Dynamic Resizer (Standalone)

Last updated by | Philipp Losbichler | 31. Jan. 2024 at 08:54 MEZ

Agravity ImageEdit

Description

This is the standalone version of the image edit endpoint leveraging the Imagemagick API. Due to the use of reflection, custom image edit requests can be crafted resulting in various different images where pretty much everything can be controlled. In addition, basic operations can be performed using a simplified API.

Two methods of editing images:

- **GET Request:** Using the GET- Endpoint a fixed subset of Imagemagick functions can be used. Most suitable for basic operations like resizing, changing format, etc. Instructions on how to use this Endpoint can be found in OpenAPI Docs.
- **POST Request:** Using the POST- Endpoint sophisticated image manipulation requests can be crafted. Everything that's possible in the Imagemagick C# SDK is also possible via this Endpoint. Usage instructions can be found in the next chapter.

Usage of dynamic imageedit

Sample request body

```
[
    {
        "operation": "resize",
        "params": [200, 400]    //also posible to provide percentage values like "30%"
    },
    {
        "operation": "format",
        "params": ["svg"]
    }
]
```

> NOTE: Operations are executed top to bottom, meaning the order of the operations can drastically impact the resulting image.

> NOTE: Operations are not case sensitive

Supported operations

Every method or property(only set of course) including all overloads that is available in the C# Magick.NET SDK on the MagickImage Type can be used as operation.

Link to IMagickImage interface on GitHub containing every possible operation

> NOTE: Only methods with void return type should be used. Or in other words: Only methods/properties that modify the image instance, have effect on the output. Therefore every other operation is omitted.

Usage of params and supported types

Params have to be provided in the order that they can be found in the C# SDK. When dealing with a property operation only a single param can be provided (obviously).

Generally, every primitive type + enums are supported. Every other type has to be implemented if needed.

Currently supported custom types:

Type	Required string format in param
ImageMagick.Percentage	"90%"
ImageMagick.MagickColor	"#FFF0" //RGBA
ImageMagick.IColorProfile	"AppleRGB" //also custom ones through uploaded icc profiles

> HINT: Look for overloads that only need primitive types to achieve the same result without having to implement a custom type resolving.