pep517

Oct 18, 2021

Contents:

This package provides an API to call the hooks defined in **PEP 517**.

CHAPTER 1

API reference

1.1 Calling the build system

class pep517.**Pep517HookCaller**(*source_dir*, *build_backend*, *backend_path=None*, *runner=None*, *python_executable=None*)

A wrapper around a source directory to be built with a PEP 517 backend.

Parameters

- **source_dir** – The path to the source directory, containing pyproject.toml.
- **build_backend** – The build backend spec, as per PEP 517, from pyproject.toml.
- **backend_path** – The backend path, as per PEP 517, from pyproject.toml.
- **runner** – A callable that invokes the wrapper subprocess.
- **python_executable** – The Python executable used to invoke the backend

The 'runner', if provided, must expect the following:

- cmd: a list of strings representing the command and arguments to execute, as would be passed to e.g. 'subprocess.check_call'.
- cwd: a string representing the working directory that must be used for the subprocess. Corresponds to the provided source_dir.
- extra_environ: a dict mapping environment variable names to values which must be set for the subprocess execution.

get_requires_for_build_sdist(*config_settings=None*)

Identify packages required for building a wheel

Returns a list of dependency specifications, e.g.:

```
["setuptools >= 26"]
```

This does not include requirements specified in pyproject.toml. It returns the result of calling the equivalently named hook in a subprocess.

get_requires_for_build_wheel(*config_settings=None*)
Identify packages required for building a wheel

Returns a list of dependency specifications, e.g.:

```
["wheel >= 0.25", "setuptools"]
```

This does not include requirements specified in pyproject.toml. It returns the result of calling the equivalently named hook in a subprocess.

get_requires_for_build_editable(*config_settings=None*)
Identify packages required for building an editable wheel

Returns a list of dependency specifications, e.g.:

```
["wheel >= 0.25", "setuptools"]
```

This does not include requirements specified in pyproject.toml. It returns the result of calling the equivalently named hook in a subprocess.

prepare_metadata_for_build_wheel(*metadata_directory*, *config_settings=None*, *_allow_fallback=True*)
Prepare a `*.dist-info` folder with metadata for this project.

Returns the name of the newly created folder.

If the build backend defines a hook with this name, it will be called in a subprocess. If not, the backend will be asked to build a wheel, and the dist-info extracted from that (unless _allow_fallback is False).

prepare_metadata_for_build_editable(*metadata_directory*, *config_settings=None*, *_allow_fallback=True*)
Prepare a `*.dist-info` folder with metadata for this project.

Returns the name of the newly created folder.

If the build backend defines a hook with this name, it will be called in a subprocess. If not, the backend will be asked to build an editable wheel, and the dist-info extracted from that (unless _allow_fallback is False).

build_sdist(*sdist_directory*, *config_settings=None*)
Build an sdist from this project.

Returns the name of the newly created file.

This calls the 'build_sdist' backend hook in a subprocess.

build_wheel(*wheel_directory*, *config_settings=None*, *metadata_directory=None*)
Build a wheel from this project.

Returns the name of the newly created file.

In general, this will call the 'build_wheel' hook in the backend. However, if that was previously called by 'prepare_metadata_for_build_wheel', and the same metadata_directory is used, the previously built wheel will be copied to wheel_directory.

build_editable(*wheel_directory*, *config_settings=None*, *metadata_directory=None*)
Build an editable wheel from this project.

Returns the name of the newly created file.

In general, this will call the 'build_editable' hook in the backend. However, if that was previously called by 'prepare_metadata_for_build_editable', and the same metadata_directory is used, the previously built wheel will be copied to wheel_directory.

subprocess_runner(*runner*)
A context manager for temporarily overriding the default subprocess runner.

1.2 Subprocess runners

These functions may be provided when creating *Pep517HookCaller*, or to *Pep517HookCaller.subprocess_runner()*.

pep517.**default_subprocess_runner**(*cmd*, *cwd=None*, *extra_environ=None*)
The default method of calling the wrapper subprocess.

pep517.**quiet_subprocess_runner**(*cmd*, *cwd=None*, *extra_environ=None*)
A method of calling the wrapper subprocess while suppressing output.

1.3 Exceptions

exception pep517.**BackendUnavailable**(*traceback*)
Will be raised if the backend cannot be imported in the hook process.

exception pep517.**BackendInvalid**(*backend_name*, *backend_path*, *message*)
Will be raised if the backend is invalid.

exception pep517.**HookMissing**(*hook_name*)
Will be raised on missing hooks.

exception pep517.**UnsupportedOperation**(*traceback*)
May be raised by build_sdist if the backend indicates that it can't.

CHAPTER 2

Changelog

2.1 0.11.1

- Fix DeprecationWarning in tomli.

2.2 0.11

- Support editable hooks (PEP 660).
- Use the TOML 1.0 compliant `tomli` parser module on Python 3.6 and above.
- Ensure TOML files are always read as UTF-8.
- Switch CI to Github actions.

2.3 0.10

- Avoid shadowing imports such as `colorlog` in the backend, by moving the `_in_process.py` script into a separate subpackage.
- Issue warnings when using the deprecated `pep517.build` and `pep517.check` modules at the command line. See the PyPA build project for a replacement.
- Allow building with flit_core 3.x.
- Prefer the standard library `unittest.mock` to `mock` for tests on Python 3.6 and above.

2.4 0.9.1

- Silence some static analysis warnings.

2.5 0.9

- Deprecated the higher level API which handles creating an environment and installing build dependencies. This was not very complete, and the PyPA build project is designed for this use case.
- New `python_executable` parameter for *`Pep517HookCaller`* to run hooks with a different Python interpreter.
- Fix for locating the script to run in the subprocess in some scenarios.
- Fix example in README to get `build-backend` correctly.
- Created documentation on Read the Docs
- Various minor improvements to testing.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index