setuptools の最近

PyCon JP 2022

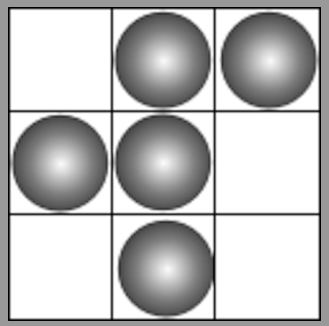
Atsushi Odagiri

2022-10-14

Outline

お前誰よ

- Atsushi Odagiri
- Open Collector
- Python は 1.5 くらいのころから



Open Collector

ケンオール

setuptools と私

- 2013 1.1.5
  - パッケージングの今と未来
- 2014 6.0
  - パッケージングの今 2014 6.0
- 2015 18.3
  - Packaging 最前線
- 2016 27.3.0
  - パッケージングを支える技術
- 2017 36.4.0
  - Python とパッケージングと私
- 2018 40.2.0
  - あなたと Python 今すぐパッケージング
- 2019 41.2.0
  - LT 今日の setuptools
- 2021 53.0.0
  - charity talk パッケージングの呼び声

今日の setuptools

- 2022 65.4.1
  - NEW! setuptools の最近

setuptools とは？

- `distutils` を拡張するもの
- 所謂 `setup.py` を作成するために使う (使われた)

setuptools の最近

- 近代化：パッケージング関連 PEP への追従
- distutils が標準ライブラリから消える予定
- いろいろ剥ぎ取られて純粋なパッケージ作成ツールになってきている

distutils の行く末

- PEP 632 – Deprecate distutils module
- 3.12 で削除 (多分)
  - `https://github.com/python/cpython/tree/main/Lib/distutils`
  - まだある...

distutils はどこにある？

- 標準ライブラリ
- setuptools の中 (`setuptools/_distutils`)
- `SETUPTOOLS_USE_DISTUTILS`
  - `"stdlib"` 標準ライブラリにある distutils を使う
  - `"local"` setuptools 内部で持っている distutils を使う
- `distutils-precedence.pth` の中で切り替え

pth ファイル

- `site-packages` に置いてある `*.pth`
- `./` などで始まる行は `sys.path` に追加
- それ以外の行は python コードとして 実行される
  - 回避不能
  - python2 のころは `-S` オプションで回避できてたかも

-S Disable the import of the module site and the site-dependent manipulations of sys.path that it entails.

Good bye setup.py?

- `setup.py` お前、消えるのか？

setuptools の近代化

- PEP への追従
- setup が持っていた機能は別のツールへ
- 純粋にパッケージ作成のツール

PEP への追従

- PEP 517
  - `setup.py` が不要に！ (でも editable するときは必要)
- PEP 621
  - `setup.cfg` が不要に！ (でも editable するときは必要)
- PEP 660
  - `pyproject.toml` だけで editable 可能に！

setup.py の役割

- パッケージメタデータを書く
- パッケージングする

setup.py でパッケージングする

```
$ python setup.py sdist bdist_wheel
$ python setup.py upload
```

setup.cfg にメタデータを書く

- `setup` 関数の引数として書いていたが `setup.cfg` にも書ける
- `setup.py` の中身は引数なしの `setup()` だけに

PEP517

- sdist から wheel を作る方法の定義
- パッケージングツールは wheel 作成の API を提供する
- パッケージング設定は `pyproject.toml` に書く
  - 設定ファイル増えたよ！？

```
[build-system]
requires = ["setuptools"]
build-backend = "setuptools.build_meta"
```

PEP621

- `pyproject.toml` にメタデータを書くためのスキーマ定義
- v61.0.0 (24 Mar 2022) で導入

パッケージメタデータ（オールドスタイル）

- setup.py

```
setup(
    name="very-useful-tool",
    version="0.1",
    author="Atsushi Odagiri",
    ...
    install_requires=[
        "pyramid",
    ],
    tests_require=[
        "pytest",
    ],
    ...
)
```

パッケージメタデータ（セミオールドスタイル）

- setup.cfg

```
[metadata]
name = very-useful-tool
version = 0.1
author Atsushi Odagiri
...

[options]
install_requires =
  pyramid
tests_require =
  pytest
```

パッケージメタデータ（PEP621）

- pyproject.toml

```
[project]
name="very-useful-tool"
version="0.1"
author="Atsushi Odagiri"
dependencies =
  ["pyramid"]

[project.optional-dependencies]
tests = ["pytest"]
```

setup.py は不要になるか

- PEP 517 対応
  - `setup.py` なしでもパッケージング作業は可能
- PEP 621 対応
  - メタデータの記述が `setup.cfg` から `pyproject.toml` に移動
  - editable インストールするときはまだ必要
  - 空の `setup.cfg` を作るはめに...
- PEP 660 で editable インストールのための API が提案された
  - poetry や flit は対応済
  - setuptools は作業中

setuptools が失ったもの

- パッケージインストーラー (easy_install)
- パッケージのマルチバージョニング (egg ディレクトリ)
- ディストリビューションフォーマット (egg フォーマット)
- パッケージメタデータの拡張 (egg_info)
- パッケージ関連のライブラリ (pkg_resources)
- PyPI へのアップロード (setup.py upload)

