entrypoints Documentation

Release 0.4

Thomas Kluyver

Feb 02, 2022

Contents

This package is in maintenance-only mode. New code should use the importlib.metadata module in the Python standard library to find and load entry points.

Entry points are a way for Python packages to advertise objects with some common interface. The most common examples are `console_scripts` entry points, which define shell commands by identifying a Python function to run.

Groups of entry points, such as `console_scripts`, point to objects with similar interfaces. An application might use a group to find its plugins, or multiple groups if it has different kinds of plugins.

The **entrypoints** module contains functions to find and load entry points. You can install it from PyPI with `pip install entrypoints`.

To advertise entry points when distributing a package, see entry_points in the Python Packaging User Guide.

The `pkg_resources` module distributed with `setuptools` provides a way to discover entrypoints as well, but it contains other functionality unrelated to entrypoint discovery, and it does a lot of work at import time. Merely *importing* `pkg_resources` causes it to scan the files of all installed packages. Thus, in environments where a large number of packages are installed, importing `pkg_resources` can be very slow (several seconds).

By contrast, `entrypoints` is focused solely on entrypoint discovery and it is faster. Importing `entrypoints` does not scan anything, and getting a given entrypoint group performs a more focused scan.

When there are multiple versions of the same distribution in different directories on `sys.path`, `entrypoints` follows the rule that the first one wins. In most cases, this follows the logic of imports. Similarly, Entrypoints relies on `pip` to ensure that only one `.dist-info` or `.egg-info` directory exists for each installed package. There is no reliable way to pick which of several *.dist-info* folders accurately relates to the importable modules.

Contents:

CHAPTER 1

entrypoints API

1.1 High-level API

`entrypoints.`**`get_single`**(*group*, *name*, *path=None*)
Find a single entry point.

Returns an *`EntryPoint`* object, or raises *`NoSuchEntryPoint`* if no match is found.

`entrypoints.`**`get_group_named`**(*group*, *path=None*)
Find a group of entry points with unique names.

Returns a dictionary of names to *`EntryPoint`* objects.

`entrypoints.`**`get_group_all`**(*group*, *path=None*)
Find all entry points in a group.

Returns a list of *`EntryPoint`* objects.

These functions will all use `sys.path` by default if you don't specify the *path* parameter. This is normally what you want, so you shouldn't need to pass *path*.

1.2 EntryPoint objects

`class` `entrypoints.`**`EntryPoint`**(*name*, *module_name*, *object_name*, *extras=None*, *distro=None*)

`name`
The name identifying this entry point

`module_name`
The name of an importable module to which it refers

`object_name`
The dotted object name within the module, or *None* if the entry point refers to a module itself.

extras
Extra setuptools features related to this entry point as a list, or *None*

distro
The distribution which advertised this entry point - a *Distribution* instance or None

load()
Load the object to which this entry point refers.

classmethod from_string(*epstr*, *name*, *distro=None*)
Parse an entry point from the syntax in entry_points.txt

Parameters

- **epstr** (*str*) – The entry point string (not including 'name =')
- **name** (*str*) – The name of this entry point
- **distro** (Distribution) – The distribution in which the entry point was found

Return type *EntryPoint*

Raises ***BadEntryPoint*** – if *epstr* can't be parsed as an entry point.

class entrypoints.**Distribution**(*name*, *version*)

name
The name of this distribution

version
The version of this distribution, as a string

1.3 Exceptions

exception entrypoints.**BadEntryPoint**(*epstr*)
Raised when an entry point can't be parsed.

exception entrypoints.**NoSuchEntryPoint**(*group*, *name*)
Raised by *get_single()* when no matching entry point is found.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index