
UNIDADE I

HTML BÁSICO

Construir um site em HTML sem usar editor html.

HTML

HTML significa **Hypertext Markup Language** e é a linguagem de descrição de documentos usada na World Wide Web. Ela é orientada por marcadores ou TAGs.

TAGs são os comandos utilizados pela linguagem HTML. Cada TAG informa ao programa visualizador ou Browser, como ele deverá formatar o texto e deve estar dentro dos sinais de menor que (<) e maior que (>). Exemplo: <HTML>, <BODY>, etc.

Os TAGs podem ser únicos ou duplos, com início e fim. Exemplos:

TAG único:
 TAG duplo:
<P>....</P>

INICIANDO UM DOCUMENTO

Todo o documento HTML fica contido entre os TAGs: <HTML> e </HTML>. Temos duas seções básicas:

HEAD

Contém parâmetros de configuração do documento.

BODY

Contém o documento em si.

A estrutura de um documento HTML é:

```
<HTML><HEAD><TITLE>Título da Home Page</TITLE></HEAD>
<BODY BACKGROUND="imagem">***
Conteúdo da Home Page ***</BODY></HTML>
```

<HTML>...</HTML>

Envolvem todas as seções de um documento (HEAD e BODY).

<HEAD>...</HEAD>

Envolvem a seção de cabeçalho do documento.

<TITLE>...</TITLE>

Indica o título do documento para o Browser. Geralmente os Browsers apresentam este título na barra de título da sua Janela no Windows.

<BODY>...</BODY>

Envolvem a seção de corpo do documento. Aqui fica o conteúdo principal da Home Page. Opcionalmente podemos indicar um arquivo de imagem para formar o fundo, usando a opção: *BACKGROUND*.

ESTRUTURA USUAL

```
<html>
<head>
    <title> Página HTML </title>
</head>
<body>
    Comandos html
</body>
</html>
```

TÍTULOS E SUBTÍTULOS

Para demarcar títulos e subtítulos, use os TAGs de HEADER (H1 a H6), juntamente com as opções CENTER ou BLINK.

Observações: As TAGs CENTER e BLINK têm a função de deixar o cabeçalho ou texto centralizado e pulsante (piscando), respectivamente (depende da versão do navegador).

Ex.:

```
<H1> acesse as fotos </H1>
```

FORMATAÇÃO DE TEXTOS

Além das TAGs <CENTER>...</CENTER> e <BLINK>...</BLINK> existem as seguintes TAGs que podem ser utilizadas para a formatação de um texto.

... - Aplica o estilo **negrito**.

<I>...</I> - Aplica o estilo *itálico*.

<U>...</U> - Aplica o estilo sublinhado (nem todos os browser o reconhecem).

`^{...}` - Faz com que o texto fique sobrescrito.

`_{...}` - Faz com que o texto fique subscrito.

`<PRE>...</PRE>` - Utiliza a pré-formatação, ou seja, deixa o texto da maneira em que foi digitado.

IMAGENS

Podemos inserir imagens dentro de um documento HTML, mas devemos ter o máximo de cuidado, para não onerar a transmissão para o usuário.

Os formatos mais usados são o GIF e o JPG, ambos com compactação de pixels. Para inserir uma imagem, uso o TAG: ``, que é único, não exigindo um TAG finalizado. Exemplo:

```
<IMG SRC="figura1.gif" height="30%"
width="300">
```

Os arquivos com as imagens deverão estar armazenados no seu Provedor de Acesso, juntamente com o documento HTML.

LINKS

Os Links servem para criar *Palavras Quentes*, que permitem a interligação entre documentos HTML e outros documentos ou até arquivos FTP.

Veja o seguinte exemplo:

```
<ul>
<li><a href=#inicio>Para o próprio
documento</a></li>
<li><a href="ivl.htm#inicio">Para outro
documento</a></li>
<li><a
href="http://www.nome_da_página.com.br/">Pa
ra minha Home Page</a></li>
<li><a href="filme1.jpg">Âncora para imagem
externa</a></li></ul>
```

Use: `Texto ou imagem...`

O parâmetro NAME serve para marcar um ponto para possíveis desvios. Quando desviamos para um determinado ponto dentro

de um documento, indicamos este nome com um "#". Por exemplo:

```
<A NAME="AQUI">Aqui é um ponto para
desvios</A>...<A HREF="#AQUI">Desvia para
o ponto "AQUI"</A>
```

LISTAS

CRIANDO LISTAS ORDENADAS

Listas ordenadas, são também denominadas listas numeradas, pois, quando um navegador encontra uma TAG, iniciando uma lista ordenada, ele passa a mostrar cada item utilizando números, como 1, 2, 3, e assim sucessivamente.

Listas ordenadas são iniciadas pela TAG ``.

Cada item utiliza a TAG ``.

Finalmente,

``.

Type – pode ser os seguintes argumentos

1
A
a
I
i

Exemplo:

```
<OL type="1" >
<LI>É fácil fazer uma Home Page
<LI>Tem que ter paciência
<LI>Bons recursos
<LI>E não exagerar em imagens
</OL>
```

Resultado:

1. É fácil fazer uma Home Page
2. Tem que ter paciência
3. Bons Recursos
4. E não exagerar em imagens.

CRIANDO LISTAS NÃO ORDENADAS

Listas não ordenadas são muito parecidas com as ordenadas. A única diferença é o fato de elas não definirem explicitamente uma ordem, como é no caso das numeradas. Elas são formadas por símbolos, que podem ser bola, quadrado, e uma bola vazia. Elas são iniciadas com a TAG `` e são

respectivamente terminadas com ****. E seus elementos são que nem as numeradas: com **** Exemplo:

<UL type="circle"> ou **square** ou **disc**

****Internet

****Intranet

****BBS

resultado:

- Internet
- Intranet
- BBS

ATRIBUTOS ADICIONAIS DO ELEMENTO UL

O Netscape introduziu o atributo **TYPE** também em listas ordenadas. Ele recebe o tipo do marcador que será utilizado ao lado dos itens da lista, o qual pode ser **CIRCLE**, **SQUARE** OU **DISC**.

ATUALIZANDO PÁGINA HTML

São páginas normalmente sem links, que chamam outras depois de um determinado tempo dentro dela, sem nenhuma interferência do internauta.

Para fazer uma página desta basta colocar no documento a seguinte linha de comando:

Ex.:

<HTML>

<HEAD>

<META HTTP-EQUIV="REFRESH"
CONTENT="segundos; URL=Documento.htm">

<TITLE> Título **</TITLE>**

</HEAD>

<BODY>

Corpo do Documento

</BODY>

</HTML>

Exercícios

1) As páginas html são programas com comandos que precisam ser interpretados pelo(a):

- () Internet
- () Navegador
- () ASP
- () Servidor de acesso

2) Escreva a estrutura usual de uma página HTML.

3) Escreva a TAG/comando html para inserir a imagem que está gravada na pasta:
E:\teste\show; com o nome de ontem.jpg. Esta imagem ficará com 300 pixel de altura e 400 de largura.

4) Defina os comandos **<SUB>** e **<SUP>** .

5) Qual é o nome dado ao recurso que clicamos em um texto ou imagem e abre uma página.

6) Faça este recurso que ao clicar no texto Abrir, apareça no navegador a página teste.html.

7) O que será exibido ao usarmos o código abaixo:

```
<ul type="square">  
<li> ola </li>  
<li> oi </li>  
<li> tudo bem </li>  
</ul>
```

8) Escreva o código para aparecer o layout abaixo:

- I. Eu vou
- II. Este ano
- III. Estudar mais

TABELAS

Tabelas correspondem a um ótimo formato para originar informações, e é por essa razão que eles foram acrescentados à linguagem HTML

CONSTRUINDO TABELAS COM O ELEMENTO TABLE

A TAG <TABLE> é utilizada para a representação de dados tabulares. A estrutura e o conteúdo da tabela devem ficar dentro das TAGs <TABLE> </TABLE>

O TÍTULO DA TABELA (ELEMENTO CAPTION)

A TAG <CAPTION> especifica o título de uma tabela. Por exemplo:

<CAPTION>Notas da primeira avaliação</CAPTION>

TABLE HEADINGS (ELEMENTO TH)

A TAG <TH> é usada para especificar as células de cabeçalho da tabela. Essas células são diferentes das outras, pois seu conteúdo aparece geralmente em negrito. O elemento TH pode ser apresentado sem conteúdo algum: isso corresponde a uma célula em branco. As tabelas podem ainda conter mais de um TH para uma dada coluna, ou linha, ou simplesmente não conter nenhum elemento TH, isto é, não conter em nenhuma célula em destaque. O TAG dela é:

<TH>texto em destaque</TH>

Observações: Elas devem ficar, assim como todas, dentro da TAG <TABLE>.

TABLE DATA (ELEMENTO TD)

A TAG <TD> especifica as células de dados de uma tabela. Por se tratar de dados comuns (e não cabeçalhos), essas células possuem seu conteúdo escrito em fonte normal, sem nenhum destaque e alinhamento à esquerda. Assim como o TH, pode-se construir células em branco, usando o elemento TD, como no exemplo a seguir:

<TD>Células de dados</TD>

Observações: A TAG de terminação, <TD>, também é opcional.

END OF TABLE ROW (ELEMENTO TR)

A TAG <TR> indica o fim de uma linha na tabela. Cada linha da tabela pode conter várias células, e portanto, é necessário que se faça uso de uma marcação que indique exatamente o ponto de quebra de uma linha e início de outra. Toda linha deve terminar com um <TR>, com exceção da última linha da tabela, que dispensa o TR porque o uso da própria marcação de fim de tabela </TABLE> torna implícito o fim da linha.

ATRIBUTOS PARA A TABELA

As marcações das tabelas podem apresentar resultados diferentes, se

acompanhadas de alguns atributos. Os principais são:

BORDER

Um atributo opcional para ser usado com TABLE é o atributo BORDER. Se ele estiver presente, a tabela será formatada com linhas de borda.

Atenção: Todas as explicações acima como as que estão por vir, foram feitas, para que você possa saber o que significa a TAG em questão.

Exemplo:

```
<TABLE BORDER>
  <CAPTION> Nota da primeira avaliação
</CAPTION>
  <TD>Notas/Alunos</TD>
  <TH>Eduardo</TH>
  <TH>Ana Lúcia</TH>
  <TH>Andréa</TH>
<TR>
  <TH>Notas</TH>
  <TD>8,0</TD>
  <TD>9.3</TD>
  <TD>7.8</TD>
<TR>
  <TH>No de Inscrição</TH>
  <TD>376234809</TD>
  <TD>387349048</TD>
  <TD>502350432</TD>
</TABLE>
```

O atributo BORDER pode também receber um valor que vai estabelecer qual a espessura (além da existência) da linha de borda da tabela (BORDER="valor"). Se o valor atribuído for 0 (zero), o BORDER funciona exatamente como o caso padrão, sem o BORDER. Dessa maneira, é possível colocar tabelas em maior destaque, atribuindo um valor maior que 1 para o BORDER.

```
<TABLE BORDER=5>
  <TD>TESTE</TD>
  <TD>TESTE2</TD>
```

```
<TD>TESTE3</TD>
<TR>
  <TD>TESTE4</TD>
  <TD>TESTE5</TD>
  <TD>TESTE6</TD>
</TABLE>
```

ALIGN

Este atributo pode ser aplicado a TH, TD ou TR e controla o alinhamento do texto dentro de uma célula, com relação as bordas laterais. Quando aplicado a TR, ele define o alinhamento de toda uma linha da tabela

O exemplo abaixo, mostra como o ALIGN aceita os valores LEFT, CENTER ou RIGHT, para alinhar à esquerda, centralizar ou alinhar à direita, respectivamente.

```
<TABLE BORDER>
  <TD>Primeira célula</TD>
  <TD>Segunda célula</TD>
  <TD>Terceira célula</TD>
<TR>
  <TD ALIGN="CENTER">Centro</TD>
  <TD ALIGN="LEFT">Esquerda</TD>
  <TD ALIGN="RIGHT">Direita</TD>
<TR>
</TABLE>
```

Veja o resultado:

Primeira célula	Segunda célula	Terceira célula
Centro	Esquerda	Direita

VALIGN

Pode ser aplicado a TH e TD e define o alinhamento do texto em relação às bordas superior e inferior.

Aceite os valores TOP, MIDDLE, E BOTTOM para alinhar na parte de cima, no meio e na parte de baixo, respectivamente.

Veja o exemplo:

```
<TABLE BORDER>
  <TD>Teste de alinhamento</TD>
  <TD VALIGN="TOP">TOP</TD>
```

```

<TD VALIGN="middle">MIDDLE</TD>
<TD VALIGN="bottom">BOTTOM</TD>

```

</TABLE>

Veja o resultado:

Teste para alinhamento	TOP	MIDDLE	BOTTOM
------------------------	-----	--------	--------

Às vezes podemos ter a necessidade de que uma célula de uma tabela ocupe mais de uma coluna ou linha da mesma, para isso utilizamos os atributos COLSPAN e ROWSPAN nas células da tabela.

COLSPAN e ROWSPAN

Em cima dessa estrutura básica vamos mesclar a primeira linha da tabela usando o COLSPAN:

```

<table border="1">
<tr>
<td colspan="3">Tabela de preços de
carros</td>
</tr>
<tr>
<td>Celta</td>
<td>Life</td>
<td>R$ 21.000</td>
</tr>
<tr>
<td>Gol</td>
<td>City</td>
<td>R$ 23.000</td>
</tr>
</table>

```

Tabela de preços de carros		
Celta	Life	R\$ 21.000
Gol	City	R\$ 23.000

Vejamos agora sobre o ROWSPAN, formatando essa nossa tabela para exibir apenas o "Celta", porém dois modelos diferentes do mesmo:

```

<table border="1">
<tr>
<td rowspan="3">Celta</td>
<td>Life</td>
<td>R$ 21.000</td>
</tr>
<tr>
<td>Super</td>
<td>R$ 23.000</td>
</tr>
</table>

```

Celta	Life	R\$ 21.000
	Super	R\$ 23.000

Usando COLSPAN e ROWSPAN.

```

<table border="1">
<tr><td colspan=3>Tabela de preços de
carros</td>
<tr>
<td rowspan="3">Celta</td>
<td>Life</td>
<td>R$ 21.000</td>
</tr>
<tr>
<td>Super</td>
<td>R$ 23.000</td>
</tr>
</table>

```

Tabela de preços de carros		
Celta	Life	R\$ 21.000
	Super	R\$ 23.000

Exercícios

1) Cite três parâmetros da TAG table, com suas respectivas definições.

2) Quando criamos uma tabela, as colunas são inseridas dentro da linha. Escreva o comando/TAG que cria a linha e a coluna.

3) Crie a tabela abaixo, sabendo que a borda da tabela possui espessura 3, cor da linha verde.

A	
B	E
C	F
D	G

4) Desenhe a tabela, após execução do código abaixo:

```
<table border="5">
<tr><td
colspan=3><center>Cursos</center></td>
<tr>
<td rowspan="3">Info</td>
<td>primeira serie</td>
<td>Diurno</td>
</tr>
<tr>
<td>terceira série</td>
<td>Diurno</td>
</tr>
<tr>
```

```
<td>Quarto período</td>
<td>noturno</td>
</tr>
</table>
```

FORMULÁRIOS

A linguagem HTML também permite que o cliente (navegador) interaja com o servidor, preenchendo campos, clicando em botões e passando informações. Por exemplo, eu tenho um Guest Book (formulário) em meu site onde eu peço a todos os visitantes que dêem suas opiniões. Essas informações, devem ser tratadas por programas, denominados scripts, que podem armazená-las para uma posterior utilização. Os scripts podem ainda retornar um outro documento HTML, uma URL, ou algum outro tipo de dado para o cliente.

O elemento **FORM**, da linguagem HTML, é justamente o responsável por tal interação. Ele provê uma maneira agradável e familiar para coletar dados do usuário através da criação de formulários com janelas de entrada de textos, botões, etc.

É preciso ter em mente que o FORM coleta dados, mas não os processa. São os scripts que entendem os dados, como mencionado. É aí que entra a necessidade da interface **CGI**. Tal interface, permite que o servidor se comunique com o script que vai atuar sobre essas informações, retornando os resultados para o navegador.

CONSTRUINDO FORMULÁRIOS COM O FORM

Para fazer formulário, você tem que colocar as TAGs <FORM> </FORM>. Todos os outros comandos, devem ficar dentro dessas TAGs. Ok?!

ATRIBUTOS PARA FORM

O elemento FORM pode conter dois atributos que determinaram para onde será mandada a entrada do FORM. Vejam como eles são:

GET

Esse atributo indica totalmente como o dado é passado para o script ou programa definido no atributo **ACTION**.

POST

Passa os dados para a entrada padrão dos sistema operacional.

Vale a pena lembrar, que será mostrado, abaixo, um exemplo completo, de como fazer sua página com formulários. Também será dado um endereço, de um servidor, que processa os dados e os retorna via e-mail. Aí poderá ser lido normalmente. Agora será explicado como colocar os campos de dados, mas se não estiver entendendo, copie o exemplo, e só altere os dados, com o seu nome, e suas informações.

INPUT

A TAG <INPUT> especifica uma variedade de campos editáveis dentro de um formulário. Ele pode receber diversos atributos que definem o tipo de mecanismo de entrada (botões, janelas de texto, etc.), o nome da variável associada com o dado da entrada, o alinhamento e o campo do valor mostrado. O atributo mais importante do **INPUT** é o **NAME**. Ele associa o valor da entrada do elemento. Por exemplo, quando você for receber os dados, já, processados, irá vir o nome : =resposta dada pelo visitante. Outro atributo importante é o **TYPE**. Ele determina o campo de entradas de dados. Veja como se usa este atributo:

```
<INPUT TYPE="TEXT" NAME="nome">
```

Para mudar o tamanho, da janela padrão, você tem que colocar o comando **SIZE**. Por exemplo:

```
<INPUT TYPE="TEXT" NAME="nome" SIZE=8>(ou número desejado)
```

Outro comando importante é o **VALUE**. Ele acrescenta uma palavra digitada no comando à janela. Por exemplo:
<INPUT TYPE="TEXT" NAME="nome" SIZE=8 VALUE="texto.">

Olhe como ficaria:

TIPOS DE ELEMENTOS TYPE

Você pode fazer várias coisas com o elemento **TYPE**. Por exemplo, para ser um campo de senha, que quando digitado, apareça o símbolo "*", ao invés das letras, você deve escrever o seguinte:

```
<INPUT TYPE="PASSWORD" NAME="nome" SIZE=8>
```

6.2.3.1.1 - TYPE="RADIO"

Quando o usuário deve escolher uma resposta em uma única alternativa, de um conjunto, utiliza-se o **RADIOButtons**. Um exemplo típico do uso de tais botões, é cuja resposta pode ser SIM ou NÃO. É preciso que todos os rádios buttons es um mesmo grupo, ou seja, referentes a mesma pergunta, tenham o mesmo atributo **NAME**. Para esse tipo de entrada, os atributos **NAME** e **VALUE**, são necessários. Veja a seguir:

```
<INPUT TYPE="RADIO" NAME="você gostou dessa home page?" VALUE="sim">sim<p>
```

```
<INPUT TYPE="RADIO" NAME="você gostou dessa home page?" VALUE="nao">não<p>
```

Repare:

☐ sim

☐ não

TYPE="PASSWORD"

Este comando serve para fazer uma campo de senhas! Quando a pessoa digitar, aparecerá o sinal de "*"! O comando é:

```
<INPUT TYPE="PASSWORD" NAME="SENHA" MAXLENGTH=6>
```

TYPE="CHECKBOX"

Esse comando é válido quando apenas uma resposta, é esperada. Mas nem sempre

está é a situação...O tipo **CHECKBOX** provê outros botões através dos quais mais de uma alternativa, pode ser escolhida.

Definição dos checkboxes:

```

<INPUT TYPE="CHECKBOX"
NAME="netscape" VALUE="net">Netscape<p>
<INPUT TYPE="CHECKBOX"
NAME="Explorer" VALUE="exp">Internet
Explorer<p>
<INPUT TYPE="CHECKBOX" NAME="Mosaic"
VALUE="mos">Mosaic<P>
<INPUT TYPE="CHECKBOX" NAME="Hot
Java" VALUE="hot"> Hot Java<P>
  
```

Veja o resultado:

☐ Netscape
☐ Internet Explorer
☐ Mosaic
☐ Hot Java

TYPE="SUBMIT"

Esse é o botão que submete os dados do formulário quando pressionados, ou seja, possibilitam, o envio, dos dados para o script que vai tratá-los. Veja como se adiciona o botão:

```
<INPUT TYPE="SUBMIT" VALUE="enviar">
```

Veja como ficará:

TYPE="RESET"

No caso dos botões **RESET**, quando o botão é clicado, ele automaticamente limpa todos os campos já preenchidos no formulário, voltando à situação inicial.

```
<INPUT TYPE="RESET" VALUE="Limpar">
```

Veja como ficará:

TEXTAREA

Para se limitar o tamanho do campo mostrado na tela, faz-se o uso dos atributos **COLS** e **ROWS** que especificam, respectivamente, o número de colunas e linhas

que se deseja mostrar para o usuário. O atributo **NAME** é obrigatório, e especifica o nome da variável, que será associadaa à entrada do cliente (navegador)O atributo value não é aceito nesse elemento, mas você pode colocar já um texto da seguinte maneira. Veja como ele é colocado:

```

<TEXTAREA NAME="nome" COLS=20
ROWS=3>texto</TEXTAREA>
  
```



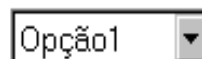
SELECT

Embora os usuários não precisem digitar sempre suas respostas, mostrar cada opção através de botões consegue um bom espaço, e facilidade. Veja como ele funciona:

```

<SELECT>
<OPTION>opção1
</SELECT>
  
```

Veja o resultado:



Abaixo, temos um exemplo completo de uma página com um formulário.

```

<form action="http://www.uki.edu/cgi-
bin/AnyForm.cgi" method="post">
<input type="Hidden" name="AnyFormModle"
value="Mail">
<input type="Hidden" name="AnyFormDisplay"
value="http://www.gun.com.br">
<input type="Hidden" name="AnyFormTo"
value="gun@gun.com.br">
<input type="Hidden" name="AnyFormSubject"
value="Dados do Formulário de HTML">
  
```

Qual o seu nome?<input type="Text" name="Nome" size="40">

Qual o seu E-mail?<input type="Text" name="E-mail" size="40"><p>

Você gostou da minha Home Page?<input type="Radio" name="Gostou" value="Sim">sim
 <input type="Radio" name="Gostou" value="Mais ou Menos"CHECKED>

Mais ou Menos <input type="Radio" name="Gostou" value="Não">Nem um pouco!<P>

Qual a página que você mais gostou??<select name="Melhor Página"><option value="Interface">Interface</option><option value="Imagens, som, etc.">Imagens, som, cores, comandos básicos, etc.</option><option value="Frames">Frames</option><option value="Ferramentas">Ferramentas</option><option value="Formulários">Formulários</option><option value="CGI">CGI</option><option value="JAVA">JAVA</option><option value="Java Script">Java Scipr</option><option value="VRML">VRML</option><option value="CHAT">CHAT</option><option value="Onde colocar">Onde colocar</option><option value="Onde divulgar">Onde Divulgar</option><option value="Bombas em Java Script">Bombas em Java Script</option><option value="Contadores de Acesso">Contadores de Acesso</option><option value="Organizando às informações">Organizando às informações</option></select><P> Deixe seus comentários sobre a minha Home Page:<textarea name="Comentários" cols="28" rows="5"></textarea>

O que está faltando? <input type="Text" name="O que está faltando?" Value="o que falta?">
Essa Home Page lhe ajudou?<input type="Radio" name="Ajudou?" value="sim">Sim<input type="Radio" name="Ajudou?" value="Não!">Não!<p><input type="Submit" value="Enviar "><input type="Reset" value="Limpar Dados">

MÚSICA

Existem 2 maneiras de colocar músicas em sua home page. A primeira é com o TAG <BGSOUND> que só é reconhecido pelo Internet Explorer. Para utilizar este TAG siga o exemplo :

<BGSOUND SRC="arquivo.mid">

Onde *arquivo.mid* é o arquivo de música. Caso você queira que a música repita-se, adicione o atributo **loop="infinite"**.

