
invenio-formatter Documentation

Release 1.1.0a1

CERN

Oct 07, 2020

Contents

1	User's Guide	3
1.1	Installation	3
1.2	Configuration	4
1.3	Usage	4
1.4	Example application	5
2	API Reference	7
2.1	API Docs	7
3	Additional Notes	11
3.1	Contributing	11
3.2	Changes	13
3.3	License	13
3.4	Contributors	14
	Python Module Index	15
	Index	17

Jinja utilities for Invenio.

Further documentation is available on <https://invenio-formatter.readthedocs.io/>

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

1.1 Installation

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

```
$ pip install invenio-formatter
```

If you want to use the badge generation feature you need explicit enable it:

```
$ pip install invenio-formatter[badges]
```

This will install [Pillow](#) and [CairoSVG](#) library. For these libraries to work you must have the following system libraries installed with development headers:

- FreeType
- Cairo
- DejaVu Sans font

1.1.1 Linux

Install the dependencies with your package manager. For Ubuntu or Debian:

```
$ sudo apt-get install libcairo2-dev
```

1.1.2 OS X

Install the dependencies above with Homebrew:

```
$ brew install cairo
```

Download the [DejaVu Sans](<https://dejavu-fonts.github.io>) font and install it on your system.

1.2 Configuration

Configuration for Invenio-Formatter.

```
invenio_formatter.config.FORMATTER_BADGES_ALLOWED_TITLES = ['DOI']
```

List of allowed titles in badges.

```
invenio_formatter.config.FORMATTER_BADGES_MAX_CACHE_AGE = 0
```

The maximum amount of time a badge will be considered fresh.

```
invenio_formatter.config.FORMATTER_BADGES_TITLE_MAPPING = {}
```

Mapping of titles.

1.3 Usage

Jinja utilities for Invenio applications.

1.3.1 Quick start

This section presents an example of the Invenio-Formatter features. In this section we will create a simple page with a time formatter and a shields.io badge.

First, let us create a new template page on which we will render some text, and save it to some location (e.g.: `/home/myuser/templates/index.html`)

```
Today is {{ mydate|to_arrow|format_arrow('YYYY-MM-DD') }}

Your book: {{ badge_svg('isbn', '9780399547331')|safe }}
```

Create a new Flask application and set some configuration for badge generation:

```
>>> from flask import Flask
>>> app = Flask('TestApp', template_folder='/home/myuser/templates')
>>> app.config.update(FORMATTER_BADGES_ALLOWED_TITLES=('isbn', 'ISBN', ))
>>> app.config.update(FORMATTER_BADGES_TITLE_MAPPING={'isbn': 'ISBN', })
```

Note: By default the `template_folder` is the directory `templates` at the root of your flask application.

Load the Invenio-Formatter extension and run the application:

```
>>> from invenio_formatter import InvenioFormatter
>>> ext_fmt = InvenioFormatter(app)
>>> app.run() # doctest: +SKIP
```

You should now be able to access the index page with formatted date and badge examples at <http://localhost:5000>.

Badges endpoints

In addition of generating an image of a shield badge, a badge-rendering endpoint is also available for easy embedding on other websites.

You can modify the template as follows:

```
{% set mydate = mydate|from_isodate -%}
Today is {{ mydate.strftime('%Y-%m-%d') }}

Your book (with badge URL: ):
</img>
```

Your badge will be visible on the page as before, and also directly accessible at <http://localhost:5000/badge/ISBN/9780399547331.svg>.

1.4 Example application

First install Invenio-Formatter by running:

```
$ pip install -e .[all]
$ cd examples
```

Next, start the development server:

```
$ export FLASK_APP=app.py FLASK_DEBUG=1
$ flask run
```

and open the example application in your browser:

```
$ open http://127.0.0.1:5000/
```


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

Jinja utilities for Invenio.

class `invenio_formatter.ext.InvenioFormatter` (*app=None*)

Invenio-Formatter extension.

Extension initialization.

Parameters `app` – The Flask application. (Default: None)

init_app (*app*)

Flask application initialization.

Parameters `app` – The Flask application.

static init_config (*app*)

Initialize configuration.

Note: If CairoSVG is installed then the configuration `FORMATTER_BADGES_ENABLE` is True.

Parameters `app` – The Flask application.

2.1.1 Context preprocessors

Badges context processor.

`invenio_formatter.context_processors.badges.badges_processor()`

Context processor for badges.

`invenio_formatter.context_processors.badges.generate_badge_png` (*title*, *value*,
color='#007ec6')

Generate the badge in PNG format.

`invenio_formatter.context_processors.badges.generate_badge_svg` (*title*, *value*,
color='#007ec6')

Generate the SVG.

Parameters

- **title** – The badge title.
- **value** – The badge content.
- **color** – The badge color. (Default: '#007ec6')

Returns The SVG badge.

`invenio_formatter.context_processors.badges.get_text_length` (**args*)

Measure the size of string rendered with a TTF no-nomospaced fonts.

Parameters **args* – List of strings to be measured.

Returns The length of the strings.

2.1.2 Filters

Datetime Jinja filters.

`invenio_formatter.filters.datetime.format_arrow` (*value*, *format_string*)

Format an arrow datetime object.

