

## Nesta seção você encontra artigos sobre a linguagem PHP e a ferramenta Delphi for PHP



# Gerando miniaturas em PHP

Utilize thumbnails em seus Web Sites PHP

### Nesse artigo veremos

Uso da biblioteca GD;  
Criação de thumbnails, miniaturas;

### Qual a finalidade

A principal finalidade é diminuir o tempo de carga de uma página de fotos.

### Quais situações utilizam esse recurso?

Todo e qualquer web site que se utilize de muitas fotos pode ter esse recurso aplicado e, com isso, ganhar em performance e velocidade.

Mesmo com o crescimento da banda larga em todo o mundo, economizar alguns bytes e largura de banda é a meta de muitos provedores, servidores e desenvolvedores de site. Isso porque quanto mais rápido um site carrega, melhor é a percepção do internauta em relação ao *website* a que se visita. É fato que ninguém gosta de ficar minutos na frente de um computador esperando que todas as imagens da *homepage* sejam carregadas.

Hoje em dia, aliás, já há muito tempo usa-se *Thumbnails* para melhorar a velocidade de carregamento de uma página. Os *Thumbnails* são miniaturas das imagens originais como podemos visualizar na **Figura 1** retirada de um portal de notícias. O próprio *Windows* utiliza esse tipo de recurso quando selecionamos a opção *Exibir Miniaturas* no *Windows XP* ao clicar com o botão direito em uma pasta ou *Ícones Médios, Grandes e Extra Grandes* no *Windows Vista* (**Figura 2**).



### Adriano Santos

falecom@adrianosantos.pro.br

É desenvolvedor Delphi desde 1998, professor e programador PHP, bacharel em Comunicação Social pela Universidade Cruzeiro do Sul, SP. É Editor Chefe das revistas *ClubeDelphi* e *WebMobile*. Gerente de TI na *SoftPark*, parceira Borland, membro fundador do *DUG-SP* ([www.dug-sp.com](http://www.dug-sp.com)) e mantenedor do blog *Delphi to Delphi* ([www.delphitodelphi.blogspot.com](http://www.delphitodelphi.blogspot.com)) com dicas, informações e tudo sobre desenvolvimento Delphi.



## Resumo do DevMan

O esforço em aumentar a banda larga em todo o país tem sido bastante grande por parte das empresas que provêem o serviço. Porém, enquanto não conseguimos velocidades altíssimas, é importante que nos preocupemos com o carregamento das páginas, a velocidade e qualidade de navegação.

Nesse artigo veremos como criar as famosas *Thumbnails*, miniaturas de figuras/fotos em álbuns de fotografias.



Figura 1. Exemplo de Thumbnails em portal de notícias

O *Windows* utiliza esse recurso para aumentar a velocidade de carregamento das imagens de determinada pasta. Imagine na *Web* você abrir um site que possui 10 ou 20 imagens de 1mb cada uma. Certamente o tempo de carga será muito alto e o internauta talvez desista de continuar acessando

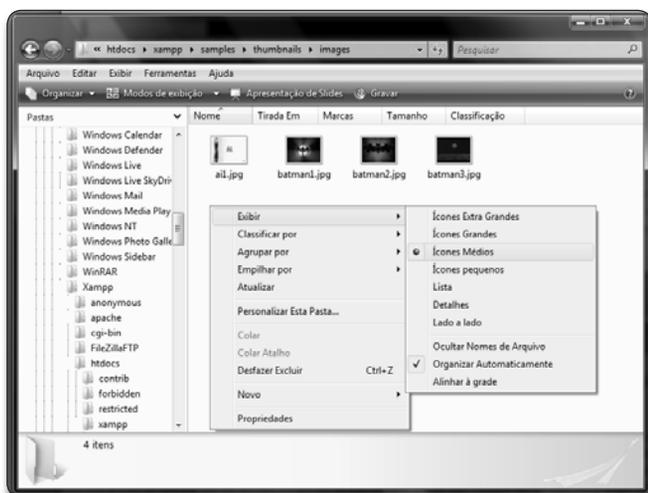


Figura 2. Exibindo miniaturas no Windows Vista

Existem algumas formas de se criar *Thumbnails*. Uma delas é criar cópias das imagens originais e reduzi-las utilizando algum programa de edição de imagens como o *Adobe Photoshop*, por exemplo. Porém, se estamos criando um álbum de fotografias, por exemplo, e queremos ter a possibilidade de enviar fotos a todo instante, alterar o tamanho de cada foto a todo instante torna-se inviável.