A segunda maneira é com o TAG <EMBED> que, por ser reconhecido pelo Internet Explorer e Netscape, é recomendado. Para utiliza-lo, siga o exemplo :<EMBED SRC="arquivo.mid">É recomendado usar arquivos midi por serem bem menores que os wavs.

CARACTERES ESPECIAIS

Á	Á	á	á	Â	Â
â	â	À	À	à	à
Ã	Å	ã	å	Ä	Ã
ä	ã	Å	Ä	å	ä
Æ	Æ	æ	æ	É	É
é	é	Ê	Ê	ê	ê
È	È	è	è	Ë	Ë
ë	ë	Ð	Ð	ð	ð
Í	Í	í	í	Î	Î
î	î	Ì	Ì	ì	ì
Ï	Ï	ï	ï	Ó	Ó
ó	ó	Ô	Ô	ô	ô
Ö	Ò	ò	ò	Ø	Ø
ø	ø	Õ	Õ	õ	õ
Ö	Ö	ö	ö	Ú	Ú
ú	ú	Û	Û	û	û
Û	Ù	ù	ù	Ü	Ü
ü	ü	Ç	Ç	ç	ç
Ñ	Ñ	ñ	ñ	<	<
>	>	&	&	"	"
®	®	©	©	Ý	Ý
ý	ý	Þ	Þ	þ	þ
ß	ß	°	º	ª	&170;
¹	¹	²	²	³	³
ƒ	ƒ	†	†	‡	‡
‰	‰	¢	¢	£	£
«	«	±	±	»	»

•	·	¼	¼	½	½
¾	¾	¿	¿	×	×
÷	÷	¡	¡	¤	¤

Exercícios

1) Cite três exemplos de sites que podem usar formulários.

2) Defina os parâmetros da tag FORM.

- a- Action
- b- Method
- c- Name

3) Quais as opções do parâmetro Method?
Qual as diferenças?

4) Crie um botão com título “enviar”.

5) Crie uma caixa de texto para permitir digitar no máximo 30 caracteres.

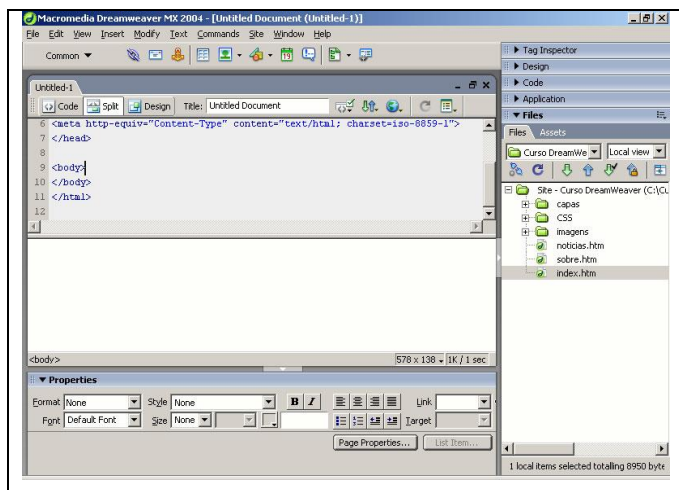
6) O que sera exibido após executar o comando: `<input type="radio" name="teste" value="acertou">`

7) Escreva os comandos par criar uma caixa select com três opções.

UNIDADE II HTML AVANÇADO

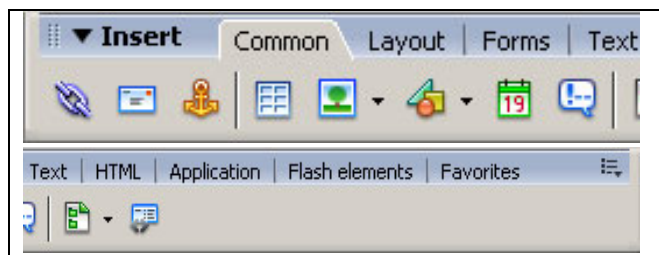
O aluno, ao final da unidade, deverá construir um site em HTML com recursos profissionais e usando um editor html (DreamWeaver).

Interface do Dreamweaver



Esta é a nova interface do programa, esta não é a primeira tela que é mostrada logo da abertura do programa, mas iremos falar da primeira tela mais adiante.

Painel Insert



O painel Insert é dividido em varias categorias:

Common: Possui os elementos comuns a serem utilizados em seus documentos, Hyperlinks, Email Link, Named Anchor, Table, Date e Comment, foram criados novos grupos na categoria Common, que são: Images , Media , Templates , que antes eram categorias.

Layout: Define o tipo de visualização, também podemos desenhar tableas, layers e frames nesta nova versão 2004.

Forms: Contém todos os elementos de

formulário.

Text: Contém elementos referentes aos textos.

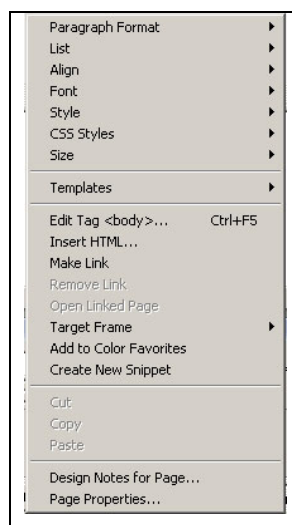
HTML: Esta nova categoria traz elementos da linguagem HTML, como Head, Tables, Frames e Script, incorporou o Horizontal Rule, que antes era da categoria Common.

Application: Contém elementos utilizados para a criação de aplicações utilizando acesso a uma base de dados.

Flash Elements: Importe um elemento de Flash em uma página e personalize as suas propriedades dentro do Dreamweaver

Favorites: Nesta nova categoria podemos personalizar com os elementos de varias categorias, ou os elementos mais usados e assim gerar uma produtividade, pois os elementos que queremos já estão todos a mão.

Barra de Ferramenta de Documentos (Toolbars Document)



Esta barra configura a visualização do documento entre outros:

Code: Exibe o código-fonte HTML

Split: Divide a janela exibindo o código-fonte HTML e ao mesmo tempo o design do documento.

Design: Exibe o design do documento, este mais usado por quem não tem nenhum conhecimento de HTML.

Title: Define o título do documento, aquele que aparece na parte superior do navegador.

No Browser Check Errors: Para checar erros de navegador, podemos informar quais navegadores.

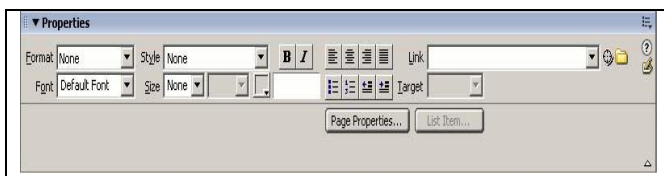
File Management: Permite gerenciar os documentos do site, principalmente quando estamos trabalhando com um provedor on-line ou off-line, para fazer transferências (put) de arquivos ou baixar (get) arquivos.

Preview / Debug in Browser: Permite uma visualização prévia no navegador antes da publicação.

Refresh: Atualiza o documento.

View Options: Opções referentes à visualização de vários elementos no documento inclusive elementos invisíveis, tais como: named anchor, layers, image map, rules, grids e etc.

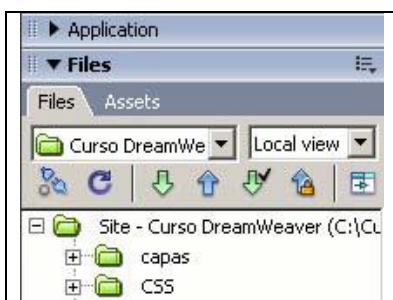
Properties Inspector



Exibe as propriedades de cada elemento usado no documento. Quando selecionamos um texto, ele mostra as propriedades de formatação do mesmo e assim por diante.

Painéis Laterais

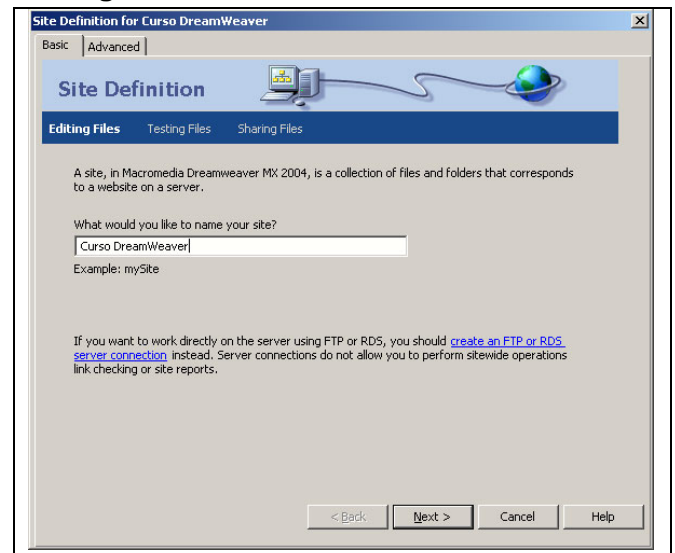
Os painéis laterais auxiliam no desenvolvimento e criação do site. Os painéis mais importantes são: Files, Application, Design, Tag Inspector e Frames. Podemos ocultar/exibir o Painel Lateral clicando na setinha que fica na linha divisória. Podemos ainda ocultar/exibir o Painel Lateral e o Properties Inspector com (F4).



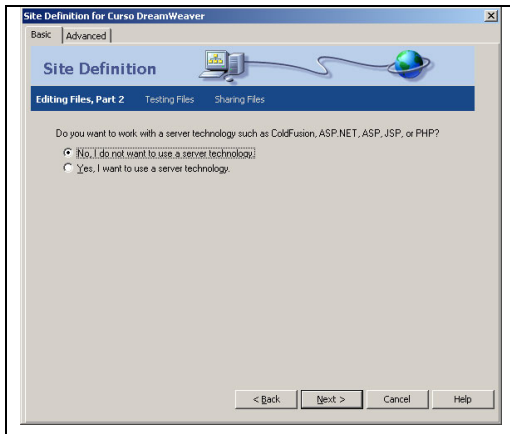
Menu Contextual

O menu contextual é acionado com o clique do botão direito do mouse, sobre um link, um texto ou área de trabalho. Os itens são variáveis de acordo com o local onde o menu é acionado.

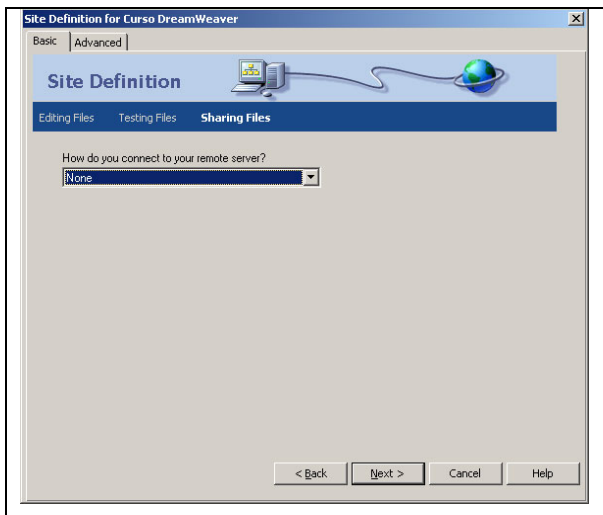
Configurando seu Site



Para que serve? Para que você possa gerenciar todos os seus documentos HTML gerados pelo Dreamweaver MX 2004. Trabalhamos com vários documentos HTML, já pensou em mudar um link em um documento que está ligado a outros dez (10). Com este gerenciamento, assim que você mudar este link o Dreamweaver pergunta se você quer atualizar também os outros dez documentos que estão apontando para este. Vamos lá criar o nosso site local. Um site local, é a área de armazenamento de todos os seus documentos HTML, suas imagens, do seu site. Precisamos dar um nome e informar uma pasta no HD para este armazenamento. Precisamos criar um novo site para cada novo projeto. Vamos lá, no painel lateral FILES, clique no ícone de opções deste painel, e escolha: Site > New Site...



Nesta primeira janela vamos definir o nome do site que iremos trabalhar, lembramos que este nome é apenas para identificar o projeto. Clique em Next >



Nesta janela vamos deixar a primeira opção marcada, pois em nosso curso não vamos trabalhar com script de servidor, esse é outro curso que o Ibratec oferece, que é Macromedia Dreamweaver com Banco de Dados. Clique em Next >

Nesta janela vamos informar a pasta local em seu HD onde vamos armazenar todo o nosso site. Use a opção acima como exemplo de pasta. Clique em Next >

Nesta janela vamos deixar como mostra, pois não vamos trabalhar com um servidor, como já foi mencionado em outro curso será abordado. Clique em Next >

Pronto, acabamos de configurar o nosso site,

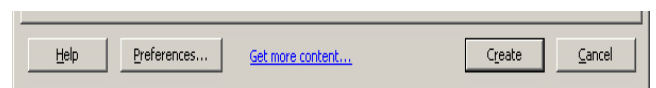
para finalizar clique em Done.

Novo documento

Para criar um novo documento HTML em branco, proceda assim: em Create New, escolha HTML.

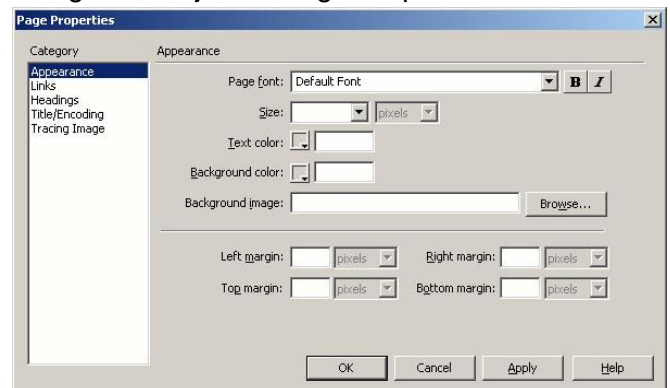


Se por acaso você já está com um documento aberto, proceda assim: Menu > File > New...

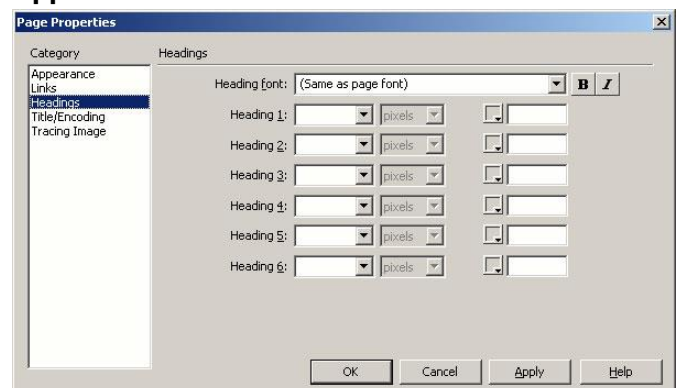


Ao abrir a janela New Document escolha na Category > Basic Page > HTML, clique no botão Create.

Pronto já estamos com o nosso novo documento, agora vamos modificar as suas propriedades. No menu > Modify > Page Properties... Vamos ver cada uma das categorias da janela Page Properties.



Appearance



Page Font: define a fonte padrão para todo o documento.

Size: define o tamanho padrão da fonte para todo o documento.

Text Color: define a cor do texto do documento.

Background Color: define a cor de fundo da página. Para escolher nova cor, clique no quadradinho de cor ao lado ou digite o código da cor.

Background Image: define a imagem de fundo da página se houver. Clique em Browse para selecionar. Se não houver imagem, deixe o campo em branco.

Left Margin: define um valor em pixels para a margem esquerda (valor aceito pelo Internet Explorer). Se não quiser margem, digite 0.

Top Margin: define um valor em pixels para a margem do alto (valor aceito pelo Internet Explorer). Se não quiser margem, digite 0.

Margin Width: especifica um valor em pixels para a largura da margem (valor aceito pelo Netscape). Se não quiser margem, digite 0.

Margin Height: especifica um valor em pixels para a altura da margem (valor aceito pelo Netscape). Se não quiser margem, digite 0.

Links

Link Font: define a fonte para todos os links.

Size: define o tamanho da fonte para todos os links.

Link Color: especifica a cor de todos os links.

Rollover Links: especifica a cor quando o mouse passar nos links.

Visited Links: define a cor dos links para os endereços que já foram visitados pelo usuário.

Active Links: especifica a cor dos links ativos no documento.

Headings

Heading Font: define a fonte para todos os título e subtítulos.

Heading 1 até o 6: estes são elementos de HTML para a formatação de títulos e subtítulos, você escolhe tamanhos e cores diferentes para cada um dos 6 itens.

Title / Encoding

Title: define o título da página. O título é exibido no alto da janela do navegador, que é o

mesmo mostrado na barra de ferramentas do Dreamweaver MX.

Encoding: define a codificação do documento.

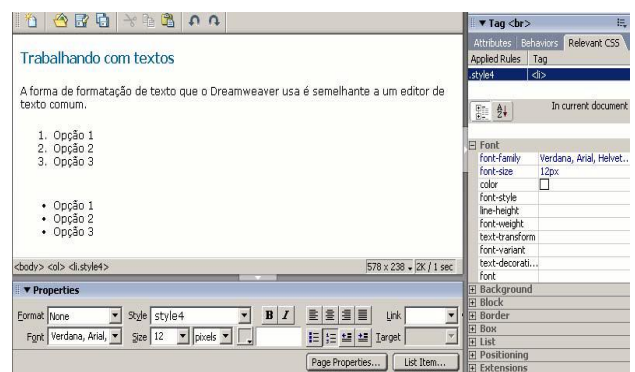
Tracing Image

Tracing Image: define a imagem a ser usada como um rascunho de fundo de página, ela não aparece no navegador. A sua função é servir como um guia de um layout para posicionar os elementos do documento exatamente onde mostra a imagem de rascunho.

Transparency: define a opacidade da imagem de rascunho, quanto menor o valor mais transparente fica.

Trabalhando com textos

A forma de formatação de texto que o Dreamweaver usa é semelhante a um editor de texto comum.



Observe que foi inserido o mesmo texto que digitei acima, como um editor de texto comum, eu selecionei o que queria e no properties inspector, formatei todo o texto. Uma grande novidade desta versão, é poder formatar todo o texto usando o painel lateral TAG, como mostra a figura. Aproveitando criei uma lista ordenada e uma não ordenada. O interessante de tudo isso é que os ícones de formatação de texto como: negrito, itálico, alinhamento à esquerda, alinhamento à direita, centralizado, justificado, marcadores, numeradores, recuo de parágrafo, são exatamente iguais a um editor de texto bem conhecido, facilitando assim o aprendizado.

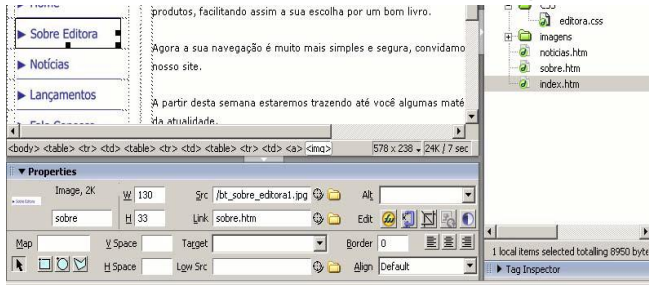
Hiperlinks

Precisamos interligar nossos documentos, para

isso o Dreamweaver oferece varias formas de criar os hipelinks para documentos, imagens, arquivo de multimídia. Temos os seguintes tipos de hiperlinks:

Link relativo: quando vinculamos documentos que estão na mesma pasta de trabalho.

Link absoluto: quando vinculamos a uma URL externa, informando pr otocolo, nome de servidor com o caminho e o nome do arquivo.

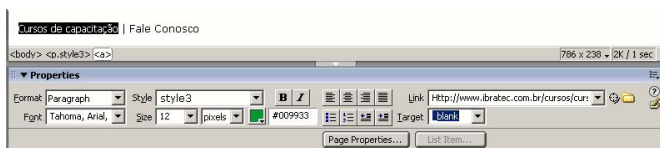


Link de email: quando vinculamos um endereço de correio eletrônico para abrir o gerenciador de email instalado na maquina do usuário.

Link interno: esse é do tipo ancora, aponta para dentro do mesmo documento.

Link relativo

Como mostra a figura abaixo, foi selecionada a imagem sobre editora e no campo Link, foi informado o nome do documento ao qual será criado o vínculo.



Absoluto

Para nosso link absoluto, vamos imaginar que em nosso documento existe um link para o site do Ibratec. Observe a figura abaixo:



Foi selecionado o texto e no campo link digitado todo o caminho absoluto. Em target, foi informado para abrir uma nova janela do

navegador, podemos ainda informar: _self (mesma janela).

Link de email



Aproveitando o mesmo documento, selecionei o texto Fale Conosco, e no painel Insert, dei um clique no ícone de um envelope .

Abriu a janela Email Link, agora é só escrever o texto e informar um email válido. Vale lembrar que este tipo de link abre um gerenciador de email instalado na maquina do usuário.

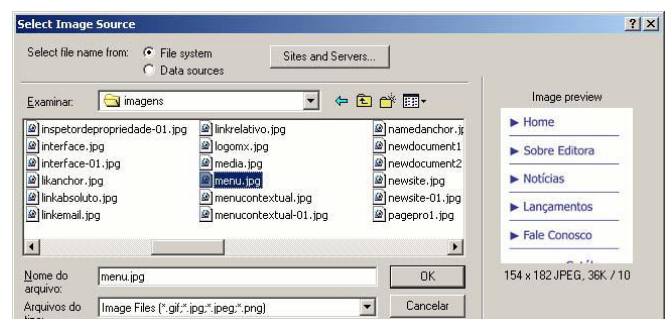
Link Interno

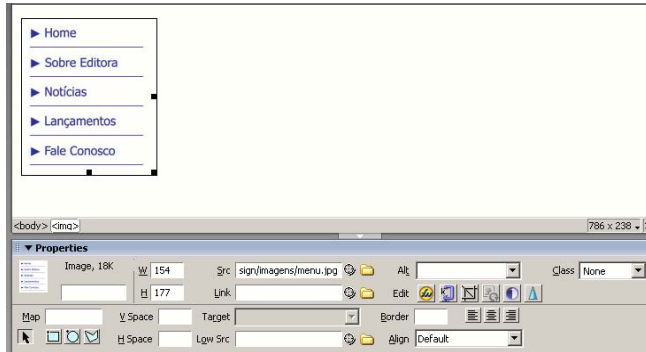
Para criar um link interno, isto é, vincular uma palavra ou imagem a outro, no mesmo documento. Para isso devemos criar o ponto de localização do alvo de nosso vinculo, uma âncora que fará a ligação entre nosso menu e o local de destino.

