Flask-Resize Documentation

Release 2.0.4

Jacob Magnusson

Dec 19, 2017

Contents

Flask extension for resizing images in code and templates. Can convert from JPEG|PNG|SVG to JPEG|PNG, resize to fit and crop. File-based and S3-based storage options are available.

CHAPTER 1

Installation

1.1 Production version

Recommended install with all features (Redis cache + S3 storage + SVG if py3.4+):

```
pip install flask-resize[full]
```

Other alternatives:

```
# File-based storage only
pip install flask-resize

# With S3 storage support
pip install flask-resize[s3]

# With Redis caching
pip install flask-resize[redis]

# With SVG source file support (only available with py3.4+)
pip install flask-resize[svg]

# Or any combination of above. E.g:
pip install flask-resize[s3,redis]
```

1.2 Development version

```
pip install -e git+https://github.com/jmagnusson/flask-resize@master#egg=Flask-Resize
```

1.3 Compatibility

Tested with Python 2.7/3.4/3.5/3.6/pypy and latest version of Flask.

Should work well on most Linux and MacOS versions.

Windows is supported on paper with the full test suite running smoothly, but the package needs some real world usage on Windows though. Please report any issues you might find to the Github issues page. Any feedback is welcome!

1.4 Running the tests

```
git clone https://github.com/jmagnusson/Flask-Resize.git
cd Flask-Resize
pip install tox
tox
```

You can also run the tests inside docker:

```
docker-compose --build up
```

Or to run a single python version's tests:

```
docker-compose run py36
```

Note that the tests in docker are run with all dependencies installed. See docker-compose.yml for more info.

1.5 Generating the docs

```
git clone https://github.com/jmagnusson/Flask-Resize.git
cd Flask-Resize
pip install -r requirements_docs.txt
python manage.py docs clean build serve
```

Now you should be able to view the docs @ localhost:8000.

1.6 Contributing

Fork the code on the Github project page then:

```
git clone git@github.com:YOUR_USERNAME/Flask-Resize.git
cd Flask-Resize
pip install '.[full,test,test_s3]' -r requirements_docs.txt
git checkout -b my-fix
# Create your fix and add any tests if deemed necessary.
# Run the test suite to make sure everything works smooth.
py.test
git commit -am 'My fix!'
git push
```

Now you should see a box on the project page with which you can create a pull request.

CHAPTER 2

Configuration

2.1 Setting up your flask app to use Flask-Resize

Using direct initialization:

```
import flask
import flask_resize

app = flask.Flask(__name__)
app.config['RESIZE_URL'] = 'https://mysite.com/'
app.config['RESIZE_ROOT'] = '/home/user/myapp/images'

resize = flask_resize.Resize(app)
```

Using the app factory pattern:

```
import flask
import flask_resize

resize = flask_resize.Resize()

def create_app(**config_values):
    app = flask.Flask()
    app.config.update(**config_values)
    resize.init_app(app)
    return app

# And later on...
app = create_app(
    RESIZE_URL='https://mysite.com/',
    RESIZE_ROOT='/home/user/myapp/images'
)
```

2.2 Setting up for standalone usage

One doesn't actually need to involve Flask at all to utilize the resizing (perhaps one day we'll break out the actual resizing into its own package). E.g:

```
import flask_resize

config = flask_resize.configuration.Config(
    url='https://mysite.com/',
    root='/home/user/myapp/images',
)

resize = flask_resize.make_resizer(config)
```

New in version 2.0.0: Standalone mode was introduced

2.3 Available settings

2.3.1 Required for `file` storage

You need to set at least two configuration options when using the default `file` storage:

```
# Where your media resides
RESIZE_ROOT = '/path/to/your/media/root/'

# The URL where your media is served at. For the best performance you
# should serve your media with a proper web server, under a subdomain
# and with cookies turned off.
RESIZE_URL = 'http://media.yoursite.com/'
```

