invenio-assets Documentation

Release 1.2.5

CERN

Aug 28, 2020

Contents

Media assets management for Invenio.

- Provides the processing and bundling of JavaScript and CSS files.
- Provides CLI for installing and building media assets for Invenio via integration with Webpack and NPM.

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

CHAPTER 1

User's Guide

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

1.1 Installation

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

```
$ pip install invenio-assets
```

Invenio-Assets depends on Flask-WebpackExt and Flask-Collect.

1.2 Configuration

Default values are set for following configuration variables:

- `COLLECT_STATIC_ROOT` - path to folder where static files will be collected to (see Flask-Collect for details). Default: `app.static_folder`.
- `COLLECT_STORAGE` - import path to Flask-Collect storage implementation (see Flask-Collect for details). Default: `'flask_collect.storage.link'` (i.e. symlinking of files from source code into `COLLECT_STATIC_ROOT`).

Note, normally in a production environment you should change `COLLECT_STORAGE` to `flask_collect.storage.file` in order to copy files instead of symlinking them.

For Webpack related configuration please see Flask-WebpackExt

1.3 Usage

Media assets management for Invenio.

Invenio-Assets helps you with integration of webassets, installation of NPM packages, and process of collecting static files.

1.3.1 Initialization

First create a Flask application:

```
>>> from flask import Flask
>>> app = Flask('myapp')
```

Next, initialize your extension:

```
>>> from invenio_assets import InvenioAssets
>>> assets = InvenioAssets(app)
```

During initialization two Flask extensions `flask_webpackext.ext.FlaskWebpackExt` and `flask_collect.Collect` are instantiated and configured.

Bundles specified in the entry point groups called `invenio_assets.webpack` are automatically registered by Invenio-Assets.

1.3.2 Using Flask-WebpackExt

Bundles

The Flask-WebpackExt package provides a class `flask_webpackext.bundle.WebpackBundle` for declaring the needed assets and NPM dependencies of each one of your modules.

```
from flask_webpackext import WebpackBundle
bundle1 = WebpackBundle(
    __name__,
    './modules/module1/static',
    entry={
        'module1-app': './js/module1-app.js',
    },
    dependencies={
        'jquery': '^3.2.1'
    }
)
```

The NPM dependencies defined in the bundles will be used to generate the `package.json` file.

Entry points loading

Invenio-Assets will automatically load bundles defined by the entry point group `invenio_assets.webpack`. Example:

```
# setup.py
setup(
    # ...
    entry_points={
        'invenio_assets.webpack': [
            'mybundle = mypackage.bundles:mybundle',
```

(continues on next page)

(continued from previous page)

```
        ],
        # ...
    }
    # ...
)
```

Command-line interface

We can now build the assets:

```
$ flask webpack buildall
```

The command will copy all files from the src folder to the application instance folder designated for the Webpack project, download the npm packages and run Webpack to build our assets.

Alternatively, we can execute each build step separately with the following flask webpack commands:

- `create` - Copy all sources to the working directory.
- `install` - Run npm install command and download all dependencies.
- `build` - Run npm run build.

Additionally if we have some static files we should collect them:

```
$ flask collect -v
```

CHAPTER 2

Upgrade to Webpack

2.1 Upgrade to Webpack

Note: Invenio-Assets v1.2.0 removed support for AMD/RequireJS and Flask-Assets build system.

In order to upgrade your module from AMD to Webpack, follow the steps below.

2.1.1 Move files to the `assets` folder

- Move files from `static/js` to the `assets/js` folder.
- Move files from `static/scss` to the `assets/scss` folder.
- Keep the rest of the static files in the `static` folder.

2.1.2 Change the way of importing modules

Since Webpack doesn't use `require.js`, you should change the way modules are imported in the JavaScript files. The example below shows how to import the modules:

```
import my-module from 'path/to/my/module'
```

2.1.3 Create a WebpackBundle

The Flask-WebpackExt package provides a class `flask_webpackext.bundle.WebpackBundle` for declaring the needed assets and NPM dependencies of each one of your modules. This class replaces the old bundles.

```python
# webpack.py
from flask_webpackext import WebpackBundle
mybundle = WebpackBundle(
    __name__,
    './modules/module1/static',
    entry={
        'module1-app': './js/module1-app.js',
    },
    dependencies={
        'jquery': '^3.2.1'
    }
)
```

2.1.4 Add a new entry point

You should remove the previous entry point (i. e. `invenio_assets.bundles`) from `setup.py`. Then, you should add the new entry point, `invenio_assets.webpack`, and include the bundle you created in the previous step.

```python
# setup.py
setup(
    # ...
    entry_points={
        'invenio_assets.webpack': [
            'mybundle = mypackage.webpack:mybundle',
        ],
        # ...
     }
     # ...
 )
```

Invenio-Assets will automatically load bundles defined by the entry point group `invenio_assets.webpack`.