No exemplo abaixo observe onde foi criado o Named Anchor e o link aponta para "#link_absoluto", (#) é o vinculo juntamente com o nome.

Inserindo imagem

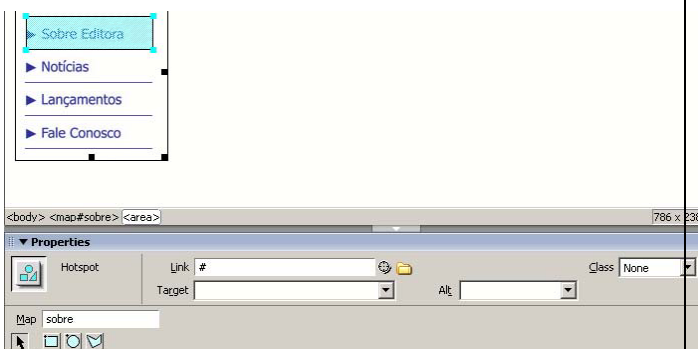
No painel insert, categoria common, clique em , para abrir a janela abaixo:





Na figura acima observamos três áreas em destaque, primeiro o Map, onde podemos criar mapas de imagem do tipo: retangular, elíptica e poligonal. A segunda o Edit, com novas ferramentas para edição de imagens dentro do Dreamweaver. A primeira opção já existia: "Edit". Ao lado, temos a ferramenta "Optimize in Fireworks". A seguinte é "Crop" (corta a imagem). A próxima é "Resample" (com esta ferramenta você pode diminuir uma imagem). Prosseguindo, teremos "Brightness and Contrast" para brilho e contraste e "Sharpen" para tratar a imagem. A terceira, são as opções de alinhamento de uma imagem, clicando no menu suspenso temos todas as opções de alinhamento. Os botões de alinhamento (À esquerda, À direita e No centro) também podem ser utilizados para colocar os elementos selecionados.

Abaixo vamos ver como configurar os pontos ativos de um mapa de imagem:



É só escolher uma das três opções de mapa e

desenhar por cima da imagem e configurar as propriedades

Trabalhando com tabelas

As tabelas são uma ferramenta de desenho de muito avançada para organizar os dados e as imagens em uma página HTML. As tabelas proporcionam aos Webdesigners os meios de adicionar uma estrutura horizontal e vertical a uma página. Elas consistem de três componentes básicos:

linhas (espaçamento horizontal)

colunas (espaçamento vertical)

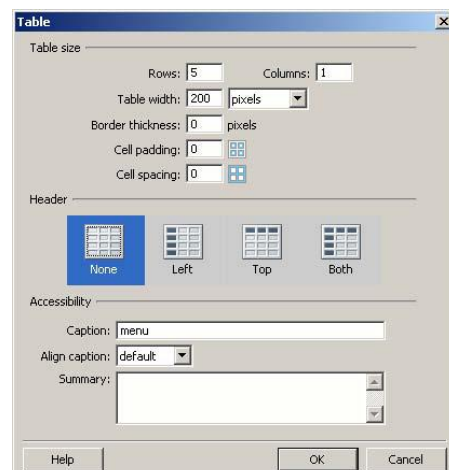
células (o recipiente criado pela interseção de uma linha e uma coluna). Utilize as tabelas para organizar os dados tabulares, desenhar colunas em um documento ou organizar texto e imagens em seus

documentos. Após a criação da tabela, a aparência e a estrutura podem ser facilmente modificadas. O conteúdo pode ser incluído; adicione, exclua, divida ou mescle as **linhas e colunas**; modifique as propriedades da tabela, das linhas ou células, para adicionar cor e alinhamento, e para copiar e colar células.

O modo de criação de uma tabela pode variar de acordo com o tipo de visualização do documento: Standart View ou Layout View.

Standart view

No painel insert, categoria common, clique no ícone table. Na versão 2004 aparece a seguinte janela para auxiliar na criação de uma tabela.



Rows: define o número total de linhas da tabela.

Columns: define o número total de colunas da tabela.

Cell Padding: indica o espaço entre o conteúdo da célula e a sua parede. Para não deixar espaço, digite o valor 0.

Cell Spacing: indica o espaço entre cada célula, excluindo a borda. Para não deixar espaço, digite 0.

Width: especifica a largura da tabela. O valor pode ser em pixels ou porcentagem. O valor em pixels é fixo, já o valor em porcentagem fará com que o tamanho da tabela varie conforme a largura do navegador e resolução de vídeo do usuário.

Border thickness: define a largura da borda. Para a borda ficar invisível, digite o valor 0.

Aperte OK, para confirmar as configurações da tabela.

Propriedade de uma tabela, pelo properties inspector.

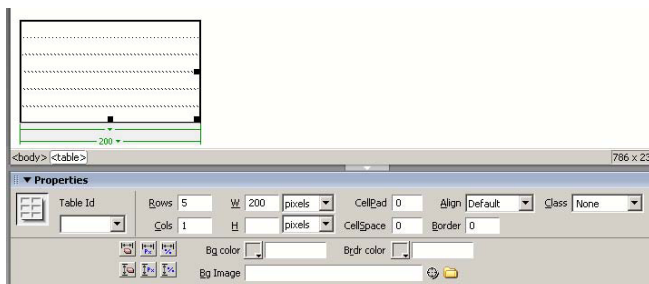


Table Id: especifica um nome para a tabela.

Rows: exibe ou define o número de linha da tabela.

Cols: exibe ou define o número de colunas da tabela.

W: largura que a tabela possui em pixels ou porcentagem.

H: altura da tabela. Se nada estiver preenchido, a altura é automática.

CellPad: define o espaçamento do conteúdo da célula em relação às suas paredes.

Cellspace: define o espaçamento entre as células.

Align: define o alinhamento da tabela. A opção

Default, geralmente, deixa a tabela alinhada à esquerda. As outras opções são: esquerda (Left), direita (Right) ou central (Center).

Border: especifica a largura da borda da tabela.

Bg Color: especifica a cor de fundo da tabela. Clique no quadrado cinza para escolher a cor.

Brdr Color: especifica a cor da borda.

Bg Image: especifica uma imagem de fundo para toda a tabela.

No campo disponível, você digita o caminho da imagem ou seleciona a imagem através do ícone de pasta ao lado do campo.

Clear Columns Widths: limpa a largura das colunas. Corresponde ao campo W.

Clear Rows Heights: limpa a altura das linhas. O campo H do properties inspector é limpo.

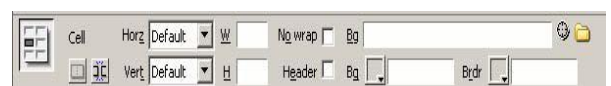
Convert Table Widths to pixels: converte a largura da tabela para o valor em pixels. Útil para fazer a conversão de porcentagem para pixels.

Convert Table Widths to Percents: converte a largura da tabela para o valor em porcentagem. Útil para fazer a conversão de pixels para porcentagem.

Convert Table Heights to Pixels: converte a altura da tabela para o valor em pixels. Útil para fazer a conversão de porcentagem para pixels.

Convert Table Heights to Percents: converte a altura da tabela para o valor em porcentagem. Útil para fazer a conversão de pixels para porcentagem.

Propriedade de uma célula, pelo properties inspector



Quando clicamos dentro de uma célula, temos as seguintes opções no properties inspector:

Horiz: define o alinhamento horizontal do conteúdo da célula. As opções são: default (esquerda), left (esquerda), center (centro) e

right (direita).

Vert: define o alinhamento vertical do conteúdo da célula. As opções são: default (normalmente, no meio), top (alto), middle (meio) bottom (embaixo), baseline (base).

No Wrap: se você deixar marcada essa opção, não ocorrerá quebra automática de linha dependendo do conteúdo da célula.

Header: formata a célula com essa opção marcada como um head de tabela. Corresponde à tag <thead> ou <th>. Útil para DHTML. Como padrão, ao aplicar esse recurso, o texto digitado ficará no centro e em negrito. Embaixo da palavra "cell", há dois ícones: enquanto o primeiro não tem função e é inativo, o segundo permite que você faça um split (divisão) na célula. Com isso, a célula fica dividida ao meio.

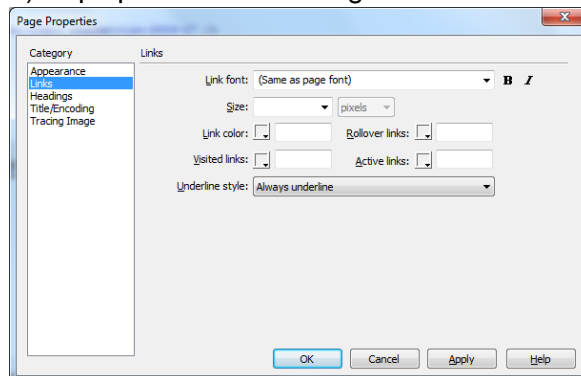
Você pode aplicar um split várias vezes em cada célula. Quando você aplica o split, abre-se uma janela perguntando se você quer dividir a célula em colunas (Columns) ou em linhas (Rows) e quantas divisões quiser fazer. O ícone que está inativo, só fica ativo quando duas ou mais células estão selecionadas, serve para mesclar as células em uma única.

Você pode determinar também a altura da célula. Basta inserir o valor no campo H. Isso serve, inclusive, para células que sofreram split. Com isso, você pode criar layouts bem interessantes.

Obs.: podemos ainda colocar uma tabela dentro de outra, para chegar ao layout que desejamos.

Exercício

- 1) Quais as opções contidas na guia Insert do menu de componentes?
- 2) Quais os passos para habilitar a paleta Insert e qual a tecla de atalho que executa a mesma tarefa?
- 3) Quais os tipos de links que podemos criar com o Dreamweaver?
- 4) Como criar uma tabela de 8 linhas e 4 colunas no Dreamweaver?
- 5) Como mesclar células em uma tabela?
- 6) Qual a importância na configuração de um site no Dreamweaver?
- 7) Explique 4 itens da imagem abaixo:



Projetando um layout de um documento

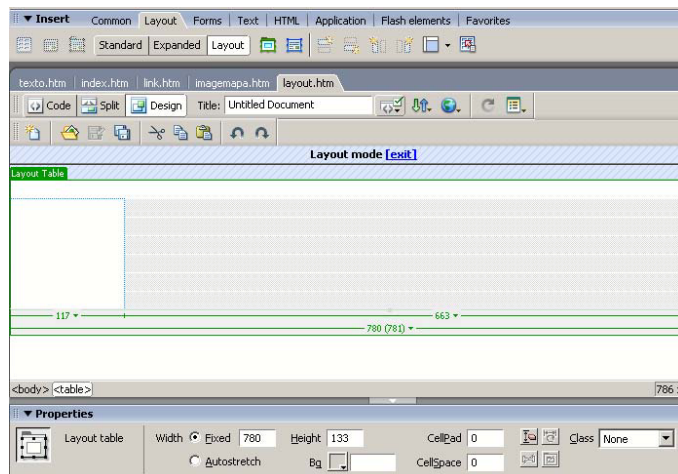
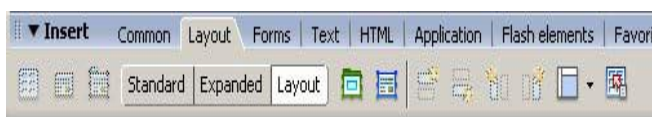
O layout de um documento é uma parte importante do projeto de um site. O termo layout de documento ou página se refere à aparência que o documento terá no navegador, como a posição de um menu ou as imagens, por exemplo. O Dreamweaver proporciona diversas maneiras de criar e controlar o layout do documento.

Um método comum de criar o layout de página consiste na utilização de tabelas HTML. No entanto, as tabelas podem ser de difícil utilização, porque elas não foram originalmente criadas para o layout de página, mas sim para exibir dados tabulares.

O Dreamweaver conta com a visualização de layout view. Na visualização de layout view é possível projetar a página utilizando tabelas como estrutura subjacente. É possível desenhar células (células de tabela) na página com facilidade e, em seguida movê-las para onde desejar. O layout pode ter uma largura fixa ou se expandir, até ocupar toda a janela do navegador.

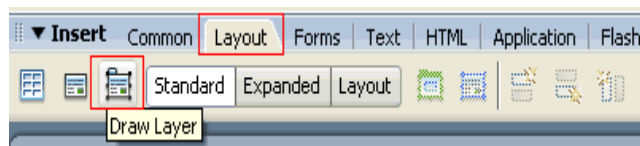
Ainda é possível dispor as suas páginas utilizando tabelas segundo a maneira tradicional (conforme vimos no capítulo anterior), ou utilizando camadas (LAYERS) e depois convertendo-as em tabelas. No entanto, a visualização de layout do Dreamweaver é a forma mais fácil de definir o layout da página ou documento.

Para utilizar a visualização de layout, é necessário sair da visualização padrão do Dreamweaver (Standard view). Mude para a categoria Layout como mostra abaixo:



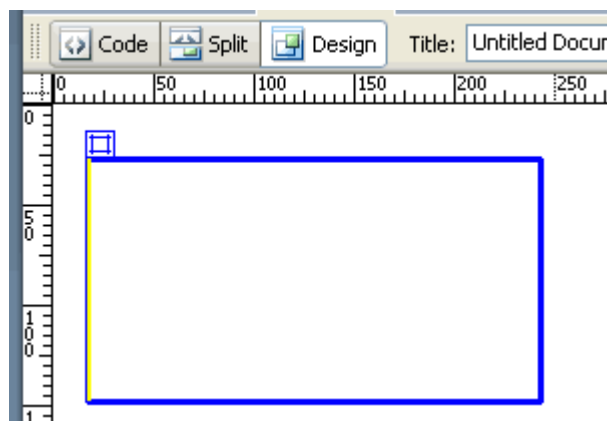
Draw Layer

Na aba **Layout** do Dreamweaver, existe a seguinte opção: **Draw Layer**, observe a imagem abaixo:



Com ele, você poderá desenhar as tais "layers" em sua página e o Dreamweaver se encarregará de posicioná-las e definir suas características como: largura, tamanho, posicionamento, entre outros para você.

Observe na imagem de exemplo abaixo, o resultado de uma "layer" desenhada no Dreamweaver:



A medida que você vai desenhando as "layers", o Dreamweaver vai nomeando-as de forma lógica e da seguinte forma:

```
<div id="Layer1"></div>
```

Ou seja, como foi a primeira "layer", o atributo **id** da tag **div** foi definido como: **Layer1**. A próxima será: **Layer2** e assim sucessivamente.

E ao mesmo tempo é criado o código CSS responsável pelas características da "layer", observe:

```

6 <style type="text/css">
7 <!--
8 #Layer1 {
9     position: absolute;
10    left: 17px;
11    top: 27px;
12    width: 226px;
13    height: 121px;
14    z-index: 1;
15 }
16 -->
17 </style>

```

O principal objetivo dos códigos CSS acima é o posicionamento da "layer". O tipo de posicionamento utilizado foi o **absolute**. O que significa dizer que tal "layer" será posicionada absolutamente com relação ao elemento body, será? Basicamente sim e é o que todos esperam.

Criando Templates ou Samples

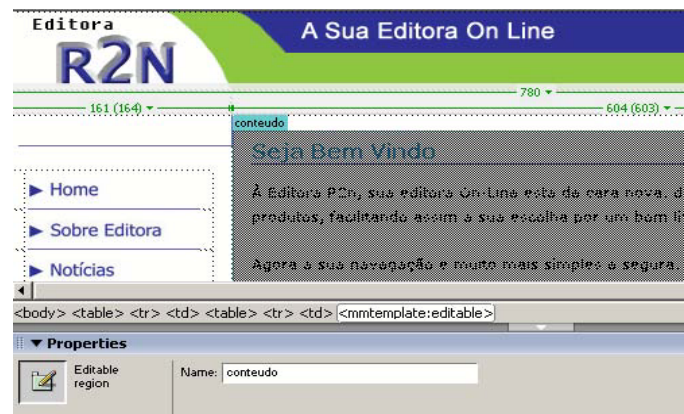
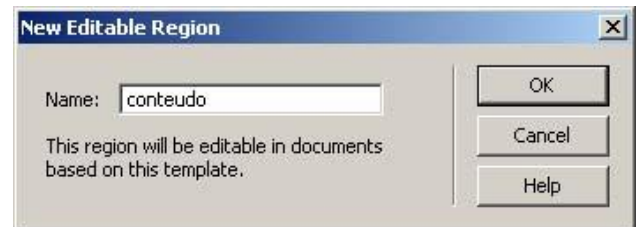
Um template (modelo) é um documento que pode ser utilizados como base para outros documentos, para que todos tenham a mesma aparência. Podemos ainda especificar áreas fixas (não editáveis) e áreas que podem sofrer alterações. Se seu site tem documentos que contém o mesmo menu, o mesmo cabeçalho, e o mesmo rodapé, enfim, a mesma estrutura, é mas rápido criar um template. O documento que vemos no exemplo abaixo tem a mesma estrutura, variando o conteúdo das informações do lado direito.

No painel insert, categoria common, temos o ícone make template, clique nele com o documento que você deseja criar como template. A janela save as template será aberta.



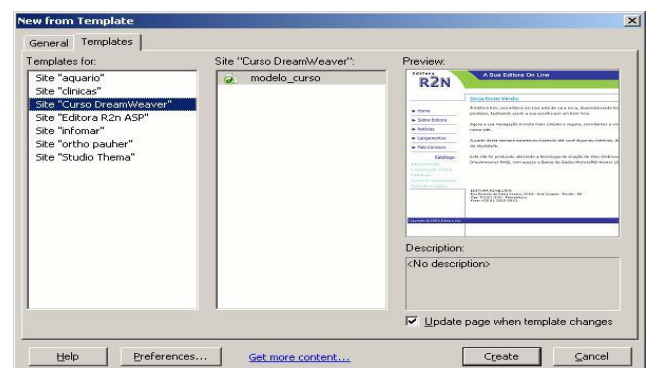
Definindo as áreas editáveis

No painel insert, categoria common, clique na setinha preta que esta ao lado do ícone Make template e escolha Editable Region. Defina um nome para sua área editável, e ok



Criando uma novo documento a partir de uma template

No menu > File > New..., quando a janela New Document abrir, selecione a guia Template e escolha o template que acabamos de criar, como mostra abaixo:



Observe que apenas a área que você criou esta disponível para ser editada, é isso aí!

Alterando um template

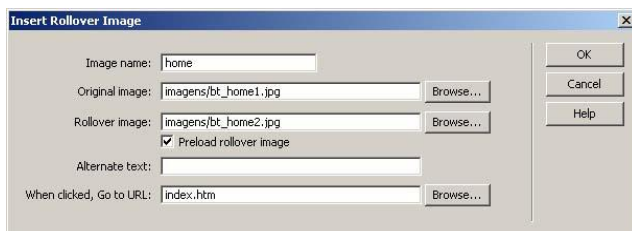
No painel lateral > Files, abra o template que você criou, ele está na pasta template. Faça as mudanças que deseja, salve e pronto!

Efeitos de botão

É muito legal poder fazer estes efeitos sem ter que programar uma linha sequer de javascript. Vamos criar um botão rollover, uma barra de navegação e de quebra botões de flash sem ter que ir fazer isso lá no flash, ou mesmo saber usar o flash.

Botão rollover

Posicione o cursor no local onde você deseja inserir o botão rollover, vale lembrar que estamos trabalhando com imagens, que pelo menos tem que ser feito antes. No painel insert, categoria common, clique na setinha preta ao lado do ícone images, escolha rollover image. Configure como mostra:



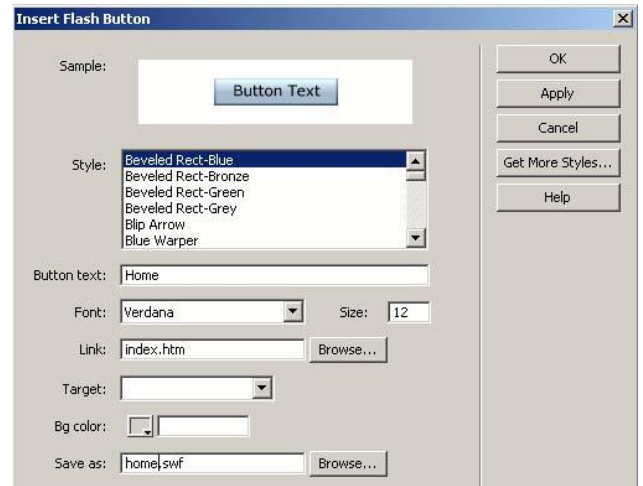
Observe que estamos trabalhando com duas imagens iguais, alterando apenas a cor da seta para a rollover image. Ok!



Botão do Flash

No painel insert, categoria common, clique na setinha preta ao lado do ícone Media e escolha

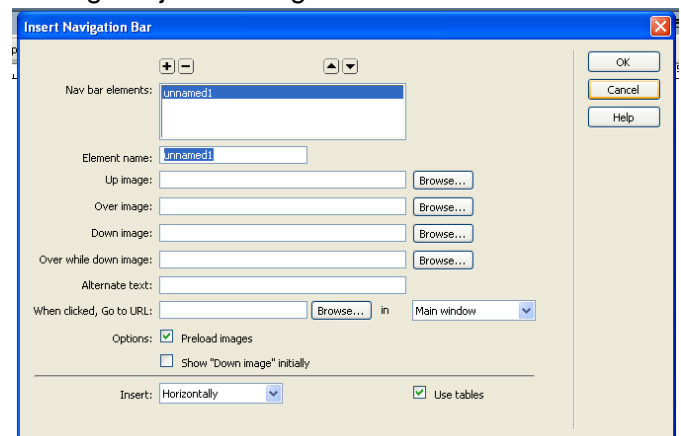
Flash Button. Configure como segue:



O arquivo *.swf, será salvo na raiz do seu site local, e é isso!

Barra De Navegação

É um recurso que nos permite criar uma barra de opções, usando imagens, para trabalhar como menu. O caminho é menu insert – Image objects – Navigation Bar



Element name – é a identificação do menu. Nome interno.

Up image – É a primeira imagem a ser exibida.

Over image – Esta imagem será exibida quando o mouse passar sobre a imagem “up image” antes do primeiro click.

Down image – Esta será exibida quando houver click na primeira imagem

Over while down image – Passa a ser a imagem “over image” após o primeiro click.

Alternate text – Quando o mouse parar sobre a imagem será exibido este texto.

When clicked, Go to URL – É o arquivo que será aberto no click, é o link.

Options

Preload images – Se marcado será carregada as imagens antes de serem exibidas

Show “Down image” initially” – Inicar com a imagem Down

Insert – Posicionamento das imagens (vertical ou horizontal)

Frames

Os frames (molduras) são compostos de dois elementos principais um conjunto de frames e os frames individuais. Um conjunto de frames é um documento HTML que define a estrutura de um conjunto de frames em um documento.