2.3.2 Required for `s3` storage

For Amazon S3 storage you only need to do the following if you've already configured Amazon Web Services with `aws configure` (or similar). Default section configuration can then be extracted using the included `botocore` package.

```
RESIZE_STORAGE_BACKEND = 's3'
RESIZE_S3_BUCKET = 'mybucket'
```

If you haven't done so then you need to manually specify the following options in addition to above:

```
RESIZE_S3_ACCESS_KEY = 'dq8rJLaMtkHEze4example3C1V'
RESIZE_S3_SECRET_KEY = 'MC9T4tRqXQexample3d1l7C9sG3M9qes0VEHiNJTG24q4a5'
RESIZE_S3_REGION = 'eu-central-1'
```

New in version 1.0.0: `RESIZE_S3_ACCESS_KEY`, `RESIZE_S3_SECRET_KEY` and `RESIZE_S3_BUCKET` were added.

New in version 1.0.1: `RESIZE_S3_REGION` was added.

2.3.3 Optional

There are also some optional settings. They are listed below, with their default values.

```
# Where the resized images should be saved. Relative to `RESIZE_ROOT`
# if using the file-based storage option.
RESIZE_TARGET_DIRECTORY = 'resized-images'

# Set to False if you want Flask-Resize to create sub-directories for
# each resize setting instead of using a hash.
RESIZE_HASH_FILENAME = True

# Change if you want to use something other than sha1 for your hashes.
# Supports all methods that hashlib supports.
RESIZE_HASH_METHOD = 'sha1'

# Useful when testing. Makes Flask-Resize skip all processing and just
# return the original image URL.
RESIZE_NOOP = False

# Which backend to store files in. Defaults to the `file` backend.
# Can be either `file` or `s3`.
RESIZE_STORAGE_BACKEND = 'file'

# Which cache store to use. Currently only redis is supported (`pip install
# flask-resize[redis]`), and will be configured automatically if the
# package is installed and `RESIZE_CACHE_STORE` hasn't been set
# explicitly. Otherwise a no-op cache is used.
RESIZE_CACHE_STORE = 'noop' if redis is None else 'redis'

# Which host to use for redis if it is enabled with `RESIZE_CACHE_STORE`
# This can also be a pre-configured `redis.StrictRedis` instance, in which
# case the redis options below are ignored by Flask-Resize.
RESIZE_REDIS_HOST = 'localhost'

# Which port to use for redis if it is enabled with `RESIZE_CACHE_STORE`
RESIZE_REDIS_PORT = 6379

# Which db to use for redis if it is enabled with `RESIZE_CACHE_STORE`
RESIZE_REDIS_DB = 0

# Which password to use for redis if it is enabled with `RESIZE_CACHE_STORE`.␣
↪Defaults to not using a password.
RESIZE_REDIS_PASSWORD = None

# Which key to use for redis if it is enabled with `RESIZE_CACHE_STORE`
RESIZE_REDIS_KEY = 0

# If True then GenerateInProgress exceptions aren't swallowed. Default is
# to only raise these exceptions when Flask is configured in debug mode.
RESIZE_RAISE_ON_GENERATE_IN_PROGRESS = app.debug
```

New in version 0.4.0: `RESIZE_NOOP` was added.

New in version 1.0.0: `RESIZE_CACHE_STORE`, `RESIZE_REDIS_HOST`, `RESIZE_REDIS_PORT`, `RESIZE_REDIS_DB` and `RESIZE_REDIS_KEY` were added.

New in version 1.0.2: `RESIZE_STORAGE_BACKEND` was added.

New in version 1.0.4: `RESIZE_RAISE_ON_GENERATE_IN_PROGRESS` was added.

New in version 2.0.3: `RESIZE_REDIS_PASSWORD` was added.

CHAPTER 3

Usage

3.1 Usage in Jinja templates

After having *installed* and *configured* Flask-Resize in your app the `resize` filter should be available in your code and jinja templates.

