invenio-iiif Documentation

Release 1.1.0

CERN

Mar 19, 2020

Contents

IIIF Image API implementation for Invenio

Features:

- Thumbnail generation and previewing of images.
- Allows to preview, resize and zoom images, by implementing the IIIF API.
- Provide celery task to create image thumbnails.

Further documentation available: https://invenio-iiif.readthedocs.io/

CHAPTER 1

User's Guide

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

1.1 Installation

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

```
$ pip install invenio-iiif
```

Invenio-IIIF uses Pillow for image processing.

To create cover images from PDFs, you need to install locally the ImageMagick image library. See https://imagemagick.org for details.

1.2 Configuration

IIIF API for Invenio.

invenio_iiif.config.**IIIF_API_PREFIX = '/iiif/'**
URL prefix to IIIF API.

invenio_iiif.config.**IIIF_PREVIEWER_PARAMS = {'size': '750,'}**
Parameters for IIIF image previewer extension.

invenio_iiif.config.**IIIF_PREVIEW_TEMPLATE = 'invenio_iiif/preview.html'**
Template for IIIF image preview.

invenio_iiif.config.**IIIF_UI_URL = '/api/iiif/'**
URL to IIIF API endpoint (allow hostname).

Invenio-IIIF depends heavily on Flask-IIIF module for images transformation. Configurations related to images formats, resize and caching are provided by `Flask-IIIF`:

- **IIIF_RESIZE_RESAMPLE** Specifies the algorithm used to resample the image. The default one is *PIL.image.BICUBIC*
- **IIIF_CACHE_HANDLER** Specifices how to cache thumbnails, e.g. in memory, Redis or any custom implementation.
- **IIIF_CACHE_TIME** Specifies for how long images will be cached.
- **IIIF_FORMATS** Specifies the supported images formats and associated MIME types

1.3 Usage

Invenio module to serve images complying with IIIF standard.

Invenio-IIIF integrates Invenio-Records-Files with Flask-IIIF to provide an endpoint for serving images complying with the International Image Interoperability Framework (IIIF) API standards.

Invenio-IIIF registers the REST API endpoint provided by Flask-IIIF in the Invenio instance through entry points. On each image request, it delegates authorization check and file retrieval to Invenio-Files-REST and it serves the image after adaptation by Flask-IIIF.

1.3.1 Basics

Preview

Invenio-IIIF can be used in a combination with Invenio-Previewer to preview images. It provides an extension to preview the most common image formats and it is exposed via the entry point named `iiif_image`.

The template used to render the image can be configured with the configuration `IIIF_PREVIEW_TEMPLATE`.

To use it with an existing Invenio instance you should update your `PREVIEWER_PREFERENCE` configuration by adding `iiif_image` to the list of available previewers with the desired precedence order. For example:

```
PREVIEWER_PREFERENCE = [
    'iiif_image',
    'json_prismjs',
    'xml_prismjs',
    'pdfjs',
    'zip',
]
```

Caching

Image caching is provided by `Flask-IIIF`. You can change cache expiration by setting the `IIIF_CACHE_TIME`.

```
# 60 seconds * 60 (1 hour) * 24 (1 day)
IIIF_CACHE_TIME = 60 * 60 * 24
```

By default `Flask-IIIF` caches images in memory. You can cache images on Redis by using the available `ImageRedisCache` handler (providing Redis URL) or point to your own custom handler by setting it in `IIIF_CACHE_HANDLER`.

```
# Cache handler
IIIF_CACHE_HANDLER = 'flask_iiif.cache.redis:ImageRedisCache'

# Redis URL
IIIF_CACHE_REDIS_URL = 'redis://localhost:6379/0'
```

PDF cover

When the retrieved file is a PDF and with ImageMagick/Wand installed in your environment (see *Installation* documentation), then a cover image with the first page of the PDF file can be generated and served it as preview.

Authorization

Permissions to retrieve the requested images are delegated to Invenio-Files-REST. At each request, authorization is checked to ensure that the user has sufficient privileges.

Thumbnails

The module can pre-cache thumbnails of requested images. It provides a celery task that will fetch a given image and resize it to create a thumbnail. It is then cached so it can be served efficiently.

```
from invenio_iiif.tasks import create_thumbnail
create_thumbnail(image_key, '250')
```

1.3.2 Initialization

First create a Flask application:

```
>>> from flask import Flask
>>> app = Flask('myapp')
>>> app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite://'
```

Then, you can define the IIIF API prefix by setting the Invenio configuration `IIIF_UI_URL`.

```
>>> app.config.update(IIIF_UI_URL='/iiif-demo')
```

The example below will demonstrate how Invenio-IIIF works with a simple image. First, initialize the needed Invenio extensions:

```
>>> from invenio_db import InvenioDB, db
>>> from invenio_files_rest import InvenioFilesREST
>>> ext_db = InvenioDB(app)
>>> ext_files_rest = InvenioFilesREST(app)
>>> app.app_context().push()
>>> db.create_all()
```

Add sample images

To be able to add the files to Invenio, let's create a default location first:

```
>>> import tempfile
>>> tmpdir = tempfile.mkdtemp()
>>> from invenio_files_rest.models import Location
>>> loc = Location(name='local', uri=tmpdir, default=True)
>>> db.session.add(loc)
>>> db.session.commit()
```

