Classe para upload de imagens em PHP com redimensionamento

Veja neste artigo como criar uma classe genérica para upload de arquivos em PHP, que no no caso do envio de imagens, já faz o redimensionamento adequado destas antes de enviá-las para o servidor.

Quem é que nunca precisou fazer algum site ou sistema que faça upload de arquivos? Porém, há muitos casos em que apenas o upload não resolve o problema, como no caso de algumas imagens, já que elas atualmente são grandes devido a alta qualidade dos dispositivos de captura, ficando inviável usar essas imagens em uma aplicação web. Também não é nada prático o usuário do sistema ter que usar uma ferramenta externa para redimensionar a imagem para poder fazer o upload. Então a ideia desse artigo é desenvolvermos uma classe genérica usada para o upload de qualquer arquivo e quando for uma imagem, que faça o redimensionamento automático sempre que necessário. Para fazer esse redimensionamento usaremos a biblioteca GD que é responsável pelo tratamento das imagens.

O envio do arquivo ocorrerá quando acontecer o clique no botão de submissão do formulário, neste momento será criada uma instância da classe e serão passados os seguintes parâmetros: arquivo a ser enviado, altura e largura máxima e pasta de destino respectivamente.

Então, mãos na massa para ver se funciona.

Vamos começar criando a estrutura de diretórios e arquivos. Será criada uma pasta chamada “fotos” que é onde os arquivos ficarão salvos, o arquivo index.php contém o formulário e por fim o Upload.class.php contém toda a lógica de funcionamento.

Observação: se você estiver usando Linux será preciso alterar a permissão da pasta de uploads para 777, permitindo a leitura, gravação e execução.



**Figura 1:** Estrutura de diretórios

**Listagem 1:** Arquivo Upload.class.php

<?php

/\*

Desenvolvido por Marco Antoni <marquinho9.10@gmail.com>

\*/

class Upload{

private $arquivo;

private $altura;

private $largura;

private $pasta;

function \_\_construct($arquivo, $altura, $largura, $pasta){

$this->arquivo = $arquivo;

$this->altura = $altura;

$this->largura = $largura;

$this->pasta = $pasta;

}

private function getExtensao(){

//retorna a extensao da imagem

return $extensao = strtolower(end(explode('.', $this->arquivo['name'])));

}

private function ehImagem($extensao){

$extensoes = array('gif', 'jpeg', 'jpg', 'png'); // extensoes permitidas

if (in\_array($extensao, $extensoes))

return true;

}

//largura, altura, tipo, localizacao da imagem original

private function redimensionar($imgLarg, $imgAlt, $tipo, $img\_localizacao){

//descobrir novo tamanho sem perder a proporcao

if ( $imgLarg > $imgAlt ){

$novaLarg = $this->largura;

$novaAlt = round( ($novaLarg / $imgLarg) \* $imgAlt );

}

elseif ( $imgAlt > $imgLarg ){

$novaAlt = $this->altura;

$novaLarg = round( ($novaAlt / $imgAlt) \* $imgLarg );

}

else // altura == largura

$novaAltura = $novaLargura = max($this->largura, $this->altura);

//redimencionar a imagem

//cria uma nova imagem com o novo tamanho

$novaimagem = imagecreatetruecolor($novaLarg, $novaAlt);

switch ($tipo){

case 1: // gif

$origem = imagecreatefromgif($img\_localizacao);

imagecopyresampled($novaimagem, $origem, 0, 0, 0, 0,

$novaLarg, $novaAlt, $imgLarg, $imgAlt);

imagegif($novaimagem, $img\_localizacao);

break;

case 2: // jpg

$origem = imagecreatefromjpeg($img\_localizacao);

imagecopyresampled($novaimagem, $origem, 0, 0, 0, 0,

$novaLarg, $novaAlt, $imgLarg, $imgAlt);

imagejpeg($novaimagem, $img\_localizacao);

break;

case 3: // png

$origem = imagecreatefrompng($img\_localizacao);

imagecopyresampled($novaimagem, $origem, 0, 0, 0, 0,

$novaLarg, $novaAlt, $imgLarg, $imgAlt);

imagepng($novaimagem, $img\_localizacao);

break;

}

//destroi as imagens criadas

imagedestroy($novaimagem);

imagedestroy($origem);

}

public function salvar(){

$extensao = $this->getExtensao();

