## Contents

1 User's Guide  
1.1 Installation  
1.2 Configuration  
1.3 Usage  

2 API Reference  
2.1 API Docs  

3 Additional Notes  
3.1 Contributing  
3.2 Changes  
3.3 License  
3.4 Contributors  

Python Module Index  

Index
WSGI, Celery and CLI applications for Invenio flavours.

Further documentation is available on https://invenio-app.readthedocs.io/
This part of the documentation will show you how to get started in using Invenio-App.

1.1 Installation

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

$ pip install invenio-app

1.2 Configuration

You can modify the default location of the instance folder and/or static folder by setting the environment variables:

- `INVENIO_INSTANCE_PATH` (default: `<sys.prefix>/var/instance/`)
- `INVENIO_STATIC_FOLDER` (default: `<instance-path>/static/`)

Instance specific configuration is loaded from:

- `<instance-path>/invenio.cfg`
- via environment variables prefixed with `INVENIO_` (e.g. `INVENIO_SQLALCHEMY_DATABASE_URI`)

Templates are loaded from:

- `<instance-path>/templates/`

Invenio App configuration.

*Invenio-App* is partially overwriting default configuration of *Limiter* and *Talisman* applications. You can find below more details about which configuration are set.

For more information, please also see *Flask-Limiter* and *Flask-Talisman* websites.
**invenio-app Documentation, Release 1.3.0**

**invenio_app.config.APP_ALLOWED_HOSTS = None**
A list of host/domain names that can be served.

This is a security measure to prevent HTTP Host header attacks, which are possible even under many seemingly-safe web server configurations.

By default all hosts are allowed. Values in this list can be fully qualified names (e.g. ‘www.example.com’). The validation only applies to request.host.

In addition to this configuration variable, you should make sure that your web server does not route requests to the application with an invalid Host header.

**invenio_app.config.APP_DEFAULT_SECURE_HEADERS = {u'content_security_policy': {u'default-src': [u'"self"'], u'object-src': [u'"none"'], ... True, u'strict_transport_security_max_age': 31556926, u'strict_transport_security_preload': False}**
Talisman default Secure Headers configuration.

As default, invenio assumes that HTTPS is enabled. If you are not using SSL, then remember to disable the force_https and session_cookie_secure configuration options related to HTTPS.

Please note that, as default Talisman behaviour, if Flask DEBUG mode is on, then also many security barriers are automatically switched off (e.g. force_https and session_cookie_secure).

---

**Note:** Overwrite Flask-Talisman configuration.

```python
from flask import Flask
from flask_talisman import Talisman

app = Flask(__name__)
app.config.update(
    SECRET_KEY='SECRET_KEY'
)
talisman = Talisman(app)

@app.route('/defenders')
@talisman(frame_options_allow_from='*')
def defenders():
    """Override policies for the specific view.""
    return 'Jessica Jones'
```

**invenio_app.config.APP_ENABLE_SECURE_HEADERS = True**
Enable Secure Headers. (Default: True)

In case you want to disable completely Talisman, you can set to False.

Remember that, for development purpose, setting `DEBUG = True` is already enough to disable any side effects such as force https.

---

**Note:** W3C

**invenio_app.config.APP_HEALTH_BLUEPRINT_ENABLED = True**
Enable the ping (healthcheck) blueprint. (Default: False)

**invenio_app.config.APP_REQUESTID_HEADER = u'X-Request-Id'**
Name of header containing a request id (max length 200 characters).

If set, the request id will be extracted from the header and set on the global Flask request object. The extracted request id can be used by other Invenio modules - e.g. Invenio-Logging could include it in log messages.

The request id can be used to trace requests between systems to make troubleshooting easier.
You can configure Nginx 1.10+ to automatically generate a request id and add it as a header to both the upstream WSGI server and downstream client:

```
add_header X-Request-ID $request_id;
```

Set to `None` to not extract a request id.

```
invenio_app.config.APP_THEME = None
```

Application-wide themes list used for template and assets lookup.