To generate an image from the supplied image URL that will fit within an area of 600px width and 400px height:

```
resized_url = resize(original_image_url, '600x400')
```

Resize and crop so that the image will fill the entire area:

```
resized_url = resize(original_image_url, '300x300', fill=1)
```

Convert to JPG:

```
resized_url = resize(original_image_url, '300x300', format='jpg')
```

The function will also be available in your jinja templates as a filter, where you'll call the resize filter like this:

```
<img src="{{ original_image_url|resize('300x300', format='jpg') }}" alt="My kittens">
```

3.2 List of arguments

3.2.1 dimensions

Default: Keep original dimensions

Can be either a string or a two item list. The format when using a string is any of `<width>x<height>`, `<width>` or `x<height>`. If width or height is left out they will be determined from the ratio of the original image. If both width and height are supplied then the output image will be within those boundries.

3.2.2 placeholder

Default: False

A placeholder image will be returned if the image couldn't be generated. The placeholder will contain text specifying dimensions, and reason for image not being generated (either `empty image path` or `<filepath> does not exist`)

3.2.3 format

Default: Keep original format

If you want to change the format. A white background color is applied when a transparent image is converted to JPEG, or the color specified with `bgcolor`. Available formats are PNG and JPEG at the moment. Defaults to using the same format as the original.

3.2.4 bgcolor

Default: Don't add a background color

Adds a background color to the image. Can be in any of the following formats:

- `#333`
- `#333333`
- `333`
- `333333`
- `(51, 51, 51)`

3.2.5 quality

Default: `80`

Only matters if output format is jpeg. Quality of the output image. 0-100.

3.2.6 upscale

Default: `True`

Disable if you don't want the original image to be upscaled if the dimensions are smaller than those of `dimensions`.

3.2.7 fill

Default: `False`

The default is to keep the ratio of the original image. With `fill` it will crop the image after resizing so that it will have exactly width and height as specified.

3.2.8 progressive

Default: True

Whether to use progressive or not. Only matters if the output format is jpeg. Article about progressive JPEGs.

CHAPTER 4

Command line usage

4.1 General

When installing flask-resize with pip you'll get a command installed along it called `flask-resize`. See below for the sub commands and what they're used for.

4.1.1 Usage

```
usage: flask-resize [-h] {generate,list,sync,clear} ...
```

Sub-commands:

generate

Generate images passed in through stdin. Return URL for resulting image

Useful to generate images outside of the regular request/response cycle of a web app = happier visitors who don't have to wait until image processing by Flask-Resize completes. Care has to be taken so that the exact same arguments are passed in as what is specified in code/templates - the smallest difference in passed in options will cause flask resize to generate a new image.

Use GNU Parallel or similar tool to parallelize the generation

```
flask-resize generate [-h] [-d DIMENSIONS] [-f FORMAT] [-q QUALITY] [-F]
                      [-b BGCOLOR] [-u] [--progressive] [--placeholder]
                      filename
```

Positional Arguments

filename None

Named Arguments

-d, --dimensions	None
-f, --format	None
-q, --quality	80
	Default: 80
-F, --fill	False
	Default: False
-b, --bgcolor	None
-u, --upscale	True
	Default: True
--progressive	True
	Default: True
--placeholder	False
	Default: False

list

Undocumented

```
flask-resize list [-h] {cache,images} ...
```

Sub-commands:

cache

List all items found in cache

```
flask-resize list cache [-h]
```

images

List all generated images found in storage backend

```
flask-resize list images [-h]
```

sync

Undocumented

```
flask-resize sync [-h] {cache} ...
```

Sub-commands:

cache

Syncs paths stored in the cache backend with what's in the storage backend

Useful when the storage backend destination is shared between multiple environments. One use case is when one has synced generated imagery from production into one's development environment (for example with *aws s3 sync –delete s3://prod-bucket s3://my-dev-bucket*). The cache can then be synced with what's been added/removed from the bucket *my-dev-bucket*.

```
flask-resize sync cache [-h]
```

clear

Undocumented

```
flask-resize clear [-h] {cache,images,all} ...
```

Sub-commands:

cache

Clear the cache backend from generated images' paths

```
flask-resize clear cache [-h]
```

images

Delete all generated images from the storage backend

```
flask-resize clear images [-h]
```

all

Clear both the cache and all generated images

```
flask-resize clear all [-h]
```

