pep610

Release 0.4.0

Edgar Ramírez-Mondragón

Mar 02, 2024

CONTENTS

A parser for PEP 610 direct URL metadata.

Release **v0.4.0**.

Python 3.10+

```
from importlib import metadata

import pep610

dist = metadata.distribution("pep610")

match data := pep610.read_from_distribution(dist):
    case pep610.DirData(url, pep610.DirInfo(editable=True)):
        print("Editable installation, a.k.a. in development mode")
    case _:
        print("Not an editable installation")
```

Python 3.8+

```
from importlib import metadata

import pep610

dist = metadata.distribution("pep610")

if (
    (data := pep610.read_from_distribution(dist))
    and isinstance(data, pep610.DirData)
    and data.dir_info.is_editable()
):
    print("Editable installation, a.k.a. in development mode")
else:
    print("Not an editable installation")
```

It can also be used to parse the direct URL download info in pip's Installation Report:

```
import json
import subprocess

import pep610

report = json.loads(
    subprocess.run(
        [
            "pip",
            "install",
            "--quiet",
            "--report",
            "-",
            "--dry-run",
```

(continues on next page)

(continued from previous page)

```
            "git+https://github.com/pypa/packaging@main",
        ],
        capture_output=True,
        text=True,
    ).stdout
)

for package in report["install"]:
    if package["is_direct"]:
        data = pep610.parse(package["download_info"])
        print(data)
```

CHAPTER

ONE

SUPPORTED FORMATS

class pep610.**ArchiveData**(*url*, *archive_info*)

Archive direct URL data.

Parameters

- **url** (*str*) – The archive URL.
- **archive_info** (ArchiveInfo) – Archive information.

class pep610.**DirData**(*url*, *dir_info*)

Local directory direct URL data.

Parameters

- **url** (*str*) – The local directory URL.
- **dir_info** (DirInfo) – Local directory information.

class pep610.**VCSData**(*url*, *vcs_info*)

VCS direct URL data.

Parameters

- **url** (*str*) – The VCS URL.
- **vcs_info** (VCSInfo) – VCS information.

CHAPTER

TWO

OTHER CLASSES

class pep610.**ArchiveInfo**(*hashes=None*, *hash=None*)

Archive information.

See the Archive URLs specification.

Parameters

- **hashes** (*dict[str, str] | None*) – Dictionary mapping a hash name to a hex encoded digest of the file.

  Any hash algorithm available via `hashlib` (specifically any that can be passed to `hashlib.new()` and do not require additional parameters) can be used as a key for the `hashes` dictionary. At least one secure algorithm from `hashlib.algorithms_guaranteed` SHOULD always be included.

- **hash** (*HashData | None*) – The archive hash (deprecated).

has_valid_algorithms()

Has valid archive hashes?

Checks that the `hashes` attribute is not empty and that at least one of the hashes is present in `hashlib.algorithms_guaranteed`.

Returns

Whether the archive has valid hashes.

Return type

bool

```
>>> archive_info = ArchiveInfo(
...     hashes={
...         "sha256":
↪"1dc6b5a470a1bde68946f263f1af1515a2574a150a30d6ce02c6ff742fcc0db9",
...         "md5": "c4e0f0a1e0a5e708c8e3e3c4cbe2e85f",
...     },
... )
>>> archive_info.has_valid_algorithms()
True
```

property all_hashes: dict[str, str]

All archive hashes.

Merges the `hashes` attribute with the legacy `hash` attribute, prioritizing the former.

Returns

All archive hashes.

```
>>> archive_info = ArchiveInfo(
...     hash=HashData(
...         "sha256",
...         "2dc6b5a470a1bde68946f263f1af1515a2574a150a30d6ce02c6ff742fcc0db8",
...     ),
...     hashes={
...         "sha256":
↪"1dc6b5a470a1bde68946f263f1af1515a2574a150a30d6ce02c6ff742fcc0db9",
...         "md5": "c4e0f0a1e0a5e708c8e3e3c4cbe2e85f",
...     },
... )
>>> archive_info.all_hashes
{'sha256': '1dc6b5a470a1bde68946f263f1af1515a2574a150a30d6ce02c6ff742fcc0db9',
↪'md5': 'c4e0f0a1e0a5e708c8e3e3c4cbe2e85f'}
```

class pep610.**DirInfo**(*editable*)

Local directory information.

See the Local Directories specification.

Parameters
editable (*bool | None*) – Whether the distribution is installed in editable mode.

is_editable()

Distribution is editable?

`True` if the distribution was/is to be installed in editable mode, `False` otherwise. If absent, default to `False`

Returns
Whether the distribution is installed in editable mode.

Return type
bool

```
>>> dir_info = DirInfo(editable=True)
>>> dir_info.is_editable()
True
```

```
>>> dir_info = DirInfo(editable=False)
>>> dir_info.is_editable()
False
```

```
>>> dir_info = DirInfo(editable=None)
>>> dir_info.is_editable()
False
```

class pep610.**VCSInfo**(*vcs*, *commit_id*, *requested_revision=None*, *resolved_revision=None*, *resolved_revision_type=None*)

VCS information.

See the VCS URLs specification.

Parameters

- **vcs** (*str*) – The VCS type.
- **commit_id** (*str*) – The exact commit/revision number that was/is to be installed.

- **requested_revision** (*str | None*) – A branch/tag/ref/commit/revision/etc (in a format compatible with the VCS).

CHAPTER

THREE

FUNCTIONS

pep610.**parse**(*data*)

Parse the direct URL data.

Parameters
data (*dict*) – The direct URL data.

Returns
The parsed direct URL data.

Return type
VCSData | ArchiveData | DirData | None

```
>>> parse(
...     {
...         "url": "https://github.com/pypa/packaging",
...         "vcs_info": {
...             "vcs": "git",
...             "requested_revision": "main",
...             "commit_id": "4f42225e91a0be634625c09e84dd29ea82b85e27"
...         }
...     }
... )
VCSData(url='https://github.com/pypa/packaging', vcs_info=VCSInfo(vcs='git', commit_
↪id='4f42225e91a0be634625c09e84dd29ea82b85e27', requested_revision='main',␣
↪resolved_revision=None, resolved_revision_type=None))
```

pep610.**read_from_distribution**(*dist*)

Read the package data for a given package.

Parameters
dist (*importlib_metadata.Distribution*) – The package distribution.

Returns
The parsed PEP 610 file.

Return type
VCSData | *ArchiveData* | *DirData* | None

```
>>> import importlib.metadata
>>> dist = importlib.metadata.distribution("pep610")
>>> read_from_distribution(dist)
DirData(url='file:///home/user/pep610', dir_info=DirInfo(editable=False))
```

pep610.**is_editable**(*distribution_name*)

Wrapper around *read_from_distribution()* to check if a distribution is editable.

Parameters
distribution_name (*str*) – The distribution name.

Returns
Whether the distribution is editable.

Raises
importlib_metadata.PackageNotFoundError – If the distribution is not found.

Return type
bool

```
>>> is_editable("pep610")
False
```

INDEX