The value is a list of theme strings applied in a fallback fashion in the order they are specified:

```
APP_THEME = ['my-overlay', 'semantic-ui']
```

From the above example, templates and assets with the `my-overlay` prefix will be looked up first, and if not found the `semantic-ui` prefix will be used. If none of the lookups are successful, a non-prefixed lookup is done.

```
invenio_app.config.RATELIMIT_APPLICATION()
```

Global rate limit.

```
invenio_app.config.RATELIMIT_AUTHENTICATED_USER = u'5000 per hour;100 per minute'
```

Rate limit for logged in users.

```
invenio_app.config.RATELIMIT_GUEST_USER = u'1000 per hour;60 per minute'
```

Rate limit for non logged in users.

```
invenio_app.config.RATELIMIT_HEADERS_ENABLED = True
```

Enable rate limit headers. (Default: `True`)

```
invenio_app.config.RATELIMIT_KEY_FUNC = None
```

Define custom key function.

This config is not part of Flask-Limiter.

This function is used to generate a unique key for each visitor to track the number of performed requests. If not defined, the default `key_func` will be used, which will create the key by concatenating the user agent and the IP address of the user.

For more information you can also see here

```
invenio_app.config.RATELIMIT_PER_ENDPOINT = {}
```

Specifically defined Flask rate limits per endpoint.

This config is not part of Flask-Limiter. Use this for endpoints that need to be whitelisted, providing the Flask blueprint path accompanied by a rate limit value.

```
RATELIMIT_PER_ENDPOINT = {
    'zenodo_frontpage.index': '10 per second',
    'security.login': '10 per second'
}
```

```
invenio_app.config.RATELIMIT_STORAGE_URL = u'memory://'
```

Storage backend to store rate-limiting information.

Memory is used by default if no value is provided. For more information regarding the mentioned above configuration values and their available options you can see the Flask-Limiter configuration.

**Note:** Provide your Redis URL if you are rate limiting a multithreaded application.
invenio_app.config.RATELIMIT_STRATEGY = u'moving-window'
The rate limiting strategy to use.

The strategy used here is the most consistant but also expensive one. If you are experiencing performance
issues due to the increased Redis traffic, you can replace it with another one from the following Flask-Limiter
strategies.

1.3 Usage

WSGI, Celery and CLI applications for Invenio flavours.

Invenio-App provides the default Flask, WSGI, Celery and CLI applications which is needed in order to run Invenio.
All which is needed to run your Invenio instance is to provide a base configuration should via the invenio_config.
module entry point.

1.3.1 Setting up your instance

Your base configuration e.g. defines the default language, name of your site as well as the data model you are using.
Usually this base configuration is put in Python module inside a package. For instance like this (assuming your
instance is called mysite)

```python
# mysite/config.py

BABEL_DEFAULT_LANGUAGE = 'en'
BABEL_DEFAULT_TIMEZONE = 'Europe/Zurich'

# ...
```

In order to tell the Invenio-App applications to load your base module, you should set the config module import path in
the invenio_config.module entry point group like the example below:

```python
# setup.py
setup(
    # ...
    entry_points={
        'invenio_config.module': [
            'mysite = mysite.config',
        ],
    }
)
```

The entry point 'mysite = mysite.config' defines an entry point named mysite pointing to the Python
module mysite.config (i.e. the config module above).

After you edited setup.py, you should always remember to install your Python package again, as otherwise the newly
added entry points won’t be picked up.

1.3.2 Applications

CLI

The command-line interface application is named invenio and you use it to e.g. run a development server or
initialize the database:
Celery

In order to start a Celery worker for Invenio, you need to point Celery to the Invenio Celery application (invenio_app.celery) in the following manner:

```
$ celery worker --app invenio_app.celery --loglevel INFO
```

WSGI

Python web applications are usually run by a WSGI server such as Gunicorn, UWSGI or mod_wsgi for Apache. Similar to Celery, you need to point the WSGI servers to the Invenio WSGI application.