CHAPTER 5

API Documentation

5.1 Resizing

class flask_resize.resizing.**Resize**(*app=None*)

Used for initializing the configuration needed for the `Resizer` instance, and for the jinja filter to work in the flask app.

Parameters **app** (*Any[flask.Flask, None]*) – A Flask app can be passed in immediately if not using the app factory pattern.

init_app(*app*)

Initialize Flask-Resize

Parameters **app** (`flask.Flask`) – The Flask app to configure.

Raises `RuntimeError` – A setting wasn't specified, or was invalid.

class flask_resize.resizing.**Resizer**(*storage_backend*, *cache_store*, *base_url*, *name_hashing_method='sha1'*, *target_directory='resized-images'*, *raise_on_generate_in_progress=False*, *noop=False*)

Factory for creating the resize function

__call__(*image_url*, *dimensions=None*, *format=None*, *quality=80*, *fill=False*, *bgcolor=None*, *upscale=True*, *progressive=True*, *placeholder=False*)

Method for resizing, converting and caching images

Parameters

- **image_url** (*str*) – URL for the image to resize. A URL relative to *base_url*
- **dimensions** (str, Sequence[`int`, `int`]) – Width and height to use when generating the new image. Uses the format of `parse_dimensions()`. No resizing is done if None is passed in.
- **format** (Optional[`str`]) – Format to convert into. Defaults to using the same format as the original image. An exception to this default is when the source image is of type SVG/SVGZ, then PNG is used as default.

- **quality** (*int*) – Quality of the output image, if the format is JPEG. Defaults to 80.
- **fill** (*bool*) – Fill the entire width and height that was specified if True, otherwise keep the original image dimensions. Defaults to False.
- **bgcolor** (Optional[str]) – If specified this color will be used as background.
- **upscale** (*bool*) – Whether or not to allow the image to become bigger than the original if the request width and/or height is bigger than its dimensions. Defaults to True.
- **progressive** (*bool*) – Whether to use progressive encoding or not when JPEG is the output format. Defaults to True.
- **placeholder** (*bool*) – Whether to show a placeholder if the specified `image_url` couldn't be found.

Raises

- `exc.EmptyImagePathError` – If an empty image path was received.
- `exc.ImageNotFoundError` – If the image could not be found.
- `exc.MissingDimensionsError` – If `fill` argument was True, but width or height was not passed.

Returns URL to the generated and cached image.

Return type str

Usage: Generate an image from the supplied image URL that will fit within an area of 600px width and 400px height:

```
resize('somedir/kittens.png', '600x400')
```

Resize and crop so that the image will fill the entire area:

```
resize('somedir/kittens.png', '300x300', fill=1)
```

Convert to JPG:

```
resize('somedir/kittens.png', '300x300', format='jpg')
```

flask_resize.resizing.**create_placeholder_image**(*width=None*, *height=None*, *message=None*)

Create a placeholder image that specified its width and height, and an optional text.

Parameters

- **width** (Optional[str]) – Width to use for the image. Will use *height* if not provided.
- **height** (Optional[str]) – Height to use for the image. Will use *width* if not provided.
- **message** (Optional[str]) – Text to add to the center of the placeholder image.

Raises `exc.MissingDimensionsError` – If neither *width* nor *height* are provided.

Returns The placeholder image.