A definição do conjunto de frames abrange informações sobre o número de frames exibidos em uma página, o tamanho dos frames, a origem da página carregada em um frame e outras propriedades que podem ser definidas. Uma página HTML com um conjunto de frames não é exibida em um navegador, ela apenas armazena as informações sobre a maneira como os frames em uma página serão exibidas.

Geralmente, os frames definem uma área de navegação e uma área de conteúdo em uma página da Web. Quando um documento do Dreamweaver é dividido em frames, são criados documentos HTML separados para o conjunto de frames e para cada novo frame. O conjunto de frames é chamado de frame-pai e um frame é chamada de frame-filho. O que o usuário percebe como uma única página da Web com dois frames constitui, na verdade, três arquivos distintos: o arquivo do conjunto de frames e dois arquivos com o conteúdo que aparece dentro dos frames. A alteração das propriedades dos frames e dos conjuntos de frames permite redimensionar os frames e utilizar os vínculos e os destinos para controlar o conteúdo de um frame.

Criando os frames (molduras)

Os frames podem ser criadas modificando em um documento existente do Dreamweaver, dividindo-o em áreas de documento adicionais. Há várias maneiras de criar um conjunto de

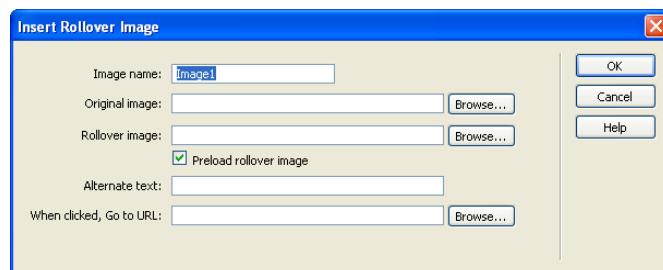
frames: você pode desenhá-lo ou selecionar entre vários conjuntos de frames predefinidos.

Como criar um conjunto de molduras

Antes de criar um conjunto de frames ou trabalhar com frames, torne visíveis as bordas do frame na janela do documento. Para exibir as bordas do frame em um documento, escolha View > Visual Aids > Frame Borders. Quando as bordas da moldura forem exibidas, será adicionado espaço em volta da borda do documento, fornecendo-lhe um indicador visual das áreas do frame no documento.

No painel Layout, clique na setinha preta ao lado do ícone Frames e escolha a opção como mostra abaixo:

Roll Over



Este recurso também cria um link, só que apenas um de cada vez.

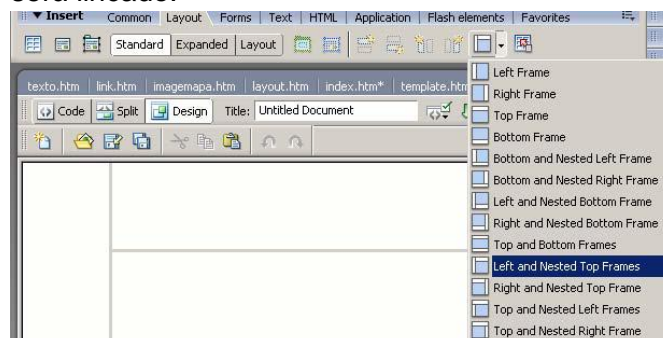
Image name – Identificação interna.

Original image – Imagem a ser exibida.

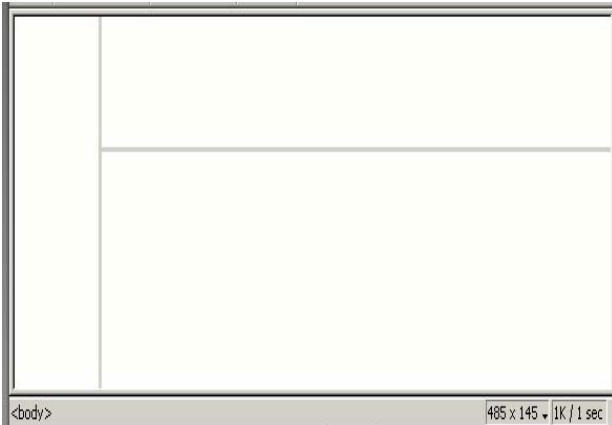
Rollover image – Será exibida quando o mouse passar sobre a imagem “original image”.

Alternate text – Texto que será exibido quando o mouse parar sobre a imagem.

When clicked, Go to URL – É o arquivo que será vincado.



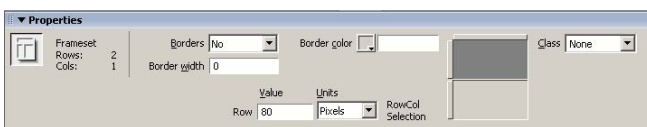
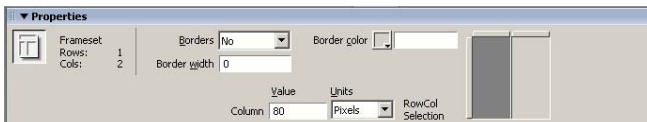
Observe abaixo a divisão dos frames, como foi explicador anteriormente.



Quando salvarmos as páginas, deveremos escolher a opção File > Save All. Isso fará com que sejam salvas os três frames filhos e também o frame pai.

Se sua página for a inicial do seu site, deverá ser salva com o nome index.htm, para que abra-se automaticamente ao ser digitado o endereço do site.

Propriedades dos Frames



Borders: especifica se o frame exibirá ou não as bordas.

Default: exibe bordas.

Yes: exibe bordas.

No: não exibe bordas.

Border Width: especifica a largura da borda. Se não quiser borda, além de especificar No Item acima, digite o número 0.

Border Color: especifica a cor da borda. Basta digitar o código da cor ou efetuar a seleção da tonalidade no painel de cores.

Columns ou Rows: conforme o tipo de frame criado, é exibido o tamanho da coluna ou linha. Esse tamanho (largura ou altura) do frame é um

valor expresso em pixels, porcentagem ou valor relativo. Uma outra maneira de especificar o tamanho do frame é arrastar a sua borda com o mouse.

No lado esquerdo do Inspetor de Propriedades, temos a definição do frame em número de linhas e colunas. No lado direito do Inspector, o layout do frame é reproduzido.

Para abrir uma página em uma divisória/frame temos as seguintes opções:

_Blank – Abrir em um novo navegador.

_Parent – Abrir no mesmo navegador, substituindo removendo as demais divisões.

_Self – Abrir na mesma divisória/frame

_Top – Abrir na divisão superior.

Obs.: Pode ser usado também nomes definidos pelo usuário.

Iframe

É um recurso parecido com frame, só que ao dividir o navegador, este recurso divide a página. Devemos ter cuidado ao usar para que a página não fique com várias barras de rolagem.

```
<IFRAME name=palco src="iframe_0.html"
frameBorder=0 width=400 height=150
scrolling=auto></IFRAME>
```

Onde:

name: é o nome da janela, ele será usado caso você queira criar links que abram dentro do iframe, é o valor do target.

src: é a página que será aberta dentro do iframe. frameborder: borda do frame.

width e height: largura e altura do iframe, respectivamente.

scrolling: barra de rolagem.

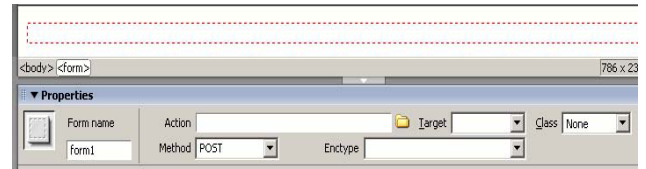
No caso de colocar um iframe num post, você deve fazer o upload do arquivo html e das figuras (se tiver) e depois colocar o código acima no post alterando os dados como você preferir. Lembre-se que o HTML trabalha

sempre com referências. Todos os arquivos vão ficar no mesmo nível, portanto não utili

Exercício

- 1) Quais as vantagens em criar sites usando templates?
- 2) Qual a principal vantagem do iframe em relação ao frame?
- 3) Escreva o código iframe para criar um moldura de nome conteúdo, com 400 pixel de largura e 500 de altura, abrindo inicialmente a página teste.html.
- 4) Explique cada opção de Target.
- 5) Fale sobre a opção Roll Over do Dreamweaver?
- 6) Escreva os passos para inserir um botão do flash pelo dreamweaver?
- 7) Como deixar a cor de fundo do botão Flash transparente?

Formulários



Os formulários permitem coletar informações dos usuários. Entre os usos mais comuns dos formulários, destacam-se as pesquisas, formulários de pedidos e interfaces de busca. Utilize o Dreamweaver para criar formulários, adicionar objetos a eles e (através da utilização de comportamentos) para validar as informações digitadas pelo usuário.

Os formulários do Dreamweaver podem incluir objetos padrão, como campos de texto, botões, campos de imagens, caixas de seleção, botões de opção, menus de lista, campos de arquivos e campos ocultos.

No painel insert, categoria Forms, para poder aplicar os objetos de formulários, selecionando o comando Forms na caixa superior. Observe o exemplo abaixo:

O primeiro botão do painel forms é o que insere o próprio form (recipiente para os elementos que formarão o formulário). É preciso clicar nele para proceder com o preenchimento.

Criando um formulário

Posicione o cursor onde será inserido o form, primeiro elemento do painel insert, categoria forms. Ao ser inserido o Dreamweaver marca a área com uma linha tracejada vermelha. Dentro da linha vermelha nosso form, insira uma tabela para melhor organizar os elementos de formulário, que vamos inserir.


O nosso formulário deve ficar assim:

Funcionário	<input type="text"/>
Salário	<input type="text"/>
Cargo	<input type="text"/>
	<input type="button" value="Enviar"/>

Cada item de nosso formulário acima, tem que ser configurado, de acordo com suas propriedades no properties inspector. Abaixo segue relação de todos os elementos de formulário.

Insert Form (Inserir Formulário) 

Insert Text Field (Campo texto) 


Insert Hidden Fields (Arquivos Ocultos) 

Insert Textarea (Texto de multiplas linhas)



Insert Checkbox (Caixa de verificação) 

Insert Radio Button (Botão de opção) 

Insert Radio Group (Grupo de botão de opção) 


Insert List/Menu (Menu de lista) 


Insert Jump Menu (Menu de links/escolha)



Insert Image Field (Campo de Imagem) 

Insert File Field (Campo de Arquivo) 

Insert Button (Botões) 

Insert Label (Etiqueta) 

Insert Fieldset 

Form: insere o formulário propriamente dito, um recipiente dos campos de formulários que segue logo abaixo.

Text Field: alfabético ou numérico. O texto digitado pode ser exibido como uma linha simples, linhas múltiplas, marcadores ou asteriscos (com a finalidade de proteger as senhas).

Hidden Fields : permitem armazenar informações (como o destinatário dos dados do formulário ou o assunto do formulário) que não forem relevantes ao usuário, mas que serão utilizadas pelo aplicativo que processa o formulário.

TextArea: campo de texto com múltiplas linhas.

Checkbox: permitem múltiplas respostas em um único grupo de opções.

Radio Button: representam opções exclusivas. A seleção de um dos botões do grupo cancela a seleção de todos os outros.

Radio Group: insere um grupo de botão de opção.

List/Menu: apresentam um conjunto de valores que os usuários poderão escolher. O objeto poderá apresentar um menu pop-up, que aparecerá apenas quando o usuário clicar no nome do objeto (e aceitará apenas uma opção), ou uma caixa de listagem, que sempre exibirá os valores em uma lista de rolagem (e aceitará mais de uma opção).

Jump Menu : permite inserir um menu no qual cada opção se vincula a um documento ou arquivo. **Image Field :** podem ser utilizados no lugar dos botões Enviar.

File Field: permitem que o usuário procure os arquivos nos discos rígidos, carregando-os como dados do formulário.

Button: realizam tarefas quando forem clicados, como o envio e redefinição dos formulários. É possível digitar um identificador personalizado para um botão ou utilizar um dos identificadores predefinidos. **Label:** insere textos entre as tags <label> e </label>

Fieldset: o texto é inserido diretamente no código-fonte, semelhante ao label.

CSS Style

Um estilo é um grupo de atributos de formatação que controla a aparência de uma faixa de texto de um único documento. A folha de estilos CSS pode ser utilizada para controlar vários documentos simultaneamente e inclui todos os estilos de um documento. A vantagem de utilizar as folhas de estilos CSS em relação aos estilos HTML é que, além de estarem

vinculadas a vários documentos, quando um estilo CSS for atualizado ou alterado, a formatação de todos os documentos que utilizam essa folha de estilos específica será também automaticamente atualizada. Os estilos CSS são identificados pelo nome ou pelo rótulo HTML, o que permite alterar o atributo de um estilo e exibir as modificações no texto inteiro ao qual esse estilo foi aplicado. Os estilos CSS nos documentos HTML podem controlar a maioria dos atributos tradicionais de formatação de texto, como fonte, tamanho e alinhamento. Eles podem especificar também atributos exclusivos de HTML, como posicionamento, efeitos especiais e imagens cambiáveis pelo mouse.

Nunca houve tanta interação com as folhas de estilo como existe nesta versão do Dreamweaver. Ao abrir as propriedades de página, já encontram opções que criam os CSS padrões do site com fonte, tamanho, cor, tipo, margem de página etc... Na edição do site, as modificações feitas por você vão sendo transformadas em novos estilos... por exemplo: Se você coloca uma cor e negrito numa parte do texto, um novo CSS é criado na hora, sempre seguindo a seguinte nomenclatura: style1, style2 ... No painel de propriedades, sempre que você selecionar um objeto, a caixa "Style" vai aparecer e nela, você terá todos os estilos criados...

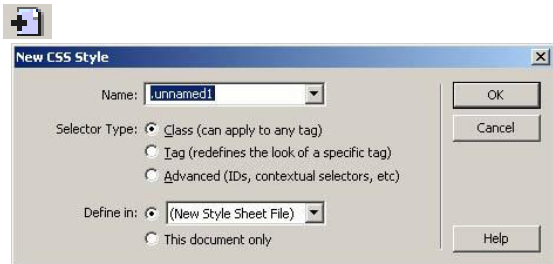


Novo CSS Style

Se o painel lateral não estiver aparecendo o painel Design, habilite para que possamos criar nossos CSS. Menu > Window > Design.



Vamos clicar no segundo ícone que aparece na parte inferior direito do painel Design, New CSS Style. Vamos analisar a janela New CSS Style.



Name: especifica o nome para o estilo. Observe que o estilo criado terá o nome unnamed1. Você pode inserir qualquer nome, desde que mantenha o ponto. Exemplo: ".titulo1".

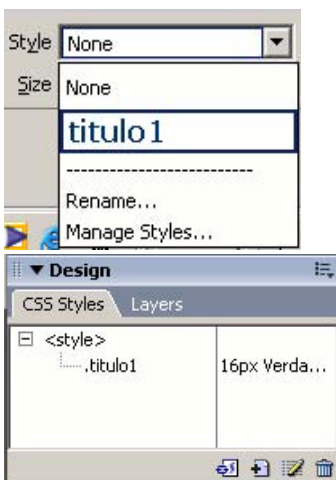
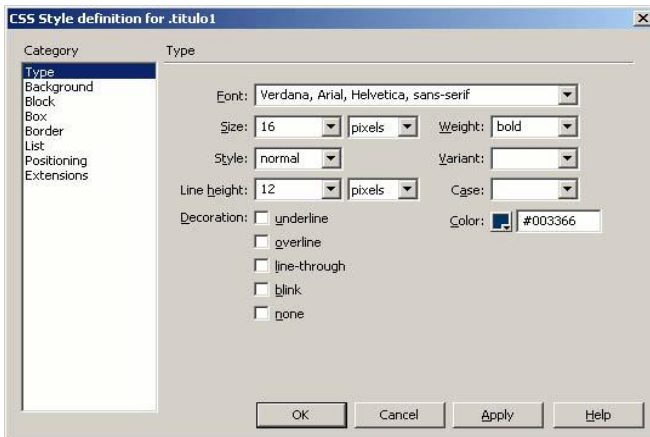
Selector Type: especifica o tipo de estilo a ser criado: Class (can apply to any tag) cria uma class, que pode ser aplicado a qualquer tag.

Tag (redefines the look of a specific tag): cria um novo visual para as tags do HTML.

Advanced (Ids, contextual selectors, etc): para a criação de style de link, os seletores link, hover, visited, etc.

Define In: especifica se o estilo será vinculado ou não. Se escolher o item New Style Sheet File, o estilo será vinculado, criando um arquivo externo para armazenar o CSS (*.CSS). Se escolher This Document Only, não será vinculado, gerando todo o código CSS junto do HTML.

Vamos escolher: Class (can apply to any tag) e em Define In: This Document Only, de um nome de título1 e ok!. Vamos formatar no .titulo1 como segue abaixo:



Após a sua confirmação, observe que no painel lateral design, já contamos com o nosso primeiro CSS Style criado. Como foi mencionando no properties inspector, também consta o nosso CSS Style, de uma forma mais visual facilitando assim a escolha dos estilos.

Alterando um CSS Style criado

No painel lateral Design, o terceiro ícone nos possibilita isto o Edit Style... Selecione o estilo que deseja alterar e pronto clique no Edit Style.



Anexando um arquivo de CSS Style ao documento

No painel lateral Design, o primeiro ícone nos possibilita isto o Attach Style Sheet



Aplicando um CSS Style em texto

Selecione o texto que você deseja aplicar o estilo, igual a um editor de texto, após a seleção escolha o CSS Style que deseja para este texto. Ao clicar no CSS Style .titulo1, por exemplo será aplicada a formatação que criamos para esse estilo no texto selecionado.

CSS programado

O CSS é uma ferramenta que dá estilo à estrutura HTML. Tudo isso é feito através de atributos aplicados a cada elemento da estrutura HTML.

Os elementos são identificados quando estão definidos por classes, ID ou pelo próprio nome da tag.

Quando identificados por classes (*div class="master"*) os elementos são referenciados através do ponto final (.). Ou seja, uma expressão CSS ficaria no seguinte formato:
.master { – Atributos – }

Quando os elementos são identificados por ID (*div id="master"*), eles devem ser referenciados através do caractere trilha (#). Ou seja, uma expressão CSS ficaria no seguinte formato:
#master { – Atributos – }

Já quando os elementos não contêm nenhuma identificação (*<label>*, *<p>*, *<div>*, etc), o CSS deve buscar o nome exato da tag HTML. Ou seja, neste caso a expressão CSS ficaria exatamente neste modelo: *label { – Atributos – }*

Definição dos principais atributos CSS

Background :Aplica cor de fundo, imagem de fundo ou transparência.

background-attachment: Rolagem do fundo pode ficar numa posição definida / Insere Marca d'água.

background-image: Insere imagem de fundo.

background-color: Insere cor de fundo ou transparência.

background-position: Posicionamento da imagem de fundo. Identifica através de coordenadas onde será o início da imagem.

background-repeat: Dá a opção de repetir a imagem na direção X (horizontal) e Y (vertical)

border: Manipula a largura, estilo e cor de todas as 4 bordas.

border-bottom: Manipula a largura, estilo e cor da borda inferior.

border-bottom-color: Manipula a cor da citada borda.

border-bottom-style: Manipula o estilo da citada borda.

border-bottom-width: Manipula a largura da citada borda.

border-color: Manipula a cor das 4 bordas.

border-left: Manipula a largura, estilo e cor da borda esquerda.

border-left-color: Manipula a cor da borda citada.

border-left-style: Manipula o estilo da borda citada.

border-left-width: Manipula a largura da borda citada.

border-right: Manipula a largura, estilo e cor da borda direita.

border-right-color: Manipula a cor da borda citada.

border-right-style: Manipula o estilo da borda citada.

border-right-width: Manipula a largura da borda citada.

border-style: Manipula o estilo de todas as 4 boras.

border-top: Manipula a largura, estilo e cor da borda superior.

border-top-color: Manipula a cor da borda citada.

border-top-style: Manipula o estilo da borda citada.

border-top-width: Manipula a largura da borda citada.

border-width: Manipula a largura de todas as 4 bordas.

clear: Elementos flutuantes à esquerda ou à direita de um elemento.

clip: Parte visível de um elemento.

color: Cor de fonte

cursor: Define o tipo de ponteiro do mouse.

display: Define se o elemento é exibido e o espaço é reservado para ele.

filter: Tipo de filtro aplicado ao elemento.

float: Define se o elemento flutua a esquerda ou a direita

font: Estilo, variante, peso, tamanho e altura da linha do tipo de fonte.

font-size: Tamanho da fonte.

font-style: Fonte itálico.

Fonte-variant: Fonte bold.

font-weight: Peso da fonte de claro a negrito.

height: Altura exibida ao elemento.

left: Posição do elemento em relação a margem esquerda da página.

letter-spacing: Distância entre as letras.

line-height: Distância entre linhas de base.

list-style: Tipo, imagem e posição do estilo da lista.

list-style-image: Marcador de item de lista.

list-style-position: Posição do marcador de item da lista.

list-style-type: Marcador de item de lista alternativo.

margin: Tamanho de todas as 4 margens.

margin-left: Tamanho da margem esquerda.

margin-right: Tamanho da margem direita.

margin-bottom: Tamanho da margem inferior.

margin-top: Tamanho da margem superior.

overflow: Define se a exibição do conteúdo que ultrapassar os limites de altura e largura ficarão visíveis ou não.

padding: Espaço em torno de um elemento em todos os lados.

padding-bottom: Espaço a partir da margem inferior de um elemento.

padding-left: Espaço à esquerda do elemento.

padding-right: Espaço à direita do elemento.

padding-top: Espaço a partir da margem superior do elemento.

page-break-after: Inserir quebra de página depois de um elemento.

page-break-before: *Inserir quebra de página antes de um elemento.*

position: *Define a posição do elemento. Se ele ficará em relação ao seu superior, ou ao <body>*

text-align: *Alinhamento do texto.*

text-decoration: *Sublinhado, sobrelinhado ou riscado.*

text-indent: *Recuo da primeira linha do parágrafo.*

text-transform: *Transformação para todas maiúsculas, minúsculas ou inicial maiúscula.*

top: *Posição do elemento em relação a parte superior da página.*

vertical-align: *Alinhamento vertical do elemento.*

visibility: *Se elemento é visível ou invisível.*

width: *Largura do elemento.*

Exemplo de mudança de cursor

```

<head>
<title>Untitled Document</title>
<style type="text/css">
<!--
body{
  cursor: crosshair;
}
img{
  cursor: wait;
}
a:link{
  cursor: pointer;
}
-->
</style></head>
  
```

Usando Behaviors



Um behavior ou comportamento, é uma combinação entre um evento e uma ação. Por exemplo: quando o usuário mover o mouse

sobre uma imagem (um evento), esta poderá ser realçada (uma ação). Uma ação consiste de código previamente escrito em JavaScript, que realiza determinadas tarefas, como a abertura da janela de um navegador.

Os eventos são definidos pelos navegadores para cada elemento da página; por exemplo: onMouseOver, onMouseOut e onClick são eventos associados a vínculos na maior parte dos navegadores, enquanto que onLoad é um evento associado a imagens e ao corpo do documento. Os eventos que podem ser utilizados para disparar uma determinada ação variam de acordo com o navegador utilizado.

