendo uma Conexão com o Bando de Dados
O AR precisa de uma conexão com o banco para executar suas operações. Por padrão, assume-se que o componente de aplicação db possui uma instância da classe CDbConnection que irá servir esta conexão. Abaixo temos um exemplo da configuração de uma aplicação:

```
return array(
    'components'=>array(
        'db'=>array(
            'class'=>'system.db.CDbConnection',
            'connectionString'=>'sqlite:path/to/dbfile',
            // habilita o cache de schema para aumentar a performance
            // 'schemaCachingDuration'=>3600,
        ),
    ),
);
```

> **Dica:** Como o Active Record depende de metadados sobre as tabelas para determinar informações sobre as colunas, gasta-se tempo lendo esses dados e os analisando. Se o schema do seu banco de dados não irá sofrer alterações, é interessante que você ative o caching de schema, configurando a propriedade CDbConnection::schemaCachingDuration para um valor maior de que 0.

O suporte para AR é limitado pelo Sistema de Gerenciamento de Banco de Dados. Atualmente, somente os seguintes SGBDs são suportados:

- MySQL 4.1 ou maior
- PostgreSQL 7.3 ou maior
- SQLite 2 e 3
- Microsoft SQL Server 2000 ou maior
- Oracle

> **Nota:** O suporte ao Microsoft SQL Server existe desde a versão 1.0.4; já o suporte ao Oracle está disponível a partir da versão 1.0.5.

Se você deseja utilizar um componente de aplicação diferente de db, ou se quiser trabalhar com vários bancos de dados utilizando AR, você deve sobrescrever o método CActiveRecord::getDbConnection(). A classe CActiveRecord é a base para todas as classes Active Record.

> **Dica:** Existem duas maneiras de trabalhar como AR utilizando vários bancos de dados. Se os schemas dos bancos são diferentes, você deve criar diferentes classes base AR, com diferentes implementações do método getDbConnection(). Caso contrário, alterar dinamicamente a variável estática CActiveRecord::db é uma idéia melhor.

2. Definindo Classes AR
Para acessar uma tabela do banco de dados, precisamos primeiro definir uma classe AR estendendo CActiveRecord. Cada classe Active Record representa uma única tabela do banco, e uma instância dessa classe representa um registro dessa tabela. O exemplo abaixo mostra o código mínimo necessário para uma classe AR que representa a tabela `Post`:

```
class Post extends CActiveRecord
```

```
{
    public static function model($className=__CLASS__)
    {
        return parent::model($className);
    }
}
```

> **Dica:** Como as classes Ar geralmente são utilizadas em diversos lugares, podemos importar todo o diretório onde elas estão localizadas, em vez de fazer a importação uma a uma. Por exemplo, se todos os arquivos de nossas classes estão em `protected/models`, podemos configurar a aplicação da seguinte maneira:
>
> ```
> return array(
>   'import'=>array(
>       'application.models.*',
>   ),
> );
> ```

Por padrão, o nome de uma classe AR é o mesmo que o da tabela do banco de dados. Sobrescreva o método tableName() caso eles sejam diferentes. O método model() deve ser declarado dessa maneira para todos as classes AR (a ser explicado em breve).

Os valores do registro de um tabela podem ser acessados pelas propriedades da instância AR correspondente. Por exemplo, o código a seguir adiciona um valor ao campo `title`:

```
$post=new Post;
$post->title='um post de exemplo';
```

Embora nunca tenhamos declarado uma propriedade `title` na classe `Post`, ainda assim podemos acessa-la no exemplo acima. Isso acontece porque `title` é uma coluna da tabela `Post`, e a classe CActiveRecord a deixa acessível por meio de uma propriedade com a ajuda do método mágico `__get()`, do PHP. Ao tentar acessar uma coluna que não existe na tabela será disparada uma exceção.

> **Informação:** Nesse guia, nomeamos as colunas utilizando o estilo camel case (por exemplo, `createTime`). Isso acontece porque acessamos essas colunas através de propriedades de objetos que também utilizam esse estilo para nomeá-las. Embora a utilização de camel case faça nosso código ter uma nomenclatura mais consistente, ele adiciona um problema relacionado aos bancos de dados que diferenciam letras maiúsculas de minúsculas. Por exemplo, o PostgreSQL, por padrão, não faz essa diferenciação nos nomes das colunas, e devemos colocar o nome da coluna entre aspas, em uma consulta, se seu nome conter letras maiúsculas e minúsculas. Por isso, é uma boa idéia nomear as colunas (e as tabelas também) somente com letras minúsculas (por exemplo, `create_time`) para evitar esse tipo de problema.

3. Criando um Registro

Para inserir um novo registro em uma tabela, criamos uma nova instância da classe AR correspondente, inserimos os valores nas propriedades relacionadas as colunas da tabela e, então, utilizamos o método save() para concluir a inserção.

```
$post=new Post;
$post->title='post de exemplo';
```

```
$post->content='conteúdo do post de exemplo';
$post->createTime=time();
$post->save();
```

Se a chave primário da tabela é auto-numérica, após a inserção, a instância da classe AR irá conter o valor atualizado da chave primária. No exemplo acima, a propriedade `id` irá refletir o valor da chave primária no novo post inserido, mesmo que não a tenhamos alterado explicitamente.

