include Documentation

Release 0.2.2

Max Fischer

Aug 02, 2018

Documentation Topics Overview:

1 include package 3

2 Automatic Bootstrapping 9

3 include 11

4 Indices and tables 13

Python Module Index 15

CHAPTER 1

include package

include.**disable**(*identifier*, *children_only=False*)
Disable an include type

Parameters

- **identifier** – module or name of the include type
- **children_only** – disable the include type only for child processes, not the current process

The `identifier` can be specified in multiple ways to disable an include type. See *disable()* for details.

include.**enable**(*identifier*, *exclude_children=False*)
Enable a previously disabled include type

Parameters

- **identifier** – module or name of the include type
- **exclude_children** – disable the include type only for child processes, not the current process

The `identifier` can be specified in multiple ways to disable an include type. See *disable()* for details.

include.**path**(*file_path*)
Include module code from a file identified by its path

Parameters **file_path** (*str*) – path to a file containing module code

Returns the imported module

Return type module

Comparable to `execfile`, but respects the rules and constraints of modules. If invoked again with the same `file_path`, the same module is returned.

```
import include
my_config = include.path('/etc/sysconfig/app_conf.py')
```

include.**source**(*source_code*)
Include module code directly from a string

Parameters **source_code** (*str*) – source code of the module

Returns the imported module

Return type module

Comparable to `exec` in a separate `globals` namespace, but respects the rules and constraints of modules. If invoked again with the same `source_code`, the same module is returned.

```
>>> import include
>>> my_module = include.source(
>>> """
... def foo():
...     return {constant}
... """.format(constant=3))
>>> my_module.foo() == 3
True
```

1.1 Subpackages

1.1.1 include.base package

Base module for import hook modules

Each include type is backed by (at least) one import hook module. This must implement the *install()* to explicitly enable the include type. Importing the import hook module should be free of side-effects, allowing modification before installation.

Notably, imported modules should live in a *separate* packagae than the import hook module. This separate package, preferably *include.mount*, is free to bootstrap the import hook module as needed.

include.base.**IMPORT_PATH = 'include.mount.base'**
package name in which to include imported modules

include.base.**install**()
Enable this type of include

Submodules

include.base.import_hook module

class include.base.import_hook.**BaseIncludeLoader**(*module_prefix*)
Bases: object

Import hook to load Python modules from an arbitrary location

Parameters **module_prefix** (*str*) – prefix for modules to import

Base class for import hooks to non-standard code sources. Implements the general structure for encoded sources: a module source translates to an artificial module path of the form `<module_prefix>.<encoded_name>`. The `module_prefix` identifies the code source type (and import hook) while the `encoded_name` contains all required information to retrieve the code.

For example, a `module_prefix` of `include.type.files` could identify a source file type, and an `encoded_name` of `SLASHtmpSLASHfooDOTpy` point to the path `/tmp/foo.py`. The resulting module would appear as `include.type.files.SLASHtmpSLASHfooDOTpy`.

Note that `module_prefix` must point to a valid package, not a module. It will be actually imported by the regular import machinery, and can be used to bootstrap hooks.

The `encoded_name` is a free form field. The base class provides means to escape invalid and reserved symbols (`/` and `.`), but subclasses are free to use them if it is suitable for them. Hooks should use `encoded_name` to store a URI (or similar) to retrieve source code. As per Python rules, including a dot (`.`) in the `encoded_name` requires the hook to import each portion separately.

find_module(*fullname*, *path=None*)
Find the appropriate loader for module `name`

Parameters

- **fullname** (*str*) – `__name__` of the module to import
- **path** (*str or None*) – `__path__` of the *parent* package already imported

load_module(*name*)
Load and return a module

Always returns the corresponding module. If the module is already loaded, the existing module is returned.

module2uri(*module_name*)
Convert an encoded module name to an unencoded source uri

module_prefix

uri2module(*uri*)
Convert an unencoded source uri to an encoded module name

1.1.2 include.encoded package

Load modules from encoded source code

This include type encodes raw source code as a module name. The resulting module is self-contained: it can be stored, loaded and transferred without the original source code. Only the dependencies of the module (including *include*) must be satisfied.

include.encoded.**install**()