Ao aplicar um comportamento a um elemento do seu documento, você estará especificando uma ação e o evento que a disparará. Vamos ao painel lateral Tag, escolha a guia Behaviors, como mostra abaixo:

Os rótulos selecionados aparecerão no alto do painel.

Ações (+) exibe uma lista de ações que podem ocorrer. A escolha de uma ação acarretará o aparecimento da caixa de diálogo Parâmetros. Excluir (-) remove uma determinada ação e o evento a ela associada da lista do inspetor de comportamentos.

SHOW EVENTS FOR especifica os navegadores nos quais o comportamento deverá funcionar. A seleção feita neste menu determinará os eventos que aparecerão no menu pop-up Events. Clique no (+) e faça agora esta especificação.

Botões de setas acima e abaixo movem a ação selecionada para cima ou para baixo na lista de comportamentos. As ações serão executadas na ordem especificada.

Events exibe todos os eventos que podem disparar a ação. Os eventos que aparecerão dependerão do objeto selecionado. Se os eventos esperados não aparecerem, certifique-se de que o objeto correto esteja selecionado.

Os diversos navegadores reconhecem de

maneira diversa os eventos relacionados aos vários objetos. Escolha os navegadores nos quais o comportamento deverá funcionar, no menu pop-up EVENTS For: Apenas os eventos reconhecidos pelos navegadores selecionados aparecerão no menu pop-up EVENTS.

Vou destacar alguns behavior interessantes:

Call Javascript: chama uma função ou código que deseja executar.

Change Property: esse é bem legal, ele muda as propriedades de alguns objetos, por exemplo cor de segundo plano de um Layer.

Check Browser: verifica o tipo do navegador do usuário.

Check Plugin: é muito útil para verificar por exemplo se o usuário tem o plugin do flash, se não direcione ele para o site da macromedia e fazer o download do mesmo.

Drag Layer: dependendo do efeito que deseja, este é bem interessante, pois possibilita o usuário arrastar e soltar uma layer (camada).

Go to URL: podemos criar um redirecionamento associado ao evento onload, para páginas que mudaram de endereço.

Open Browser Window: quem nunca sonhou em criar aquelas adoráveis janelas pop-up, quando entramos em um site, abre em média umas 200 (risos). Também serve para ao clicar em um link abrir uma nova janela do navegador.

Set Text: se deseja colocar um texto na barra de status do navegador ou em um frame.

Show Pop-Up Menu: muito bom este behavior, para habilitar, ele tem que ser associado a uma imagem.

Valid Form: esse behavior garante que os campos de formulário sejam realmente preenchidos, fazendo assim uma crítica antes que os dados sejam enviados, avisando quais campos precisam ainda ser preenchido. Eu não esqueci do behavior Timeline, é que nesta versão 2004 ele foi retirado, só mandando um Email-assinado para a macromedia para reclamar. O behavior Show Hide Layers, será abordado no próximo Capítulo.

Exercício

- 1) O que é CSS?
- 2) Quais as formas de mudar um estilo de comando/tag?
- 3) Crie um página html com css e formate 4 comando, com pelo menos 3 atributos.
- 4) Escreva sobre o recurso Show Pop-Up Menu?
- 5) Na configuração de formulário, diferencie as opções GET e POST da propriedade Method.

UNIDADE III

ASP INTRODUTÓRIO

CONSTRUIR UM SITE COM FORMULÁRIOS E VALIDANDO OS DADOS USANDO ASP

INTRODUÇÃO

A Internet é um conjunto de redes de computadores interligados pelo mundo inteiro, que têm em comum um conjunto de protocolos e serviços, de forma que os usuários a ela conectados podem usufruir de serviços de informação e comunicação de alcance mundial tais como: e-mail, servidores Web, ftp, irc, icq etc.

Trata-se da mais bem sucedida aplicação prática do conceito de interoperabilidade, que consiste em conectividade de redes de tecnologias distintas. Isso só foi conseguido graças ao conjunto de protocolos conhecidos como TCP/IP (Transmission Protocol/Internet Protocol).

Mas o que popularizou mesmo a Internet foi a criação da World Wide Web. Trata-se de um serviço para a transmissão multimídia de informações implementado pelo protocolo de aplicação HTTP(Hypertext Transfer Protocol).

Um Cliente HTTP(Browser WEB) se comunica com um servidor HTTP(Servidor WEB) requisitando arquivos. Geralmente esses arquivos estão no formato HTML (Hypertext Markup Language) que pode conter referências para outros arquivos diversos(imagens, sons, vídeos etc). Ao receber o arquivo HTML o cliente verifica cada referência, solicitando ao servidor HTTP os arquivos indicados.

Esse modelo de funcionamento limita bastante o uso da Web uma vez que as páginas HTML têm um conteúdo estático, ou seja, sempre são exibidas da mesma forma e não possibilitam nenhuma interação com o usuário.

Para deixar a Web mais dinâmica e interativa, criou-se o CGI(Common Gateway

Interface). Agora podemos ter programas num servidor Web que podem ser requisitados por um cliente Web. O programa é processado e o resultado desse processamento é enviado pelo servidor Web ao cliente, geralmente no formato HTML. É importante percebermos onde está o dinamismo do CGI: o processamento de tais programas pode retornar diferentes resultados, dependendo dos parâmetros informados pelo cliente(interação) ao programa CGI.

Apesar de dar mais “vida” a web, programas CGI possuem uma série de desvantagens técnicas, sendo a principal delas o fato de tais programas executarem num processo diferente do Web Server. Sendo assim, um servidor web que recebesse várias requisições simultâneas, facilmente se sobrecarregava e parava.

Por isso surgiram, e ainda surgem a cada dia, tecnologias alternativas ao uso do CGI: ISAPI, NISAPI, IDC/HTX, Cold Fusion, Java Server Pages(JSP), Personal Home Page(PHP), Active Server Pages(ASP) etc.

ACTIVE SERVER PAGES

ASP é um tecnologia da Microsoft que disponibiliza um conjunto de componentes para o desenvolvimento de páginas Web dinâmicas. Tais páginas consistem em arquivos de extensão *.asp no formato texto(ASCII) que contém combinações de scripts e tags HTML.

Um servidor Web que suporta ASP funciona da seguinte forma:

- Cliente solicita página *.asp
- Servidor abre a página e lê seu conteúdo
- Se encontra tags HTML, envia direto ao cliente
- Se encontra comandos de script:
- Para o envio
- Processa os comandos
- Envia o resultado HTML ao cliente.

Como todo código de programação existente em páginas Asp é executado no servidor, e este só retorna ao cliente respostas em HTML, aplicações ASP têm seu código fonte totalmente preservado além de poderem

ser acessadas por “qualquer” browser existente no mercado.

Entre os recursos que podem ser implementados com ASP, podemos citar:

- Programação com Visual Basic Script e Java Script
- Acesso a banco de dados
- Envio de e-mail

Para utilizar ASP em suas homepages certifique-se que o computador que a hospedará roda Windows NT Server 4.0(ou superior) com o Internet Information Server 3.0(ou superior). Esse último é um programa Servidor Web da Microsoft. Se pretende usar os recursos de acesso a banco de dados, você precisará de um driver de ODBC instalado e funcionando no servidor.

ASP também “funciona” com o MS Personal Web Server(PWS), para Windows NT WorkStation e para Windows 9x, muito embora essa não seja a plataforma mais recomendada. Para os amantes das plataformas Unix/Linux, já existem módulos no mercado que garantem o suporte ao ASP nessa plataforma.

ALGUNS SITES BRASILEIROS QUE UTILIZAM ASP

Ex.: www.microsoft.com
www.receita.fazenda.gov.br
www.clubedelphi.com.br

INTERNET INFORMATION SERVICES

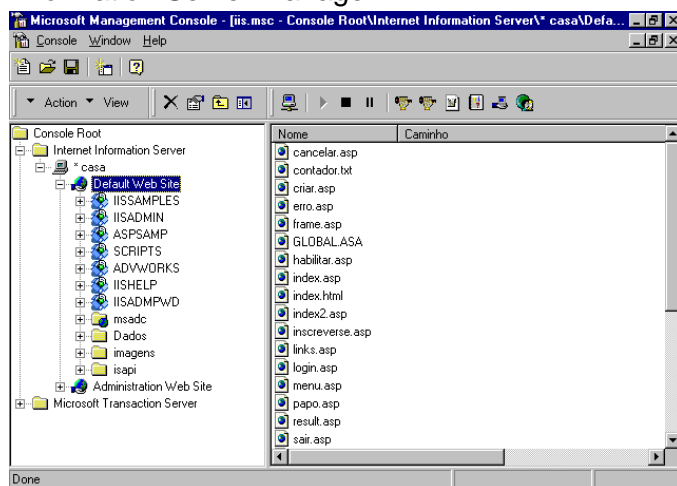
Acesso de Leitura
Acesso de Script
Acesso de Execução
Acesso de Gravação
Pesquisa em pasta

Atualmente na versão 4.0, o IIS é o servidor Web mais recomendada pela Microsoft para desenvolvimento de sites dinâmicos com ASP, pois roda num ambiente mais seguro e robusto:

Windows NT Server. Ele pode ser encontrado no CD-ROOM do Windows NT 4.0 Optional Pack.

Sua instalação é bastante simples, cabendo ao usuário informar apenas quais os componentes do Optional Pack 4.0 deseja instalar e em que diretório serão instalados, bem como informar os diretórios principais para os serviços de publicação WWW e FTP.

A configuração do IIS é feita através do Microsoft Management Console: Iniciar → Programas → Windows NT 4.0 Optional Pack → Microsoft Internet Information Server → Information Server Manager.



A primeira coisa a se fazer é criar um novo Web site. Basta ir no menu Action→New→Site Web, indicar a descrição do novo site, indicar o IP da máquina e o número da porta do servidor Web(80), indicar o caminho para o diretório home do site e, para finalizar, informar as permissões de acesso:

Depois de criar o novo site, você pode alterar suas configurações clicando no mesmo com o botão direito do mouse, escolhendo a opção propriedades:

Opção	Descrição
Site da Web	Configuração do IP, portas de conexão, limite de conexões, Ativação do Log
Operadores	Designar contas administrativas p/ site
Desempenho	Otimização
Filtros ISAPI	Adicionar/Remover filtros ISAPI
Pasta Base	Configurar permissões de acesso e diretório base
Erros Personalizados	Personalizar os erros do Servidor Web
Pasta de Segurança	Configurar autenticação de usuários, segurança de comunicação e restrições a endereços IP
Documentos	Definir arquivos padrões

O IIS também possui o recurso de Diretórios Virtuais. Eles servem para que os usuários Web possam acessar o seu conteúdo, sem precisar saber sua localização física no disco do servidor. Camuflando a estrutura real do disco, nos prevenimos contra possíveis ataques de hackers e facilitamos a vida dos usuários Web, pois se mudarmos a estrutura interna de armazenamento, o endereço virtual não será afetado. Cada diretório virtual criado possui suas próprias configurações de segurança, permissões de acesso, erros personalizados, documentos padrões etc.

Para criar um diretório virtual, clique com o botão direito do mouse no Web Site onde este deverá se localizar → New → Pasta Virtual. Informe então um nome para o diretório virtual, especifique o diretório físico onde estarão as páginas desse diretório, especifique as permissões de acesso de seus usuários. Para que páginas ASP sejam executadas, deve-se pelo menos marcar a permissão de Script. Caso contrário, o servidor Web retornará as páginas ASP desse diretório aos clientes da mesma forma que estão armazenadas (com todos os comandos de scripts visíveis ao usuário).

Quando você desejar publicar uma página na Web basta criar o arquivo ASP ou HTML e salvá-lo em algum diretório físico relacionado a algum diretório virtual. Para acessar esse

arquivo por um navegador, informe o seguinte URL:

**http://
servidor/alias_diretorio_virtual/nome_página**

ROTINAS DE SCRIPT

Script é um programa escrito numa determinada linguagem de programação que não necessita ser compilado para ser posteriormente executado. Scripts são interpretados, ou seja, seus comandos são lidos em tempo de execução por um Script Engine, processados e seus resultados passados para a saída padrão da aplicação (monitor de vídeo, impressora, servidor web etc).

Toda a funcionalidade e “Inteligência” de uma página ASP é controlada através de comandos de Script. Teoricamente, o ASP pode utilizar qualquer Script Engine (interpretador), mas na prática a Microsoft só disponibiliza dois:

- Visual Basic Script (VBScript) - default
- MS Java Script (JScript)

Ao escrevermos páginas *.ASP a primeira coisa que devemos fazer é indicar em qual dessas linguagens disponíveis elas serão escritas:

<% @ LANGUAGE=VBScript %>

ou

**<SCRIPT LANGUAGE="VBScript"
RUNAT=SERVER>...</SCRIPT>**

O Web browser não executará comandos HTML, somente scripts. Mas como ele reconhece um script? Simples, se o Web Server encontrar na página *.ASP a tag **<%** ou **<SCRIPT LANGUAGE="VBScript" RUNAT=SERVER>** ele entende que daquela posição até **%>** ou **</SCRIPT>** existem comandos de scripts a serem executados. Sendo assim, ele os executa e só retornará para o cliente o resultado HTML. É importante observar que o Web Server só tentará

interpretar uma página se a mesma estiver salva com a extensão .asp, caso contrário, o servidor Web enviará a página como se fosse um arquivo de texto normal. Logo, não adianta criar scripts altamente eficientes e esquecer de salvar corretamente o arquivo.

A seguir nossa primeira página ASP. Ela simplesmente retorna a data e hora atual do Servidor Web. Crie o seguinte arquivo chamado now.asp no seu diretório virtual:

Exemplo nº 1 – now.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<%=NOW%> <!--função do vbscript que
retorna hora/data-->
</BODY></HTML>
```

Obs : Tende abrir essa página utilizando o caminho físico da mesma e veja o que acontece quando você tenta visualizar o código fonte. Depois, execute através do URL http://servidor/diretorio_virtual/now.asp e faça o mesmo teste.

VISUAL BASIC SCRIPT (VBSCRIPT)

VBScript é uma linguagem criada a partir do Visual Basic, mas com algumas limitações, por motivos de segurança, além de ser interpretada e não compilada. Dentre outros pontos, permite a manipulação de strings, datas, numéricos e objetos Active X do servidor. Ela é a linguagem de script mais utilizada para desenvolvimento de páginas ASP. Sendo assim, é de fundamental importância conhecermos seus principais comandos antes de prosseguir com o nosso estudo.

Diferente de linguagens como Delphi ou C++, **VBScript só aceita um comando por linha**. O seguinte código retornaria um erro:

Exemplo 2.1: erro_linha.asp

```
<% @ LANGUAGE=VBSCRIPT %>
```

```
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% a = 2 b = a*2 %>
Valor de B = <%=b%>
</BODY></HTML>
```

Executando esse script podemos averiguar que quando a página ASP encontra um erro, o mesmo é bem ilustrado no browser. Isso facilita bastante o trabalho de depuração para os programadores ASP. Mas por outro lado, essa informação de erro não diz muita coisa para o usuário WEB. Temos a opção de tratar os diversos possíveis erros como veremos mais adiante ou simplesmente configurar nosso Web Server para retornar uma mensagem de erro padrão (MS Controle Manager → Diretório Virtual → Propriedades → Pasta Base → Configurações → App Debugging → Script Error Messages → Send Text Error Message To Client → Digite a Mensagem).

Existem duas formas de consertar esse erro: colocando um comando por linha ou separar os comandos por **:(dois pontos)**

Exemplo 2.2: erro_linha2.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% a = 2 : b = a*2 %>
Valor de B = <%=b%>
</BODY></HTML>
```

Um comando não pode existir em mais de uma linha.

Existem duas formas de se escrever essa página corretamente: colocar o comando numa única linha ou usar o caractere **_ (underline)**.

Exemplo 2.3: erro_linha3.asp

```
<% @
LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% texto = "Lupernitácio Jacó " +
"de Oliveira"%>
```

```
Nome = <%=texto%>
</BODY></HTML>
```

Exemplo 2.4: erro_linha4.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% texto = "Lupernitácio Jacó " + _
    "de Oliveira"%>
Nome = <%=texto%>
</BODY></HTML>
```

VARIÁVEIS

São identificadores alfanuméricos que "apontam" para posições de memória onde existem valores armazenados temporariamente, sendo que estes podem ser alterados durante o processamento de uma aplicação. Não nos interessa saber como esse valor será armazenado na memória, nem onde. Basta apenas sabermos o nome e o tipo do valor armazenado em tal variável.

Em VBScript os nomes de variáveis devem começar obrigatoriamente com uma letra e não podem exceder 255 caracteres. Ao contrário da maioria das linguagens de programação, uma variável do VBScript não necessita ser declarada antes de ser utilizada.

Dim Nome_da_Variavel

Exemplo 3.1: var1.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% Dim v1
    v1 = 100
    v2 = 200
    v3 = 300 %>
V1=<%=v1%><BR>
V2=<%=v2%><BR>
V3=<%=v3%>
</BODY></HTML>
```

A mesma variável não pode ser declarada mais de uma vez no mesmo escopo do script: O código a seguir está errado

Exemplo 3.2: var2.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% Dim v1
    v1 = 100
    v2 = 200
    Dim v1 'Redeclaração da Variável v1
    v1 = 900
    v3 = 300 %>
V1=<%=v1%><BR>
V2=<%=v2%><BR>
V3=<%=v3%>
</BODY></HTML>
```

Obs: O "tempo de vida" de uma variável vai desde sua declaração explícita (Dim) ou implícita (sem Dim) até o final do script ou sub-rotina.

Talvez você não perceba isso com exemplos tão simples mas scripts com declarações implícitas de variáveis são mais difíceis de ser entendidos, além de estarmos mais vulneráveis a erros de digitação. Observe o seguinte exemplo e tire suas conclusões.

Exemplo 3.3: var3.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% teste = "Lupernitácio Jacó" %>
    Nome do Usuário =
<%=teste%>
</BODY></HTML>
```

Para evitar esse tipo de erro, podemos utilizar a declaração **Option Explicit**. Ela informa ao interpretador do script que variáveis só poderão ser utilizadas se antes forem declaradas explicitamente.

Exemplo 3.4: var4.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<% Option Explicit %>
```



```
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% Dim teste
    teste = "Lupernitácio Jacó" %>
Nome do Usuário = <%=tste%>
</BODY></HTML>
```

Observe que agora a execução do script retornará um erro, uma vez que a variável **tste** (erro de digitação) não foi explicitamente declarada.

TIPOS DE DADOS

O VBScript contém apenas um tipo de variável chamado de **Variant**. Na verdade, uma variável pode armazenar valores de qualquer tipo. Só que num determinado instante, uma variável possui apenas um tipo implícito. O que determina o subtipo de uma variável é o valor a ela atribuído.

Subtipos
Integer
Long
Single
Double
Date
String
Boolean
Null
Empty
Object

Exemplo 4: tipo.asp

```
<% @ Language = VBScript%>
<% Option Explicit %>
<HTML><HEAD><TITLE>Curso ASP</TITLE>
<BODY>
<% Dim A
    A = "Lupernitácio Jacó de Oliveira" %>
Valor de A como String = <%=A%><BR>
<% A = 2 %>
```

```
Valor de A como Inteiro = <%=A%><BR>
<% A=#02/10/1978# %>
Valor de A como Date = <%=A%><BR>
<% A = #06:30:00# %>
Valor de A como Time = <%=A%><BR>
<% A = 10.20 %>
Valor de A como Real = <%=A%><BR>
<% A = True %>
Valor de A como Booleano = <%=A%><BR>
</BODY>
</HTML>
```

É possível em VBScript declarar uma variável para armazenar mais de um valor: Array. Ao declararmos uma variável array devemos informar seu nome e a quantidade de valores que a mesma pode armazenar:

Dim Nome_Array(Quantidade)

Para acessar determinado valor de uma variável array, informamos o nome e a posição de tal valor. Esse índice começa em 0(zero) e vai até o valor especificado na declaração(Quantidade).

Exemplo 5.1: array1.asp

```
<% @ Language = VBScript%>
<HTML><HEAD><TITLE>Curso ASP</TITLE>
<BODY>
    <% Dim MeuArray(3)
        MeuArray(0)=Date
        MeuArray(1)=" Lupernitácio Jacó de
Oliveira"
        MeuArray(2)=12.45
        MeuArray(3)=Now
    %>
Posição 1 = <%=MeuArray(0)%><BR>
Posição 2 = <%=MeuArray(1)%><BR>
Posição 3 = <%=MeuArray(2)%><BR>
Posição 4 = <%=MeuArray(3)%>
</BODY>
</HTML>
```

Observe que o valor armazenado em cada elemento de um array pode ser de um subtipo diferente dos demais. Outra observação

importante: **arrays têm que ser declarados explicitamente.**

Arrays não são limitados para uma única dimensão. Em VBScript pode-se declarar Arrays de até 60 dimensões.

Exemplo 5.2: array2.asp

```
<% @ Language = VBScript%>
<HTML><HEAD><TITLE>Curso ASP</TITLE>
<BODY>
  <% Dim MeuArray(1,1)
    MeuArray(0,0)= "MacDias"
    MeuArray(0,1)=2225240
    MeuArray(1,0)= "Escola Técnica"
    MeuArray(1,1)=2172000
  %>
  <B>Nome</B>      =      <%=MeuArray(0,0)%>
  <B>Telefone</B>   =
  <%=MeuArray(0,1)%><BR>
  <B>Nome</B>      =      <%=MeuArray(1,0)%>
  <B>Telefone</B>   =
  <%=MeuArray(1,1)%><BR>
</BODY>
</HTML>
```

Pode-se declarar um array cujo tamanho é alterado durante a interpretação do script. Para tal, basta declarar o array com **Dim** sem informar a quantidade de elementos. Depois deve-se utilizar a declaração **ReDim** para determinar o número de elementos. Caso seja necessário redimensionar o array, utiliza-se novamente a declaração **ReDim**. Se houver a necessidade de preservar o conteúdo do array a ser redimensionado, utiliza-se a declaração **ReDim Preserve**.

Exemplo 5.3: array3.asp

```
<% @ Language = VBScript%>
<HTML><HEAD><TITLE>Curso ASP</TITLE>
<BODY>
Sem Preserve <BR>
  <% Dim MeuArray()
    ReDim MeuArray(1)
    MeuArray(0)= " Lupernitácio Jacó de Oliveira
  "
    MeuArray(1)=12.95
    ReDim MeuArray(2)
    MeuArray(2)=9090
  %>
```

```
<B>Posição 0</B> = <%=MeuArray(0)%><BR>
<B>Posição 1</B> = <%=MeuArray(1)%><BR>
<B>Posição 2</B> = <%=MeuArray(2)%><BR>
<HR>
```

Com Preserve


```
  <% ReDim MeuArray(1)
    MeuArray(0)= " Lupernitácio Jacó de
Oliveira "
    MeuArray(1)=12.95
    ReDim Preserve MeuArray(2)
    MeuArray(2)=0909
  %>
  <B>Posição 0</B> = <%=MeuArray(0)%><BR>
  <B>Posição 1</B> = <%=MeuArray(1)%><BR>
  <B>Posição 2</B> = <%=MeuArray(2)%>
</BODY>
</HTML>
```

CONSTANTES

Uma constante representa um valor fixo através de um identificador alfanumérico. Sua diferença para variáveis é que seu valor uma vez definido, não pode ser modificado. Para definir uma função em VBScript utiliza-se a declaração Const:

Const Nome_Constante = Valor_Constante

Exemplo 6 : constante.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
  <% Const nome = "Lupernitácio"
    Const data = #02/10/1978# %>
  <B>Nome: </B><%=nome%><BR>
  <B>Nascido em :</B><%=data%><BR>
</BODY></HTML>
```

OPERADORES

De nada adiantaria termos valores armazenados em nossas variáveis de memória, se não pudéssemos fazer cálculos, comparações ou qualquer outra operação com eles. Em VBScript temos um conjunto de símbolos alfanuméricos para efetuar tais operações:

Operador	Descrição	
=	Atribuição / Igual	também, ou seja, que a soma de um número e uma string alfanumérica resultaria numa concatenação. Verifique o próximo exemplo e tire suas conclusões:
<>	Diferente	
<	Menor que	
<=	Menor ou igual que	
>	Maior que	<u>Exemplo 7.2</u> : operador2.asp
>=	Maior ou igual que	
+	Soma numérica/ Concatenação de Strings	<% @ LANGUAGE=VBSCRIPT %> <% Option Explicit %> <HTML><HEAD><TITLE>Curso ASP</TITLE></HEAD> <BODY> <% Dim a,b a = 1 b = "Alfanumérica" %> A + B = <%=a+b%> </BODY></HTML>
-	Subtração ou negativo Numérico	
*	Multiplicação	
/	Divisão	
\	Efetua a divisão entre dois números e retorna um número inteiro	
Mod	Retorna o Resto de uma divisão entre inteiros	
^	Exponenciação	
&	Concatenação de Strings	
Is	Comparação de Igualdade entre dois Objetos	

Pelo exposto, não fica difícil perceber que as variáveis serão manipuladas de acordo com o seus subtipos. Os valores envolvidos numa mesma operação devem ser do mesmo subtipo ou de subtipos compatíveis.