Se alguma coluna é definida com um valor padrão estático (por exemplo, uma string ou um número) no schema da tabela, a propriedade correspondente na instância AR terá, automaticamente, este valor, assim que criada. Uma maneira de alterar esse valor padrão é declarar explicitamente a propriedade na classe AR:

```
class Post extends CActiveRecord
{
    public $title='por favor insira um título';
    ......
}

$post=new Post;
echo $post->title;  // irá exibir: por favor insira um título
```

A partir da versão 1.0.2, pode-se atribuir a um atributo um valor do tipo CDbExpression, antes que o registro seja salvo (tanto na inserção, quanto na atualização) no banco de dados. Por exemplo, para salvar um timestamp retornado pela função `NOW()` do MySQL, podemos utilizar o seguinte código:

```
$post=new Post;
$post->createTime=new CDbExpression('NOW()');
// $post->createTime='NOW()'; não irá funcionar porque
// 'NOW()' será tratado como uma string
$post->save();
```

> **Dica:** Embora o Active Record torne possível que sejam realizadas operações no banco de dados sem a necessidade de escrever consultas em SQL, geralmente queremos saber quais consultas estão sendo executadas pelo AR. Para isso, ative o recurso de registros de logs do Yii. Por exemplo, podemos ativar o componente CWebLogRoute na configuração da aplicação, e, então poderemos ver as instruções SQL executadas exibidas no final de cada página. Desde a versão 1.0.5, podemos alterar o valor da propriedade CDbConnection::enableParamLogging para true, na configuração da aplicação, assim os valores dos parâmetros vinculados a instrução também serão registrados.

4. Lendo um Registro

Para ler dados de uma tabela, podemos utilizar um dos métodos `find`:

```
// encontra o primeiro registro que atenda a condição especificada
$post=Post::model()->find($condition,$params);
// encontra o registro com a chave primária especificada
$post=Post::model()->findByPk($postID,$condition,$params);
// encontra o registro com os atributos tendo os valores especificados
$post=Post::model()->findByAttributes($attributes,$condition,$params);
// encontra o primeiro registro, utilizando o comando SQL especificado
$post=Post::model()->findBySql($sql,$params);
```

No exemplo acima, utilizamos o método `find` em conjunto com `Post::model()`. Lembre-se que o método estático `model()` é obrigatório em todas as classes AR. Esse método retorna uma instância AR que é utilizada para acessar métodos a nível de classe (algo parecido com métodos estáticos de classe) em um contexto de objeto.

Se o método `find` encontra um registro que satisfaça as condições da consulta, ele irá retornar uma instância cujas propriedades irão conter os valores do registro específico. Podemos então ler os valores carregados normalmente como fazemos com as propriedades de um objeto. Por exemplo, `echo $post->title;`.

O método `find` irá retornar null se nenhum registro for encontrado.

Ao executar o método `find`, utilizamos os parâmetros `$condition` e `$params` para especificar as condições desejadas. Nesse caso, `$condition` pode ser uma string representando uma cláusula WHERE, do SQL, e `$params` é um vetor com parâmetros cujos valores devem ser vinculados a marcadores na `$condition`. Por exemplo:

```
// encontra o registro com postID=10
$post=Post::model()->find('postID=:postID', array(':postID'=>10));
```

> **Nota:** No exemplo acima, precisamos escapar a referência para a coluna `postID`, em certos SGBDs. Por exemplo, se estivermos utilizando o PostgreSQL, deveríamos ter escrito a condição como `"postID"=:postID`, porque este banco de dados, por padrão, não diferencia letras maiúsculas e minúsculas nos nomes de colunas.

Podemos também utilizar o parâmetro `$condition` para especificar condições de pesquisa mais complexas. Em vez de uma string, `$condition` pode ser uma instância de CDbCriteria, o que permite especificar outras condições além do WHERE. Por exemplo:

```
$criteria=new CDbCriteria;
$criteria->select='title';  // seleciona apenas a coluna title
$criteria->condition='postID=:postID';
$criteria->params=array(':postID'=>10);
$post=Post::model()->find($criteria); // $params não é necessario
```

Note que, ao utilizar CDbCriteria como condição para a pesquisa, o parâmetro `$params` não é mais necessário, uma vez que ele pode ser especificado diretamente na instância de CDbCriteria, como no exemplo acima.

Uma maneira alternativa de utilizar CDbCriteria é passar um vetor para o método `find`. As chaves e valores do vetor correspondem as propriedades e valores da condição, respectivamente. O exemplo acima pode ser reescrito da seguinte maneira:

```
$post=Post::model()->find(array(
    'select'=>'title',
    'condition'=>'postID=:postID',
    'params'=>array(':postID'=>10),
));
```

> **Informação:** Quando a condição de uma consulta deve casar com colunas com um determinado valor, utilizamos o método findByAttributes(). Fazemos com que o parâmetro `$attributes` seja um vetor, onde os atributos são indexados pelos nomes das colunas. Em alguns frameworks, essa tarefa é feita utilizando-se métodos como `findByNameAndTitle`. Apesar de parecer uma maneira mais atrativa, normalmente esses métodos causam confusão, conflitos

e problemas em relação aos nomes de colunas com maiúsculas e minúsculas.

Quando uma condição encontra diversos resultados em uma consulta, podemos traze-los todos de uma vez utilizando os seguintes métodos `findAll`, cada um com sua contraparte na forma de métodos `find`, como já descrito.

```
// encontra todos os registros que satisfação a condição informada
$posts=Post::model()->findAll($condition,$params);
// encontra todos os registros com a chave primária informada
$posts=Post::model()->findAllByPk($postIDs,$condition,$params);
// encontra todos os registros com campos com o valor informado
$posts=Post::model()->findAllByAttributes($attributes,$condition,$params);
// encontra todos os registros utilizando a consulta SQL informada
$posts=Post::model()->findAllBySql($sql,$params);
```

Se nenhum registro for encontrada, `findAll` irá retornar um vetor vazio, diferente dos métodos `find` que retornam null quando nada é encontrado.