Nesse artigo, veremos como criar *Thumbnails* utilizando a biblioteca *GD* do *PHP* que por sinal facilita bastante o nosso trabalho. Mão na massa.

### Considerações iniciais

Como dito anteriormente, utilizaremos a biblioteca *GD* do *PHP*. A biblioteca nada mais é do que uma *DLL* que possui as funções necessárias para a criação da miniatura. Para habilitar a *DLL* em seu *PHP*, é necessário abrir o arquivo *PHP.INI*, normalmente localizado no diretório *System32* do *Windows* e logo após localizar a seguinte linha:

```
#extension=php_gd2.dll
```

Perceba que temos um sinal # na frente da linha, indicando que a mesma encontra-se comentada. Basta então retirar o sinal e salvar o arquivo *.ini*. Feito isso já estamos habilitados para

usar *GD* em nossas aplicações *PHP*.

Na edição 101 de *ClubeDelphi*, publiquei um artigo falando como configurar um bom ambiente de testes para *PHP* utilizando o *Xampp*. Caso esteja usando esse ambiente, não é necessário fazer a configuração, pois por padrão o *GD* já encontra-se ativado, como vemos na **Figura 3**.

gd	
GD Support	enabled
GD Version	bundled (2.0.34 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.1.9
T1Lib Support	enabled
GIF Read Support	enabled
GIF Create Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled
XBM Support	enabled

Figura 3. Relatório phpinfo() mostrando GD ativado

### Criando o arquivo de script

Vamos iniciar criando e entendendo o arquivo de *script* responsável por criar as miniaturas. Para esse exemplo precisaremos apenas de dois arquivos. Um contendo o *script* das miniaturas e o outro para exibição das imagens. Abra então o bloco de notas do *Windows* ou qualquer outro editor de textos para que possamos digitar o arquivo *PHP*.

A primeira coisa que faremos no *script* é definir as constantes de *Endereço das Imagens*, *Largura Máxima* e *Altura Máxima*. Para isso usaremos o método *define()*. Comece o *script* digitando as linhas de programação abaixo:

```
<?PHP
define('PATH_IMG', '<caminho>');
define('MAX_WIDTH', 160);
define('MAX_HEIGHT', 120);
...
```

Veja que na definição da constante *PATH\_IMG* coloquei propositalmente o texto *<caminho>*. Pois nesse momento podemos atribuir o local correto das imagens de nosso *WebSite*. Precisamos apenas trocar *<caminho>*, pelo caminho certo das fotos. Veja um exemplo:

```
define('PATH_IMG', 'C:\xampp\exemplos\
thumbnails\images');
```

Aqui estamos apontando para a pasta *images* presente no diretório da aplicação. É importante colocar as fotos em

Figura 4. Topo do arquivo index.php

locais diferentes, pois faremos uma varredura do diretório transformaremos em *thumbnails* todos os arquivos *jpg* e *jpeg* encontrados. Isso significa que se tivermos outras imagens no mesmo diretório, como os ícones que compõem a aplicação em si, elas também serão convertidas.

