
dk-tasklib Documentation

Release 0.3.0

December 21, 2018

1	dktasklib package	1
1.1	Subpackages	1
1.2	Submodules	2
1.3	dktasklib.clean module	2
1.4	dktasklib.commands module	2
1.5	dktasklib.concat module	2
1.6	dktasklib.docs module	2
1.7	dktasklib.environment module	2
1.8	dktasklib.executables module	2
1.9	dktasklib.help module	3
1.10	dktasklib.jstools module	3
1.11	dktasklib.lessc module	3
1.12	dktasklib.manage module	3
1.13	dktasklib.npm module	3
1.14	dktasklib.pset module	4
1.15	dktasklib.publish module	4
1.16	dktasklib.rule module	4
1.17	dktasklib.runners module	5
1.18	dktasklib.upversion module	5
1.19	dktasklib.urlinliner module	5
1.20	dktasklib.utils module	5
1.21	dktasklib.version module	6
1.22	dktasklib.watch module	6
1.23	dktasklib.wintask module	7
1.24	Module contents	8
2	Developing dktasklib	9
2.1	Uploading to PyPI	9
2.2	Running tests	9
2.3	Building documentation	10
3	Indices and tables	11
	Python Module Index	13

dktasklib package

Subpackages

dktasklib.entry_points package

Submodules

dktasklib.entry_points.dktasklibcmd module

Commands installed by setup.py

```
dktasklib.entry_points.dktasklibcmd.add_django_to_docs_conf()
```

```
dktasklib.entry_points.dktasklibcmd.create_docs_cmd(args)
```

```
dktasklib.entry_points.dktasklibcmd.install_cmd(args)
```

Install a basic task.py to the current directory.

```
dktasklib.entry_points.dktasklibcmd.main(args=None)
```

dktasklib.entry_points.taskbase module

Module contents

dktasklib.package package

Submodules

dktasklib.package.package_interface module

```
class dktasklib.package.package_interface.Package(ctx=None)
```

Bases: dkpkg.directory.Package

```
config()
```

```
get(key, default=None)
```

```
overridables = set(['source_less', 'tests', 'build_docs', 'django_templates', 'package_name', 'docs', 'build_coverage',
```

```
overrides (**res)
```

```
dktasklib.package.package_interface.pfind(path, *fnames)
```

Module contents

Submodules

dktasklib.clean module

dktasklib.commands module

```
class dktasklib.commands.Command(name, argspec='', requirements=(), policy={'list_join': ' ',  
                                'negative_bool': 'omit'}, **optdefs)  
    Bases: object
```

dktasklib.concat module

```
dktasklib.concat.chomp(s)  
    Remove line terminator if it exists.  
  
dktasklib.concat.concat(ctx, dest, *sources, **kw)  
  
dktasklib.concat.copy(ctx, source, dest, force=False)  
    Copy source to dest, which can be a file or directory.  
  
dktasklib.concat.fix_line_endings(fname, eol='\n')  
    Change all line endings to eol.  
  
dktasklib.concat.line_endings(fname)  
    Return all line endings in the file.
```

dktasklib.docs module

dktasklib.environment module

Global state.

```
class dktasklib.environment.Environment  
    Bases: object
```

dktasklib.executables module

```
class dktasklib.executables.Executables  
    Bases: object  
  
    Class for finding executables on the host system.  
  
    ctx  
  
    find(name, requires=(), install_txt='')  
        Find the executable named name on the PATH.  
  
        Parameters
```

- **name** (*str*) – name of executable to find.
- **requires** (*List[str]*) – list of executables to find first.
- **install_txt** (*str*) – instructions for how to install the executable if it is not found.

find_babel()

find_babili()

find_browserify()

find_nodejs()

Find node.

find_npm()

Find the node package manager (npm).

find_twine()

find_uglify()

find_wheel()

require (*dependencies)

Ensure that all dependencies are available. You should not need to call this yourself, use the *requires*() decorator instead.

exception `dktasklib.executables.MissingCommand`

Bases: `exceptions.Exception`

Exception thrown when a command (executable) is not found.