Em conjunto com os métodos `find` e `findAll`, já descritos, os seguintes métodos também são fornecidos:

```
// pega o número de registros que satisfaz a condição informada
$n=Post::model()->count($condition,$params);
// pega o número de registros que satisfaz a instrução SQL informada
$n=Post::model()->countBySql($sql,$params);
// verifica se há pelo menos um registro que satisfaz a condição informada
$exists=Post::model()->exists($condition,$params);
```

5. Atualizando Registros
Depois que uma instância AR tenha sido preenchida com os valores dos campos da tabela, podemos atualiza-los e salva-los de volta para o banco de dados.

```
$post=Post::model()->findByPk(10);
$post->title='novo título do post';
$post->save(); // salva as alterações para o banco de dados
```

Como podemos ver, utilizamos o mesmo método save() para fazer a inserção e atualização dos dados. Se uma instância AR é criada por meio do operador `new`, executar o método save() irá inserir um novo registro no banco de dados; se a instância é o resultado de um `find` ou `findAll`, executar o método save() irá atualizar o registro existente na tabela. Podemos utilizar a propriedade CActiveRecord::isNewRecord para verificar se uma instância AR é nova ou não.

Também é possível atualizar um ou vários registros em uma tabela do banco, sem ter que carrega-lo primeiro. Existem os seguinte métodos para efetuar essas operações de uma maneira mais conveniente:

```
// atualiza os registros que satisfação a condição informada
Post::model()->updateAll($attributes,$condition,$params);
// atualiza os registros que tenha a chave primária informada, e satisfação a
condição
Post::model()->updateByPk($pk,$attributes,$condition,$params);
// atualiza uma coluna counter (contagem) que satisfaça a condição informada
Post::model()->updateCounters($counters,$condition,$params);
```

No exemplo acima, `$attributes` é um vetor com os valores das colunas, indexados pelos nomes delas. `$counter` é um vetor com as colunas que terão seus valores

incrementados, indexadas pelos seus nomes. `$condition` e `$paramns` estão descritos nos itens anteriores.

6. Excluindo um Registro
Podemos também excluir um registro se a instância AR já estiver preenchida com ele.

```
$post=Post::model()->findByPk(10); // assumindo que há um post com ID 10
$post->delete(); // exclui o registro da tabela no banco de dados.
```

Note que, depois da exclusão, a instância AR continua inalterada, mas o registro correspondente no banco de dados já foi excluído.

Os seguintes métodos são utilizados para excluir registros sem a necessidade de carrega-los primeiro:

```
// exclui os registros que satisfação a condição informada
Post::model()->deleteAll($condition,$params);
// exclui os registros com a chave primária e condição informada
Post::model()->deleteByPk($pk,$condition,$params);
```

7. Validação de Dados
Ao inserir ou atualizar um registro, geralmente precisamos verificar ser o valor está de acordo com certas regras. Isso é especialmente importante nos casos em que os valores das colunas são informados pelos usuários. No geral, é bom nunca confiar em nenhum dado vindo do lado do cliente (usuário).

O AR efetua a validação automaticamente quando o método save() é executado. A validação é baseada em regras especificadas pelo método rules() da classe AR. Para mais detalhes sobre como especificar regras de validação consulte Declarando Regras de Validação. Abaixo temos o fluxo típico necessário para salvar um registro:

```
if($post->save())
{
    // dados são validos e são inseridos/atualizados no banco
}
else
{
    // dados são inválidos. utilize getErrors() para recuperar as mensagens de
erro
}
```

Quando os dados para inserção ou atualização são enviados pelos usuários através de um formulário HTML, precisamos atribuí-los as propriedades correspondentes da classe AR. Podemos fazer isso da seguinte maneira:

```
$post->title=$_POST['title'];
$post->content=$_POST['content'];
$post->save();
```

Se existirem muitos campos, teríamos uma longa lista dessas atribuições. Esse trabalho pode ser aliviado, por meio da propriedade attributes, como feito no exemplo abaixo. Mais detalhes podem ser consultados em Atribuição de Atributos Seguros e Criando uma Ação.

```
// assumindo que $_POST['Post'] é um vetor com os valores das colunas, indexados
pelos seus nomes
$post->attributes=$_POST['Post'];
$post->save();
```

8. Comparando Registros
Assim como registros de uma tabela, as instâncias AR também são unicamente identificadas pelos valores de suas chaves primárias. Portanto, para comparar duas instâncias AR, precisamos apenas comparar os valores de suas chaves, assumindo que ambas pertencem a mesma classe. Entretanto, existe uma maneira mais simples de compara-las, que é utilizar o método CActiveRecord::equals().

> **Informação:** Diferente das implementações de Active Record em outros frameworks, o Yii suporta chaves primárias compostas em seu AR. Uma chave primária composta é formada por duas ou mais colunas. De forma correspondente, a chave primária é representada por um vetor no Yii. A propriedade primaryKey retorna a chave uma instância AR.

9. Personalização
A classe CActiveRecord possui alguns métodos que podem ser sobrescritos por suas classes derivadas, para personalizar seu fluxo de funcionamento.

