Flask-IIIIF Documentation

Release 0.5.0

CERN

May 23, 2018
## Contents

1 Contents .......................................................... 3
   1.1 Installation .................................................... 4
   1.2 Quickstart ....................................................... 4
   1.3 Configuration .................................................. 5
   1.4 API .............................................................. 5
   1.5 Changes ......................................................... 11
   1.6 Contributing .................................................... 13
   1.7 License .......................................................... 13
   1.8 Authors ........................................................ 13

Python Module Index .................................................. 15
Flask-IIIF is a Flask extension permitting easy integration with the International Image Interoperability Framework (IIIF) API standards.
CHAPTER 1

Contents

• Installation
  – Requirements
• Quickstart
  – A Minimal Example
• Configuration
• API
  – Flask-IIIF
  – Cache
  – RESTful
• Changes
  – Version 0.5.0 (released 2018-05-18)
  – Version 0.4.0 (released 2018-04-17)
  – Version 0.3.2 (released 2018-04-09)
  – Version 0.3.1 (released 2017-08-18)
  – Version 0.3.0 (released 2017-08-17)
  – Version 0.2.0 (released 2015-05-22)
  – Version 0.1.0 (released 2015-04-28)
• Contributing
• License
• Authors
1.1 Installation

Flask-IIIF is on PyPI so all you need is:

```
$ pip install flask-iiif
```

The development version can be downloaded from its page at GitHub.

```
$ git clone https://github.com/inveniosoftware/flask-iiif.git
$ cd flask-iiif
$ python setup.py develop
$ ./run-tests.sh
```

1.1.1 Requirements

Flask-IIIF has the following dependencies:

- Flask
- blinker
- six

Flask-IIIF requires Python version 2.6, 2.7 or 3.3+

1.2 Quickstart

This part of the documentation will show you how to get started in using Flask-IIIF with Flask.

This guide assumes that you have successfully installed Flask-IIIF and that you have a working understanding of Flask framework. If not, please follow the installation steps and read about Flask at http://flask.pocoo.org/docs/.

1.2.1 A Minimal Example

A minimal Flask-IIIF usage example looks like this.

First, let’s create the application and initialise the extension:

```python
from flask import Flask, session, redirect
from flask_iiif import IIIF
app = Flask("myapp")
ext = IIIF(app=app)
```

Second, let’s create Flask-RESTful api instance and register image resource.

```python
from flask_restful import Api
api = Api(app=app)
ext.init_restful(api)
```
1.3 Configuration

IIIF configuration.

flask_iiif.config.IIIF_CACHE_HANDLER
Add the preferred cache adaptor.

See also:
ImageCache

flask_iiif.config.IIIF_CACHE_TIME
How much time the image would be cached.

flask_iiif.config.IIIF_QUALITIES
The supported image qualities.

See also:
IIIF Image API

flask_iiif.config.IIIF_CONVERTERS
The supported image converters.

flask_iiif.config.IIIF_VALIDATIONS
The IIIF Image API validation.

See also:
IIIF Image API v1 and IIIF Image API v2

flask_iiif.config.IIIF_API_INFO_RESPONSE_SKELETON
Information request document for the image.

See also:
IIIF Image API v1 Information request and IIIF Image API v2 Information request

1.4 API

This documentation section is automatically generated from Flask-IIIF source code.

1.4.1 Flask-IIIF

Multimedia Image API.

class flask_iiif.api.IIIFImageAPIWrapper(image)
IIIF Image API Wrapper.

apply_api(**kwargs)
Apply the IIIF API to the image.

Example to apply the IIIF API:

```python
from flask_iiif.api import IIIFImageAPIWrapper
image = IIIFImageAPIWrapper.from_file(path)
image.apply_api()
```

(continues on next page)
Note:

- If the version is not specified it will fallback to version 2.0.
- Please note the `validate_api()` should be run before `apply_api()`.

```python
apply_quality(value)
IIIF apply quality.
Apply quality().

apply_region(value)
IIIF apply crop.
Apply crop().

apply_rotate(value)
IIIF apply rotate.
Apply rotate().

apply_size(value)
IIIF apply resize.
Apply resize().

classmethod open_image(source)
Create an `MultimediaImage` instance.

Parameters

- `source` (BytesIO object) – The image image string
- `source_type` (str) – the type of data

Returns a `MultimediaImage` instance

static validate_api(**kwargs)
Validate IIIF Image API.

Example to validate the IIIF API:

```python
from flask_iiif.api import IIIFImageAPIWrapper

IIIFImageAPIWrapper.validate_api(
    version=version,
    region=region,
    size=size,
    rotation=rotation,
    quality=quality,
    image_format=image_format
)
```
Note: If the version is not specified it will fallback to version 2.0.

class flask_iiif.api.MultimediaImage(image)
Multimedia Image API.

Initializes an image api with IIIF standards. You can:

- Resize `resize()`.
- Crop `crop()`.
- Rotate `rotate()`.
- Change image quality `quality()`.

Example of editing an image and saving it to disk:

```python
from flask_iiif.api import MultimediaImage

image = IIIFImageAPIWrapper.from_file(path)
# Rotate the image
image.rotate(90)
# Resize the image
image.resize('300,200')
# Crop the image
image.crop('20,20,400,300')
# Make the image black and white
image.quality('grey')
# Finally save it to /tmp
image.save('/tmp')
```

Example of serving the modified image over http:

```python
from flask import current_app, Blueprint
from flask_iiif.api import MultimediaImage

@blueprint.route('/serve/<string:uuid>/<string:size>')</def serve_thumbnail(uuid, size):
    """Serve the image thumbnail.
    :param size: The desired image size.
    """
    # Initialize the image with the uuid
    path = current_app.extensions['iiif'].uuid_to_path(uuid)
    image = IIIFImageAPIWrapper.from_file(path)
    # Resize it
    image.resize(size)
    # Serve it
    return send_file(image.serve(), mimetype='image/jpeg')
```

crop(coordinates)
Crop the image.

Parameters coordinates (str) – The coordinates to crop the image

Note:
- `coordinates` must have the following pattern:
- ‘x,y,w,h’: in pixels.
- ‘pct:x,y,w,h’: percentage.

classmethod from_file(path)
Return the image object from the given path.

Parameters
path (str) – The absolute path of the file
Returns a MultimediaImage instance

classmethod from_string(source)
Create an MultimediaImage instance.

Parameters
source (BytesIO object) – the image string
Returns a MultimediaImage instance

static percent_to_number(number)
Calculate the percentage.

quality(quality)
Change the image format.

Parameters
quality (str) – The image quality should be in (default, grey, bitonal, color)

Note: The library supports transformations between each supported mode and the “L” and “RGB” modes. To convert between other modes, you may have to use an intermediate image (typically an “RGB” image).

static reduce_by(nominally, dominator)
Calculate the ratio.

resize(dimensions, resample=None)
Resize the image.

Parameters

* dimensions (str) – The dimensions to resize the image
* resample (PIL.Image algorithm) – The algorithm to be used

Note:
* dimensions must be one of the following:
  - ‘w,’ The exact width, height will be calculated.
  - ‘,h’ The exact height, width will be calculated.
  - ‘pct:n’ Image percentage scale.
  - ‘w,h’ The exact width and height.
  - ‘!w,h’ Best fit for the given width and height.

rotate(degrees, mirror=False)
Rotate the image by given degrees.

Parameters

* degrees (float) – The degrees, should be in range of [0, 360]
• **mirror** (*bool*) – Flip image from left to right

**static sanitize_format_name** (*value*)
Lowercase formats and make sure that jpg is written as jpeg.

**save** (*path, image_format='jpeg', quality=90*)
Store the image to the specific path.

  **Parameters**

  • **path** (*str*) – absolute path
  • **image_format** (*str*) – (gif, jpeg, pdf, png, tif)
  • **quality** (*int*) – The image quality; [1, 100]

**Note:** *image_format* = jpg will not be recognized by **PIL.Image** and it will be changed to jpeg.

**serve** (*image_format='png', quality=90*)
Return a BytesIO object to easily serve it thought HTTP.

  **Parameters**

  • **image_format** (*str*) – (gif, jpeg, pdf, png, tif)
  • **quality** (*int*) – The image quality; [1, 100]

**Note:** *image_format* = jpg will not be recognized by **PIL.Image** and it will be changed to jpeg.

**size()**
Return the current image size.

**Returns**  the image size

**class flask_iiif.api.MultimediaObject**
The Multimedia Object.

### 1.4.2 Cache

Abstract simple cache definition.

All cache adaptors must at least implement **get()** and **set()** methods.

**class flask_iiif.cache.cache.ImageCache**
Abstract cache layer.

  **delete** (*key*)
  Delete the specific key.

  **flush()**
  Flush the cache.

  **get** (*key*)
  Return the key value.

  **Parameters** **key** – the object’s key

  **set** (*key, value, timeout=None*)
  Cache the object.

  **Parameters**
• **key** – the object’s key
• **value** (*StringIO.StringIO* object) – the stored object
• **timeout** – the cache timeout in seconds

**timeout**
Return default timeout from config.

Implement a simple cache.