Return type PIL.Image

flask_resize.resizing.**format_to_ext**(*format*)

Return the file extension to use for format

flask_resize.resizing.**image_data**(*img*, *format*, ***save_options*)
Save a PIL Image instance and return its byte contents

flask_resize.resizing.**make_opaque**(*img*, *bgcolor*)
Apply a background color to image

Parameters

- **img** (*PIL.Image*) – Image to alter.
- **bgcolor** (*str*) – A `parse_rgb()` parseable value to use as background color.

Returns A new image with the background color applied.

Return type PIL.Image

flask_resize.resizing.**make_resizer**(*config*)
Resizer instance factory

5.2 Storage

class flask_resize.storage.**FileStorage**(*base_path*)
Local file based storage

Note that to follow the same convention as for *S3Storage*, all methods' passed in and extracted paths must use forward slash as path separator. the path separator. The only exception is the *base_path* passed in to the constructor.

Parameters **base_path** (*str*) – The directory where files will be read from and written to. Expected to use the local OS's path separator.

delete(*key*)
Delete file at specified key

Parameters **key** (*str*) – The key / relative file path to delete

delete_tree(*subdir*)
Recursively deletes all regular files in specified sub-directory

Parameters **subdir** (*str*) – The subdirectory to delete files in

Returns Yields deleted subdirectory's filenames

Return type Generator[str, str, None]

exists(*key*)
Check if the key exists in the backend

Parameters **key** (*str*) – The key / relative file path to check

Returns Whether the file exists or not

Return type bool

get(*key*)
Get binary file data for specified key

Parameters **key** (*str*) – The key / relative file path to get data for

Returns The file's binary data

Return type bytes

list_tree(*subdir*)
Recursively yields all regular files' names in a sub-directory of the storage backend.

Keys/filenames are returned *self.base_path*-relative, and uses forward slash as a path separator, regardless of OS.

Parameters **subdir** (*str*) – The subdirectory to list paths for

Returns Yields subdirectory's filenames

Return type Generator[str, str, None]

save(*key*, *bdata*)
Store binary file data at specified key

Parameters

- **key** (*str*) – The key / relative file path to store data at
- **bdata** (*bytes*) – The file data

class flask_resize.storage.**S3Storage**(*bucket*, *access_key=None*, *secret_key=None*, *region_name=None*, *file_acl='public-read'*)
Amazon Web Services S3 based storage

Parameters

- **bucket** (*str*) – Bucket name
- **access_key** (*Any[str, None]*) – The access key. Defaults to reading from the local AWS config.
- **secret_key** (*Any[str, None]*) – The secret access key. Defaults to reading from the local AWS config.
- **region_name** (*Any[str, None]*) – The name of the bucket's region. Defaults to reading from the local AWS config.
- **file_acl** (*str*) – The ACL to set on uploaded images. Defaults to "public-read"

base_url
The base URL for the storage's bucket

Returns The URL

Return type str

delete(*relative_path*)
Delete file at specified key

Parameters **key** (*str*) – The key to delete

delete_tree(*subdir*)
Recursively deletes all keys in specified sub-directory (prefix)

Parameters **subdir** (*str*) – The subdirectory to delete keys in

Returns Yields subdirectory's deleted keys

Return type Generator[str, str, None]

exists(*relative_path*)
Check if the key exists in the backend

Parameters **key** (*str*) – The key to check

Returns Whether the key exists or not

Return type bool

get(*relative_path*)
Get binary file data for specified key

Parameters **key** (*str*) – The key to get data for

Returns The file's binary data

Return type bytes

list_tree(*subdir*)
Recursively yields all keys in a sub-directory of the storage backend

Parameters **subdir** (*str*) – The subdirectory to list keys for

Returns Yields subdirectory's keys

Return type Generator[str, str, None]

save(*relative_path*, *bdata*)
Store binary file data at specified key

Parameters

- **key** (*str*) – The key to store data at
- **bdata** (*bytes*) – The file data

class flask_resize.storage.**Storage**(**args*, ***kwargs*)
Storage backend base class

flask_resize.storage.**make**(*config*)
Generate storage backend from supplied config

Parameters **config** (*dict*) – The config to extract settings from

Returns A *Storage* sub-class, based on the *RESIZE_STORAGE_BACKEND* value.