//gera um nome unico para a imagem em funcao do tempo

$novo\_nome = time() . '.' . $extensao;

//localizacao do arquivo

$destino = $this->pasta . $novo\_nome;

//move o arquivo

if (! move\_uploaded\_file($this->arquivo['tmp\_name'], $destino)){

if ($this->arquivo['error'] == 1)

return "Tamanho excede o permitido";

else

return "Erro " . $this->arquivo['error'];

}

if ($this->ehImagem($extensao)){

//pega a largura, altura, tipo e atributo da imagem

list($largura, $altura, $tipo, $atributo) = getimagesize($destino);

// testa se é preciso redimensionar a imagem

if(($largura > $this->largura) || ($altura > $this->altura))

$this->redimensionar($largura, $altura, $tipo, $destino);

}

return "Sucesso";

}

}

?>

Entendendo o código

Repare que foram criados quatro atributos privados que são:

* $arquivo: armazena a estrutura de uma variável do tipo $\_FILE do PHP;
* $altura e $largura: armazenam um número inteiro indicando a altura e largura máxima que a imagem terá após o upload.
* $pasta: armazena uma string contendo a pasta de destino dos uploads.

Toda a lógica de funcionamento da classe está no método salvar. Ele funciona da seguinte maneira: obtém a extensão do arquivo, gera um novo nome baseado no tempo, se o arquivo é uma imagem do tipo permitidos é efeito um teste para saber se é preciso redimensionar e então o arquivo estará salvo.

Agora que temos o caminho absoluto do arquivo ($destino), será feito o envio para o servidor. Se houver algum erro durante esse processo, a execução do script será interrompida devido ao return. Neste ponto o arquivo já está salvo no servidor, se tudo acorrer certo o próximo passo é descobrir se o arquivo é uma imagem, para isso foi implementada a função ehImagem, que recebe como parâmetro uma string contendo a extensão do arquivo. Nela existe um vetor contendo as extensões que são “permitidas”, ou seja, as imagens que poderão serem redimensionadas (através do método redimensionar) posteriormente caso excedam os limites permitidos, esse método retorna verdadeiro.

Então, se o arquivo é uma imagem, iremos pegar a largura, altura, tipo e atributo da imagem usando a funcão getimagesize. Depois testamos se a largura ou altura da imagem excedem os limites (atributos da classe), caso um deles exceda, a imagem tem que ser redimensionada e esse procedimento é feito através do método redimensionar, que recebe como parâmetros a largura, altura, tipo e o endereço absoluto da imagem.

Finalmente o método redimensionar testa as dimensões da foto gerando o novo tamanho sem perder a proporção da imagem. A variável $novaimagem recebe uma imagem vazia recém criada com as novas dimensões. O processamento da imagem agora é feito com base no tipo da imagem (Ex: 1 é um GIF) .

* $origem: recebe uma cópia da imagem que está sendo redimensionada (imagefromgif, imagemfromjpg, etc..).

A função imagecopyresampled copia uma imagem para dentro de outra com os respectivos tamanhos, após esse procedimento a variável $novaimagem contém a imagem redimensionada. Agora usamos uma das funcões (imagegif, imagejpg, etc) para substituir a imagem antiga pela nova.

Nesse ponto o método vai encerrar, retornando ao método salvar que retornará “Sucesso”.

O próximo passo é criar o nosso formulário dentro do arquivo index.php, vamos ao código.

**Listagem 2:** Arquivo index.php

<html>

<form method='post' enctype='multipart/form-data'><br>

<input type='file' name='foto' value='Cadastrar foto'>

<input type='submit' name='submit'>

</form>

<?php

include "Upload.class.php";

if ((isset($\_POST["submit"])) && (! empty($\_FILES['foto']))){

$upload = new Upload($\_FILES['foto'], 1000, 800, "fotos/");

echo $upload->salvar();

}

?>

</body>

</html>

O formulário com envio de arquivos deve obrigatoriamente conter o atributo **enctype='multipart/form-data'**. No input temos que definir o tipo **file**. No trecho de código do PHP testamos se foi clicado no botão submit e se foi preenchido o campo file, caso as duas condições sejam verdadeiras, será criada uma instância da classe upload e então será chamado o método para enviar o arquivo.

Agora vamos ver se funciona.



**Figura 2:** Formulário pronto



**Figura 3:** Arquivo enviado com sucesso