- beforeValidate e afterValidate: esses métodos são executados antes e depois que uma validação é executada
- beforeSave e afterSave: esses métodos são executados antes e depois que um registro é salvo.
- beforeDelete e afterDelete: esses métodos são executados antes e depois que uma instância AR é excluída.
- afterConstruct: esse método é utilizado para toda instância AR criada por meio do operador new.
- beforeFind: esse método é chamado antes que um objeto AR finder seja utilizado para executar uma consulta (por exemplo, `find()`, `findAll()`). Ele está disponível a partir da versão 1.0.9.
- afterFind: esse método é chamado após cada instância AR criada como resultado de um consulta.

10. Utilizando Transações com AR
Todas as instâncias AR contém uma propriedade chamada dbConnection que é uma instância da classe CDbConnection. Podemos então, utilizar o recurso de transações existente no DAO do Yii para trabalhar com Active Record.

```
$model=Post::model();
$transaction=$model->dbConnection->beginTransaction();
try
{
    // find e save são dois passos que podem ser interrompidos por outra
requisição
    // portanto utilizamos uma transação para garantir e a consistência a
integridade dos dados
    $post=$model->findByPk(10);
    $post->title='novo título para o post';
    $post->save();
    $transaction->commit();
}
catch(Exception $e)
{
    $transaction->rollBack();
```

```
}
```

11. Named Scopes (Escopos com Nomes)

> **Nota:** O suporte a named scopes está disponível a partir da versão 1.0.5. A idéia original dos named scopes veio do Ruby on Rails.

Um *named scope* representa um critério de consulta com um nome, que pode ser combinado com outros named scopes e ser aplicado em uma consulta com active record.

Named scopes são declarados, normalmente, dentro do método CActiveRecord::scopes(), como pares nome-critério. O código a seguir, declara dois named scopes, `published` e `recently`, dentro da classe `Post`:

```
class Post extends CActiveRecord
{
    ......
    public function scopes()
    {
        return array(
            'published'=>array(
                'condition'=>'status=1',
            ),
            'recently'=>array(
                'order'=>'createTime DESC',
                'limit'=>5,
            ),
        );
    }
}
```

Cada named scope é declarado como um vetor que pode ser utilizado para iniciar uma instância da classe CDbCriteria. Por exemplo, o named scope `recently` especifica que o valor da propriedade `order` seja `createTime DESC` e o da propriedade `limit` seja 5, que será traduzido em um critério de consulta que retornará os 5 posts mais recentes.

Na maioria das vezes, named scopes são utilizados como modificadores nas chamadas aos métodos `find`. Vários named scopes podem ser encadeados para gerar um conjunto de resultados mais restrito. Por exemplo, para encontrar os posts publicados recentemente, podemos fazer como no código abaixo:

```
$posts=Post::model()->published()->recently()->findAll();
```

Geralmente, os named scopes aparecem a esquerda da chamada ao método `find`. Então, cada um deles fornece um critério de pesquisa que é combinado com outros critérios, incluindo o que é passado para o método `find`. Esse encadeamento é como adicionar uma lista de filtros em um consulta.

A partir da versão 1.0.6, named scopes também podem ser utilizados com os métodos `update` e `delete`. Por exemplo, no código a seguir vemos como deletar todos os posts publicados recentemente:

```
Post::model()->published()->recently()->delete();
```

> **Nota:** Named scopes podem ser utilizados somente como métodos a nível de classe. Por esse motivo, o método deve ser executando utilizando `NomeDaClasse::model()`.

Named Scopes Parametrizados
Named scopes podem ser parametrizados. Por exemplo, podemos querer personalizar o número de posts retornados no named scope `recently`, Para isso, em vez de declarar o named scope dentro do método CActiveRecord::scopes, precisamos definir um novo método cujo nome seja o mesmo do escopo:

```
public function recently($limit=5)
{
    $this->getDbCriteria()->mergeWith(array(
        'order'=>'createTime DESC',
        'limit'=>$limit,
    ));
    return $this;
}
```

Então, para recuperar apenas os 3 posts publicados recentemente, utilizamos:

```
$posts=Post::model()->published()->recently(3)->findAll();
```

Se não tivéssemos informado o parâmetro 3 no exemplo acima, iriamos recuperar 5 posts, que é a quantidade padrão definida no método.

Named Scope Padrão
A classe de um modelo pode ter um named scope padrão, que é aplicado para todas as suas consultas (incluindo as relacionais). Por exemplo, um website que suporte vários idiomas, pode querer exibir seu conteúdo somente no idioma que o usuário especificar. Como devem existir muitas consultas para recuperar esse conteúdo, podemos definir um named scope para resolver esse problema. Para isso sobrescrevemos o método CActiveRecord::defaultScope, como no código a seguir:

```
class Content extends CActiveRecord
{
    public function defaultScope()
    {
        return array(
            'condition'=>"idioma='".Yii::app()->idioma."'",
        );
    }
}
```

Assim, a chamada de método a seguir irá utilizar automaticamente o critério definido acima:

```
$contents=Content::model()->findAll();
```

Note que o named scope padrão é aplicado somente as consultas utilizando `SELECT`. Ele é ignorado nas consultas com `INSERT`, `UPDATE` e `DELETE`.

3-MySQL

TUTORIAL RÁPIDO SOBRE O MYSQL

Instalação
Instale com o pacote Xampp (xampp.sf.net)

Executando

Uma boa opção de administração do MySQL é o phpmyadmin, que também acompanha o Xampp.

