Impulse Documentation

Release 1.0b1

David Seddon

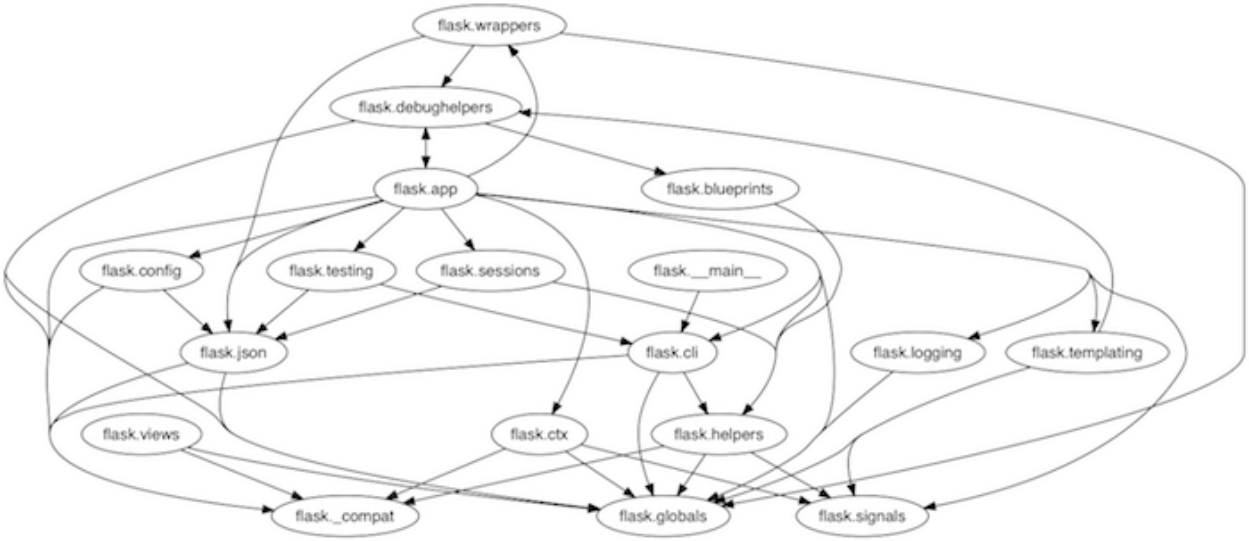
Feb 09, 2019

Contents:

- Free software: BSD license

Impulse is a command line tool for exploring the imports in a Python package.

It can be used to produce dependency graphs such as this:



Warning: This software is currently in beta. It is undergoing active development, and breaking changes may be introduced between versions.

CHAPTER 1

Installation

Install Impulse:

```
pip install impulse
```

Install the Python package you wish to analyse:

```
pip install somepackage
```

CHAPTER 2

Command overview

There is currently only one command, feel free to suggest more by opening an issue or pull request.

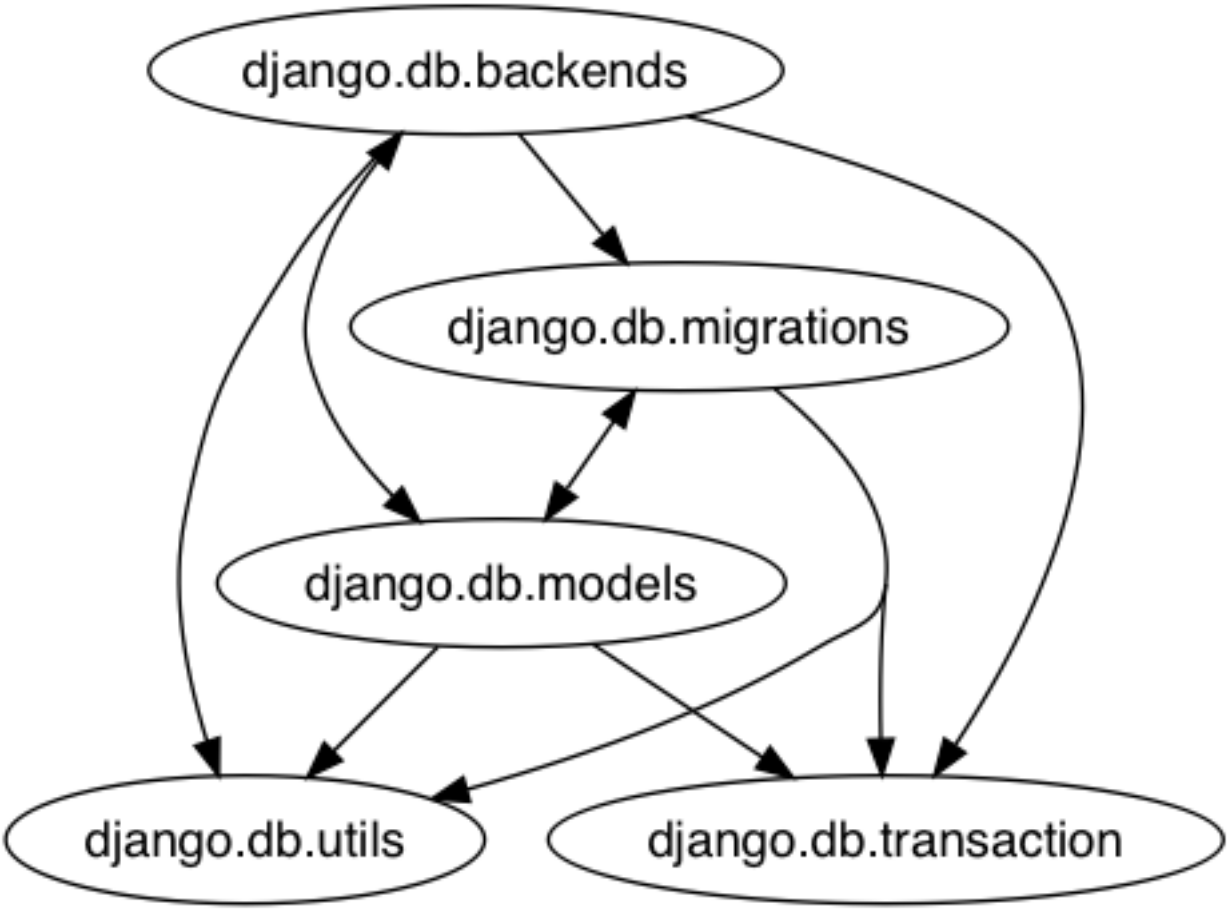
2.1 drawgraph

Draw a graph of the dependencies within any installed Python package or subpackage.

Command:

```
impulse drawgraph django.db
```

Output:



2.1.1 Contributing

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

Bug reports

When 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.

Documentation improvements

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

Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/seddonym/impulse/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 code contributions are welcome :)

Development

To set up *impulse* for local development:

1. Fork impulse (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/impulse.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

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

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

Tips

To run a subset of tests:

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

```
tox -e envname -- pytest -k test_myfeature
```

2.1.2 Authors

- David Seddon - http://seddonym.me

2.1.3 Changelog

1.0a1 (2019-2-1)

- Initial release.

1.0b1 (2019-2-9)

- Added documentation.