mork Documentation

Release 0.0.0.dev0

Dan Ryan <dan@danryan.co>

Nov 12, 2018

Contents:

1 mork: A project for installing packages across the virtualenv boundary. 1

1.1 Summary . . . 1

1.1.1 See What's Installed . . . 1

1.1.2 Install A Package . . . 1

1.1.3 Uninstall a Package . . . 1

1.1.4 Display Information about Python . . . 2

1.1.5 Run Commands Inside the Virtualenv . . . 2

2 mork package 3

2.1 Submodules . . . 6

2.1.1 mork.virtualenv module . . . 6

3 Indices and tables 11

Python Module Index 13

CHAPTER 1

mork: A project for installing packages across the virtualenv boundary.

1.1 Summary

Mork_ is a library designed for installing and querying python packages inside virtual environments.

1.1.1 See What's Installed

```
>>> import mork
>>> venv = mork.VirtualEnv.from_project_path('/home/user/git/pipenv')
>>> dists = venv.get_distributions()
>>> [dist for dist in dists][:3]
[wheel 0.31.1 (/home/user/.virtualenvs/pipenv-MfOPs1lW/lib/python3.7/site-packages),␣
↪Werkzeug 0.14.1 (/home/user/.virtualenvs/pipenv-MfOPs1lW/lib/python3.7/site-
↪packages), vistir 0.1.4 (/home/user/.virtualenvs/pipenv-MfOPs1lW/lib/python3.7/site-
↪packages)]
```

1.1.2 Install A Package

```
>>> from requirementslib.models.requirements import Requirement
>>> r = Requirement.from_line("requests")
>>> venv.install(r, editable=False)
```

1.1.3 Uninstall a Package

```
>>> pkg = "pytz"
>>> with venv.uninstall(pkg, auto_confirm=True) as uninstall:
        if uninstall.paths:
            cleaned = pkg
>>> print("Removed package: %s" % cleaned)
```

1.1.4 Display Information about Python

```
>>> venv.python
'/home/user/.virtualenvs/pipenv-MfOPs1lW/bin/python'
>>> venv.python_version
'3.7'
```

1.1.5 Run Commands Inside the Virtualenv

```
>>> cmd = venv.run("env")
>>> [line for line in cmd.out.splitlines() if line.startswith("VIRTUAL_ENV")]
['VIRTUAL_ENV=/user/hawk/.virtualenvs/pipenv-MfOPs1lW']
>>> cmd = venv.run_py(["import os; print(os.environ.get('VIRTUAL_ENV'))"])
Deactivating virtualenv...
>>> cmd.out
'/home/user/.virtualenvs/pipenv-MfOPs1lW\n'
>>> with venv.activated():
        print(os.environ["VIRTUAL_ENV"])
/home/hawk/.virtualenvs/pipenv-MfOPs1lW
```

Read the documentation.

CHAPTER 2

mork package

class mork.**VirtualEnv**(*venv_dir*)

Bases: object

activated(*include_extras=True*, *extra_dists=[]*)

A context manager which activates the virtualenv.

Parameters **extra_dists** (*list*) – Paths added to the context after the virtualenv is activated.

This context manager sets the following environment variables:

- *PYTHONUSERBASE*
- *VIRTUAL_ENV*
- *PYTHONIOENCODING*
- *PYTHONDONTWRITEBYTECODE*

In addition, it activates the virtualenv inline by calling *activate_this.py*.

base_paths

classmethod filter_sources(*requirement*, *sources*)

classmethod from_project_path(*path*)

Utility for finding a virtualenv location based on a project path

get_distributions()

Retrives the distributions installed on the library path of the virtualenv

Returns A set of distributions found on the library path

Return type iterator

get_monkeypatched_pathset()

Returns a monkeypatched *UninstallPathset* for using to uninstall packages from the virtualenv

Returns A patched *UninstallPathset* which enables uninstallation of venv packages

Return type `pip._internal.req.req_uninstall.UninstallPathset`

get_setup_install_args(*pkgname*, *setup_py*, *develop=False*)

Get setup.py install args for installing the supplied package in the virtualenv

Parameters

- **pkgname** (*str*) – The name of the package to install
- **setup_py** (*str*) – The path to the setup file of the package
- **develop** (*bool*) – Whether the package is in development mode

Returns The installation arguments to pass to the interpreter when installing

Return type list

classmethod get_sys_path(*python_path*)

Get the `sys.path` data for a given python executable.