Para administração pela linha de comando use:

mysql -h host -u user -p (o super usuário default é root)
mysql -u root (quando estiver sem senha)

TROCANDO SENHA DO USUÁRIO ROOT

mysql -u root teste (Usuário root acessar banco teste)
use mysql;
UPDATE user SET Password=PASSWORD("novasenha") WHERE user="root";
FLUSH PRIVILEGES;

Ou

mysql -u root clientes
SET PASSWORD FOR root=PASSWORD('senhadoroot');

CRIANDO USUÁRIOS

mysql --user=root mysql
GRANT ALL PRIVILEGES ON *.* TO super@localhost
IDENTIFIED BY 'senha' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO super@"%"
IDENTIFIED BY 'some_pass' WITH GRANT OPTION;

super - é um total super usuário que pode se conectar no localhost e
de qualquer lugar ("%"), mas precisa usar senha

GRANT RELOAD,PROCESS ON *.* TO admin@localhost;

admin - usuário que pode se conectar no localhost sem senha.
Pode executar os comandos mysqladmin reload, mysqladmin refresh, and mysqladmin flush-*
e mysqladmin processlist . Não tem nenhum privilégio relacionado aos bancos.

GRANT USAGE ON *.* TO fraco@localhost;

fraco - pode conectar somente via localhost sem senha mas sem privilégios, somente para uso.

Exemplo:
GRANT ALL PRIVILEGES ON *.* TO ribafs@localhost IDENTIFIED BY 'ribafs' WITH GRANT OPTION;
mysql -u ribafs // Dá erro de senha

mysql -u ribafs -p //Funciona após entrar a senha ribafs

REMOVENDO USUÁRIOS

DROP USER nomeusuario;

PRIVILÉGIOS

REVOKE GRANT ALL ON nomebancooutabelaou*ou*.* FROM nomeusuario
* - todas as tabelas
. todos os bancos e todas as tabelas
banco.* - todas as tabelas do banco

GRANT SELECT,INSERT,UPDATE ON nomebanco.* TO nomeuser;

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON nomebanco.* TO usuario@localhost
IDENTIFIED BY 'senha';

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON nomebanco.* TO usuario@dominio.com.br
IDENTIFIED BY 'senha';

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON nomebanco.* TO usuario@'%'
IDENTIFIED BY 'senha';

INSTALAR COMO SERVIÇO

Instalar MySQL como serviço no Windows para trabalhar com Java (J2EE):
mysqld-nt --install --ansi --sql-mode=ANSI_QUOTES

Instalar como serviço:
bin\mysqld-nt --install mysql

Remover o serviço:
bin\mysqld --remove mysql

Remover serviço ansi:
bin\mysqld --remove --ansi

CRIAR TABELAS COM RELACIONAMENTOS

```
create table produto(
	codigo int not null primary key,
	nome varchar(50) not null unique,
	descricao varchar(200),
	valor real(6, 2)
) ENGINE=INNODB;

create table cliente(
	codigo int not null primary key,
	nome varchar(50) not null,
	email varchar(100) not null unique,
	cpf varchar(11) not null
) ENGINE=INNODB;

create table pedido(
	numero int not null primary key auto_increment,
	codigocliente int not null references cliente(codigo),
	valortotal real(7,2) DEFAULT '0.00' NOT NULL
) ENGINE=INNODB;

create table item(
	numeropedido int not null references pedido(numero),
	codigoproduto int not null references produto(codigo),
	quantidade int not null,
	primary key(numeropedido, codigoproduto)
) ENGINE=INNODB;

CREATE TABLE product (
	category INT NOT NULL,
	id INT NOT NULL,
	price DECIMAL,
	PRIMARY KEY(category, id)
) ENGINE=INNODB;

CREATE TABLE product_order (
	no INT NOT NULL AUTO_INCREMENT,
	product_category INT NOT NULL,
	product_id INT NOT NULL,
	customer_id INT NOT NULL,
	PRIMARY KEY(no),
```

```
        INDEX (product_category, product_id),
        FOREIGN KEY (product_category, product_id)
        REFERENCES product(category, id) ON UPDATE CASCADE ON DELETE
RESTRICT,
        INDEX (customer_id),
        FOREIGN KEY (customer_id)
        REFERENCES customer(id)
) ENGINE=INNODB;
```

O tipo InnoDb dá suporte à constraint Foreign Key (references).

REMOVER SERVIÇO NO WINDOWS NT/XP

```
mysql\bin\mysqld -- remove         (remove o serviço mysql)
                 -- remove --ansi  (remover o serviço ansi)
```

RESUMO DE USO

1) mysql -u root -p ou mysql -u root

mysql -h host -u user -p banco

Obs: Caso receba a mensagem: Can't connect to MySQL server on 'localhost'
Falta startar o MySQL

2) create database nomebanco;

3) use nomebanco;

4) create table nometabela(campos tipos...);

5) select * from nometabela;

6) show databases;

7) show tables;

8) describe nometabela;

IMPORTAR E EXPORTAR

Exportando:
bin\mysqldump -u user -p passwd banco > banco.sql

Importando:

bin\mysql -u user -p password banco < banco.sql

Mudar Conjunto de Caracters para LATIN1

musql -u root
\C latin1

POPULANDO TABELAS APÓS A CRIAÇÃO

O comando LOAD DATA pode ser utilizado para popular tabelas, trazendo de arquivos:

LOAD DATA LOCAL INFILE '/path/arquivo.txt' INTO TABLE nometabela;

SELECT DATABASE();

SHOW CHARACTER SET;