Exemplo 7.1 : operador1.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<% Option Explicit %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% Dim a,b
    a = 1
    b = "2"
%>
A+B = <%=a+b%><HR>
<% a="Número de Variáveis"%>
A+B = <%=a+b%>
</BODY></HTML>
```

Esse exemplo funciona corretamente pois pela regra de compatibilidade a soma de um número inteiro e uma "string numérica" resulta realmente na soma numérica dos dois. Seria lógico pensar que o contrário estaria correto

Podem ocorrer confusões de outros tipos:

Exemplo 7.3 : operador3.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<% Option Explicit %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY>
<% Dim a,b
    a = "1"
    b = "2" %>
A - B = <%=a-b%><BR>
A + B = <%=a+b%>
</BODY></HTML>
```

Para resolver esses problemas, poderíamos ser mais cautelosos na escrita de nossos programas efetuando exaustivos testes. Outra solução menos "dolorosa" seria utilizarmos conversões de tipos explícitas nos nossos programas.

Função de Conversão	Descrição
CStr	Converte uma expressão para o subtipo String
Cint	Tenta converter uma expressão para o subtipo

	Integer
CLng	Tenta converter uma expressão para o subtipo Long
Cbool	Tenta Converter para Booleano
Cbyte	Tenta converter para o subtipo Byte
Cdate	Tenta converter para o subtipo Date
CDbl	Tenta converter para o subtipo Double
CSng	Tenta converter para o subtipo Single

Exemplo 8.1 : converte1.asp

```
<% @ LANGUAGE=VBSCRIPT %><% Option
Explicit %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY>
<% Dim a,b
      a = "1" : b = "2" %>
A - B = <%=Cint(a)-Cint(b)%><BR>A + B =
<%=Cint(a)+Cint(b)%>
</BODY></HTML>
```

Analisando o próximo programa podemos pensar que o mesmo gera uma exceção:

Exemplo 8.2 : converte2.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<% Option Explicit %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY>
<% Dim a,b
      a = 1.1
      a = CDate(a) %>
<%=a%><BR>
<% a = CDbl(#02/10/2000#) %>
<%=a%>
</BODY></HTML>
```

O erro não ocorreu pois na verdade um subtipo Date é implementado como um número real.

Ele armazena a quantidade de dias desde 31/12/1899. Valores a esquerda do decimal representam a data e a direita o horário. Sendo assim, operações aritméticas também podem ser feitas em cima desse subtipo sem problema algum.

Exemplo 8.3 : converte3.asp

```
<% @ LANGUAGE=VBSCRIPT %>;
<% Option Explicit %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY>
<% Dim Datan,hoje
      Datan = #02/10/1978#
      Hoje = Date%>
Eu tenho <%=CLng(Hoje-Datan)%> dias de
vida<BR>
Farei 10000 dias de vida em
<%=Cdate(Datan+10000)%>
</BODY></HTML>
```

O próximo exemplo faz uso do operador (^) exponenciação:

Exemplo 9.1 : exp1.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<% Option Explicit %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY>
<% Dim Resultado
      Resultado = 2^2^3 %>
2^2^3 = <%=Resultado%>
</BODY></HTML>
```

Esse script dá a falsa impressão de estarmos calculando o valor da expressão da direita para a esquerda, ou seja, $\text{Resultado} = 2^{2^3} = 2^8 = 256$. Mas o que percebemos foi que o resultado gerado é igual a 64, ou seja, $\text{Resultado} = 2^{2^3} = 4^3 = 64$. Para evitar esse tipo de confusão com relação à ordem de cálculo de uma expressão devemos utilizar parênteses. A sub expressão que estiver dentro de parênteses têm prioridade de cálculo em cima das demais.

Exemplo 9.2 : exp2.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<% Option Explicit %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY>
<% Dim Resultado
      Resultado = 2^(2^3) %>
```

2^(2^3) = <%=Resultado%>
</BODY></HTML>

Exercício

- 1) O que é e para que serve?
- 2) O que representa o alias na configuração do IIS?
- 3) Declare variáveis para os dados abaixo:
 - a- Peso de um barril de alumínio
 - b- Salário do Presidente da LLX
 - c- Nome de uma empresa
- 4) Explique com exemplos, a declaração de variável implícita e explícita.
- 5) Explique três funções de conversão de dados.
- 6) Observe o código abaixo e escreva o que será impresso:


```
Dim numero(2),a,b
Numero(0)=1
Numero(1)=2
Numero(2)=0
A=numero(0) + 5
B=numero(1)+numero(0)
Response.write a+b
Response.write " numero(2)"
```

SUBROTINAS

VBScript possui dois tipos de subrotinas: Sub e Function. Um Sub é um conjunto de comandos associados a um identificador alfanumérico. Uma Function possui a mesma definição só que além de executar os comandos a ela associados, pode gerar um valor como resultado. Essas subrotinas podem ainda receber algum(s) valor(s) como parâmetro. Veja a seguir suas respectivas sintaxes:

Sub NomeDoProcedimento([Parâmetro(s)])

Comando(s)

End Sub

Function NomeDaFunção([Parâmetro(s)])

Comando(s)

End Function

Dentro da Function devemos implementar um mecanismo para passar o valor calculado para fora da function. Basta atribuir tal valor ao identificador da Function.

Na verdade, o identificador da Function é uma expressão pois retorna um valor. Para verificarmos o valor retornado por uma Function, colocamos a mesma no lado direito de uma atribuição:

Exemplo 10.1: sub1.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY>
<% Function Soma(a,b)
    Dim Resultado
    Resultado = a + b
    Soma = Resultado
end Function

Dim c,d,e
e = Soma(10,20) %>
Soma(10,20)= <%=e%><BR>
```



```
Soma(100,200)=<%=Soma(100,200)%>
</BODY></HTML>
```

Esse exemplo serve para ilustrar as vantagens de se trabalhar com subrotinas : programa fica mais estruturado e dependendo do número de comandos de uma subrotina, existe uma economia de linhas de código.

Cabem aqui algumas observações. A variável declarada dentro da Function é dita variável local da Subrotina, pois seu valor não pode ser “enxergado” fora da mesma. Por outro lado, as variáveis c,d,e podem ser “vistas” pela Function pois estas são variáveis de escopo global.

Exemplo 10.2: sub2.asp

```
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY>
<% Dim A,B
      A = "Variável Global A"
      B = "Variável Global B"
      MudaB %>
Valor de A = <%=A%><BR>
Valor de B = <%=B%>

<% Sub MudaB()
      Dim A
      A = "Variável A no SUB"
      B = "Variável B no SUB"
End Sub
%>
</BODY></HTML>
```

Observe que o código para o SUB(ou Function) não precisa ser escrito antes de sua chamada.

INCLUDE FILES

Essa é mais uma das formas que existe para poupar trabalho dos programadores economizando linhas de código. A idéia é criar um arquivo texto de qualquer extensão que contenha um conjunto de subrotinas. Essas, estarão disponíveis a qualquer página asp que faça referência a esse arquivo.

Existem duas formas de referenciar tal arquivo numa página ASP:

```
<!-- #INCLUDE
VIRTUAL="Path_Virtual/Nome_Arquivo" --> ou
<!-- #INCLUDE FILE="Path_FÍSICO/Nome_Arquivo"
-->
```

Exemplo 11 : subrotinas.inc

```
<SCRIPT LANGUAGE=VBSCRIPT RUNAT=SERVER>
```

```
Function Dias(Dial,DiaF)
Dias = CDate(DiaF)-CDate(Dial)
End Function
```

```
Function Horas(Dial,DiaF)
Horas = Dias(Dial,DiaF)*24
End Function
```

```
</SCRIPT>
```

Exemplo 11: includefiles.asp

```
<% @ LANGUAGE=VBSCRIPT %>
<!-- #INCLUDE VIRTUAL="curso/subrotinas.inc" -->
<HTML><BODY>
      Eu tenho <%=Dias(#02/10/1978#,Date)%> dias de
vida !<BR>
      Ou seja, mais ou menos
<%=Horas(#02/10/1978#,Date)%> horas de vida!
</BODY></HTML>
```

FUNÇÕES PRÉ-DEFINIDAS

Função	Descrição
Abs(valor_n umérico)	Retorna o módulo de um número
Fix (valor_numé rico)	Retorna a parte inteira de um número
Int(valor_nu mérico)	Retorna a parte inteira de um número. Quando o valor numérico é menor que zero, é retornado o valor imediatamente menor. Ex: Fix(-1,4) = -2 Int(1.4) = 1
Sin(valor_nu)	Retorna o seno de um

mérico)	determinado angulo (radianos)
Cos (valor_numérico)	Retorna o coseno de um determinado angulo (radiando)
Tan (valor_numérico)	Retorna a tangente de um determinado angulo (radiando)
Atn (valor_numérico)	Retorna o arco tangente(radiando) de um determinado valor
Log (valor_numérico)	Retorna o Logaritmo Neperiano de um número
Exp (valor_numérico)	Retorna $e^{\text{valor_numérico}}$
Sqr (valor_numérico)	Retorna a raiz quadrada de um valor numérico maior ou igual a zero
Date	Retorna a Data atual
Time	Retorna a Hora atual
Now	Retorna a Data/Hora atual
Day (valor_data)	Retorna o dia de uma determinada data
Month (valor_data)	Retorna o mês de uma determinada data
Year (valor_data)	Retorna o ano de uma determinada data
Weekday (valor_data)	Retorna o dia da semana no formato numérico de uma determinada data
Hour (tempo)	Retorna a hora de uma determinada expressão de tempo
Minute (tempo)	Retorna os minutos de uma determinada expressão de tempo
Second (tempo)	Retorna os segundos de uma determinada expressão de tempo
TimeSerial(h ora,minuto,s	Retorna uma expressão de

egundo)	tempo
DateSerial(a no,mes,dia)	Retorna caracteres são de data
Asc (caractere)	Retorna o número correspondente na tabela ASCII do caractere informado
Chr (valor inteiro)	Retorna o caractere ASCII correspondente ao valor inteiro informado
Lcase (string)	Converte todos os caracteres de uma string para minúsculas
Ucase(String)	Converte todos os caracteres de uma string para maiúsculas
Len (String)	Retorna o número de caracteres de uma determinada string
InStr (pos_inicial,s tring,substrin g)	Retorna a posição do primeiro caractere de uma substring numa determinada string. Pos_inicial indica a posição da string onde devemos começar a busca. Ex : InStr(1,"Lineu","eu") = 4 InStr(3,"Lineu","eu") = 4
Mid (string,pos_i nicial,Taman ho)	Retorna uma substring a partir de uma string informada , bastando apenas especificarmos qual a posição inicial da substring e seu tamanho. Ex : Mid ("Lineu",4,2) = eu
Left (string,Tama nho)	Corta uma string a partir do lado esquerdo. Tamanho é o número de caracteres da nova string Ex: Left ("Lineu",3) = Lin
Right(string, Tamanho)	Corta uma string a partir do lado direito. Tamanho é o número de caracteres da nova string Ex: Right ("Lineu",2) = eu
Ltrim (String)	Retira caracteres de espaços

	que possam existir no lado esquerdo de uma string Ex : Ltrim(" Lineu S ") = "Lineu S "
Rtrim (String)	Retira caracteres de espaços que possam existir no lado direito de uma string Ex : Ltrim(" Lineu S ") = "Lineu S"
Trim (String)	Retira caracteres de espaço do inicio e do fim de uma string Ex : Trim(" Lineu S ") = "Lineu S"
FormatCurrency(Valor Numérico)	Formata um valor numérico para o padrão moeda configurado no computador servidor.
FormatNumber(Valor Numérico)	Formata um valor numérico para o padrão numérico configurado no computador servidor.

Exemplo 12.1 : funcoes1.asp

```
<% @ Language = VBScript%><% Option Explicit%>
<HTML><HEAD><TITLE>Curso
ASP</TITLE><BODY>
  <% Dim s
    S = " MacDias "
    S = Trim(S) %>
  Caracteres da palavra <%=UCCase(s)%> :<BR>
  1 - <%=Mid(s,1,1)%><BR>    2 -
  <%=Mid(s,2,1)%><BR>
  3 - <%=Mid(s,3,1)%><BR>    4 -
  <%=Mid(s,4,1)%><BR>
  5 - <%=Mid(s,5,1)%><BR>
</BODY></HTML>
```

Verifique no próximo exemplo a ocorrência de erros:

Exemplo 12.2 : funcoes2.asp

```
<% @ Language = VBScript%>
```

```
<% Option Explicit%>
<HTML><HEAD><TITLE>Curso ASP</TITLE>
<BODY>
  <% Dim dia,mes,ano
    dia = 2
    mes = 31
    ano = 2000 %>
  <%=DateSerial(ano,mes,dia) %>
  <BR><%=CDate(dia & "/" & mes & "/" & ano)%>
</BODY></HTML>
```

Exemplo 12.3 : funcoes3.asp

```
<% @ Language = VBScript%>
<% Option Explicit%>
<HTML><HEAD><TITLE>Curso ASP</TITLE>
<BODY>
  Eu          gostaria          de          ganhar
  <%=FormatCurrency(50000)%> por mês!<BR>
  Hoje eu ganho <%=FormatNumber(35/3)%> por
  dia...
</BODY></HTML>
```

Exercício

- 1) Escreva um código para exibir o dia da semana por extenso.
- 2) Defina três comandos da data ou hora.
- 3) Escreva uma utilidade para a função TRIM.
- 4) Qual a função que usamos para retornar a quantidade de caracteres de uma variável do tipo string?
- 5) O que faz a função UCASE?
- 6) Qual a diferença entre Function e Sub?

ESTRUTURAS DE CONTROLE DE DECISÃO

São estruturas embutidas numa linguagem de programação que nos permite executar determinado conjunto de comandos de acordo com uma determinada condição. O primeiro comando do VBScript que se encaixa nessa descrição é o IF:

If <Condição> Then

Comando(s)

End if

Esse comando testa a Condição, se mesma for verdadeira, o bloco de comandos entre If e End If será executado.

Exemplo 13.1 : if1.asp

```
<% @ Language = VBScript%>  
<HTML><HEAD><TITLE>Curso  
ASP</TITLE></HEAD>  
<BODY>
```

```
<% if WeekDay(Date)=1 then %>  
    Hoje é Domingo  
<% end if %>  
<% if WeekDay(Date)<>1 then %>  
    Hoje não é Domingo  
<% end if %>  
</BODY></HTML>
```

O exemplo anterior apesar de correto poderia ser escrito de uma forma mais eficiente. Para tal, devemos utilizar o comando If associado a Else:

If <Condição> Then

Comandos_1

Else

Comandos_2

End if

Quando o interpretador encontra esse comando testa a condição, se for verdadeira, executa o bloco de Comandos_1. Se a condição for

falsa(0), o interpretador pula para Else e executa o bloco de comandos_2.

Exemplo 13.2 : if2.asp

```
<% @ Language = VBScript%>
<HTML>
<HEAD>
<TITLE>Curso ASP</TITLE>
</HEAD>
<BODY>
  <% if WeekDay(Date)=1 then %>
    Hoje é Domingo
  <% else %>
    Hoje não é Domingo
  <% end if %>
</BODY></HTML>
```

Como dá pra perceber, há aqui uma economia de processador uma vez que o teste da condição agora só é realizado uma vez.

Podemos também ter vários If..Then...Else aninhados, sendo que cada clausula Else estará ligada ao If .. Then imediatamente anterior.

Exemplo 13.3 : if3.asp

```
<% @ Language = VBScript%>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY>
  <% if WeekDay(Date)=1 then %>
    Hoje é Domingo
  <% else
    if WeekDay(Date)=7 then %>
      Hoje é Sábado
  <% else %>
    Hoje é dia de trabalhar e estudar
  <% end if
  end if %>
</BODY> </HTML>
```

No exemplo anterior temos um **End IF** para cada **If .. then .. Else**. Isso dificulta bastante a leitura do código por parte do programador. Para melhorar a nossa vida, o VBScript possui uma estrutura derivada da anterior: o **If Then Elseif**. A diferença dessa estrutura para a

anterior é que agora só precisamos de um **endif** ao final de todos os comandos. Sintaxe:

If <Condição> Then

Comandos_1

Elseif <Condição>

Comandos_2

Else

Comando_3

End if

Exemplo 13.4 : if4.asp

```
<% @ Language = VBScript%>
<HTML><HEAD><TITLE>Curso ASP</TITLE>
<BODY>
  <% if WeekDay(Date)=1 then %>
    Hoje é Domingo
  <% elseif WeekDay(Date)=7 then %>
    Hoje é Sábado
  <% else %>
    Hoje é dia de trabalhar e estudar
  <% end if %>
</BODY></HTML>
```

Ainda existe uma estrutura alternativa ao If..Then..Else, mais flexível e mais elegante: o **Select Case**. Sintaxe:

Select Case Expressao

Case Condição

Comandos_1

Case Condição 2

Comandos_2

Case Else

Comandos_3

End Select

O resultado da expressão será comparado com uma série de condições, até encontrar uma que

case com ele. Quando isso ocorre, os comandos que estiverem associados à condição serão executados. Se nenhuma das condições for satisfeita e houver a cláusula Case Else então os comandos associados a ela serão executados.

Exemplo 14 : SelCase.asp

```
<% @LANGUAGE=VBSCRIPT%>

<HTML>

<HEAD><TITLE>Curso de
ASP</TITLE></HEAD>

<BODY>

<%    Dim DiaS

        Select Case WeekDay(Date)

            Case 1

                DiaS = "Domingo"

            Case 2

                DiaS = "Segunda-feira"

            Case 3

                DiaS = "Terça-feira"

            Case 4

                DiaS = "Quarta-feira"

            Case 5

                DiaS = "Quinta-feira"

            Case 6

                DiaS = "Sexta-feira"

            Case Else

                DiaS = "Sábado"

        End Select    %>

Hoje é <%=DiaS%>

</BODY>

</HTML>
```

ESTRUTURAS DE CONTROLE DE REPETIÇÃO

É possível repetir um bloco de instruções dentro de um programa escrito em VBScript. Para

isso existem sete tipos diferentes de estruturas de repetição, denominadas de loop:

Do Until <Condição> Comandos Loop	Executa um bloco de instruções até que a Condição se torne verdadeira
Do Comandos Loop Until <Condição>	Só difere da estrutura anterior pois a condição só é testada no final. Sendo assim, o comando do laço será executado pelo menos uma vez.
Do While <Condição> Comandos Loop	Executa um bloco de comandos enquanto a condição for verdadeira
Do Comandos Loop While <Condição>	Só difere da estrutura anterior pois a condição só é testada no final. Sendo assim, o comando do laço será executado pelo menos uma vez.
While <Condição> Comandos Wend	Mesma coisa que Do While ... Loop
For Variável=limitel to limiteF Step N Comandos Next	A Variável será iniciada com o valor limitel. A cada execução do laço ela é incrementada em N. Quando seu valor ultrapassar de limiteF, o laço se encerra.
For Each Elemento In Coleção Comandos Next	Parecido com a estrutura anterior, só que aqui o bloco de instruções é executado para cada elemento existente numa matriz ou numa coleção de objetos.

Exemplo 15.1 : loop1.asp

```
<% @LANGUAGE=VBSCRIPT%>

<HTML>

<HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
```

<BODY>

Números pares menores que 100:
*

<% Dim i

For i=0 to 100 step 2 %>

<%=i%> *

<% Next

i = 1 %>

<P>Números Impares menos que 100:
*

<% Do While i<=100 %>

<%=i%> *

<% i = i + 2

Loop %>

</BODY>

</HTML>

Exemplo 15.2 : loop2.asp

<% @LANGUAGE=VBSCRIPT%>

<HTML>

<HEAD><TITLE>Curso
ASP</TITLE></HEAD>

<BODY>

<% Dim Nomes(8), Nome

Nomes(0)="Macário"

Nomes(1)="Flávia"

Nomes(2)="Rose"

Nomes(3)="Márcio"

Nomes(4)="Vagner"

Nomes(5)="Flávio"

Nomes(6)="Gilvan"

Nomes(7)="Álvaro"

For each nome in Nomes %>

<%=nome%>

<% Next%>

</BODY></HTML>

Para encerrar esse capítulo sobre laços, falaremos um pouco do comando EXIT. Ele permite que se saia permanentemente de um loop, de uma função ou de um Sub. A tabela seguinte apresenta o seu formato para cada situação:

Exit Do	Dentro de laços que começam com DO
Exit For	Dentro de laços que começam com FOR
Exit Function	Para sair de uma Function
Exit Sub	Para abandonar um Sub

Exemplo 16 : exit.asp

<% @LANGUAGE=VBSCRIPT%>

<HTML>

<HEAD><TITLE>Curso
ASP</TITLE></HEAD>

<BODY>

<% Dim i

i = 0

Do

i=i+1

if i<=10 then %>

<%=CStr(i) & " "%>

<% else

Exit Do

Loop Until False%>

</BODY></HTML>

TRATAMENTO DE ERROS

No VBScript existe um objeto interno utilizado para indicar situações de erro em tempo de

interpretação: o objeto ERR. É um objeto bastante simples que contém somente duas propriedades:

- ✓ **Description** : Mensagem original do erro
- ✓ **Number** : Número associado ao erro

Quando não fazemos uso desse objeto, os usuários do nosso site podem receber mensagens de erro “indecifráveis”. A idéia então é escrever um código capaz de detectar os erros e enviar mensagens amigáveis aos usuários.