`dktasklib.executables.exe = <dktasklib.executables.Executables object>`

public interface to the *Executables* class

`dktasklib.executables.requires` (*deps)

Decorator to declare global dependencies/requirements.

Usage (@task must be last):

```
@requires('nodejs', 'npm', 'lessc')
@task
def mytask(..)
```

dktasklib.help module

dktasklib.jstools module

dktasklib.lessc module

dktasklib.manage module

dktasklib.npm module

`dktasklib.npm.global_package` (*pkgname*)

Check if an npm package is installed globally.

```
dktasklib.npm.npm(cmdline)
```

dktasklib.pset module

```
class dktasklib.pset.pset(items=(), **attrs)
```

Bases: `dict`

Property Set class. A property set is an object where values are attached to attributes, but can still be iterated over as key/value pairs. The order of assignment is maintained during iteration. Only one value allowed per key.

```
>>> x = pset()
>>> x.a = 42
>>> x.b = 'foo'
>>> x.a = 314
>>> x
pset(a=314, b='foo')
```

`__add__(other)`

self + other

`__eq__(other)`

Equal iff they have the same set of keys, and the values for each key is equal. Key order is not considered for equality.

`__radd__(other)`

other + self

`items()`

`keys()`

`remove(key)`

Remove key from client vars.

`values()`

dktasklib.publish module

dktasklib.rule module

```
class dktasklib.rule.BuildRule(*args, **kwargs)
```

Bases: `object`

`after = []`

`needs_to_run()`

`requires = []`

`run(ctx)`

`topsort(tasklist)`

Topological sort

dktasklib.runners module

class dktasklib.runners.**Result**

Bases: `str`

cmd = None

returncode = None

dktasklib.runners.**command** (*executable, dryrun=False*)

dktasklib.runners.**run** (*cmdline, throw=False*)

dktasklib.upversion module

Update a package's version number.

class dktasklib.upversion.**UpdateTemplateVersion** (**args, **kwargs*)

Bases: `dktasklib.rule.BuildRule`

dktasklib.upversion.**files_with_version_numbers** ()

dktasklib.urlinliner module

dktasklib.urlinliner.**inline_data** (*data, type='image/png', name=''*)

Inline (encode) the data.

dktasklib.urlinliner.**inline_file** (*fname*)

Inline from a file source named fname.

dktasklib.urlinliner.**inline_url** (*uri*)

Fetch uri and inline.

dktasklib.utils module

dktasklib.utils.**cd** (**args, **kws*)

Context manager to change directory.

Usage:

```
with cd('foo/bar'):
    # current directory is now foo/bar
    # current directory restored.
```

dktasklib.utils.**dest_is_newer_than_source** (*src, dst*)

Check if destination is newer than source.

Usage:

```
if not force and dest_is_newer_than_source(source, dest):
    print 'babel:', dest, 'is up-to-date.'
    return dest
```

dktasklib.utils.**env** (**args, **kws*)

Context amanger to temporarily override environment variables.

`dktasklib.utils.filename(fname)`

Return only the file name (removes the path)

`dktasklib.utils.find_pymodule(dotted_name)`

Find the directory of a python module, without importing it.

`dktasklib.utils.fmt(s, ctx)`

Use the mapping *ctx* as a formatter for the {new.style} formatting string *s*.

`dktasklib.utils.message(*args, **kws)`

`dktasklib.utils.switch_extension(fname, ext='', old_ext=None)`

Switch file extension on *fname* to *ext*. Returns the resulting file name.

Usage:

```
switch_extension('a/b/c/d.less', '.css')
```

dktasklib.version module

`dktasklib.version.add_version(ctx, source, outputdir=None, kind='pkg', force=False)`

Copy source with version number to *outputdir*.

The version type is specified by the *kind* parameter and can be either “pkg” (package version), “svn” (current subversion revision number), or “hash” (the md5 hash of the file’s contents).