CREATE DATABASE db_name
[[DEFAULT] CHARACTER SET charset_name]
[[DEFAULT] COLLATE collation_name]
ALTER DATABASE db_name
[[DEFAULT] CHARACTER SET charset_name]
[[DEFAULT] COLLATE collation_name]

CREATE TABLE tbl_name (column_list)
[[DEFAULT] CHARACTER SET charset_name] [COLLATE collation_name]]
ALTER TABLE tbl_name
[[DEFAULT] CHARACTER SET charset_name] [COLLATE collation_name]
Example:
CREATE TABLE t1 (...) CHARACTER SET latin1 COLLATE latin1_danish_ci;

col_name {CHAR | VARCHAR | TEXT} (col_length)
[CHARACTER SET charset_name] [COLLATE collation_name]

Example:
CREATE TABLE Table1
(
column1 VARCHAR(5) CHARACTER SET latin1 COLLATE latin1_german1_ci
);

FUNÇÕES COM DATAS

DATE_SUB
SELECT something FROM tbl_name WHERE DATE_SUB(CURDATE(),INTERVAL 30 DAY) <= date_col;
SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');

DATE_ADD
SELECT DATE_ADD('2006-05-00',INTERVAL 1 DAY);

SELECT CURDATE();

SELECT CURTIME();

DATE_FORMAT
SELECT date_format('2006-04-30', '%d/%m/%Y'); -- 30/04/2006
SELECT DATE_FORMAT('2003-10-03',GET_FORMAT(DATE,'EUR')); -- 03.10.2003
SELECT DATE_FORMAT('2006-06-00', '%d/%m/%Y');

SELECT NOW();

SELECT TO_DAYS('1997-10-07'); -- RETORNA DIAS

SELECT YEAR('2000-01-01');

Transações no MySQL

Alterar o /etc/mysql/my.cnf, comentando as linhas:

skip-bdb
skip-innodb

E adicionando as linhas abaixo na seção [mysqld]:

default-table-type=innodb

Então reiniciar o mysql.

Iniciar uma trabsação:

set autocommit=0;
start transaction; -- Para versões anteriores a 4 usa-se, ao invés, begin;

insert into conteudo values (21, 'teste','Teste', 'TESTE');

commit; -- Caso queira confirmar
rollback; -- Caso queira cancelar os comandos

Introdução ao MySQL

Wikipédia - http://pt.wikipedia.org/wiki/MySQL
O **MySQL** é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês *Structured Query Language*) como interface. É atualmente um dos bancos de dados mais populares, com mais de 10 milhões de instalações pelo mundo.

Entre os usuários do banco de dados MySQL estão: NASA, Friendster, Banco Bradesco, Dataprev, HP, Nokia, Sony, Lufthansa, U.S Army, US. Federal Reserve Bank, Associated Press, Alcatel, Slashdot, Cisco Systems, CanaVialis S.A. e outros.

História

O **MySQL** foi criado na Suécia por dois suecos e um finlandês: David Axmark, Allan Larsson e Michael "Monty" Widenius, que têm trabalhado juntos desde a década de 1980. Hoje seu desenvolvimento e manutenção empregam aproximadamente 400 profissionais no mundo inteiro, e mais de mil contribuem testando o software, integrando-o a outros produtos, e escrevendo a respeito dele.

No dia 16 de Janeiro de 2008, a MySQL AB, desenvolvedora do MySQL foi adquirida pela Sun Microsystems, por US$ 1 bilhão, um preço jamais visto no setor de licenças livres. No dia 20 de Abril de 2009 a Oracle compra a Sun Microsystems e todos o seu produtos, incluindo o MySQL.

O sucesso do MySQL deve-se em grande medida à fácil integração com o PHP incluído, quase que obrigatoriamente, nos pacotes de hospedagem de sites da Internet oferecidos atualmente. Empresas como Yahoo! Finance, MP3.com, Motorola, NASA, Silicon Graphics e Texas Instruments usam o MySQL em aplicações de missão crítica[3]. A Wikipédia é um exemplo de utilização do MySQL em sites de grande audiência.

O **MySQL** hoje suporta Unicode, Full Text Indexes, replicação, Hot Backup, GIS, OLAP e muitos outros recursos.

Aprendendo a instalar e configurar o phpMyAdmin

Introdução

O que é o phpMyAdmin?

phpMyAdmin é pretendido para administração do MySQL sobre aWeb.Com o phpMyAdmin instalado em seu servidor você podera críar banco de dados, copiar tabelas, criar, apagar e editar. Facil configuração e administração.

O que é necessario para instalar o phpMyAdmin?

mySQL: www.mysql.com

Apache: www.apache.org

PHP: www.php.net

Instalando

Entre na url www.phpwizard.net e entre na seção phpMyAdmin, você podera optar por duas opções do phpMyAdmin:

1-) Extensão *.php3

2-) Extensão *.php

Aconselho a pega o phpMyAdmin de extensão *.php, porque a maioria dos servidores reconhece essa extensão.

Configurando

Após fazer o download descompacte o mesmo e copie todos os arquivos para uma pasta no seu servidor ex: mysql e edite o arquivo config.inc

$cfgServers[1]['host'] = 'localhost'; // MySQL hostname
mantenha localhost

$cfgServers[1]['port'] = ''; // MySQL port - leave blank for default port
mantenha

$cfgServers[1]['adv_auth'] = true; // Use advanced authentication?
aconselho a deixar true pois assim ao acessar a url no browser ele aparecera uma caixa de login e senha do usuário, para sua melhor segurança. caso deixe false ele não aparecera nada e entrara direto.