Return type Any[*FileStorage*, *S3Storage*]

Raises RuntimeError – If another *RESIZE_STORAGE_BACKEND* value was set

5.3 Cache

class flask_resize.cache.**Cache**
Cache base class

class flask_resize.cache.**NoopCache**
No-op cache, just to get the same API regardless of whether cache is used or not.

add(*unique_key*)
Add key to cache

Parameters **unique_key** (*str*) – Add this key to the cache

Returns Whether key was added or not

Return type bool

all()
List all keys in cache

Returns All the keys in the set, as a list

Return type List[str]

clear()
Remove all keys from cache

Returns Whether any keys were removed or not

Return type bool

exists(*unique_key*)
Check if key exists in cache

Parameters **unique_key** (*str*) – Unique key to check for

Returns Whether key exist in cache or not

Return type bool

remove(*unique_key*)
Remove key from cache

Parameters **unique_key** (*str*) – Remove this key from the cache

Returns Whether key was removed or not

Return type bool

transaction(*unique_key*, *ttl=600*)
No-op context-manager for transactions. Always yields *True*.

class flask_resize.cache.**RedisCache**(*host='localhost'*, *port=6379*, *db=0*, *password=None*, *key='flask-resize'*)
A Redis-based cache that works with a single set-type key

Basically just useful for checking whether an expected value in the set already exists (which is exactly what's needed in Flask-Resize)

add(*unique_key*)
Add key to cache

Parameters **unique_key** (*str*) – Add this key to the cache

Returns Whether key was added or not

Return type bool

all()
List all keys in cache

Returns All the keys in the set, as a list

Return type List[str]

clear()
Remove all keys from cache

Returns Whether any keys were removed or not

Return type bool

exists(*unique_key*)
Check if key exists in cache

Parameters **unique_key** (*str*) – Unique key to check for

Returns Whether key exist in cache or not

Return type bool

remove(*unique_key*)
Remove key from cache

Parameters **unique_key** (*str*) – Remove this key from the cache

Returns Whether key was removed or not

Return type bool

transaction(*unique_key*, *ttl=600*)
Context-manager to use when it's important that no one else handles *unique_key* at the same time (for example when saving data to a storage backend).

Parameters

- **unique_key** (*str*) – The unique key to ensure atomicity for
- **ttl** (*int*) – Time before the transaction is deemed irrelevant and discarded from cache. Is only relevant if the host forcefully restarts.

flask_resize.cache.**make**(*config*)
Generate cache store from supplied config

Parameters **config** (*dict*) – The config to extract settings from

Returns A *Cache* sub-class, based on the *RESIZE_CACHE_STORE* value.

Return type Any[*RedisCache*, *NoopCache*]

Raises RuntimeError – If another *RESIZE_CACHE_STORE* value was set

5.4 Configuration

class flask_resize.configuration.**Config**(**config*)
The main configuration entry point

classmethod from_dict(*dct*, *default_overrides=None*)
Turns a dictionary with *RESIZE_* prefix config options into lower-case keys on this object.

Parameters

- **dct** (*dict*) – The dictionary to get explicitly set values from
- **default_overrides** (*Any[dict,None]*) – A dictionary with default overrides, where all declared keys' values will be used as defaults instead of the "app default", in case a setting wasn't explicitly added to config.