A próxima etapa do *script* é responsável por receber o nome do arquivo de imagem, retirar o símbolo "...", caso haja, e concatenar com a constante *PATH\_IMG*. Por fim atribuímos esse endereço de imagem a variável *\$image\_path*. A última linha desse trecho atribui o valor *null* a variável *\$img*. Não chega a ser a criação e/ou declaração da variável, assim como fazemos em Delphi, mas é uma forma de iniciá-la. Digite o código a seguir no mesmo arquivo que estamos criando.

```
# Pega onde está a imagem
$image_file = str_replace('.', '', $_SERVER['QUERY_STRING']);
$image_path = PATH_IMG . '/' . $image_file;
# Carrega a imagem
$img = null;
```

Perceba o uso da função *str\_replace* no início. Essa função faz a substituição de um dado caractere por outro. Nesse caso estamos informando à função que o "." encontrado na *QUERY\_STRING*, deve ser substituído por "", ou seja, nenhum caractere. Em Delphi temos essa função também sob o nome de *ReplaceString*. Em outras palavras, nosso *script* receberá algo como a linha a seguir:

```

```

Nossa constante receberá o caminho físico das fotos e o concatenará com o nome da foto recebido na *QueryString*. Dessa forma teremos algo como vemos a seguir. Com isso em "mãos" o *script* é capaz de transformar em *thumbnails*.

```
$image_path = 'C:\xampp\exemplos\thumbnails\images\image.jpg';
```

Agora precisamos tomar outra providência. Nem todos os tipos de imagem podem ser convertidos em *miniaturas* e além do mais, cada tipo de extensão precisa de um tratamento especial. Por isso, extraímos a extensão do arquivo e o tratamos chamando a função adequada. Veja na **Listagem 1**.

**Listagem 1.** Analisando a extensão e tipo de arquivo

```
$extensao = strtolower(end(explode('.', $image_path)));

if ($extensao == 'jpg' || $extensao == 'jpeg') {
    $img = @imagecreatefromjpeg($image_path);
} elseif ($extensao == 'png') {
    $img = @imagecreatefrompng($image_path);
} elseif ($extensao == 'gif') {
    $img = @imagecreatefromgif($image_path);
}
```

A variável *\$extensao* está recebendo o resultado de várias funções, uma validando a outra. Primeiro separamos literalmente o nome do arquivo e sua extensão usando o método *explode*, que na verdade o transforma em um *array*. Enviamos como parâmetro para ele um ponto, que é o separador entre nome e extensão como já estamos acostumados. Depois pegamos o último elemento do *array*, nesse caso a extensão, usando o método *end*. Por fim, transformamos tudo em minúsculo com a função *strtolower*.

Na seqüência testamos cada valor e chamamos as respectivas funções para criação das miniaturas. Veja, se a imagem for um *jpg* ou *jpeg*, a função *@imagecreatefromjpeg* é chamada. Case seja *png*, então chamamos *@imagecreatefrompng* e assim sucessivamente. Evidentemente que não há todas aqui, mas basta uma boa lida na documentação do PHP para encontrar as demais funções. Repare também que o resultado da função é passado diretamente para variável *\$img*.

Vamos continuar a codificação do *script*. Após criar a imagem, precisamos testar se ela foi criada com sucesso e então começar a manipular suas propriedades. Veja o código da **Listagem 2**. Logo de início pegamos o valor de largura (*\$width*), altura (*\$height*) e sua escala (*\$scale*). Com isso analisamos a imagem. Se a escala for maior que o permitido, então encolhemos a imagem (*Linha 5 a 7*). Criamos então uma imagem temporária na *linha 8* e copiamos para ela (*Linha 9*) o mesmo tamanho da imagem original. Destruímos a imagem que estava gravada em *\$img* e atribuímos a ela a imagem temporária.

**Listagem 2.** Tratamento da imagem

```
...
01 if ($img){
02     $width = imagesx($img);
03     $height = imagesy($img);
04     $scale = min(MAX_WIDTH / $width, MAX_HEIGHT / $height);
05     if ($scale < 1) {
06         $new_width = floor($scale * $width);
07         $new_height = floor($scale * $height);
08         $tmp_img = imagecreatetruecolor($new_width, $new_height);
09         imagecopyresized($tmp_img, $img, 0, 0, 0, 0, $new_width,
10             $new_height, $width, $height);
11         imagedestroy($img);
12         $img = $tmp_img;
13     }
14 }
```