Returns (str) output file name

`dktasklib.version.copy_to_version(ctx, source, outputdir=None, kind='pkg', force=False)`

Copy source with version number to *outputdir*.

The version type is specified by the *kind* parameter and can be either “pkg” (package version), “svn” (current subversion revision number), or “hash” (the md5 hash of the file’s contents).

Returns (str) output file name

`dktasklib.version.get_version(ctx, fname, kind='pkg')`

Return the version number for *fname*.

`dktasklib.version.min_name(fname, min='.min')`

Adds a *.min* extension before the last file extension.

`dktasklib.version.version_name(fname)`

Returns a template string containing {*version*} in the correct place.

`dktasklib.version.versioned_name(fname)`

Returns a template string containing {*version*} in the correct place.

dktasklib.watch module

Usage:

```
@task
def watch(ctx):
    watcher = Watcher(ctx)
    watcher.watch_file(
        name='{pkg.source}/less/{pkg.name}.less',
        action=lambda e: build(ctx, less=True)
```

```

    )
    watcher.watch_directory(
        path='{pkg.source}/js', ext='.jsx',
        action=lambda e: build(ctx, js=True)
    )
    watcher.watch_directory(
        path='{pkg.docs}', ext='.rst',
        action=lambda e: build(ctx, docs=True)
    )
    watcher.start()

ns = Collection(..., watch, ...)
ns.configure({
    'pkg': Package()
})

```

```
class dktasklib.watch.DirectoryModified(ctx, path, ext, action)
```

```
    Bases: watchdog.events.FileSystemEventHandler
```

```
    on_modified(event)
```

```
class dktasklib.watch.FileModified(ctx, fname, action)
```

```
    Bases: watchdog.events.FileSystemEventHandler
```

```
    on_modified(event)
```

```
class dktasklib.watch.Watcher(ctx)
```

```
    Bases: object
```

```
    start()
```

```
    watch_directory(path, ext, action)
```

```
    watch_file(name, action)
```

dktasklib.wintask module

```
dktasklib.wintask.task(*args, **kwargs)
```

Marks wrapped callable object as a valid Invoke task.

May be called without any parentheses if no extra options need to be specified. Otherwise, the following keyword arguments are allowed in the parenthese'd form:

- name:** Default name to use when binding to a *Collection*. Useful for avoiding Python namespace issues (i.e. when the desired CLI level name can't or shouldn't be used as the Python level name.)
- aliases:** Specify one or more aliases for this task, allowing it to be invoked as multiple different names. For example, a task named `mytask` with a simple `@task` wrapper may only be invoked as `"mytask"`. Changing the decorator to be `@task(aliases=['myothertask'])` allows invocation as `"mytask"` or `"myothertask"`.
- positional:** Iterable overriding the parser's automatic "args with no default value are considered positional" behavior. If a list of arg names, no args besides those named in this iterable will be considered positional. (This means that an empty list will force all arguments to be given as explicit flags.)
- optional:** Iterable of argument names, declaring those args to have optional values. Such arguments may be given as value-taking options (e.g. `--my-arg=myvalue`, wherein the task is given `"myvalue"`) or as Boolean flags (`--my-arg`, resulting in `True`).
- iterable:** Iterable of argument names, declaring them to build iterable values.

- `incrementable`: Iterable of argument names, declaring them to increment their values.
- `default`: Boolean option specifying whether this task should be its collection's default task (i.e. called if the collection's own name is given.)
- `auto_shortflags`: Whether or not to automatically create short flags from task options; defaults to `True`.
- `help`: Dict mapping argument names to their help strings. Will be displayed in `--help` output.
- `pre, post`: Lists of task objects to execute prior to, or after, the wrapped task whenever it is executed.
- `autoprint`: Boolean determining whether to automatically print this task's return value to standard output when invoked directly via the CLI. Defaults to `False`.
- `klass`: Class to instantiate/return. Defaults to `.Task`.