Submodules

include.encoded.import_hook module

class include.encoded.import_hook.**EncodedModuleLoader**(*module_prefix*)
Bases: *include.base.import_hook.BaseIncludeLoader*

Load python modules from their encoded content

This import hook allows storing and using module content as a compressed data blob.

load_module(*name*)
Load and return a module

Always returns the corresponding module. If the module is already loaded, the existing module is returned.

module2uri(*module_name*)
Convert an encoded module name to an unencoded source uri

uri2module(*uri*)
Convert an unencoded source uri to an encoded module name

1.1.3 include.files package

Load modules from files

include.files.**install**()

Submodules

include.files.import_hook module

class include.files.import_hook.**FilePathLoader**(*module_prefix*)
Bases: *include.base.import_hook.BaseIncludeLoader*

Load python file from their path

This import hook allows using encoded paths as module names to load modules directly from their file.

load_module(*name*)
Load and return a module

Always returns the corresponding module. If the module is already loaded, the existing module is returned.

1.1.4 include.mount package

class include.mount.**MountLoader**(*mount_prefix*, *module_prefix*)
Bases: object

find_module(*name*, *path=None*)

is_module(*name*)
Test that *name* is a module name

is_mount(*name*)
Test that *name* is a mount name

load_module(*name*)
Load and return a module

mount2name(*mount*)
Convert a mount name to a module name

name2mount(*name*)
Convert a module name to a mount name

1.2 Submodules

1.2.1 include.inhibit module

include.inhibit.**DISABLED_TYPES = <include.inhibit.DisabledIncludeTypes for '', children ''>**
Default interface to disabled types, see *DisabledIncludeTypes*

exception include.inhibit.**DisabledIncludeError**
Bases: exceptions.ImportError

An import operation failed due to its include type being disabled

class include.inhibit.**DisabledIncludeTypes**
Bases: object

Interface for disabling include types

Meta-container to control disabled include types. The methods *disable()* and *enable()* allow to control which include types can be used.

Disabled types cannot be used to import code, be it explicitly or implicitly via bootstrapping. Once a type is disabled, attempts to import code with it raise *DisabledIncludeError*.

This is provided as a two-level filter: each type can be disabled either for both the current and any child process, or only for child processes. It is not possible to enable an include type for child processes but not the current process. Note that child processes inherit disabled types only on startup.

Note This is a singleton, as it controls interpreter state.

disable(*identifier*, *children_only=False*)
Disable an include type

Parameters

- **identifier** – module or name of the include type
- **children_only** – disable the include type only for child processes, not the current process

The identifier can be specified in multiple ways to disable an include type:

module (include.files or include.mount.files) The base module implementing the include type. These modules have a module.IMPORT_PATH attribute.

implementation path ("include.files") Import path of the module implementing the include type.

mount path ("include.mount.files") Mount path of the module implementing the include type.

short path ("files") Relative path of the module implementing the include type.

enable(*identifier*, *exclude_children=False*)
Enable a previously disabled include type

Parameters

- **identifier** – module or name of the include type
- **exclude_children** – disable the include type only for child processes, not the current process

The identifier can be specified in multiple ways to disable an include type. See *disable()* for details.

environment_key = 'PY_INCLUDE_DISABLE'
Key of the environment storing include types disabled for child processes

class include.inhibit.**DisabledTypeLoader**(*module_prefix*)
Bases: *include.base.import_hook.BaseIncludeLoader*

load_module(*name*)
Load and return a module

If the module is already loaded, the existing module is returned. Otherwise, raises *DisabledIncludeError*.

uri2module(*uri*)
Convert an unencoded source uri to an encoded module name

Always raises *DisabledIncludeError*.

CHAPTER 2

Automatic Bootstrapping

Modules imported by *include* have an automatic bootstrapping mechanism: when imported in another process, the required import hook is automatically loaded. This allows un/pickling and transferring objects, without explicitly preparing *include* in the receiver. The only restriction is that *include* must be importable *before* any module code is run.

2.1 Disabling Bootstrap Hooks

At times, bootstrapping arbitrary include types is not desired. For example, a public web service may desire to run existing, local configuration code files during startup only. As a simple remedy, individual include types can be disabled.