5.5 Constants

flask_resize.constants.**DEFAULT_NAME_HASHING_METHOD = 'sha1'**
Default filename hashing method for generated images

flask_resize.constants.**DEFAULT_REDIS_KEY = 'flask-resize'**
Default key to store redis cache as

flask_resize.constants.**DEFAULT_TARGET_DIRECTORY = 'resized-images'**
Default target directory for generated images

flask_resize.constants.**JPEG = 'JPEG'**
JPEG format

flask_resize.constants.**PNG = 'PNG'**
PNG format

flask_resize.constants.**SUPPORTED_OUTPUT_FILE_FORMATS = ('JPEG', 'PNG')**
Image formats that can be generated

flask_resize.constants.**SVG = 'SVG'**
SVG format

5.6 Exceptions

exception flask_resize.exc.**Boto3ImportError**
Raised when S3 is configured, but the *boto3* library is not installed.

exception flask_resize.exc.**CacheMiss**
Raised when a cached image path could not be found

exception flask_resize.exc.**CairoSVGImportError**
Raised when an SVG input file is encountered but CairoSVG is not installed.

exception flask_resize.exc.**EmptyImagePathError**
Raised if an empty image path was encountered.

exception flask_resize.exc.**GenerateInProgress**
The image is currently being generated

exception flask_resize.exc.**ImageNotFoundError**
Raised if the image could not be fetched from storage.

exception flask_resize.exc.**InvalidDimensionsError**
Raised when a dimension string/tuple is improperly specified.

exception flask_resize.exc.**InvalidResizeSettingError**
Raised when a resize argument such as if *width* was invalid.

exception flask_resize.exc.**MissingDimensionsError**
Raised when width and/or height is missing.

exception flask_resize.exc.**RedisImportError**
Raised when Redis cache is configured, but the *redis* library is not installed.

exception flask_resize.exc.**UnsupportedImageFormatError**
Raised when an unsupported output image format is encountered.

CHAPTER 6

Changelog

6.1 2.0.4 (2017-12-19)

- **Feature** Add the ability to pass in a pre-configured redis host with option *RESIZE_REDIS_HOST*.

6.2 2.0.3 (2017-04-19)

- **Feature** *RESIZE_REDIS_PASSWORD* added

6.3 2.0.2 (2017-03-25)

- **Bugfix** *RESIZE_RAISE_ON_GENERATE_IN_PROGRESS* wasn't respected - error was always raised.

6.4 2.0.1 (2017-03-23)

- **Bugfix** *s3* was erroneously added as a dependency instead of *boto3* in the recommended installation option *flask-resize[full]*

6.5 2.0.0 (2017-03-22)

- **Feature** Added commands for generating images, listing/clearing cache and generated images (see *Command line usage*)
- **Feature** Ability to run resizing without a Flask app being involved (see *Setting up for standalone usage*)

- **Enhancement** Windows is now supported, at least on paper. The test suite runs on a Windows host after each pushed commit, using the excellent CI service AppVeyor. The project still needs some real world usage though. Please report any issues you might find to the Github issues page. (see *Compatibility*)
- **Enhancement** *dimensions* is now an optional argument. Useful when just converting to a different format for example. (see *dimensions*)
- **Enhancement** Added some basic logging statements for debugging purposes
- **Enhancement** Pypy and Python 3.6 are now officially supported
- **Bugfix** Concurrent generation of the same image wasn't handled properly
- **Bugfix** There was a logic error when installing *flask-resize[full]*. *redis* and *s3* weren't installed below python3.4.
- **Breaking** *flask_resize.resize* is no longer available. It's instead accessible when creating the Flask app extension. I.e: *resize = flask_resize.Resize(app)* or *resize = flask_resize.Resize(); resize.init_app(app)*

6.6 1.0.3 (2017-03-17)

- **Improvement** Add image to cache when the path is found while checking for its existence, even though it wasn't in the local cache since before. This way a path can be cached even though the generation was done on another computer, or after the local cache has been cleared.

6.7 1.0.2 (2017-03-17)

- **Breaking** Removed option *RESIZE_USE_S3* and replaced it with *RESIZE_STORAGE_BACKEND*, more explicit and simple.
- **Improvement** Automatically load *RESIZE_S3_ACCESS_KEY*, *RESIZE_S3_SECRET_KEY* and *RESIZE_S3_REGION* settings using *botocore.session*, in case they are not explicitly specified.