```python
class flask_iiif.cache.simple.ImageSimpleCache
    Simple image cache.
    
def delete(key)
        Delete the specific key.
    
def flush()
        Flush the cache.
    
def get(key)
        Return the key value.
        
        **Parameters**
        • **key** – the object’s key
        
        **Returns**
        the stored object
        
        **Return type**
        *BytesIO* object
    
def set(key, value, timeout=None)
        Cache the object.
        
        **Parameters**
        • **key** – the object’s key
        • **value** (*BytesIO* object) – the stored object
        • **timeout** – the cache timeout in seconds
```

### 1.4.3 RESTful

Multimedia IIIF Image API.

```python
class flask_iiif.restful.IIIFImageAPI
    IIIF API Implementation.
```

**Note:**

• **IIIF IMAGE API v1.0**
  – For more infos please visit <http://iiif.io/api/image/>.

• **IIIF Image API v2.0**
  – For more infos please visit <http://iiif.io/api/image/2.0/>.

• The API works only for GET requests
• The image process must follow strictly the following workflow:
  – Region
  – Size
  – Rotation
- Quality
  
- Format

```python
get (version, uuid, region, size, rotation, quality, image_format)
Run IIIF Image API workflow.
```

```python
class flask_iiif.restful.IIIFImageBase
IIIF Image Base.

get (version, uuid)
Get IIIF Image Base.
```

**Note:** It will redirect to iiifimageinfo endpoint with status code 303.

```python
class flask_iiif.restful.IIIFImageInfo
IIIF Image Info.

get (**kwargs)
Get IIIF Image Info.
```

### 1.5 Changes

Here you can see the full list of changes between each Flask-IIIF release.

#### 1.5.1 Version 0.5.0 (released 2018-05-18)

- Fixes
  - wrong ratio calculation for best fit
- New features
  - adds black background to requested best fit thumbnail or gif if the image does not cover the whole window of requested size

#### 1.5.2 Version 0.4.0 (released 2018-04-17)

- Fixes unicode filename issues.
- Changes default resampling algorithm to BICUBIC for better image quality.
- Adds support for _external, _scheme etc parameters for iiif_image_url.

#### 1.5.3 Version 0.3.2 (released 2018-04-09)

- Security
  - Fixed missing API protection on image metadata endpoint.
1.5.4 Version 0.3.1 (released 2017-08-18)

- Deployment changes.

1.5.5 Version 0.3.0 (released 2017-08-17)

- New features
  - Adds TIFF image support to the default config.
  - Adds proper GIF resize.
  - Adds optional Redis cache.
- Notes
  - Minimum Pillow version is update to 3.4.

1.5.6 Version 0.2.0 (released 2015-05-22)

- Incompatible changes
  - Removes `uuid_to_path_handler` callback.
  - Updates error classes names (MultimediaImageResizeError and MultimediaImageCropError).
- New features
  - Adds image information request endpoint `<uuid>/info.json` which contains available metadata for the image, such as the full height and width, and the functionality available for the image, such as the formats in which it may be retrieved, and the IIIF profile used.
  - Adds new signals to REST API that permits to have access before and after process of the request as well as after the validation of IIIF.
  - Adds a configurable decorator to the REST API which can configure with the `api_decorator_handler`.
  - Adds the `uuid_to_image_opener_handler` which can handle both `fullpath` and `bytestream` as source.
- Improved features
  - Improves the initialisation of the REST API by adding a possibility to override the default API prefix `/api/multimedia/image/`.
  - Adds better testing cases and increases the overall test efficiency.
- Notes
  - The decorator can be used to restrict access to the REST API.

1.5.7 Version 0.1.0 (released 2015-04-28)

- Initial public release.
1.6 Contributing

Bug reports, feature requests, and other contributions are welcome. If you find a demonstrable problem that is caused by the code of this library, please:

1. Search for already reported problems.
2. Check if the issue has been fixed or is still reproducible on the latest master branch.
3. Create an issue with a test case.

If you create a feature branch, you can run the tests to ensure everything is operating correctly:

```
$ ./run-tests.sh
```

1.7 License

Flask-IIIF is free software; you can redistribute it and/or modify it under the terms of the Revised BSD License quoted below.

Copyright (C) 2014, 2016 CERN.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an Intergovernmental Organization or submit itself to any jurisdiction.

1.8 Authors

Flask-IIIF was originally developed for use in Invenio digital library software.

Contact us at info@inveniosoftware.org
1.8.1 Contributors

- Alexander Ioannidis <a.ioannidis@cern.ch>
- Harris Tzovanakis <drjova@cern.ch>
- Jiri Kuncar <jiri.kuncar@cern.ch>
- Orestis Melkonian <melkon.or@gmail.com>
- Tibor Simko <tibor.simko@cern.ch>
Python Module Index

f
flask_iiif.api, 5
flask_iiif.cache.cache, 9
flask_iiif.cache.simple, 10
flask_iiif.config, 5
flask_iiif.restful, 10
A
apply_api() (flask_iiif.api.IIIFImageAPIWrapper method), 5
apply_quality() (flask_iiif.api.IIIFImageAPIWrapper method), 6
apply_region() (flask_iiif.api.IIIFImageAPIWrapper method), 6
apply_rotate() (flask_iiif.api.IIIFImageAPIWrapper method), 6
apply_size() (flask_iiif.api.IIIFImageAPIWrapper method), 6

C
crop() (flask_iiif.api.MultimediaImage method), 7

D
delete() (flask_iiif.cache.cache.ImageCache method), 9
delete() (flask_iiif.cache.simple.ImageSimpleCache method), 10

F
flask_iiif.api (module), 5
flask_iiif.cache.cache (module), 9
flask_iiif.cache.simple (module), 10
flask_iiif.config (module), 5
flask_iiif.restful (module), 10
flush() (flask_iiif.cache.cache.ImageCache method), 9
flush() (flask_iiif.cache.simple.ImageSimpleCache method), 10
from_file() (flask_iiif.api.MultimediaImage class method), 8
from_string() (flask_iiif.api.MultimediaImage class method), 8

G
get() (flask_iiif.cache.cache.ImageCache method), 9
get() (flask_iiif.cache.simple.ImageSimpleCache method), 10
get() (flask_iiif.restful.IIIFImageAPI method), 11

I
IIIF_API_INFO_RESPONSE_SKELETON (in module flask_iiif.config), 5
IIIF_CACHE_HANDLER (in module flask_iiif.config), 5
IIIF_CACHE_TIME (in module flask_iiif.config), 5
IIIF_CONVERTERS (in module flask_iiif.config), 5
IIIF_QUALITIES (in module flask_iiif.config), 5
IIIF_VALIDATIONS (in module flask_iiif.config), 5
IIIFImageAPI (class in flask_iiif.restful), 10
IIIFImageAPIWrapper (class in flask_iiif.api), 5
IIIFImageBase (class in flask_iiif.restful), 11
IIIFImageInfo (class in flask_iiif.restful), 11
ImageCache (class in flask_iiif.cache.cache), 9
ImageSimpleCache (class in flask_iiif.cache.simple), 10

M
MultimediaImage (class in flask_iiif.api), 7
MultimediaObject (class in flask_iiif.api), 9

O
open_image() (flask_iiif.api.IIIFImageAPIWrapper class method), 6

P
percent_to_number() (flask_iiif.api.MultimediaImage static method), 8

Q
quality() (flask_iiif.api.MultimediaImage method), 8

R
reduce_by() (flask_iiif.api.MultimediaImage static method), 8
resize() (flask_iiif.api.MultimediaImage method), 8
rotate() (flask_iiif.api.MultimediaImage method), 8
S
sanitize_format_name() (flask_iiif.api.MultimediaImage static method), 9
save() (flask_iiif.api.MultimediaImage method), 9
serve() (flask_iiif.api.MultimediaImage method), 9
set() (flask_iiif.cache.cache.ImageCache method), 9
set() (flask_iiif.cache.simple.ImageSimpleCache method), 10
size() (flask_iiif.api.MultimediaImage method), 9

T
timeout (flask_iiif.cache.cache.ImageCache attribute), 10

V
validate_api() (flask_iiif.api.IIIFImageAPIWrapper static method), 6