Here is e.g. an example with Gunicorn:

```
$ pip install gunicorn
$ gunicorn invenio_app.wsgi
```

Invenio-App provides three different WSGI applications depending on your needs:

- **invenio_app.wsgi**: Combined UI + REST API application with the REST API mounted under /api.
- **invenio_app.wsgi_ui**: UI-only application.
- **invenio_app.wsgi_rest**: REST API-only application.

The individual UI and REST API applications are useful if you want to run the UI and API on different servers or domains (e.g. www.example.org and api.example.org), whereas the combined application is useful if you want to run everything on the same server (e.g. www.example.org and www.example.org/api/).

1.3.3 Deployment

Deploying the applications in a production environment among other things involve:

- daemonizing the applications
- setting instance specific configuration (e.g. database hostname)
- securing your instance

**Daemonizing the applications**

In a production system you would usually need to daemonize the WSGI and Celery application. A full guide on doing this is outside the scope of this documentation, but usually it involves setting up e.g. Supervisord or another process management tool to manage the automatic starting/stopping of the application.

For a full guide on how to deploy Invenio, please see the general Invenio documentation.
Instance configuration

The instance configuration defines e.g. hostnames of your database server and other values which change depending on where you are running the applications (e.g. local development machine vs. production system). In comparison the base configuration is the same no matter where you run the application (e.g. site name).

The instance configuration can be provided either in <instance-path>/invenio.cfg or via environment variables prefixed with INVENIO_, e.g.:

```
$ export INVENIO_SQLALCHEMY_DATABASE_URI = ...
```

The instance path is defined by the environment variable INVENIO_INSTANCE_PATH or if not set defaults to <sys.prefix>/var/instance/ where <sys.prefix> is your Python root prefix (e.g. /usr/)

Securing your instance

The Invenio documentation explains in details how to secure an Invenio instance. It is important to note that if you deploy your Invenio instance with at least one reverse proxy in front of it, then you will have to set the configuration variable WSGI_PROXIES accordingly to correctly handle the X-Forwarded-* headers.

Templates and static files

You can add templates and static files in the following folders respectively:

- <instance-path>/templates/
- <instance-path>/static/

This should only be done for small number of templates/files, as it is usually better to provide them via an installable Python package. Do take care not to overwrite any existing Invenio files.
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 Flask application factories

### 2.1.2 WSGI applications

### 2.1.3 Celery application factory

### 2.1.4 CLI application factory

### 2.1.5 Flask limiter functions

Flask Limiter functions.

```python
invenio_app.limiter.set_rate_limit()
```

Set rates for Flask limiter.

**Returns** a rate limit string with the Flask-Limiter format.

For more information regarding the format you can see [here](#).

The order in which the rate will be evaluated is the following:

1) Initially the endpoint is going to be evaluated against the whitelisted ones. If it has been marked as whitelisted then the custom limit for this endpoint will be the one to be returned.

2) If the endpoint is not whitelisted and the flask-security package is installed, it will evaluate if the user is logged in and if this is the case it will also check if there is an explicitly defined rate limit for them by comparing their ID with the ones present in the `RATELIMIT_PER_USER` mapping. If it is present then the custom rate limit value will be returned, otherwise the one returned will be the `RATELIMIT_AUTHENTICATED_USER`
3) Finally if none of the above is our case then the `RATELIMIT_GUEST_USER` will be the one to be returned.

invenio_app.limiter.useragent_and_ip_limit_key()
Create key for the rate limiting.
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


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-App could always use more documentation, whether as part of the official Invenio-App 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-app/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-app for local development.

1. Fork the inveniosoftware/invenio-app repo on GitHub.

2. Clone your fork locally:

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

3.2 Changes

Version 1.3.0 (released 2020-05-13)
- Adds new template theming via allowing Jinja to load templates from different theme folders via the new configuration variable \texttt{APP\_THEME}.
- Removes the ChoiceLoader used to load templates from the instance folder in favour of using Flask instead. Invenio-App sets the application’s root path to the instance folder, which makes Flask create the same behavior previously achieved with the ChoiceLoader.