$cfgServers[1]['stduser'] = 'usuário'; // MySQL standard user (only needed with advanced auth)
nome do usuário

$cfgServers[1]['stdpass'] = 'senha'; // MySQL standard password (only needed with advanced auth)
senha

$cfgServers[1]['user'] = 'usuário'; // MySQL user (only needed with basic auth)
nome do usuário

$cfgServers[1]['password'] = 'senha'; // MySQL password (only needed with basic auth)
senha

Pronto a configuração basica do phpMyAdmin já esta pronta.

Agora é só acessar via browser seu phpMyAdmin ex: www.seusite.com.br/mysql/
Qualquer duvida ou comentário, poste uma mensagem no forum.
Tutorial feito por:
Eduardo Luis de Medeiros - eduardo@superphp.com.br

Introdução ao MySQL

Teste de SQL Online
http://www.w3schools.com/sql/default.asp
http://www.w3schools.com/sql/sql_tryit.asp

Instalação
Sugestão: Instale com o pacote Xampp (http://xampp.sf.net)

Executando
Uma boa opção de administração do MySQL é o phpmyadmin, que também acompanha o Xampp.

Administrando pela linha de comando

mysql -h host -u root -p (o super usuário default é root)
mysql -u root (sintaxe para acessar o MySQL local e quando estiver sem senha)

Trocando a senha do Usuário root
mysql -u root teste (Usuário root acessar banco teste)
use mysql;
UPDATE user SET Password=PASSWORD("novasenha") WHERE user="root";
FLUSH PRIVILEGES;

Ou

mysql -u root clientes
SET PASSWORD FOR root=PASSWORD('senhadoroot');

Criando Usuários
mysql --user=root mysql
GRANT ALL PRIVILEGES ON *.* TO super@localhost
IDENTIFIED BY 'senha' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO super@"%"
IDENTIFIED BY 'some_pass' WITH GRANT OPTION;

super - é um total super usuário que pode se conectar no localhost e
de qualquer lugar ("%"), mas precisa usar senha

GRANT RELOAD,PROCESS ON *.* TO admin@localhost;

admin - usuário que pode se conectar no localhost sem senha.
Pode executar os comandos mysqladmin reload, mysqladmin refresh, and mysqladmin flush-*
e mysqladmin processlist . Não tem nenhum privilégio relacionado aos bancos.

GRANT USAGE ON *.* TO fraco@localhost;

fraco - pode conectar somente via localhost sem senha mas sem privilégios, somente para uso.

Exemplo:
GRANT ALL PRIVILEGES ON *.* TO ribafs@localhost IDENTIFIED BY 'ribafs' WITH GRANT OPTION;
mysql -u ribafs // Dá erro de senha

mysql -u ribafs -p //Funciona após entrar a senha ribafs

Removendo Usuários
DROP USER nomeusuario;

Privilégios
REVOKE GRANT ALL ON nomebancooutabelaou*ou*.* FROM nomeusuario
* - todas as tabelas
. todos os bancos e todas as tabelas
banco.* - todas as tabelas do banco

GRANT SELECT,INSERT,UPDATE ON nomebanco.* TO nomeuser;

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON nomebanco.* TO usuario@localhost
IDENTIFIED BY 'senha';

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON nomebanco.* TO usuario@dominio.com.br
IDENTIFIED BY 'senha';

GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP ON nomebanco.* TO usuario@'%'
IDENTIFIED BY 'senha';

Remover o serviço:
bin\mysqld --remove mysql

Criar Tabelas Relacionadas

```
create table produto(
        codigo int not null primary key,
        nome varchar(50) not null unique,
        descricao varchar(200),
        valor real(6, 2)
) ENGINE=INNODB;
```

```
create table cliente(
        codigo int not null primary key,
        nome varchar(50) not null,
        email varchar(100) not null unique,
        cpf varchar(11) not null
) ENGINE=INNODB;

create table pedido(
        numero int not null primary key auto_increment,
        codigocliente int not null references cliente(codigo),
        valortotal real(7,2) DEFAULT '0.00' NOT NULL
) ENGINE=INNODB;

create table item(
        numeropedido int not null references pedido(numero),
        codigoproduto int not null references produto(codigo),
        quantidade int not null,
        primary key(numeropedido, codigoproduto)
) ENGINE=INNODB;

CREATE TABLE product (
        category INT NOT NULL,
        id INT NOT NULL,
        price DECIMAL,
        PRIMARY KEY(category, id)
) ENGINE=INNODB;

CREATE TABLE product_order (
        no INT NOT NULL AUTO_INCREMENT,
        product_category INT NOT NULL,
        product_id INT NOT NULL,
        customer_id INT NOT NULL,
        PRIMARY KEY(no),
        INDEX (product_category, product_id),
        FOREIGN KEY (product_category, product_id)
        REFERENCES product(category, id) ON UPDATE CASCADE ON DELETE
RESTRICT,
        INDEX (customer_id),
        FOREIGN KEY (customer_id)
        REFERENCES customer(id)
) ENGINE=INNODB;
```

O tipo InnoDb dá suporte à constraint Foreign Key (references).