2.1.5 Run the webpack commands

In order to build the assets you need to run the following command:

```
$ flask webpack buildall
```

This command will copy all files from the `assets` folder to the application instance folder designated for the Webpack project, download the npm packages and run Webpack to build the assets.

To collect the static files from the `static` folder, you need to run the command below:

```
$ flask collect -v
```

CHAPTER 3

API Reference

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

3.1 API Docs

Media asset management for Invenio.

class invenio_assets.ext.**InvenioAssets**(*app=None*, ***kwargs*)

Invenio asset extension.

Extension initialization.

Parameters

- **app** – An instance of `Flask`.
- ****kwargs** – Keyword arguments are passed to `init_app` method.

init_app(*app*, ***kwargs*)

Initialize application object.

Parameters **app** – An instance of `Flask`.

Changed in version 1.0.0b2: The *entrypoint* has been renamed to *entry_point_group*.

init_config(*app*)

Initialize configuration.

Parameters **app** – An instance of `Flask`.

3.1.1 Webpack

Default Webpack project for Invenio.

class invenio_assets.webpack.**UniqueJinjaManifestEntry**(*name*, *paths*)

Manifest entry which avoids double output of chunks.

Initialize manifest entry.

class invenio_assets.webpack.**UniqueJinjaManifestLoader**(*manifest_cls=<class 'flask_webpackext.manifest.JinjaManifest'>*, *entry_cls=<class 'invenio_assets.webpack.UniqueJinjaManifestEntry'>*)

Factory which uses the Jinja manifest entry.

Initialize manifest loader.

class invenio_assets.webpack.**WebpackThemeBundle**(*import_name*, *folder*, *default=None*, *themes=None*)

Webpack themed bundle.

Initialize webpack bundle.

Parameters

- **import_name** – Name of the module where the WebpackBundle class is instantiated.
- **folder** – Relative path to the assets.
- **default** – The default theme to be used when APP_THEME is not set.
- **themes** – Dictionary where the keys are theme names and the values are the keyword arguments passed to the WebpackBundle class.

3.1.2 Command Line Interface

Click command-line interface for assets and collect.

CHAPTER 4

Additional Notes

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

4.1 Contributing

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

4.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/invenio-assets/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-Assets could always use more documentation, whether as part of the official Invenio-Assets 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-assets/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 :)

4.1.2 Get Started!

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

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

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

4.1.3 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-assets/pull_requests and make sure that the tests pass for all supported Python versions.

4.2 Changes

Version 1.2.5 (released 2020-06-24)

- Updates `package.json` dev dependencies.

Version 1.2.4 (released 2020-06-24)

- Pins less-loader to version 6.1.0. See https://github.com/inveniosoftware/invenio-assets/issues/130.

Version 1.2.3 (released 2020-05-27)

- Fixes an alias issue with jQuery.

Version 1.2.2 (released 2020-05-26)

- Fixes an issue with attribute access and application context errors.

Version 1.2.1 (released 2020-05-25)

- Adds support for adding Webpack aliases to theme bundles.

Version 1.2.0 (released 2020-05-13)

- Uses `webpack-bundle-tracker` for the generating the Webpack manifest.
- Disables the vendor chunk grouping in Webpack config. Since now the manifest exposes entry chunk dependencies, the newly added `UniqueJinjaManifestLoader` renders (only once) each chunk.
- Adds a `WebpackThemeBundle` which uses the `APP_THEME` variable to determine which bundle will be used.
- Removes support for Flask-Assets and Webassets which was deprecated with the release of Invenio v3.0.

Version 1.1.5 (released 2020-04-28)

- Webpack now uses by default in debug/development mode folder-level symlinking
- Enabled source maps for Webpack development builds.
- Patched the `watchpack` library to support symlink watching via using `patch-package`.

Version 1.1.4 (released 2019-02-20)

- Webpack Live-reload plugin.
- Webpack *@templates* alias.
- Webpack fix symlinks issue.

Version 1.1.3 (released 2019-07-29)

- Turn off webpack warnings

Version 1.1.2 (released 2019-02-15)

- Removes NPM warnings.

Version 1.1.1 (released 2018-12-14)

Version 1.1.0 (released 2018-11-06)

- Introduces webpack support.

Version 1.0.0 (released 2018-03-23)

- Initial public release.

4.3 License

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.

4.4 Contributors

- Alexander Ioannidis
- Alizee Pace
- Harris Tzovanakis
- Ivan Masár
- Jacopo Notarstefano
- Javier Delgado
- Jiri Kuncar
- Krzysztof Nowak
- Lars Holm Nielsen
- Leonardo Rossi

- Rémi Ducceschi
- Sami Hiltunen
- Tibor Simko
- Yoan Blanc

Python Module Index

Index