Version 1.2.6 (released 2020-05-06)
- Deprecated Python versions lower than 3.6.0. Now supporting 3.6.0 and 3.7.0.

Version 1.2.5 (released 2020-02-26)

Version 1.2.4 (released 2019-11-20)
- Disable ratelimit for celery.

Version 1.2.3 (released 2019-10-10)
- Make \texttt{static\_url\_path} configurable through environment variable.

Version 1.2.2 (released 2019-08-29)
- Unpins Invenio packages versions.

Version 1.2.1 (released 2019-08-21)
- Exempts the “/ping” view from rate limiting.

Version 1.2.0 (released 2019-07-29)
- Fixes issue with instance_path and static_folder being globals. Depends on change in Invenio-Base v1.1.0
- Improves rate limiting function to have limits per guest and per authenticated users.

Version 1.1.1 (released 2019-07-15)
- Fixes a security issue where \texttt{APP\_ALLOWED\_HOSTS} was not always being checked, and thus could allow host header injection attacks.

NOTE: you should never route requests to your application with a wrong host header. The \texttt{APP\_ALLOWED\_HOSTS} exists as an extra protective measure, because it is easy to misconfigure your web server.

The root cause was that Werkzeug’s trusted host feature only works when request.host is being evaluated. This means that for instance when only url_for (part of the routing system) is used, then the host header check is not performed.
Version 1.1.0 (released 2018-12-14)
  • The Flask-DebugToolbar extension is now automatically registered if installed.

Version 1.0.5 (released 2018-12-05)
  • Add health check view
  • Fix response headers assertion in tests

Version 1.0.4 (released 2018-10-11)
  • Fix Content Security Policy headers when set empty in DEBUG mode.

Version 1.0.3 (released 2018-10-08)
  • Fix Content Security Policy headers when running in DEBUG mode.

Version 1.0.2 (released 2018-08-24)
  • Allows use of Flask-DebugToolbar when running in DEBUG mode.

Version 1.0.1 (released 2018-06-29)
  • Pin Flask-Talisman.

Version 1.0.0 (released 2018-03-23)
  • Initial public release.

### 3.3 License

MIT License

Copyright (C) 2017-2018 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
- Dinos Kousidis
• Harris Tzovanakis
• Lars Holm Nielsen
• Leonardo Rossi
• Nikos Filippakis
• Rémi Ducèschi
• Tibor Simko
Python Module Index

i

invenio_app, 6
invenio_app.config, 3
invenio_app.limiter, 9
Index

A
APP_ALLOWED_HOSTS (in module invenio_app.config), 3
APP_DEFAULT_SECURE_HEADERS (in module invenio_app.config), 4
APP_ENABLE_SECURE_HEADERS (in module invenio_app.config), 4
APP_HEALTH_BLUEPRINT_ENABLED (in module invenio_app.config), 4
APP_REQUESTID_HEADER (in module invenio_app.config), 4
APP_THEME (in module invenio_app.config), 5

I
invenio_app (module), 6
invenio_app.config (module), 3
invenio_app.limiter (module), 9

R
RATELIMIT_APPLICATION() (in module invenio_app.config), 5
RATELIMIT_AUTHENTICATED_USER (in module invenio_app.config), 5
RATELIMIT_GUEST_USER (in module invenio_app.config), 5
RATELIMIT_HEADERS_ENABLED (in module invenio_app.config), 5
RATELIMIT_KEY_FUNC (in module invenio_app.config), 5
RATELIMIT_PER_ENDPOINT (in module invenio_app.config), 5
RATELIMIT_STORAGE_URL (in module invenio_app.config), 5
RATELIMIT_STRATEGY (in module invenio_app.config), 5

S
set_rate_limit() (in module invenio_app.limiter), 9

U
useragent_and_ip_limit_key() (in module invenio_app.limiter), 10