Resumo de Uso

1) mysql -u root -p ou mysql -u root

mysql -h host -u user -p banco

Obs: Caso receba a mensagem: Can't connect to MySQL server on 'localhost'
Falta startar o MySQL

2) create database nomebanco;

3) use nomebanco;

4) create table nometabela(campos tipos...);

5) select * from nometabela;

6) show databases;

7) show tables;

8) describe nometabela;

Importar e Exportar

Exportando:
bin\mysqldump -u user -p passwd banco > banco.sql

Importando:
bin\mysql -u user -p password banco < banco.sql

Mudar Conjunto de Caracters para LATIN1

musql -u root
\C latin1

Populando Tabelas após a Criação

O comando LOAD DATA pode ser utilizado para popular tabelas, trazendo de arquivos:

LOAD DATA LOCAL INFILE '/path/arquivo.txt' INTO TABLE nometabela;

SELECT DATABASE();

SHOW CHARACTER SET;

```
CREATE DATABASE db_name
[[DEFAULT] CHARACTER SET charset_name]
[[DEFAULT] COLLATE collation_name]
ALTER DATABASE db_name
[[DEFAULT] CHARACTER SET charset_name]
[[DEFAULT] COLLATE collation_name]

CREATE TABLE tbl_name (column_list)
[[DEFAULT] CHARACTER SET charset_name] [COLLATE collation_name]]
ALTER TABLE tbl_name
[[DEFAULT] CHARACTER SET charset_name] [COLLATE collation_name]
Example:
CREATE TABLE t1 ( ... ) CHARACTER SET latin1 COLLATE latin1_danish_ci;

col_name {CHAR | VARCHAR | TEXT} (col_length)
[CHARACTER SET charset_name] [COLLATE collation_name]

Example:
CREATE TABLE Table1
(
column1 VARCHAR(5) CHARACTER SET latin1 COLLATE latin1_german1_ci
);
```

Funções com Datas

```
DATE_SUB
SELECT something FROM tbl_name WHERE DATE_SUB(CURDATE(),INTERVAL 30
DAY) <= date_col;
SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');

DATE_ADD
SELECT DATE_ADD('2006-05-00',INTERVAL 1 DAY);

SELECT CURDATE();

SELECT CURTIME();

DATE_FORMAT
SELECT date_format( '2006-04-30', '%d/%m/%Y' ); -- 30/04/2006
SELECT DATE_FORMAT('2003-10-03',GET_FORMAT(DATE,'EUR')); -- 03.10.2003
SELECT DATE_FORMAT('2006-06-00', '%d/%m/%Y');

SELECT NOW();

SELECT TO_DAYS('1997-10-07'); -- RETORNA DIAS

SELECT YEAR('2000-01-01');
```

dbDESIGNER

Gerenciamento e modelagem do SGBD MySQL.

Site oficial - http://fabforce.net/dbdesigner4/

Download - http://fabforce.net/downloads.php
O programa e a documentação.

Tutorial usando com o PostgreSQL:
http://www.flaviobrito.com.br/cursos/bd/apostilas/GuiaDBDesign4PosgreSQL.pdf

Triggers no MySQL

Criando uma tabela de logs
Que guardará informações dos registros excluídos e atualizados

Exemplo de tabela, cujas operações (delete e update) queremos monitorar

```
create table clientes
(
        cliente int primary key,
        cpf char(11),
        nome char(45) not null,
        credito_liberado char(1) not null,
        data_nasc date,
        email varchar(50)
)engine innodb;
```

Exemplo de tabela onde queremos guardar as informações (logs) sobre as alterações, a data e hora e quem efetuou

```
create table logs
(
        code int auto_increment primary key,
        tablel char(20) not null,
        operation char(6) not null,
        login char(12) not null,
        date_time datetime not null
)engine innodb;
```

Exemplo de trigger que fará o serviço automaticamente a cada operação de delete

```
delimiter |
DROP TRIGGER IF EXISTS tg_delete
CREATE TRIGGER tg_delete BEFORE DELETE ON clientes
FOR EACH ROW
BEGIN
INSERT INTO logs SET tablel = 'clientes', operation = 'delete',login = CURRENT_USER,
```

```
date_time = NOW();
END;
| delimiter;
```

Exemplo de trigger que fará o serviço automaticamente a cada operação de update

```
delimiter |
CREATE TRIGGER tg_update BEFORE UPDATE ON clientes
  FOR EACH ROW BEGIN
    INSERT INTO logs SET cliente = OLD.cliente, nome=OLD.nome;
  END;
|
delimiter ;
```

Triggers no MySQL

Criando uma tabela de logs

Que guardará informações dos registros excluídos e atualizados

Exemplo de tabela, cujas operações (delete e update) queremos monitorar

```
create table clientes
(
        cliente int primary key,
        cpf char(11),
        nome char(45) not null,
        credito_liberado char(1) not null,
        data_nasc date,
        email varchar(50)
)engine innodb;
```

Exemplo de tabela onde queremos guardar as informações (logs) sobre as alterações, a data e hora e quem efetuou

```
create table logs
(
        code int auto_increment primary key,
        tablel char(20) not null,
        operation char(6) not null,
        login char(12) not null,
        datet datetime not null
)engine innodb;
```

Exemplo de trigger que fará o serviço automaticamente a cada operação de delete

```
delimiter |
DROP TRIGGER IF EXISTS tg_delete
CREATE TRIGGER tg_delete BEFORE DELETE ON clientes
```

```
FOR EACH ROW
BEGIN
INSERT INTO logs SET tablel = 'clientes', operation = 'delete',login = CURRENT_USER,
datet = NOW();
END;
| delimiter;
```

Exemplo de trigger que fará o serviço automaticamente a cada operação de update

```
delimiter |
CREATE TRIGGER tg_update BEFORE UPDATE ON clientes
  FOR EACH ROW BEGIN
    INSERT INTO logs SET cliente = OLD.cliente, nome=OLD.nome;
  END;
|
delimiter ;
```