Parameters **python_path** (*str*) – Path to a specific python executable.

Returns The system path information for that python runtime.

Return type list

get_working_set()

Retrieve the working set of installed packages for the virtualenv.

Returns The working set for the virtualenv

Return type `pkg_resources.WorkingSet`

classmethod get_workon_home()

initial_working_set

install(*req*, *editable=False*, *sources=[]*)

Install a package into the virtualenv

Parameters

- **req** (`requirementslib.models.requirement.Requirement`) – A requirement to install
- **editable** (*bool*) – Whether the requirement is editable, defaults to False
- **sources** (*list*) – A list of pip sources to consult, defaults to []

Returns A return code, 0 if successful

Return type int

is_installed(*pkgname*)

Given a package name, returns whether it is installed in the virtual environment

Parameters **pkgname** (*str*) – The name of a package

Returns Whether the supplied package is installed in the environment

Return type bool

libdir

classmethod normalize_path(*path*)

paths

python
Path to the virtualenv python

python_version

pyversion

classmethod resolve_dist(*dist*, *working_set*)
Given a local distribution and a working set, returns all dependencies from the set.

Parameters

- **dist** (`pkg_resources.Distribution`) – A single distribution to find the dependencies of
- **working_set** (`pkg_resources.WorkingSet`) – A working set to search for all packages

Returns A set of distributions which the package depends on, including the package

Return type set(`pkg_resources.Distribution`)

run(*cmd*, *cwd='.'*)
Run a command with `Popen` in the context of the virtualenv

Parameters

- **cmd** (*`str or list`*) – A command to run in the virtual environment
- **cwd** (*`str`*) – The working directory in which to execute the command, defaults to `os.curdir`

Returns A finished command object

Return type `Popen`

run_py(*cmd*, *cwd='.'*)
Run a python command in the virtualenv context.

Parameters

- **cmd** (*`str or list`*) – A command to run in the virtual environment - runs with *python -c*
- **cwd** (*`str`*) – The working directory in which to execute the command, defaults to `os.curdir`

Returns A finished command object

Return type `Popen`

classmethod safe_import(*name*)
Helper utility for reimporting previously imported modules while inside the venv

script_basedir
Path to the virtualenv scripts dir

scripts_dir

setuptools_install(*chdir_to*, *pkg_name*, *setup_py_path=None*, *editable=False*)
Install an sdist or an editable package into the virtualenv

Parameters

- **chdir_to** (*`str`*) – The location to change to
- **setup_py_path** (*`str`*) – The path to the setup.py, if applicable defaults to None

- **editable** (*bool*) – Whether the package is editable, defaults to False

sys_path

The system path inside the virtualenv

Returns The `sys.path` from the virtualenv

Return type list

sys_prefix

The prefix run inside the context of the virtualenv

Returns The python prefix inside the virtualenv

Return type `sys.prefix`

system_paths

uninstall(*pkgname*, **args*, ***kwargs*)

A context manager which allows uninstallation of packages from the virtualenv

Parameters **pkgname** (*str*) – The name of a package to uninstall

```
>>> venv = VirtualEnv("/path/to/venv/root")
>>> with venv.uninstall("pytz", auto_confirm=True, verbose=False) as␣
↪uninstaller:
        cleaned = uninstaller.paths
>>> if cleaned:
        print("uninstalled packages: %s" % cleaned)
```

2.1 Submodules

2.1.1 mork.virtualenv module

class mork.virtualenv.**PatchedUninstaller**

Bases: `object`

class mork.virtualenv.**VirtualEnv**(*venv_dir*)

Bases: `object`

activated(*include_extras=True*, *extra_dists=[]*)

A context manager which activates the virtualenv.

Parameters **extra_dists** (*list*) – Paths added to the context after the virtualenv is activated.

