Flask-IIIF Documentation

Release 0.5.3

CERN

Mar 02, 2020

Contents

Flask-IIIF is a Flask extension permitting easy integration with the International Image Interoperability Framework (IIIF) API standards.

CHAPTER 1

Contents

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:

```
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.

```
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_REDIS_PREFIX`**
Sets prefix for redis keys, default: *iiif*

`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_FORMATS`**
The supported image formats with their MIME type.

`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:

```
from flask_iiif.api import IIIFImageAPIWrapper

image = IIIFImageAPIWrapper.from_file(path)

image.apply_api(
    version=version,
    region=region,
    size=size,
    rotation=rotation,
    quality=quality
)
```

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()*.

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:

```
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:

```
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')
# Finaly save it to /tmp
image.save('/tmp')
```

Example of serving the modified image over http:

```
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 uuid: The document uuid.
    :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)
```

(continues on next page)

(continued from previous page)

```
    # 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 HTTTP.

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

1.4.3 RESTful

Multimedia IIIF Image API.

class flask_iiif.restful.**IIIFImageAPI**
IIIF API Implementation.

Note:

- **IIF 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

---

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

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.

---

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 0.5.3 (released 2019-11-21)

- Adds Last-Lodified and If-Modified-Since to imageapi
- Removes warning message for LocalProxy
- Fixes werkzeug deprecation warning

1.5.2 Version 0.5.2 (released 2019-07-25)

- Sets Redis cache prefix
- Fixes cache control headers

1.5.3 Version 0.5.1 (released 2019-05-23)

- Fixes syntax error in documentation
- Fixes import sorting

1.5.4 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

1.5.5 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.

1.5.6 Version 0.3.2 (released 2018-04-09)

- Security
  - Fixed missing API protection on image metadata endpoint.

1.5.7 Version 0.3.1 (released 2017-08-18)

- Deployment changes.

1.5.8 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.

1.5.9 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 be 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.

1.5.10 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

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

Index