6.8 1.0.1 (2017-03-17)

- **Improvement** Added the option *RESIZE_S3_REGION*. Can be set if the S3 region has to be specified manually for some reason.

6.9 1.0.0 (2017-03-17)

Flask-Resize 1.0

The big changes are:

- **Feature** Support Amazon S3 as a storage backend
- **Feature Support Redis caching. The usefulness of this is threefold:**
    1. When using S3 as a storage backend to save on HTTP requests
    2. No need to check if file exists on file storage backends - perhaps noticable on servers with slow disk IO
    3. Much lower chance of multiple threads/processes trying to generate the same image at the same time.

- **Breaking** Please note that this release forces all generated images to be recreated because of a change in the creation of the "unique key" which is used to determine if the image has already been generated.
- **Breaking** Drop RESIZE_HASH_FILENAME as an option - always hash the cache object's filename. Less moving parts in the machinery.
- **Breaking** Rename RESIZE_CACHE_DIR to RESIZE_TARGET_DIRECTORY to better reflect what the setting does, now that we have Redis caching.
- **Breaking** Use SHA1 as default filename hashing method for less likelyhood of collision

6.10 0.8.0 (2016-10-01)

- **Improvement** Release as a universal python wheel

6.11 0.8.0 (2016-09-07)

- **Feature** Support SVG as input format by utilizing [CairoSVG](http://cairosvg.org/).

6.12 0.7.0 (2016-09-01)

- **Improvement** Keep ICC profile from source image
- **Minor fix** Clarify that Python 3.5 is supported

6.13 0.6.0 (2015-10-01)

- **Bugfix** Fill doesn't cut the image any more

6.14 0.5.2 (2015-06-12)

- **Bugfix** Fix Python 2 regression

6.15 0.5.1 (2015-06-12)

- **Improvement** Tests that actually convert images with the `flask_resize.resize()` command.
- **Improvement** Validates that `RESIZE_ROOT` and `RESIZE_URL` are strings.

6.16 0.5.0 (2015-06-10)

- **Improvement** Proper documentation, hosted on `RTD`
- **Improvement** Properly documented all functions and classes
- **Improvement** Continuous integration with `Travis CI`

- **Improvement** Code coverage with `coveralls`
- **Improvement** More tests
- **Change** Dropped `nose` in favor of `py.test`
- **Change** Removed unused method `Resize.teardown`

6.17 0.4.0 (2015-04-28)

- **Feature** Adds the setting `RESIZE_NOOP` which will just return the passed in image path, as is. This was added to ease the pain of unit testing when Flask-Resize is a part of the project.
- **Change** Added more tests

6.18 0.3.0 (2015-04-23)

- **Feature** Adds the `bgcolor` option for specifying a background color to apply to the image.

6.19 0.2.5 (2015-03-20)

- **Bugfix** Because of a logic error no exception was raised when file to resize didn't exist

6.20 0.2.4 (2015-03-19)

- **Bugfix** Fix for pip parse_requirements syntax change (fixes #6)

6.21 0.2.3 (2015-01-30)

- **Feature** Python 3.4 support (might work in other Pythons as well)

6.22 0.2.2 (2014-02-01)

- **Bugfix** Placeholders were being regenerated on each page load.

6.23 0.2.1 (2013-12-09)

- **Bugfix** Same placeholder reason text was used for all resizes with identical dimensions

6.24 0.2.0 (2013-12-04)

- **Feature** Support for generating image placeholders

6.25 0.1.1 (2013-11-09)

- **Bugfix** Format argument wasn't respected
- **Change** Bumped default JPEG quality to 80

6.26 0.1.0 (2013-11-09)

- Initial version

CHAPTER 7

To-do

7.1 Automatic fitting of placeholder text

See issue #7.

7.2 Support for signals

Generate images directly instead of when web page is first hit.

CHAPTER 8

About

Created by Jacob Magnusson.

Python Module Index

f

Index

Symbols

A

B

C

D

E

F

G

I

J

L

M