インストーラーは easy_install から pip へ

- PEP 453 – Explicit bootstrapping of pip in Python installations
  - python インストールと同時に pip もインストールされるようになった
- easy_install と pip
  - PyPI からダウンロードしてインストール
  - 対象ライブラリが依存するライブラリもインストールする
- `easy_install` の弱点
  - atomic 性の欠如
    - 複数パッケージインストール中にエラーが発生すると中途半端な状態に
- egg ディレクトリへのインストール

インストール先は egg ディレクトリから venv へ

- PEP 405 – Python Virtual Environments
- `site-packages` 以下にディストリビューションごとのディレクトリ (= egg ディレクトリ) を作成してその下に展開
  - 例えば `site-packages/pyramid-1.4-egg/pyramid/`
- pth ファイルを使って `sys.path` に追加
- egg zip safe
  - egg ディレクトリと同じ構造で zip 化した状態
  - zip_safe=True なら egg ファイルのまま site-packages にコピー

egg ディレクトリで multi versioning してたのに！

- `easy_install -m` で multi versioning 対象に
  - pth ファイルから対象の egg ディレクトリを削除
  - このままでは sys.path に追加されなくなる
- `setuptools.Require` で特定バージョンを有効化
- `venv` で分離すればいいよね

バイナリディストリビューションは egg から wheel へ

- PEP 427 – The Wheel Binary Package Format 1.0
- PEP 491 – The Wheel Binary Package Format 1.9
- PEP 425 – Compatibility Tags for Built Distributions
- PEP 513 – A Platform Tag for Portable Linux Built Distributions
- PEP 571 – The manylinux2010 Platform Tag
- PEP 599 – The manylinux2014 Platform Tag
- PEP 600 – Future ‘manylinux’ Platform Tags for Portable Linux Built Distributions
- wheel/egg2wheel

egg_info から dist_info に！

- PEP 241 – Metadata for Python Software Packages
- PEP 314 – Metadata for Python Software Packages v1.1
- PEP 345 – Metadata for Python Software Packages 1.2
- PEP 566 – Metadata for Python Software Packages 2.1
  - description-content-type
- PEP 643 – Metadata for Package Source Distributions
- 2.3
- PEP 685 – Comparison of extra names for optional distribution dependencies
- PEP 508 – Dependency specification for Python Software Packages
- PEP 386 – Changing the version comparison module in Distutils
- PEP 376 – Database of Installed Python Distributions

pkg_resources とその後継

- distlib
- packaging
- pkg_resources の機能が標準ライブラリへ
  - importlib.metadata
  - importlib.resource

インストールされているパッケージ一覧を表示する (freeze) 例

pkg_resources の例

```
import site
import pkg_resources
pkg_resources.find_distributions(
    site.getsitepackages()[0])
```

distlib の例

```
from distlib.database import DistributionPath
dist_path = DistributionPath()
[d.name for d in dist_path.get_distributions()]
```

importlib.metadata の例

```
from importlib import metadata
[d.name for d in metadata.distributions()]
```

PyPI へのアップロードフロー

- wheel 作成は `build` を使う
- PyPI へのアップロードは `twine` を使う

```
$ python setup.py register sdist upload
```

```
$ python -m build .
$ python -m twine upload dist/*.whl
```

まとめ

- distutils が標準ライブラリから消えるので setuptools に同梱
  - ハックがひどい
- setuptools の近代化
  - PEP 517
    - `setup.py` が不要に！ (でも editable するときは必要)
  - PEP 621
    - `setup.cfg` が不要に！ (でも editable するときは必要)
  - PEP 660
    - `pyproject.toml` だけで editable 可能に！
- setuptools が失ってきたもの
  - インストーラー
  - egg
  - pkg_resources
  - などなど

参考 (1)

- PEPs `https://peps.python.org/topic/packaging/`
  - PEP 405 – Python Virtual Environments
  - PEP 420 – Implicit Namespace Packages
  - PEP 425 – Compatibility Tags for Built Distributions
  - PEP 440 – Version Identification and Dependency Specification
  - PEP 453 – Explicit bootstrapping of pip in Python installations
  - PEP 491 – The Wheel Binary Package Format 1.9
  - PEP 513 – A Platform Tag for Portable Linux Built Distributions
  - PEP 517 – A build-system independent format for source trees
  - PEP 571 – The manylinux2010 Platform Tag

参考 (2)

- PEPs
  - PEP 599 – The manylinux2014 Platform Tag
  - PEP 600 – Future ‘manylinux’ Platform Tags for Portable Linux Built Distributions
  - PEP 621 – Storing project metadata in pyproject.toml
  - PEP 632 – Deprecate distutils module
  - PEP 660 – Editable installs for pyproject.toml based builds (wheel based)
- The Python Standard Library
  - pkgutil
  - importlib
  - importlib.metadata
- setuptools documentation, `https://setuptools.pypa.io/en/latest/`
- Python Packaging User Guide, `https://packaging.python.org/en/latest/`
- setuptools - The Peak Developer’s Center, `http://peak.telecommunity.com/DevCenter/setuptools`