invenio-mail Documentation

Release 1.0.2

CERN

Dec 06, 2018

Contents

Invenio-Mail is an integration layer between Invenio and Flask-Mail.

Further documentation is available on https://invenio-mail.readthedocs.io/

CHAPTER 1

User's Guide

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

1.1 Installation

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

```
$ pip install invenio-mail
```

1.2 Configuration

Configuration for Invenio-Mail.

Invenio-Mail is a tiny integration layer between Invenio and Flask-Mail, so please refer to Flask-Mail's list of configuration variables.

1.3 Usage

Invenio mail module.

The Invenio-Mail module is a tiny wrapper around Flask-Mail that provides printing of emails to standard output when the configuration variable `MAIL_SUPPRESS_SEND` is `True`.

Invenio-Mail also takes care of initializing Flask-Mail if not already initialized.

First, initialize the extension:

```
>>> from flask import Flask
>>> from invenio_mail import InvenioMail
>>> app = Flask('myapp')
>>> app.config.update(MAIL_SUPPRESS_SEND=True)
>>> InvenioMail(app)
<invenio_mail.ext.InvenioMail ...>
```

Next, let's send an email:

```
>>> from flask_mail import Message
>>> msg = Message('Hello', sender='from@example.org',
...     recipients=['to@example.com'], body='Hello, World!')
>>> with app.app_context():
...     app.extensions['mail'].send(msg)
Content-Type: text/plain; charset="utf-8"...
```

1.3.1 Template based messages

A simple API lets you create a message from a template, so you just have to give the right arguments to get the full message. Moreover, it can create a complete e-mail with both HTML and text content.

First, you need to instantiate the *TemplatedMessage* class, just like you would do with a standard `flask_mail.Message`:

```
>>> from invenio_mail.api import TemplatedMessage
>>> with app.app_context():
...     msg = TemplatedMessage(
...         template_html='', # path to your template
...         template_body='', # path to your template
...         subject='Hello',
...         sender='from@example.org',
...         recipients=['to@example.com'],
...         ctx={
...             'content': 'Hello, World!',
...             'logo': 'logo.png',
...             'sender': 'Sender',
...             'user': 'User',
...         })
```

You just need to add the templates to use and a `ctx` dictionary, containing the values useful to fill the templates. If you ommit these three arguments, you will have the same result as you would with the standard `flask_mail.Message` class.

Note that you must be in the application context in order to be able to render the templates. Once you have created a message, you can send it the standard way:

```
>>> with app.app_context():
...     app.extensions['mail'].send(msg)
Content-Type: text/plain; charset="utf-8"...
```

1.3.2 Writing extensions

By default you should just depend on Flask-Mail if you are writing an extension which needs email sending functionality:

```
from flask import current_app
from flask_mail import Message

def mystuff():
    msg = Message('Hello', sender='from@example.org',
                  recipients=['to@example.com'], body='Hello, World!')
    current_app.extensions['mail'].send(msg)
```

Remember to add Flask-Mail to your `setup.py` file as well:

```
setup(
    # ...
    install_requires = ['Flask-Mail>=0.9.1',]
    #...
 )
```

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

2.1.1 Extension

Invenio mail module.

class invenio_mail.ext.**InvenioMail**(*app=None*, *stream=None*)

Invenio-Mail extension.

Extension initialization.

Mails are only printed to the stream if MAIL_SUPPRESS_SEND is True.

Parameters

- **app** – Flask application object.
- **stream** – Stream to print emails to. Defaults to sys.stdout.

init_app(*app*)

Flask application initialization.

The initialization will:

- Set default values for the configuration variables.
- Initialise the Flask mail extension.
- Configure the extension to avoid the email sending in case of debug or MAIL_SUPPRESS_SEND config variable set. In this case, the email will be written in the stream configured in the extension.

Parameters **app** – Flask application object.

static init_config(*app*)
Initialize configuration.

Parameters **app** – Flask application object.

invenio_mail.ext.**print_email**(*message*, *app*)
Print mail to stream.

Signal handler for email_dispatched signal. Prints by default the output to the stream specified in the constructor of InvenioMail.

Parameters

- **message** – Message object.
- **app** – Flask application object.

2.1.2 Tasks

(task) invenio_mail.tasks.**send_email**(*data*)
Celery task for sending emails.

Warning: Due to an incompatibility between MessagePack serialization and Message, support for attachments and dates is limited. Consult the tests for details.

2.1.3 Templated Messages

Template based messages.

class invenio_mail.api.**TemplatedMessage**(*template_body=None*, *template_html=None*, *ctx=None*, ***kwargs*)
Siplify creation of templated messages.

Build message body and HTML based on provided templates.

Provided templates can use keyword arguments `body` and `html` respectively.

Parameters

- **template_body** – Path to the text template.
- **template_html** – Path to the html template.
- **ctx** – A mapping containing additional information passed to the template.
- ****kwargs** – Keyword arguments as defined in `flask_mail.Message`.

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

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

```
$ git clone git@github.com:your_name_here/invenio-mail.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-mail
$ cd invenio-mail/
$ 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-mail/pull_requests and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 1.0.2 (released 2018-12-05)

- Fixes issue with passing None context value to the e-mail template

Version 1.0.1 (released 2018-04-12)

- Fixes issue with task running in request context, when only the app context is needed. This causes issues when e.g host header injection protection is turned on.

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

- Alizee Pace
- Dinos Kousidis
- Jacopo Notarstefano
- Javier Delgado
- Jiri Kuncar
- Krzysztof Nowak
- Lars Holm Nielsen
- Rémi Ducceschi
- Sami Hiltunen
- Tibor Simko

Python Module Index

i

Index