eventviva / **php-image-resize**

PHP class to re-size and scale images

README.md

php-image-resize

PHP class to resize and scale images.

Setup

This package is available through Packagist with the vendor and package identifier the same as this repo.

If using Composer, in your `composer.json` file add:

```
{
    "require": {
        "eventviva/php-image-resize": "1.5.*"
    }
}
```

Otherwise:

```
include '/path/to/ImageResize.php';
```

Because this class uses namespacing, when instantiating the object, you need to either use the fully qualified namespace:

```
$image = new \Eventviva\ImageResize();
```

Or alias it:

```
use \Eventviva\ImageResize;
```

```
$image = new ImageResize();
```

Note: This library uses GD class which do not support resizing animated gif files

Usage

To scale an image, in this case to half it's size (scaling is percentage based):

```
$image = new ImageResize('image.jpg');
$image->scale(50);
$image->save('image2.jpg')
```

To resize an image according to one dimension (keeping aspect ratio):

```
$image = new ImageResize('image.jpg');
$image->resizeToHeight(500);
$image->save('image2.jpg');

$image = new ImageResize('image.jpg');
$image->resizeToWidth(300);
$image->save('image2.jpg');
```

To resize an image to best fit a given set of dimensions (keeping aspet ratio):

```
$image = new ImageResize('image.jpg');
$image->resizeToBestFit(500, 300);
$image->save('image2.jpg');
```

To to crop an image:

```
$image = new ImageResize('image.jpg');
$image->crop(200, 200);
$image->save('image2.jpg');
```

This will scale the image to as close as it can to the passed dimensions, and then crop and center the rest.

In the case of the example above, an image of 400px × 600px will be resized down to 200px × 300px, and then 50px will be taken off the top and bottom, leaving you with 200px × 200px.

If you are happy to handle aspect ratios yourself, you can resize directly:

```
$image = new ImageResize('image.jpg');
$image->resize(800, 600);
$image->save('image2.jpg');
```

This will cause your image to skew if you do not use the same width/height ratio as the source image.

Loading and saving images from string

To load an image from a string:

```
$image = ImageResize::createFromString(base64_decode('R0lGODlhAQABAIAAAAQCBP///yH5BAEAAAEALAAAAAABAAEAAAICRAEAOw=='))
$image->scale(50);
$image->save('image.jpg');
```

You can also return the result as a string:

```
$image = ImageResize::createFromString(base64_decode('R0lGODlhAQABAIAAAAQCBP///yH5BAEAAAEALAAAAAABAAEAAAICRAEAOw=='))
$image->scale(50);
echo $image->getImageAsString();
```

Magic `__toString()` is also supported:

```
$image = ImageResize::createFromString(base64_decode('R0lGODlhAQABAIAAAAQCBP///yH5BAEAAAEALAAAAAABAAEAAAICRAEAOw=='))
$image->resize(10, 10);
echo (string)$image;
```

Displaying

As seen above, you can call `$image->save('image.jpg');`

To render the image directly into the browser, you can call `$image->output()`;

Image Types

When saving to disk or outputting into the browser, the script assumes the same output type as input.

If you would like to save/output in a different image type, you need to pass a (supported) PHP `IMAGETYPE_*` constant:

- `IMAGETYPE_GIF`
- `IMAGETYPE_JPEG`
- `IMAGETYPE_PNG`

This allows you to save in a different type to the source:

```
$image = new ImageResize('image.jpg');
$image->resize(800, 600);
$image->save('image.png', IMAGETYPE_PNG);
```

Quality

The properties `$quality_jpg` and `$quality_png` are available for you to configure:

```
$image = new ImageResize('image.jpg');
$image->quality_jpg = 100;
$image->resize(800, 600);
$image->save('image2.jpg');
```

By default they are set to 75 and 0 respectively. See the manual entries for `imagejpeg()` and `imagepng()` for more info.

You can also pass the quality directly to the `save()`, `output()` and `getImageAsString()` methods:

```
$image = new ImageResize('image.jpg');
$image->crop(200, 200);
$image->save('image2.jpg', null, 100);

$image = new ImageResize('image.jpg');
$image->resizeToWidth(300);
$image->output(IMAGETYPE_PNG, 4);

$image = new ImageResize('image.jpg');
$image->scale(50);
$result = $image->getImageAsString(IMAGETYPE_PNG, 4);
```

We're passing `null` for the image type in the example above to skip over it and provide the quality. In this case, the image type is assumed to be the same as the input.

Interlacing

By default, image interlacing is turned off. It can be enabled by setting `$interlace` to `1`:

```
$image = new ImageResize('image.jpg');
$image->scale(50);
```

```
$image->interlace = 1;
$image->save('image2.jpg')
```

Chaining

When performing operations, the original image is retained, so that you can chain operations without excessive destruction.

This is useful for creating multiple sizes:

```
$image = new ImageResize('image.jpg');
$image
    ->scale(50)
    ->save('image2.jpg')

    ->resizeToWidth(300)
    ->save('image3.jpg')

    ->crop(100, 100)
    ->save('image4.jpg')
;
```