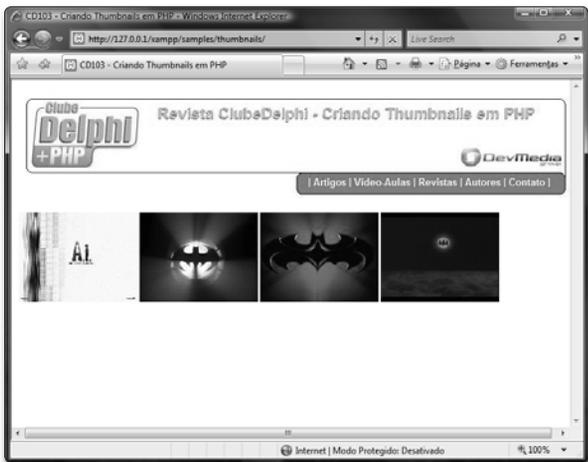


Figura 5. Exemplo funcionando com os thumbnails

Como podemos ver, não há grandes segredos no desenvolvimento de aplicações que se utilizem de *Thumbnails*, miniaturas. O mais importante é que agora temos uma página carregando em grande velocidade.

## Conclusão

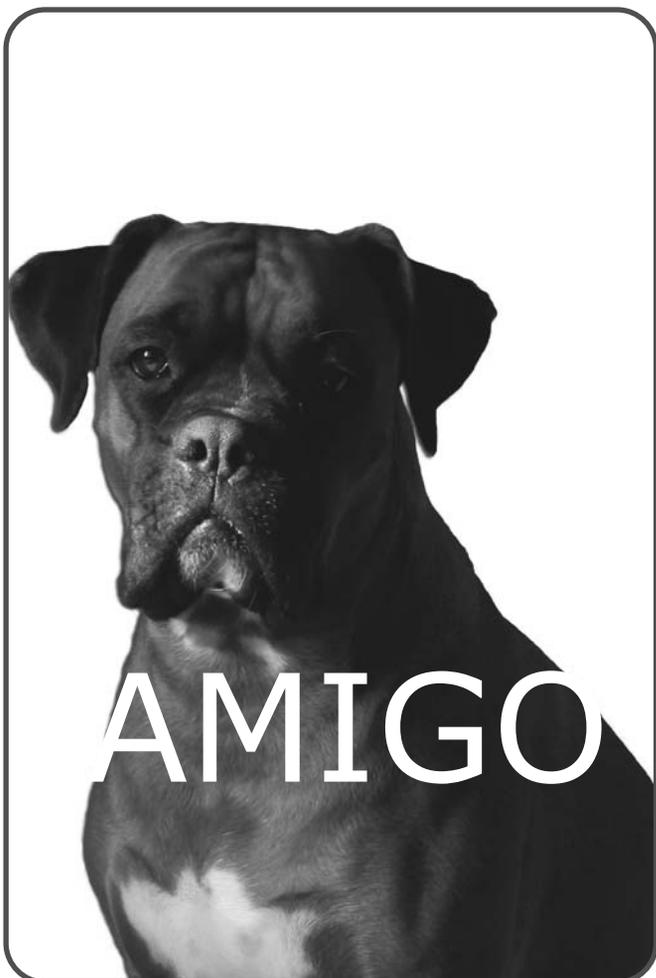
O desenvolvimento de web sites no geral, tem evoluído significativamente. As tecnologias e linguagens de programação têm melhorado cada vez mais, e certamente com isso todos nós ganhamos. Com apenas o uso da biblioteca GD criamos um sistema capaz de exibir *thumbnails*, pequenas imagens que representam as fotos originais de um álbum de fotografias. Há outras bibliotecas capazes de realizar as mais diversas tarefas do dia-a-dia. Confira a documentação do PHP. Até a próxima e boa sorte. ●

### Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

[www.devmedia.com.br/clubedelphi/feedback](http://www.devmedia.com.br/clubedelphi/feedback)



Existem coisas  
que não conseguimos  
ficar sem!

...só pra lembrar,  
sua assinatura pode  
estar acabando!

**Renove Já!**

[www.devmedia.com.br/renovacao](http://www.devmedia.com.br/renovacao)

Para mais informações:  
[www.devmedia.com.br/central](http://www.devmedia.com.br/central)