And a bucket with the previously created location:

```
>>> from invenio_files_rest.models import Bucket
>>> b1 = Bucket.create(loc)
```

Now, add a few sample images:

```
>>> import os
>>> from invenio_files_rest.models import ObjectVersion
>>> demo_files_path = 'examples/demo_files'
>>> demo_files = (
...     'img.jpg',
...     'img.png')
>>> for f in demo_files:
...     with open(os.path.join(demo_files_path, f), 'rb') as fp:
...         img = ObjectVersion.create(b1, f, stream=fp)
>>> db.session.commit()
```

1.3.3 Serving an image

While Flask-IIIF requires the UUID of the image to retrieve as part of the URL, Invenio needs the bucket id, version id and the key of the file so that it can be retrieved via Invenio-Files-REST. Invenio-IIIF provides an utility to prepare such URLs, e.g. `/v2/<uuid>/<path>`, and convert the `uuid` parameter to a concatenation of `<bucket_id>:<version_id>:<key>`.

Given a previously created image object:

```
>>> img_obj = ObjectVersion.get_versions(bucket=b1,
...                                      key=demo_files[1]).first()
```

you can create the corresponding IIIF URL:

```
>>> from invenio_iiif.utils import ui_iiif_image_url
>>> image_url = ui_iiif_image_url(
...       obj=img_obj, version='v2', region='full', size='full', rotation=0,
...       quality='default', image_format='png')
```

The result will be `/iiif-demov2/<bucket_id>:<version_id>:img.png/full/full/0/default.png`

CHAPTER 2

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 API Docs

IIIF API for Invenio.

class invenio_iiif.ext.**InvenioIIIF**(*app=None*)
Invenio-IIIF extension.

Extension initialization.

init_app(*app*)
Flask application initialization.

init_config(*app*)
Initialize configuration.

class invenio_iiif.ext.**InvenioIIIFAPI**(*app=None*)
Invenio-IIIF extension.

Extension initialization.

init_app(*app*)
Flask application initialization.

2.1.1 Handlers

Handler functions for Flask-IIIF to open image and protect API.

invenio_iiif.handlers.**image_opener**(*key*)
Handler to locate file based on key.

Note: If the file is a PDF then only the first page will be returned as an image.

Parameters **key** – A key encoded in the format "<bucket>:<version>:<object_key>".

Returns A file-like object.

invenio_iiif.handlers.**protect_api**(*uuid=None*, ***kwargs*)
Retrieve object and check permissions.

Retrieve ObjectVersion of image being requested and check permission using the Invenio-Files-REST permission factory.

2.1.2 Previewer

IIIF image previewer.

invenio_iiif.previewer.**blueprint = <flask.blueprints.Blueprint object>**
Blueprint to allow loading of templates.

invenio_iiif.previewer.**can_preview**(*file*)
Determine if the given file can be previewed by its extension.

Parameters **file** – The file to be previewed.

Returns Boolean

invenio_iiif.previewer.**preview**(*file*)
Render appropriate template with embed flag.

Note: Any non .png image is treated as .jpg

Parameters **file** – The file to be previewed.

Returns Template with the preview of the provided file.

2.1.3 Tasks

Background tasks to prepare cache with thumbnails.

2.1.4 Utils

Utilities for IIIF.

invenio_iiif.utils.**iiif_image_key**(*obj*)
Generate a unique IIIF image key, using the images DB location.

Parameters **obj** – File object instance.

Returns Image key 'u'(str)

invenio_iiif.utils.**ui_iiif_image_url**(*obj*, *version='v2'*, *region='full'*, *size='full'*, *rotation=0*, *quality='default'*, *image_format='png'*)
Generate IIIF image URL from the UI application.

Parameters **obj** – File object instance.

Returns URL to retrieve the processed image from.

CHAPTER 3

Additional Notes

Notes on how to contribute, legal information and changes are here for the interested.

3.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

3.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/invenio-iiif/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Invenio-IIIF could always use more documentation, whether as part of the official Invenio-IIIF docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/invenio-iiif/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.1.2 Get Started!

Ready to contribute? Here's how to set up *invenio-iiif* for local development.

1. Fork the *inveniosoftware/invenio-iiif* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/invenio-iiif.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv invenio-iiif
$ cd invenio-iiif/
$ pip install -e .[all]
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass tests:

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

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -s
    -m "component: title without verbs"
    -m "* NEW Adds your new feature."
    -m "* FIX Fixes an existing issue."
    -m "* BETTER Improves and existing feature."
    -m "* Changes something that should not be visible in release notes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

3.1.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.5 and 3.6. Check https://travis-ci.org/inveniosoftware/invenio-iiif/pull_requests and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 1.1.0 (released 2020-03-19)

- Remove Python 2.7 support.
- Bump Flask-IIIF to v0.6.

Version 1.0.0 (released 2019-07-24)

- Initial public release.

3.3 License

3.4 Authors

- Chiara Bigarella
- Esteban J. G. Gabancho

- Harris Tzovanakis
- Lars Holm Nielsen
- Nikos Filippakis
- Sebastian Witowski

Python Module Index

i

Index