Exemplo 17.1 : erro1.asp

```
<% @LANGUAGE=VBSCRIPT%>
<HTML>
<HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
<BODY>
<% Dim i
    For i=10 to 0 step -1 %>
        100 Dividido por <%=i%> =
<%=(100/i)%><BR>
<% next %>
</BODY></HTML>
```

Para usar nosso objeto devemos adicionar o seguinte comando no início do nosso script:

ON ERROR RESUME NEXT . Esse comando informa ao interpretador que se ocorrer algum erro na interpretação de algum comando do script, deve-se descartar esse comando, atualizar o objeto ERR e executar a próxima linha. Basta agora escrever código em qualquer lugar do script para verificar a ocorrência do erro para posterior tratamento.

Exemplo 17.2 : erro2.asp

```
<% @LANGUAGE=VBSCRIPT%>
<HTML>
<HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
<BODY>
<% On Error Resume Next
    Dim i
    For i=10 to 0 step -1 %>
        100 Dividido por <%=i%> =
<%=(100/i)%><BR>
<% next
    if err then %>
        <B>OCORREU UM ERRO
[<%=Err.Number%>] : <%=Err.Description%>
<%end if%>
</BODY></HTML>
```

UNIDADE IV ASP AVANÇADO

USAR OBJETOS INTERNOS DA LINGUAGEM ASP, REDIRECIONAR PÁGINAS, ACESSAR BANCO DE DADOS E USAR SQL.

OBJETOS INTERNOS

Além da programação com VBScript, ASP disponibiliza cinco objetos “internos” que promovem a interação entre nossos scripts e o ambiente. Trata-se de estruturas especiais que possuem propriedades, métodos, eventos, coleções etc. Dentre outras coisas, tais objetos servem para:

- Verificar dados informados pelo clientes Web
- Enviar respostas HTML para tais clientes
- Instanciar objetos ActiveX em seus scripts
- Permitem a comunicação entre clientes conectados ao aplicativo ASP

Objetos internos do ASP são:

Application	Representa um conjunto de páginas de um mesmo diretório virtual
Session	Representa uma Sessão aberta com um Cliente via Web Browser
Server	Representa o Servidor Web em si, permitindo acesso a algumas propriedades do mesmo e a criação de instâncias de Objetos Activex
Response	Representa as respostas HTML enviadas ao cliente
Request	Representa os dados enviados por um formulário HTML ao Servidor Web

APPLICATION

Ao conjunto de páginas ASP de um mesmo diretório virtual damos o nome de Aplicação ASP. Tal aplicação será iniciada na primeira vez que um usuário tentar acessar alguma página desse diretório virtual. Será finalizada quando o servidor web for desligado.

O objeto Application existe para nos possibilitar o armazenamento e recuperação de valores relacionadas a uma aplicação ASP. Com ele podemos criar variáveis de qualquer subtipo cujo valor pode ser acessado ou modificado por qualquer usuário conectado ao diretório virtual.

Para criar uma variável do nível de aplicação, devemos escrever comandos com seguinte sintaxe:

Application(“NOME_DA_VARIAVEL”) = VALOR_DA_VARIAVEL

Uma vez criada, tal variável estará acessível a qualquer usuário da aplicação. Seu valor ficará armazenado até que o servidor web seja desligado.

Como o conteúdo desse tipo de variável pode ser modificado por qualquer usuário conectado à aplicação, poderia haver alguma confusão se vários usuários tentassem alterar esse valor ao mesmo tempo. Para evitar possíveis problemas com a “concorrência”, o objeto application disponibiliza dois métodos: LOCK e UNLOCK.

O primeiro bloqueia as variáveis de nível de aplicação para o usuário que invoca tal método. Se qualquer outro “usuário” tentar acessar variáveis desse nível, ficará esperando até a aplicação ser desbloqueada.

A aplicação só será desbloqueada quando o script que a bloqueou termina sua execução, ou quando ocorre o “TimeOut”, ou quando o script invoca o método UNLOCK.

Ainda relacionado a esse objeto existem dois eventos:

Application_OnStart	Ocorre quando a aplicação é iniciada, ou seja, quando um diretório virtual é acessado pela primeira vez.
Application_OnEnd	Ocorre quando a aplicação é finalizada, ou seja, quando o web server é desligado.

Um evento é uma subrotina automaticamente chamada quando o sistema sofre alguma ação específica. Tais subrotinas não são escritas diretamente nas páginas ASP mas num arquivo a parte nomeado de GLOBAL.ASA.

Sendo assim, quando um diretório virtual for acessado pela primeira vez, o Servidor Web procura em tal diretório a existência desse arquivo. Se encontra, abre o arquivo e procura

a subrotina Application_OnStart para executar seus comandos. A mesma coisa acontece quando desligamos o servidor web, só que ele chama a subrotina Application_OnEnd.

No exemplo a seguir criamos uma variável de nível de aplicação chamada DataHoral para armazenar a Data/Hora em que a aplicação foi iniciada. Outra variável chamada Titulo para armazenar o título da aplicação ASP. E uma variável chamada Correio que armazena o e-mail do Web Master:

Exemplo 18.1: Global.asa

```
<SCRIPT          LANGUAGE=VBSCRIPT
RUNAT=SERVER>
```

```
Sub Application_OnStart()
```

```
    Application("DataHoral")=Now
```

```
    Application("Titulo")="Curso de ASP"
```

```
Application("Correio")="mailto:macario@macdia
s.com.br"
```

```
End Sub
```

```
</SCRIPT>
```

Exemplo 18.2 : Application1.asp

```
<% @LANGUAGE=VBSCRIPT %>
```

```
<HTML><HEAD><TITLE><%=Application("Titul
o")%></TITLE></HEAD>
```

```
<BODY>
```

```
Essa aplicação ASP foi iniciada em
<B><%=Application("DataHoral")%></B><BR>
```

```
<A Href="<%=Application("Correio")%>">Web
Master</a>
```

```
</BODY>
```

```
</HTML>
```

Obs : Só poderá existir um arquivo Global.asa em cada diretório virtual

Observe que o trabalho de manutenção do site pode ficar facilitado. Imagine que todas as

páginas asp do seu diretório virtual possuem um padrão de cores, links, cabeçalho, etc. Sendo assim, as páginas teriam muito código em comum. Se desejarmos modificar os padrões do nosso site, teríamos que fazer alterações em todos os arquivos do diretório virtual. Mas se utilizarmos variáveis de nível de aplicação para armazenar essas configurações, não precisamos mudar todos os arquivos do diretório, mas só o arquivo GLOBAL.ASA.

Exemplo 18.3: Global.asa

```
<SCRIPT          LANGUAGE=VBSCRIPT
RUNAT=SERVER>
```

```
Sub Application_OnStart()
```

```
    Application("DataHoral")=Now
```

```
    Application("Titulo")="I Curso de ASPI"
```

```
    Application("Correio")=mailto:
macario@macdias.com.br
```

```
    Application("CorFundo")="Black"
```

```
    Application("CorTexto")="Yellow"
```

```
    Application("TamFonte")="4"
```

```
End Sub
```

```
</SCRIPT>
```

Exemplo 18.4: Application2.asp

```
<% @LANGUAGE=VBSCRIPT %>
```

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

```
<%=Application("Titulo")%>
```

```
</TITLE></HEAD>
```

```
<BASEFONT
```

```
SIZE=<%=Application("TamFonte")%>
```

```
COLOR=<%=Application("CorTexto")%>>
```

```
<BODY
```

```
BGCOLOR=<%=Application("CorFundo")%>>
```

```
Essa aplicação ASP foi iniciada em
<B><%=Application("DataHoral")%></B><BR>
```

```
<A Href="<%=Application("Correio")%>">Web
Master</a></BODY></HTML>
```

SESSION

Toda vez que um usuário Web se conecta a um aplicativo ASP é iniciada uma sessão para o mesmo no servidor Web. Para representar tal

sessão, o ASP possui um objeto interno chamado Session.

Na verdade, ele é muito parecido com o objeto Application. A diferença está em dizer que esse objeto pode armazenar valores ligados apenas a um único visitante do site (o dono da sessão). Com ele podemos criar variáveis de qualquer subtipo cujo valor pode ser acessado ou modificado somente pelo “dono” da sessão.

Para criar uma variável do nível de sessão, devemos escrever comandos com seguinte sintaxe:

Session(“NOME_DA_VARIAVEL”) = VALOR_DA_VARIAVEL

As variáveis de sessão permanecerão na memória (ativas) até a sessão ser encerrada. Isso pode acontecer quando o usuário fechar o web browser, quando ocorre o “TIMEOUT” da sessão, ou quando o script invoca o método ABANDON do objeto Session.

A propriedade TIMEOUT é usada quando o usuário fica parado sem fazer nada no Browser. O default são vinte minutos, mas esse valor pode ser modificado da seguinte forma:

Session.Timeout = VALOR_MINUTOS

Ainda relacionado a esse objeto existem dois eventos:

Session_OnStart	Ocorre quando a sessão é iniciada
Session_OnEnd	Ocorre quando a sessão é finalizada.

A exemplo dos eventos do objeto Application, eles também devem ser escritos como subrotinas de um arquivo GLOBAL.ASA.

O exemplo ilustra o conceito de sessão. Temos uma variável a nível de aplicação chamada contador. Ela serve para informar a quantidade de pessoas que acessaram essa aplicação ASP. A ideia é incrementar o valor dessa variável toda vez que uma sessão é iniciada. Também utilizamos uma variável de sessão que informa a hora em que a sessão em questão foi aberta.

Exemplo 19.1: Global.asa

```
<SCRIPT LANGUAGE=VBSCRIPT
RUNAT=SERVER>
```

```
Sub Application_OnStart()
    Application(“DataHoral”) = Now
    Application(“Titulo”) = “I Curso de ASP”
    Application(“Correio”) = mailto:
macario@macdias.com.br
    Application(“CorFundo”) = “Black”
    Application(“CorTexto”) = “Yellow”
    Application(“TamFonte”) = “4”
    Application(“Contador”) = 0
End Sub
```

```
Sub Session_OnStart()
```

```
Application(“Contador”) = Application(“Contador”)
+1
```

```
Session(“HoraS”) = Time
```

```
End Sub
```

```
</SCRIPT>
```

A página a seguir encerra a sessão através do método ABANDON:

Exemplo 19.2 sessao1.asp

```
<% @LANGUAGE=VBSCRIPT %>
```

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

```
</TITLE></HEAD>
```

```
Essa sessão foi iniciada às
<%=Session(“HoraS”) %><BR>
```

```
Você é o visitante de número
<%=Session(“Contador”) %><BR>
```

```
Desde <%=Application(“DataHoral”) %>
```

```
<HR>
```

```
<A HREF=“sessao2.asp”>Encerrar Sessão</A>
```

```
</BODY>
```

```
</HTML>
```

RESPONSE

Antes de uma página ASP ser enviada ao cliente o Web Server lê seu conteúdo.

Encontra-se TAGS HTML ou texto, envia diretamente ao cliente. Encontra-se scripts, executa-os e repassa o resultado HTML para o Web Browser. Para representar essas respostas HTML, ASP possui o objeto interno RESPONSE.

Com esse objeto podemos enviar simples comandos HTML ou texto, podemos redirecionar o browser para buscar informações em outro URL, podemos bufferizar a resposta ao cliente, bem como interromper o envio da página ASP.

Para tanto, esse objeto possui as seguintes propriedades e métodos:

Write Texto	Com esse método podemos enviar qualquer texto ao web browser.
End	Método que termina o envio da página ASP ao cliente, mesmo que ela não tenha chagado ao final.
Buffer = Valor Booleano	Quando Buffer=True, o servidor web só enviará a página ASP ao cliente quando encerrar toda a leitura da mesma, ou quando utilizarmos nesse script o método Flush. Essa propriedade só pode ser utilizada antes de qualquer comando que gere respostas HTML ao cliente.
Redirect URL	Permite redirecionar o Browser do Cliente para outra página(URL). Se a propriedade Buffer for diferente de True, esse método só poderá ser chamado antes de qualquer comando que gere respostas HTML ao cliente.
Clear	Esse método apaga todo o conteúdo do Buffer se o mesmo estiver ativo.
Flush	Esse método pode ser utilizado para enviar o conteúdo do Buffer para o cliente WEB.
Cookies ("nome_cookie") = valor	Cria cookies ou altera seu valor. Se a propriedade Buffer for diferente de True, essa propriedade só poderá ser chamado antes de qualquer comando que gere respostas HTML ao cliente.
Expires = minutos	Essa propriedade informa o tempo(minutos) em que uma

	página ASP pode permanecer ativa na Cache do Web Browser.
ExpiresAbsolute = #data hora#	Essa propriedade informa a data e a hora exata em que a página expira da Cache do Web Browser.
AppendToLog Texto	Esse método escreve um texto no arquivo de LOG do Web Server.

Com o objeto response podemos escrever uma página ASP 100% script:

Exemplo 20.1 : Response1.asp

```
<SCRIPT LANGUAGE=VBSCRIPT
RUNAT=SERVER>

RESPONSE.WRITE
"<HTML><HEAD><TITLE>Curso de
ASP</TITLE></HEAD>"

RESPONSE.WRITE
"<BODY><CENTER>Testando objeto
Response</CENTER>"

RESPONSE.WRITE "</BODY></HTML>"

</SCRIPT>
```

Podemos também escrever uma página ASP que simplesmente Redireciona o usuário para outro site:

Exemplo 20.2 : Response2.asp

```
<% @LANGUAGE=VBSCRIPT %>

<% RESPONSE.REDIRECT
"http://www.agenciasdorio.com"%>
```

Quando utilizamos Buffer=True, temos mais controle do que será enviado como resposta ao cliente:

Exemplo 20.3 Response3.asp

```
<% @LANGUAGE=VBSCRIPT %>

<% RESPONSE.BUFFER=TRUE

DIM Inicio,Fim
```

```
Inicio = "<HTML><HEAD><TITLE>Curso de
ASP</TITLE></HEAD><BODY>"
```

```
Fim = "</BODY></HTML>"
```

```
RESPONSE.WRITE Inicio
```

```
RESPONSE.WRITE "Esse texto não será
enviado ao Cliente!"
```

```
RESPONSE.WRITE Fim
```

```
RESPONSE.CLEAR
```

```
RESPONSE.WRITE Inicio & Chr(13)
```

```
RESPONSE.WRITE "Esse texto será enviado
ao cliente!" & Chr(13)
```

```
RESPONSE.WRITE Fim
```

```
%>
```

FORMULÁRIOS HTML

Antes de continuar com os objetos internos do ASP devemos começar a nos preocupar com a construção de páginas interativas: os formulários Web. São interfaces HTML utilizadas para obter dados de um usuário Web para um programa que roda no Web Server.

Num formulário podemos ter os seguintes objetos: caixas de texto, botões, comboboxes, checked boxes e radio buttons. A idéia é permitir que o usuário digite dados ou escolha opções e com um simples clique num objeto botão, tais dados serão passados para um determinado programa do Web Server.

Para montar um formulário HTML usamos a Tag <FORM> </FORM>. Essa tag possui ainda alguns atributos:

METHOD

Indica o método pelo quais os dados informados no formulário serão passados ao programa do Web Server.

Quando METHOD=POST, os dados serão passados junto com a requisição do cliente.

Quando METHOD=GET, os dados serão passados depois da requisição do cliente.

ACTION

Indica a URL da aplicação do servidor Web.

NAME

Indica o nome do formulário em questão. (Opcional)

Depois dessa tag devemos informar quais são os objetos de interface do formulário.

CAIXA DE TEXTO SIMPLES

```
<INPUT      TYPE=Tipo      NAME=NomeC
VALUE=Valor      MAXLENGTH=Tam_Max
READONLY>
```

- Tipo: PASSWORD(só aparecem asteriscos), TEXT(aparecem caracteres normais)
- NomeC : nome da caixa de texto em questão;
- Valor : texto que aparece na caixa (opcional);
- Tam_Max : indica quantos caracteres podem ser digitados(Opcional);
- READONLY : indica que a caixa de texto é somente leitura (Opcional);

Exemplo 21.1 Form1.asp

```
<% @ LANGUAGE= VBSCRIPT %>
```

```
<HTML><HEAD><TITLE>Curso      de
ASP</TITLE></HEAD>
```

```
<BODY>
```

```
<FORM ACTION="form2.asp" METHOD=GET>
```

```
  Usuário<BR>
```

```
  <INPUT      TYPE=TEXT      NAME="user"
VALUE="MacDias" MAXLENGTH=10><BR>
```

```
  Senha<BR>
```

```
  <INPUT TYPE=PASSWORD NAME="senha"
MAXLENGTH=10><BR>
```

```
  Data<BR>
```

```
  <INPUT      TYPE=TEXT      NAME="Data"
VALUE="<%=Date%>" READONLY>
```

```
</FORM>
```

```
</BODY>
```

```
</HTML>
```

ÁREA DE TEXTO

```
<TEXTAREA NAME=NomeA COLS=NumCols
ROWS=NumLin READONLY>
```

Texto Default

```
</TEXTAREA>
```

- NomeA : Nome da área de texto;
- NumCols: Quantidade de colunas;
- NumLin : Quantidade de linhas;
- READONLY : indica que a área é somente leitura (Opicional);
- Texto Default : texto que aparece na área;

Exemplo 21.2 Form2.asp

```
<% @ LANGUAGE= VBSCRIPT %>
<HTML><HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
<BODY>
<FORM ACTION="form2.asp" METHOD=GET>
<TEXTAREA NAME="Obs" COLS=50
ROWS=5>
<% For i=0 to 100
Response.Write "Linha número " & i &
chr(13)
Next %>
</TEXTAREA>
</FORM>
</BODY>
</HTML>
```

CAIXAS DE COMBINAÇÃO

```
<SELECT SIZE=Tam NAME=NomeCC>
<OPTION VALUE=valor_passado
SELECTED> Valor Mostrado </OPTION>
<\SELECT>
```

- Tam :Quantidade de linhas (default=1);
- NomeCC : Nome do objeto;

- Valor Passado : valor correspondente a essa opção;
 - SELECTED: Indica que essa será a opção Default;
 - Valor Mostrado : valor mostrado no Web Browser;
- Obs : Podem existir várias opções..

Exemplo 21.3 Form3.asp

```
<% @ LANGUAGE= VBSCRIPT %>
<HTML><HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
<BODY>
<FORM ACTION="form2.asp" METHOD=GET>
<SELECT SIZE=1 NAME="Ano">
<OPTION VALUE=1998>1998</OPTION>
<OPTION VALUE=1999>1999</OPTION>
<OPTION VALUE=2000
SELECTED>2000</OPTION>
<OPTION VALUE=2001
>2001</OPTION>
</SELECT>
</FORM>
</BODY></HTML>
```

BOTÕES DE SELEÇÃO E CHECAGEM

```
<INPUT TYPE=tipo NAME=NomeR
VALUE=Valor CHECKED>
```

- Tipo : RADIO(Marca somente um) CHECKBOX(pode marcar mais de um)
- NomeR : nome do objeto;
- Valor : Valor relacionado ao botão de seleção;
- CHECKED: deixa o botão selecionado quando o formulário for carregado.

A idéia é ter vários objetos desses com o mesmo NAME.

Exemplo 21.4 Form4.asp

```
<% @ LANGUAGE= VBSCRIPT %>
<HTML><HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
<BODY>
<FORM ACTION="form2.asp" METHOD=GET>
    Sexo<BR>
    Homem <INPUT TYPE=RADIO
NAME="Sexo" VALUE="HOMEM"><BR>
    Mulher <INPUT TYPE=RADIO
NAME="Sexo" VALUE="MULHER"><BR>
    Indeciso <INPUT TYPE=RADIO
NAME="Sexo" VALUE="HOMO"> <P>
    Ocupação <BR>
    Estuda<INPUT TYPE= CHECKBOX
NAME="Ocp" VALUE="E"><BR>
    Trabalha<INPUT TYPE= CHECKBOX
NAME="Ocp" VALUE="T">
</FORM>
</BODY>
</HTML>
```

BOTÕES

<INPUT TYPE=tipo VALUE=Valor>

Tipo : SUBMIT (Chama página indicada em ACTION) RESET(Limpa formulário)

Valor : Rótulo do Botão

Exemplo 21.5 Form5.asp

```
<% @ LANGUAGE= VBSCRIPT %>
<HTML><HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
<BODY>
<FORM ACTION="form5.asp" METHOD=GET>
```

```
<INPUT TYPE=TEXT NAME="user"
MAXLENGTH=10><BR>
<INPUT TYPE=SUBMIT VALUE="Enviar">
<INPUT TYPE=RESET VALUE="Limpar">
</FORM>
</BODY>
</HTML>
```

Quando pressionamos o botão Enviar observamos que a página é recarregada. Isso acontece porque o ACTION do nosso formulário é igual a Form5.asp. Se olharmos mais detalhadamente, a caixa de URL do browser vai conter um endereço mais ou menos parecido

Isso acontece pois METHOD = GET, user é o nome do nosso objeto caixa de texto do formulário e Lineu foi o valor digitado. Percebeu a importância que tem em nomearmos os objetos dos nossos formulários?

REQUEST

Esse objeto serve para possibilitar a captura em nossas páginas ASP dos dados passados pelos formulários HTML ao Servidor Web.

Existem três formas diferentes de fazermos essa captura, dependendo do METHOD:

GET	Request.QueryString(Nome_Objeto)
POST	Request.Form(Nome_Objeto)
GET OU POST	Request(Nome_Objeto)

Exemplo 22.1 Request1.asp

```
<% @ LANGUAGE= VBSCRIPT %>
<HTML><HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
<BODY>
<FORM ACTION="Request2.asp"
METHOD=GET>
Usuário <INPUT TYPE=TEXT NAME="user"
MAXLENGTH=10><BR>
```



```
Senha      <INPUT      TYPE=PASSWORD
NAME="senha" MAXLENGTH=10><BR>
<INPUT TYPE=SUBMIT VALUE="Enviar">
<INPUT TYPE=RESET VALUE="Limpar">
</FORM>
</BODY>
</HTML>
```

Request2.asp

```
<% @ LANGUAGE=VBScript %>
<HTML><HEAD><TITLE>Curso      de
ASP</TITLE></HEAD>
<BODY>
Usuário      =
<%=Request.QueryString("user")%><BR>
Senha = <%=Request("senha")%>
</BODY>
</HTML>
```

Observe que o método GET não é adequado para formulário que exigem alguma informação secreta(senha) pois o que foi digitado no mesmo é mostrado na caixa de endereço do Browser. Sendo assim, procure adaptar o exemplo anterior ao método POST.