This context manager sets the following environment variables:

- *PYTHONUSERBASE*
- *VIRTUAL_ENV*
- *PYTHONIOENCODING*
- *PYTHONDONTWRITEBYTECODE*

In addition, it activates the virtualenv inline by calling *activate_this.py*.

base_paths

classmethod filter_sources(*requirement*, *sources*)

classmethod from_project_path(*path*)
Utility for finding a virtualenv location based on a project path

get_distributions()
Retrives the distributions installed on the library path of the virtualenv

Returns A set of distributions found on the library path

Return type iterator

get_monkeypatched_pathset()
Returns a monkeypatched *UninstallPathset* for using to uninstall packages from the virtualenv

Returns A patched *UninstallPathset* which enables uninstallation of venv packages

Return type `pip._internal.req.req_uninstall.UninstallPathset`

get_setup_install_args(*pkgname*, *setup_py*, *develop=False*)
Get setup.py install args for installing the supplied package in the virtualenv

Parameters

- **pkgname** (*str*) – The name of the package to install
- **setup_py** (*str*) – The path to the setup file of the package
- **develop** (*bool*) – Whether the package is in development mode

Returns The installation arguments to pass to the interpreter when installing

Return type list

classmethod get_sys_path(*python_path*)
Get the `sys.path` data for a given python executable.

Parameters **python_path** (*str*) – Path to a specific python executable.

Returns The system path information for that python runtime.

Return type list

get_working_set()
Retrieve the working set of installed packages for the virtualenv.

Returns The working set for the virtualenv

Return type `pkg_resources.WorkingSet`

classmethod get_workon_home()

initial_working_set

install(*req*, *editable=False*, *sources=[]*)
Install a package into the virtualenv

Parameters

- **req** (`requirementslib.models.requirement.Requirement`) – A requirement to install
- **editable** (*bool*) – Whether the requirement is editable, defaults to False
- **sources** (*list*) – A list of pip sources to consult, defaults to []

Returns A return code, 0 if successful

Return type int

is_installed(*pkgname*)
Given a package name, returns whether it is installed in the virtual environment

Parameters **pkgname** (*str*) – The name of a package

Returns Whether the supplied package is installed in the environment

Return type bool

libdir

classmethod normalize_path(*path*)

paths

python
Path to the virtualenv python

python_version

pyversion

classmethod resolve_dist(*dist*, *working_set*)
Given a local distribution and a working set, returns all dependencies from the set.

Parameters

- **dist** (pkg_resources.Distribution) – A single distribution to find the dependencies of
- **working_set** (pkg_resources.WorkingSet) – A working set to search for all packages

Returns A set of distributions which the package depends on, including the package

Return type set(pkg_resources.Distribution)

run(*cmd*, *cwd='.'*)
Run a command with Popen in the context of the virtualenv

Parameters

- **cmd** (*str or list*) – A command to run in the virtual environment
- **cwd** (*str*) – The working directory in which to execute the command, defaults to os.curdir

Returns A finished command object

Return type Popen

run_py(*cmd*, *cwd='.'*)
Run a python command in the virtualenv context.

Parameters

- **cmd** (*str or list*) – A command to run in the virtual environment - runs with *python -c*
- **cwd** (*str*) – The working directory in which to execute the command, defaults to os.curdir

Returns A finished command object

Return type Popen

classmethod safe_import(*name*)
Helper utility for reimporting previously imported modules while inside the venv

script_basedir
Path to the virtualenv scripts dir

scripts_dir

setuptools_install (*chdir_to*, *pkg_name*, *setup_py_path=None*, *editable=False*)
Install an sdist or an editable package into the virtualenv

Parameters

- **chdir_to** (*str*) – The location to change to
- **setup_py_path** (*str*) – The path to the setup.py, if applicable defaults to None
- **editable** (*bool*) – Whether the package is editable, defaults to False

sys_path
The system path inside the virtualenv

Returns The `sys.path` from the virtualenv

Return type list

sys_prefix
The prefix run inside the context of the virtualenv

Returns The python prefix inside the virtualenv

Return type `sys.prefix`

system_paths

uninstall (*pkgname*, **args*, ***kwargs*)
A context manager which allows uninstallation of packages from the virtualenv