If any non-keyword arguments are given, they are taken as the value of the `pre` kwarg for convenience's sake. (It is an error to give both `*args` and `pre` at the same time.)

New in version 1.0.

Changed in version 1.1: Added the `klass` keyword argument.

Module contents

Developing dktasklib

Uploading to PyPi

- only source distribution:

```
python setup.py sdist upload
```

- source and windows installer:

```
python setup.py sdist bdist_wininst upload
```

- source, windows, and wheel installer:

```
python setup.py sdist bdist_wininst bdist_wheel upload
```

- create a documentation bundle to upload to PyPi:

```
python setup.py build_sphinx  
python setup.py upload_docs
```

Note: if you're using this as a template for new projects, remember to *python setup.py register <projectname>* before you upload to PyPi.

Running tests

One of:

```
python setup.py test  
py.test dktasklib
```

with coverage:

```
py.test --cov=dktasklib .
```

Building documentation

```
python setup.py build_sphinx
```

Indices and tables

- `genindex`
- `modindex`
- `search`

d

- `dktasklib`, 8
- `dktasklib.clean`, 2
- `dktasklib.commands`, 2
- `dktasklib.concat`, 2
- `dktasklib.docs`, 2
- `dktasklib.entry_points`, 1
- `dktasklib.entry_points.dktasklibcmd`, 1
- `dktasklib.environment`, 2
- `dktasklib.executables`, 2
- `dktasklib.help`, 3
- `dktasklib.lessc`, 3
- `dktasklib.manage`, 3
- `dktasklib.npm`, 3
- `dktasklib.package`, 2
- `dktasklib.package.package_interface`, 1
- `dktasklib.pset`, 4
- `dktasklib.publish`, 4
- `dktasklib.rule`, 4
- `dktasklib.runners`, 5
- `dktasklib.upversion`, 5
- `dktasklib.urlinliner`, 5
- `dktasklib.utils`, 5
- `dktasklib.version`, 6
- `dktasklib.watch`, 6
- `dktasklib.wintask`, 7

Symbols

`__add__()` (dktasklib.pset.pset method), 4
`__eq__()` (dktasklib.pset.pset method), 4
`__radd__()` (dktasklib.pset.pset method), 4

A

`add_django_to_docs_conf()` (in module `tasklib.entry_points.dktasklibcmd`), 1
`add_version()` (in module `dktasklib.version`), 6
`after` (dktasklib.rule.BuildRule attribute), 4

B

`BuildRule` (class in `dktasklib.rule`), 4

C

`cd()` (in module `dktasklib.utils`), 5
`chomp()` (in module `dktasklib.concat`), 2
`cmd` (dktasklib.runners.Result attribute), 5
`Command` (class in `dktasklib.commands`), 2
`command()` (in module `dktasklib.runners`), 5
`concat()` (in module `dktasklib.concat`), 2
`config()` (dktasklib.package.package_interface.Package method), 1
`copy()` (in module `dktasklib.concat`), 2
`copy_to_version()` (in module `dktasklib.version`), 6
`create_docs_cmd()` (in module `tasklib.entry_points.dktasklibcmd`), 1
`ctx` (dktasklib.executables.Executables attribute), 2

D

`dest_is_newer_than_source()` (in module `dktasklib.utils`), 5
`DirectoryModified` (class in `dktasklib.watch`), 7
`dktasklib` (module), 8
`dktasklib.clean` (module), 2
`dktasklib.commands` (module), 2
`dktasklib.concat` (module), 2
`dktasklib.docs` (module), 2
`dktasklib.entry_points` (module), 1
`dktasklib.entry_points.dktasklibcmd` (module), 1

`dktasklib.environment` (module), 2
`dktasklib.executables` (module), 2
`dktasklib.help` (module), 3
`dktasklib.lessc` (module), 3
`dktasklib.manage` (module), 3
`dktasklib.npm` (module), 3
`dktasklib.package` (module), 2
`dktasklib.package.package_interface` (module), 1
`dktasklib.pset` (module), 4
`dktasklib.publish` (module), 4
`dktasklib.rule` (module), 4
`dktasklib.runners` (module), 5
`dktasklib.upversion` (module), 5
`dktasklib.urlinliner` (module), 5
`dktasklib.utils` (module), 5
`dktasklib.version` (module), 6
`dktasklib.watch` (module), 6
`dktasklib.wintask` (module), 7