Com o objeto Request somos capazes também de:

Ler Cookies	Request.Cookies("NomeCookie")
Saber o endereço IP do usuário	Request.ServerVariables("REMOTE_ADDR")
Saber o METHOD utilizado	Request.ServerVariables("REQUEST_METHOD")

Exemplo 22.2 Request3.asp

```
<% @ LANGUAGE=VBScript %>
<HTML><HEAD><TITLE>Curso      de
ASP</TITLE></HEAD>
<BODY>
Endereço IP do usuário :
```

```
<%=Request.servervariables("REMOTE_ADDR")%>
</BODY>
</HTML>
```

Obs : Como Request.QueryString, Request.Form, Request.Cookies, Request.ServerVariables possuem coleções de variáveis, as mesmas podem ser lidas com um comando For Each.

Exemplo 22.3 Request4.asp

```
<% @LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Curso      de
ASP</TITLE></HEAD>
<BODY>
<%FOR      EACH      ITEM      IN
Request.ServerVariables
Response.write ITEM & " - " &
Request.ServerVariables(ITEM) & "<BR>"
NEXT%>
</BODY></HTML>
```

Exercício

- 1) Quais as três formas para resgatar dados de um formulário.
- 2) O que faz o comando `response.redirect`?
- 3) Diferencie variável local de variável de sessão.
- 4) Escreva uma aplicação para variável de sessão.
- 5) O que faz o comando `response.end`?
- 6) Qual a importância o nome de um objeto de formulário na hora de resgatar os dados?
- 7) Resgate o dado do objeto "txtnum" e abra a página `par.html` se o conteúdo da caixa de texto for um número par, ou abra a página `impar.html`, se for um número ímpar.

SERVER

Representa o Servidor Web, permitindo acesso a algumas de suas propriedades. Além disso, ele possibilita a instânciação em páginas ASP de componentes ActiveX escritos por outros programadores.

Os objetos ActiveX fazem parte da tecnologia COM para criação de componentes de software reutilizáveis. A ideia é desenvolver objetos com esse padrão para que possam ser reutilizados por "qualquer" linguagem de programação.

Em particular, o ASP pode trabalhar com componentes COM, só que os mesmos não podem possuir interface visual. Sendo assim, em páginas ASP usamos os chamados ActiveX DLL's, componentes que não fornecem nenhuma interface com o usuário, somente com aplicativos (métodos, propriedades).

Esses componentes estendem bastante nosso poder de programação. Dentre outras coisas, eles nos permitem:

- Acessar Banco de Dados
- Criar/Ler/Alterar arquivos
- Enviar e-mail

ScriptTimeout	Propriedade que determina o tempo máximo(segundos) que um script ASP pode ficar executando, antes que o Servidor Web o termine. Isso serve para proibir que scripts ASP fiquem executando "infinitamente".
HTMLEncode	Método que codifica uma string para o formato HTML.
MapPath	Método que retorna o PATH real de um determinado diretório virtual do servidor Web.
URLEncode	Esse método transforma uma string para o formato padrão de URL.
CreateObject	Cria uma instância de um componente ActiveX na página ASP: Set Obj =

	Server.CreateObject("IDObjeto")
--	---------------------------------

Para exemplificar o uso do objeto Server trabalharemos com o ActiveX FileSystemObject. Tal componente possui uma serie de propriedades e métodos para manipulação de arquivos e diretórios do servidor Web.

Para instanciar tal objeto numa página ASP escrevemos o seguinte código:

Dim Objeto

```
Set Objeto = Server.CreateObject("Scripting.FileSystemObject")
```

Esse objeto recém instanciado representa o Sistema de Arquivo do servidor Web. Devemos agora criar outro objeto(TextStream), a partir desse, para representar um determinado arquivo. Fazemos isso, utilizando o método OpenTextFile do FileSystemObject:

```
Set Arquivo = Objeto.OpenTextFile(Nome,modo,cria,formato)
```

Onde:

Nome	Nome do arquivo a ser utilizado pelo script
Modo	Modo de Abertura do arquivo. 1 para leitura, 2 para gravação por cima, 3 para gravação adicional.
Cria	Valor Booleano que indica se o arquivo deve ser criado(true) ou não(false) caso não exista.
Formato	Indica o formato de gravação do arquivo a ser utilizado. -1 Unicode, 0 Ascii

Para lermos o conteúdo de um arquivo, podemos utilizar os seguintes métodos do objeto TextStream:

Read (quantidade)	Lê um determinado número de caracteres do arquivo
ReadLine	Lê uma linha inteira do

	arquivo
ReadAll	Lê o arquivo inteiro de uma só vez

Mas se nos interessar gravar informações no arquivo, os métodos disponíveis são:

Write	Grava uma string no arquivo
WriteLine	Grava um string no arquivo, incluindo a quebra de linha
WriteBlankLines	Grava um determinado número de linhas em branco num arquivo

Esse componente ActiveX ainda possui as seguintes propriedades:

AtEndOfLine	Indica o fim de uma determinada linha do arquivo
AtEndOfStream	Indica o Final do Arquivo
Column	Indica em que coluna do arquivo estamos
Line	Indica o número da linha atual do arquivo

O código a seguir mostra como abrimos um arquivo localizado no servidor WEB e exibimos seu conteúdo:

Exemplo 23.1 : Arquivo1.asp

```
<% @Language=vbScript %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY><CENTER>
<% dim final
final = "</CENTER></BODY></HTML>"
On Error Resume Next
Set Obj = Server.CreateObject("Scripting.FileSystemObject")
Set arquivo = Obj.OpenTextFile("D:\pessoas.txt",1)
```

```

if Err then
    Response.write "Ocorreu um erro tentando
    abrir o arquivo!"
    Response.write final
    Response.End
end if
Response.Write "Lista de E-mails<BR>"
Response.write "<HR>"
do while arquivo.AtEndOfStream=false
    a = arquivo.Readline
    response.write a & "<BR>"
    a = arquivo.Readline
    response.write a & "<HR>"
loop
Response.Write "Nova Entrada"
Response.Write "<FORM
ACTION=Arquivo2.asp>"
Response.Write "NOME : <INPUT
TYPE=TEXT NAME=NOME><BR>"
Response.Write "EMAIL : <INPUT
TYPE=TEXT NAME=EMAIL><BR>"
Response.Write "<INPUT TYPE=SUBMIT
VALUE=ENVIAR></FORM>"
Response.Write final
arquivo.close
%>

```

A próxima página ASP mostra como escrever num arquivo localizado no servidor WEB. Lembrando que essa página deve ser acessada por um formulário WEB com um objeto de NAME=Nome e outro NAME=Email:

Exemplo 23.2 Arquivo2.asp

```

<% @Language=vbScript %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD>
<BODY><CENTER>
<% dim final
    final = "</CENTER></BODY></HTML>"
    On Error Resume Next
    Set Obj =
Server.CreateObject("Scripting.FileSystemObj
ect")
    Application.Lock
    Set arquivo =
Obj.OpenTextFile("D:\pessoas.txt",8)

```

```

if Err then
    Response.write "Ocorreu um erro tentando
    abrir arquivo!"
    Response.write final
    Response.End
end if
Arquivo.WriteLine(Request("Nome"))
Arquivo.WriteLine(Request("Email"))
arquivo.close
if Err then
    Response.write "Ocorreu um erro tentando
    gravar no arquivo!"
    Response.write final
    Response.End
else
    Response.Write "Dados inseridos com
    sucesso!" & "<BR>"
    Response.Write "Nome:" &
Request("Nome") & "<BR>"
    Response.Write "Email : " &
Request("Email")
    end if
    Response.Write final
%>

```

ACESSANDO BANCO DE DADOS

É possível criar páginas ASP que tenham a habilidade de recuperar ou alterar informações em um banco de dados. Para tal, devemos utilizar o componente Microsoft Data Access que já vem instalado com o IIS.

Para implementar esse acesso a um banco de dados relacional o Data Access utiliza componentes inseridos no pacote ADO(Activex Data Object), uma tecnologia baseada no ODBC(Open DataBase Connectivity).

ACESSANDO ACCESS

```

<%
1 DIM CONEXAO,TABELA
2 SET CONEXAO =
SERVER.CREATEOBJECT("ADODB.CONNECTION")
3 conexao.open
"PROVIDER=Microsoft.ACE.OleDB.12.0;
Data Source=" &
server.mappath(".") &
"/nomedobanco.accdb"
4 SET TABELA = CONEXAO.EXECUTE
("commando SQL")

```

ACESSANDO POSTGRE

```

1 Set Conexao =
Server.CreateObject("ADODB.Connection")
2 conexao.CursorLocation = 3

```

```

3 conexao.Open
"DRIVER={PostGreSQL};SERVER=localhost;D
ATABASE=DATABASE;UID=postgres;PWD=PWD;
OPTION=3;CursorType=1"
4 Set tabela = conexao.execute("SQL")
  
```

ACESSANDO SQL SERVER 12

```

1 Set Conexao =
Server.CreateObject("ADODB.Connection")
2 conexao.CursorLocation = 3
3 conexao.Open
"Provider=SQLNCLI11;Server=IPSERVIDOR;D
atabase=BASEDEDEADOS;Uid=USUARIO;Pwd=SEN
HA;"
  
```

4 Set tabela = conexao.execute("SQL")

ODBC

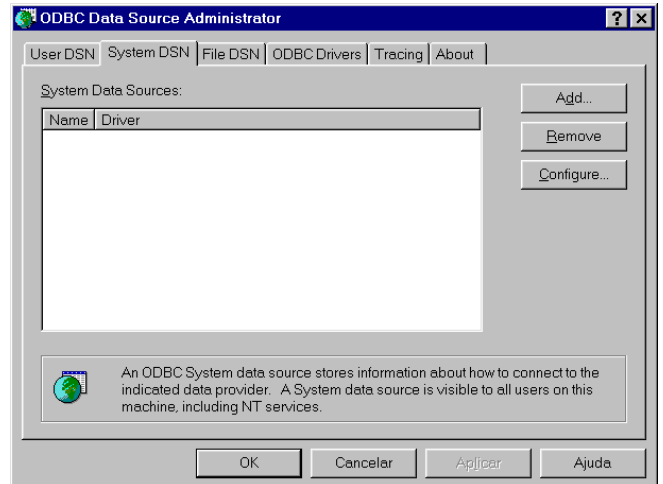
O ODBC é o meio mais conhecido para acessar um banco de dados em ambiente Windows. Ele facilita a vida do desenvolvedor mascarando as particularidades de implementação dos Banco de Dados com que os mesmos irão trabalhar.

A idéia é que cada fabricante de banco de dados crie um drive ODBC. Sendo assim, quando um programa tenta acessar determinado banco de dados via ODBC o windows procura um drive apropriado para este, verifica como deve proceder e então efetua as operações requisitadas.

Antes de colocar a mão na massa e desenvolver páginas ASP com acesso a banco de dados devemos:

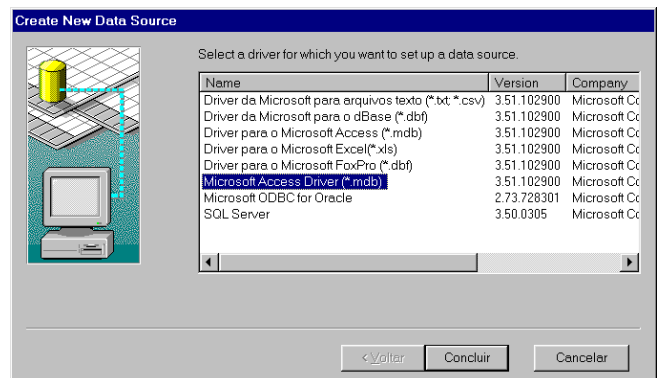
- ✓ Criar um Banco compatível com ODBC
- ✓ Registrar o Banco como uma origem de dados ODBC

Esse último passo permite que aplicações que acessem Banco de Dados via ODBC precisem apenas saber o nome dessa origem de dados, dispensando-se as preocupações com a localização exata e o tipo de banco de dados acessado.



Para criar uma origem de dados devemos executar o programa ODBC do Pannel de Controle do Windows.

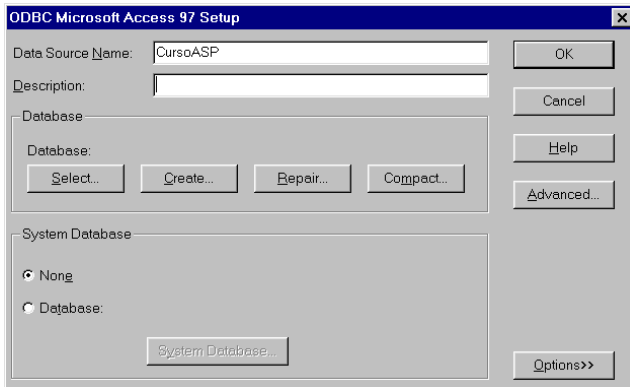
Geralmente para uma aplicação Web é criada uma Origem de dados na opção System DSN. Clicando no botão adicionar temos:



Aqui devemos escolher o driver ODBC a ser utilizado para acessar o Banco de Dados. Para o nosso exemplo usaremos o Microsoft Access. O nome do nosso Banco será Curso.mdb e terá uma tabela chamada Pessoal com os campos:

Código	AutoIncremental (Chave Primária)
Nome	Texto(50) (Não Nulo)
Email	Texto(50)
DataN	Data/Hora

O nome da nossa Fonte de Dados será CursoAsp.



Nessa janela especificaremos o nome da nossa fonte e o seu PATH(Select).

Feito isso, nossa fonte de dados foi criada. Quando nossas aplicações necessitarem abrir tal banco de dados, basta informar agora o nome dessa fonte(CursoASP).

ADO

Para utilizar banco de dados em nossas páginas ASP devemos instanciar os objetos do ADO:

- ✓ **Connection** : representa uma conexão ativa com um banco de dados ODBC. Esse é o componente mais importante do ADO, sendo que os demais só poderão ser criados a partir dele.
- ✓ **Command** : representa um comando a ser executado pela fonte de dados ODBC. Com ele podemos criar comandos SQL compilados no servidor(preparados).
- ✓ **Recordset**: representa um conjunto de registros resultantes do processamento de um comando SQL em um objeto Connection. É esse objeto que permite a navegação numa tabela ou consulta do banco de dados.
- ✓ **Fields** : Representa os campos de um RecordSet.

O primeiro objeto que deve ser instanciado é o Connection:

Set Conexao =
Server.CreateObject("ADODB.Connection")

Após sua definição, precisamos ativar a conexão com a fonte de dados ODBC. Para isso, usamos o método Open do Connection. Sintaxe:

ObjConexao.Open FonteODBC , Usuário,
senha

Onde :

FonteODBC : é o nome da fonte ODBC ou uma string de conexão com o Banco

Usuário : é o nome do usuário da Banco de Dados(opcional)

Senha : é a senha desse usuário(opcional)

Conexão por string

Exemplo:

Conexao.Open "CursoASP" // com criação da fonte de dados

Conexão.Open "DRIVER=Microsoft Access Driver (*.mdb);dbq=c:\banco.mdb" // com string de conexão

O objeto Connection possui vários outros métodos, sendo que os principais são:

Close	Fecha a conexão e libera recursos no Servidor de Banco de Dados.
Execute (ComandoSQL)	Executa um comando SQL na Conexão aberta. Quando o comando SQL for de seleção, o método retorna um objeto do tipo RecordSet com o resultado da consulta. Quando o comando SQL for de execução esse método não retorna nada.
BeginTrans	Inicia uma transação no Banco de Dados
CommitTrans	Finaliza com sucesso uma transação iniciada
RollBackTrans	Desfaz todos os comandos de uma transação aberta.

Para instanciarmos um RecordSet procedemos da seguinte forma:

Set Tabela =
Server.CreateObject("ADODB.RecordSet")

Em seguida devemos indicar a Fonte de Dados ODBC para o Recordset:

Tabela.ActiveConnection = "CursoASP"

Por fim, abrimos o recordset utilizando o método Open e informando o comando SQL:

Tabela.Open "Select * From Pessoa"

Mas a forma mais fácil de se instanciar um Recordset é pela utilização do método Execute do Connection:

Set Tabela = Conexao.Execute("Select * from pessoa")

Os principais métodos e propriedades do objeto RecordSet são:

BOF	True indica início do RecordSet
EOF	True indica o final do RecordSet
Close	Fecha o recordset aberto
MoveFirst	Move o cursor para o primeiro registro de um RecordSet
MoveLast	Move o cursor para o último registro de um RecordSet
MoveNext	Move o cursor para o próximo registro de um RecordSet
MovePrevious	Move o cursor para registro anterior de um RecordSet
Update	Salva as alterações feitas no RecordSet no Banco de Dados. Também serve para alterar o registro corrente do RecordSet Exemplo: Tabela.Update "email", "lsantos@ufpi.br "
AddNew	Adiciona um registro ao RecordSet Ex: Tabela.AddNew Tabela("Nome")="Kelly" Tabela("DataN")=#26/05/1980# Tabela.Update
Delete	Elimina um registro de um Recordset consequentemente, de uma tabela do Banco de Dados.

Para exibir os valores dos campos de todos os registros de um Recordset usamos a coleção de objetos Fields:

```

Do While not Tabela.EOF
    Response.Write
    Tabela.Fields(0).Value & "<BR>"
    Response.Write
    Tabela.Fields("Nome") & "<BR>"
    Response.Write
    Tabela.Fields("email") & "<BR>"
    Response.Write
    Tabela.Fields("DataN") & "<HR>"
    Tabela.MoveNext
Loop

```

Ou simplesmente

```

Do While not Tabela.EOF
    Response.Write
    Tabela("codigo") & "<BR>"
    Response.Write
    Tabela("Nome") & "<BR>"

```

```

Response.Write Tabela("email")
& "<BR>"
Response.Write
Tabela("DataN") & "<HR>"
Tabela.MoveNext
Loop

```

Exemplo 24.1 banco1.asp

```

<% @LANGUAGE=VBSCRIPT %>
<% OPTION EXPLICIT %>
<% DIM CONEXAO,TABELA
SET CONEXAO =
SERVER.CREATEOBJECT("ADODB.CONNECTION")
CONEXAO.OPEN "DRIVER=Microsoft
Access Driver (*.mdb);dbq=" &
server.mappath(".") & "\banco.mdb"
SET TABELA = CONEXAO.EXECUTE
("SELECT * FROM PESSOA") %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY><CENTER>
<H1> RELAÇÃO DE PESSOAS </H1>
<%
    Do While not Tabela.EOF
        Response.Write
        Tabela.Fields(0).Value & "<BR>"
        Response.Write
        Tabela.Fields("Nome") & "<BR>"
        Response.Write
        Tabela.Fields("email") & "<BR>"
        Response.Write
        Tabela.Fields("DataN") & "<HR>"
        Tabela.MoveNext
    Loop
%>
</CENTER></BODY></HTML>

```

Para efetuar operações de Inclusão, Deleção ou Modificação num Banco de Dados utilizamos comandos SQL do tipo INSERT, DELETE e UPDATE, respectivamente, no método Execute do Connection:

Exemplo 24.2 banco2.asp?Codigo=1 APAGAR

```

<% @LANGUAGE=VBSCRIPT %>
<% OPTION EXPLICIT %>
<% DIM CONEXAO,SQL
SET CONEXAO =
SERVER.CREATEOBJECT("ADODB.CONNECTION")
CONEXAO.OPEN "DRIVER=Microsoft
Access Driver (*.mdb);dbq=" &
server.mappath(".") & "\banco.mdb"

```

```
SQL = "DELETE FROM PESSOA WHERE
CODIGO=" & REQUEST("CODIGO")
CONEXAO.EXECUTE (SQL) %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY><CENTER>
Pessoa de Código igual a
<%=REQUEST("CODIGO")%> foi excluída do
Banco!
</CENTER></BODY></HTML>
```

Exemplo 24.3 ALTERAR

```
<% @LANGUAGE=VBSCRIPT %>
<% OPTION EXPLICIT %>
<% DIM CONEXAO,SQL
SET CONEXAO =
SERVER.CREATEOBJECT("ADODB.CONNEC
TION")
CONEXAO.OPEN "CursoASP"
SQL = "UPDATE PESSOA SET "
SQL = SQL & "EMAIL = '" &
REQUEST("EMAIL") & "' "
SQL = SQL & "WHERE CODIGO=" &
REQUEST("CODIGO")
CONEXAO.EXECUTE (SQL) %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY><CENTER>
Novo email da pessoa
<%=REQUEST("CODIGO")%> é
<%=REQUEST("email")%>
</CENTER></BODY></HTML>
```

Exemplo 24.4 INSERE

```
<% @LANGUAGE=VBSCRIPT %>
<% OPTION EXPLICIT %>
<% DIM CONEXAO,SQL
SET CONEXAO =
SERVER.CREATEOBJECT("ADODB.CONNEC
TION")
CONEXAO.OPEN "DRIVER=Microsoft
Access Driver (*.mdb);dbq=" &
server.mappath(".") & "\banco.mdb"
SQL = "INSERT INTO
PESSOA(NOME,EMAIL,DATAN)VALUES("
SQL = SQL & "'" & REQUEST("NOME") & "',"
SQL = SQL & "'" & REQUEST("EMAIL") & "',"
SQL = SQL & "#" & REQUEST("DATAN") &
"#)"
CONEXAO.EXECUTE (SQL) %>
<HTML><HEAD><TITLE>Curso
ASP</TITLE></HEAD><BODY><CENTER>
<%=REQUEST("NOME")%> foi inserido(a) no
Banco!
</CENTER></BODY></HTML>
```

A seguir, código da página FORMBANCO.asp que contém formulários para acessar as páginas anteriores.

FORMBANCO.asp

```
<% @ LANGUAGE= VBSCRIPT %>
<HTML><HEAD><TITLE>Curso de
ASP</TITLE></HEAD>
<BODY><CENTER>
<H1>INSERIR USUÁRIO</H1>
<FORM ACTION="Banco4.asp"
METHOD=GET>
Nome do usuário <BR>
<INPUT TYPE=TEXT NAME="Nome"
MAXLENGTH=50><BR>
Email do usuário <BR>
<INPUT TYPE=TEXT NAME="email"
MAXLENGTH=50><BR>
Data de Nascimento <BR>
<INPUT TYPE=TEXT NAME="datan"
MAXLENGTH=10
VALUE="<%=DATE%>"><BR>
<INPUT TYPE=SUBMIT
VALUE="INSERIR"><BR>
</FORM>
<HR>
<H1>APAGAR USUÁRIO</H1>
<FORM ACTION="Banco2.asp"
METHOD=GET>
Código do usuário <BR>
<INPUT TYPE=TEXT NAME="Codigo"
MAXLENGTH=5><BR>
<INPUT TYPE=SUBMIT
VALUE="APAGAR"><BR>
</FORM>
<HR>
<H1>ALTERAR EMAIL</H1>
<FORM ACTION="Banco3.asp"
METHOD=GET>
Código do usuário <BR>
<INPUT TYPE=TEXT NAME="Codigo"
MAXLENGTH=5><BR>
E-mail do usuário <BR>
<INPUT TYPE=TEXT NAME="email"
MAXLENGTH=50><BR>
<INPUT TYPE=SUBMIT
VALUE="MODIFICAR"><BR>
</FORM>
<HR>
</CENTER>
</BODY>
</HTML>
```

Exercício prático

- 1) Crie um banco de dados com os seguintes campos: código, produto, validade e preço. Construa um site para manipular este banco dados, fazendo cadastro, alteração, exclusão, pesquisa exata e pesquisa

aproximada. Fazer controle de login com variável de sessão.