Parameters **pkgname** (*str*) – The name of a package to uninstall

```
>>> venv = VirtualEnv("/path/to/venv/root")
>>> with venv.uninstall("pytz", auto_confirm=True, verbose=False) as␣
↪uninstaller:
        cleaned = uninstaller.paths
>>> if cleaned:
        print("uninstalled packages: %s" % cleaned)
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

mork, 3
mork.virtualenv, 6

Index

A

activated() (mork.VirtualEnv method), 3
activated() (mork.virtualenv.VirtualEnv method), 6

B

base_paths (mork.VirtualEnv attribute), 3
base_paths (mork.virtualenv.VirtualEnv attribute), 6

F

filter_sources() (mork.VirtualEnv class method), 3
filter_sources() (mork.virtualenv.VirtualEnv class method), 6
from_project_path() (mork.VirtualEnv class method), 3
from_project_path() (mork.virtualenv.VirtualEnv class method), 6

G

get_distributions() (mork.VirtualEnv method), 3
get_distributions() (mork.virtualenv.VirtualEnv method), 7
get_monkeypatched_pathset() (mork.VirtualEnv method), 3
get_monkeypatched_pathset() (mork.virtualenv.VirtualEnv method), 7
get_setup_install_args() (mork.VirtualEnv method), 4
get_setup_install_args() (mork.virtualenv.VirtualEnv method), 7
get_sys_path() (mork.VirtualEnv class method), 4
get_sys_path() (mork.virtualenv.VirtualEnv class method), 7
get_working_set() (mork.VirtualEnv method), 4
get_working_set() (mork.virtualenv.VirtualEnv method), 7
get_workon_home() (mork.VirtualEnv class method), 4
get_workon_home() (mork.virtualenv.VirtualEnv class method), 7

I

initial_working_set (mork.VirtualEnv attribute), 4
initial_working_set (mork.virtualenv.VirtualEnv attribute), 7
install() (mork.VirtualEnv method), 4
install() (mork.virtualenv.VirtualEnv method), 7
is_installed() (mork.VirtualEnv method), 4
is_installed() (mork.virtualenv.VirtualEnv method), 7

L

libdir (mork.VirtualEnv attribute), 4
libdir (mork.virtualenv.VirtualEnv attribute), 8

M

mork (module), 3
mork.virtualenv (module), 6

N

normalize_path() (mork.VirtualEnv class method), 4
normalize_path() (mork.virtualenv.VirtualEnv class method), 8

P

PatchedUninstaller (class in mork.virtualenv), 6
paths (mork.VirtualEnv attribute), 4
paths (mork.virtualenv.VirtualEnv attribute), 8
python (mork.VirtualEnv attribute), 4
python (mork.virtualenv.VirtualEnv attribute), 8
python_version (mork.VirtualEnv attribute), 5
python_version (mork.virtualenv.VirtualEnv attribute), 8
pyversion (mork.VirtualEnv attribute), 5
pyversion (mork.virtualenv.VirtualEnv attribute), 8

R

resolve_dist() (mork.VirtualEnv class method), 5
resolve_dist() (mork.virtualenv.VirtualEnv class method), 8
run() (mork.VirtualEnv method), 5
run() (mork.virtualenv.VirtualEnv method), 8
run_py() (mork.VirtualEnv method), 5
run_py() (mork.virtualenv.VirtualEnv method), 8

S

safe_import() (mork.VirtualEnv class method), 5
safe_import() (mork.virtualenv.VirtualEnv class method), 8
script_basedir (mork.VirtualEnv attribute), 5
script_basedir (mork.virtualenv.VirtualEnv attribute), 8
scripts_dir (mork.VirtualEnv attribute), 5
scripts_dir (mork.virtualenv.VirtualEnv attribute), 9
setuptools_install() (mork.VirtualEnv method), 5
setuptools_install() (mork.virtualenv.VirtualEnv method), 9
sys_path (mork.VirtualEnv attribute), 6
sys_path (mork.virtualenv.VirtualEnv attribute), 9
sys_prefix (mork.VirtualEnv attribute), 6
sys_prefix (mork.virtualenv.VirtualEnv attribute), 9
system_paths (mork.VirtualEnv attribute), 6
system_paths (mork.virtualenv.VirtualEnv attribute), 9

U

uninstall() (mork.VirtualEnv method), 6
uninstall() (mork.virtualenv.VirtualEnv method), 9

V

VirtualEnv (class in mork), 3
VirtualEnv (class in mork.virtualenv), 6