E

`env()` (in module `dktasklib.utils`), 5
`Environment` (class in `dktasklib.environment`), 2
`environment variable`
 `PATH`, 2
`exe` (in module `dktasklib.executables`), 3
`Executables` (class in `dktasklib.executables`), 2

F

`FileModified` (class in `dktasklib.watch`), 7
`filename()` (in module `dktasklib.utils`), 5
`files_with_version_numbers()` (in module `dktasklib.upversion`), 5
`find()` (dktasklib.executables.Executables method), 2
`find_babel()` (dktasklib.executables.Executables method), 3
`find_babili()` (dktasklib.executables.Executables method), 3
`find_browserify()` (dktasklib.executables.Executables method), 3
`find_nodejs()` (dktasklib.executables.Executables method), 3

find_npm() (dktasklib.executables.Executables method), 3
 find_pymodule() (in module dktasklib.utils), 6
 find_twine() (dktasklib.executables.Executables method), 3
 find_uglify() (dktasklib.executables.Executables method), 3
 find_wheel() (dktasklib.executables.Executables method), 3
 fix_line_endings() (in module dktasklib.concat), 2
 fmt() (in module dktasklib.utils), 6

G

get() (dktasklib.package.package_interface.Package method), 1
 get_version() (in module dktasklib.version), 6
 global_package() (in module dktasklib.npm), 3

I

inline_data() (in module dktasklib.urlinliner), 5
 inline_file() (in module dktasklib.urlinliner), 5
 inline_url() (in module dktasklib.urlinliner), 5
 install_cmd() (in module dktasklib.entry_points.dktasklibcmd), 1
 items() (dktasklib.pset.pset method), 4

K

keys() (dktasklib.pset.pset method), 4

L

line_endings() (in module dktasklib.concat), 2

M

main() (in module dktasklib.entry_points.dktasklibcmd), 1
 message() (in module dktasklib.utils), 6
 min_name() (in module dktasklib.version), 6
 MissingCommand, 3

N

needs_to_run() (dktasklib.rule.BuildRule method), 4
 npm() (in module dktasklib.npm), 3

O

on_modified() (dktasklib.watch.DirectoryModified method), 7
 on_modified() (dktasklib.watch.FileModified method), 7
 overridables (dktasklib.package.package_interface.Package attribute), 1
 overrides() (dktasklib.package.package_interface.Package method), 1

P

Package (class in dktasklib.package.package_interface), 1
 PATH, 2
 pfind() (in module dktasklib.package.package_interface), 1
 pset (class in dktasklib.pset), 4

R

remove() (dktasklib.pset.pset method), 4
 require() (dktasklib.executables.Executables method), 3
 requires (dktasklib.rule.BuildRule attribute), 4
 requires() (in module dktasklib.executables), 3
 Result (class in dktasklib.runners), 5
 returncode (dktasklib.runners.Result attribute), 5
 run() (dktasklib.rule.BuildRule method), 4
 run() (in module dktasklib.runners), 5

S

start() (dktasklib.watch.Watcher method), 7
 switch_extension() (in module dktasklib.utils), 6

T

dk-task() (in module dktasklib.wintask), 7
 topsort() (dktasklib.rule.BuildRule method), 4

U

UpdateTemplateVersion (class in dktasklib.upversion), 5

V

values() (dktasklib.pset.pset method), 4
 version_name() (in module dktasklib.version), 6
 versioned_name() (in module dktasklib.version), 6

W

watch_directory() (dktasklib.watch.Watcher method), 7
 watch_file() (dktasklib.watch.Watcher method), 7
 Watcher (class in dktasklib.watch), 7