To disable include types, both a code and environment interface are available:

Environment Variable `PY_INCLUDE_DISABLE` A list of include types to disable. Types are indicated by their module name, such as `files` for the `include.files` type. Multiple types may be delimited by commas, with optional whitespace. For example, the list `files, encoded` disables imports from files and encoded source.

Code Interface *include.disable()* and *include.enable()* Functions to disable or enable individual include types. Types may be indicated by a variety of identifiers. See *disable()* for details.

2.2 Bootstrapping Mechanism

Each *include* corresponds to an include type *implementation* and *namespace*. The former provides the actual import machinery, such as hooks and name translation. The later is a namespace module, into which all modules imported by *include* are inserted.

For each include type, a separate namespace is used. The namespace itself ensures that its include type is installed. Thus, re-importing a module in a separate process first loads its namespace and thus its import machinery.

For example, a file based import is implemented in `include.files`, and the default namespace is `include.mount`. When importing a file `foo`, it is inserted as `include.mount.files.foo`. Re-importing `include.mount.files.foo` first imports `include.mount.files`, which in turn installs `include.files`.

CHAPTER 3

include

The *include* library is built to use code from arbitrary sources in your application. In contrast to `eval`, `exec` and other code execution, *include* creates fully featured modules. This makes code viable for pickling, multiprocessing, IPC and more.

Using *include* is straight forward - you only need the top level functions of the module:

```
import include
my_config = include.path('/etc/sysconfig/app_conf.py')
```

All ugly bits are handled by *include* - once a module is imported, it works as normal. Even in other processes, the import machinery is bootstrapped without manual intervention.

For all supported import methods, check out the public inteface of the *include* module.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Documentation built from include 0.2.2 at Aug 02, 2018.

Python Module Index

i

include, 3
include.base, 4
include.base.import_hook, 4
include.encoded, 5
include.encoded.import_hook, 5
include.files, 6
include.files.import_hook, 6
include.inhibit, 7
include.mount, 6

Index

B

BaseIncludeLoader (class in include.base.import_hook), 4

D

disable() (in module include), 3
disable() (include.inhibit.DisabledIncludeTypes method), 7
DISABLED_TYPES (in module include.inhibit), 7
DisabledIncludeError, 7
DisabledIncludeTypes (class in include.inhibit), 7
DisabledTypeLoader (class in include.inhibit), 8

E

enable() (in module include), 3
enable() (include.inhibit.DisabledIncludeTypes method), 7
EncodedModuleLoader (class in include.encoded.import_hook), 5
environment variable
    PY_INCLUDE_DISABLE, 9
environment_key (include.inhibit.DisabledIncludeTypes attribute), 7

F

FilePathLoader (class in include.files.import_hook), 6
find_module() (include.base.import_hook.BaseIncludeLoader method), 5
find_module() (include.mount.MountLoader method), 6

I

IMPORT_PATH (in module include.base), 4
include (module), 3
include.base (module), 4
include.base.import_hook (module), 4
include.encoded (module), 5
include.encoded.import_hook (module), 5
include.files (module), 6
include.files.import_hook (module), 6
include.inhibit (module), 7
include.mount (module), 6
install() (in module include.base), 4
install() (in module include.encoded), 5
install() (in module include.files), 6
is_module() (include.mount.MountLoader method), 6
is_mount() (include.mount.MountLoader method), 6

L

load_module() (include.base.import_hook.BaseIncludeLoader method), 5
load_module() (include.encoded.import_hook.EncodedModuleLoader method), 5
load_module() (include.files.import_hook.FilePathLoader method), 6
load_module() (include.inhibit.DisabledTypeLoader method), 8
load_module() (include.mount.MountLoader method), 6

M

module2uri() (include.base.import_hook.BaseIncludeLoader method), 5
module2uri() (include.encoded.import_hook.EncodedModuleLoader method), 5
module_prefix (include.base.import_hook.BaseIncludeLoader attribute), 5
mount2name() (include.mount.MountLoader method), 6
MountLoader (class in include.mount), 6

N

name2mount() (include.mount.MountLoader method), 6

P

path() (in module include), 3
PY_INCLUDE_DISABLE, 9

S

source() (in module include), 3

U

uri2module() (include.base.import_hook.BaseIncludeLoader method), 5
uri2module() (include.encoded.import_hook.EncodedModuleLoader method), 6
uri2module() (include.inhibit.DisabledTypeLoader method), 8