Parameters

- **value** – The arrow datetime object.
- **format_string** – The date format string

Returns Returns a string representation of the given arrow datetime object, formatted according to the given format string.

Note: Do not use this filter to format date/times presented to an end user. Instead use `datetimeformat` or `dateformat` from Invenio-I18N.

`invenio_formatter.filters.datetime.from_isodate` (*value*, *strict=False*)

Convert an ISO formatted date into a Date object.

Parameters

- **value** – The ISO formatted date.
- **strict** – If value is None, then if strict is True it returns the Date object of today, otherwise it returns None. (Default: False)

Returns The Date object or None.

`invenio_formatter.filters.datetime.from_isodatettime` (*value*, *strict=False*)

Convert an ISO formatted datetime into a Date object.

Parameters

- **value** – The ISO formatted datetime.

- **strict** – If value is `None`, then if `strict` is `True` it returns the `Date` object of today, otherwise it returns `None`. (Default: `False`)

Returns The `Date` object or `None`.

`invenio_formatter.filters.datetime.to_arrow(value)`

Convert a `Date` object to an arrow datetime object.

HTML sanitisation Jinja filters.

`invenio_formatter.filters.html.sanitize_html(value, tags=None, attributes=None)`

Sanitize HTML.

Parameters

- **tags** – Allowed HTML tags. Configuration set by `Invenio-Config`.
- **attributes** – Allowed HTML attributes. Configuration set by `Invenio-Config`.

Use this function when you need to include unescaped HTML that contain user provided data.

2.1.3 Views

View method for `Invenio-Formatter`.

`invenio_formatter.views.create_badge_blueprint(allowed_types)`

Create the badge blueprint.

Parameters `allowed_types` – A list of allowed types.

Returns A Flask blueprint.

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-formatter/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-Formatter could always use more documentation, whether as part of the official Invenio-Formatter 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-formatter/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-formatter* for local development.

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

```
$ git clone git@github.com:your_name_here/invenio-formatter.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-formatter
$ cd invenio-formatter/
$ 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, 3.3, 3.4 and 3.5. Check https://travis-ci.org/inveniosoftware/invenio-formatter/pull_requests and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 1.1.0a1 (released 2020-05-15)

- Integrates Semantic-UI templates

Version 1.0.3 (released 2020-05-06)

- Deprecated Python versions lower than 3.6.0. Now supporting 3.6.0 and 3.7.0

Version 1.0.2 (released 2019-07-29)

- Adds HTML sanitization Jinja filter.

Version 1.0.1 (released 2018-05-25)

- Fixes issue with badges being cached.

Version 1.0.0 (released 2018-03-23)

- Initial public release.

3.3 License

MIT License

Copyright (C) 2015-2018 CERN.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Note: 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.

3.4 Contributors

- Alexander Ioannidis
- Alizee Pace
- Eamonn Maguire
- Harri Hirvonsalo
- Javier Delgado
- Jiri Kuncar
- Krzysztof Nowak
- Lars Holm Nielsen
- Leonardo Rossi
- Orestis Melkonian
- Sami Hiltunen
- Sebastian Witowski
- Tibor Simko

i

- `invenio_formatter`, 4
- `invenio_formatter.config`, 4
- `invenio_formatter.context_processors.badges`,
7
- `invenio_formatter.ext`, 7
- `invenio_formatter.filters.datetime`, 8
- `invenio_formatter.filters.html`, 9
- `invenio_formatter.views`, 9

B

`badges_processor()` (in module `invenio_formatter.context_processors.badges`), 7

C

`create_badge_blueprint()` (in module `invenio_formatter.views`), 9

F

`format_arrow()` (in module `invenio_formatter.filters.datetime`), 8

`FORMATTER_BADGES_ALLOWED_TITLES` (in module `invenio_formatter.config`), 4

`FORMATTER_BADGES_MAX_CACHE_AGE` (in module `invenio_formatter.config`), 4

`FORMATTER_BADGES_TITLE_MAPPING` (in module `invenio_formatter.config`), 4

`from_isodate()` (in module `invenio_formatter.filters.datetime`), 8

`from_isodatettime()` (in module `invenio_formatter.filters.datetime`), 8

G

`generate_badge_png()` (in module `invenio_formatter.context_processors.badges`), 7

`generate_badge_svg()` (in module `invenio_formatter.context_processors.badges`), 8

`get_text_length()` (in module `invenio_formatter.context_processors.badges`), 8

I

`init_app()` (`invenio_formatter.ext.InvenioFormatter` method), 7

`init_config()` (`invenio_formatter.ext.InvenioFormatter` static method), 7

`invenio_formatter` (module), 4

`invenio_formatter.config` (module), 4

`invenio_formatter.context_processors.badges` (module), 7

`invenio_formatter.ext` (module), 7

`invenio_formatter.filters.datetime` (module), 8

`invenio_formatter.filters.html` (module), 9

`invenio_formatter.views` (module), 9

`InvenioFormatter` (class in `invenio_formatter.ext`), 7

S

`sanitize_html()` (in module `invenio_formatter.filters.html`), 9

T

`to_arrow()` (in module `invenio_formatter.filters.datetime`), 9