Dirtools Documentation

Release stable

August 07, 2015

Contents

Dirtools is a little Python package aimed to provide the following features:

- Exclude/ignore files in a directory, using .gitignore like syntax (unix filename pattern matching).
- Generate a hash for a directory tree in order to check if a directory has been modified.
- Search recursively for all subidirs containing a given filename (all projects directory inside a dir).
- Track changes in a directory over time (without duplicating it or without having direct access to it).

CHAPTER 1

Installation

CHAPTER 2

Getting Started

2.1 Excluding files

Dirtools let you exlude files using .gitignore like syntax (unix filename pattern matching), by default `dirtools` will look for a `.exclude` file at root.

Here is how to check if a file should be excluded:

```
from dirtools import Dir

d = Dir('/path/to/dir', exclude_file='.gitignore')
d.is_excluded('/path/to/dir/script.pyc')
```

2.2 Hashdir

The hashdir represent the state of every files in a directory. It compute the hash of the hash of each file recursively.

Here is how to compute the hash of a directory, excluded files ares skipped if any.

```
from dirtools import Dir

d = Dir('/path/to/dir')
hashdir = d.hash()
```

2.3 Find directories containing a file

We'll call these directories **project**, `find_projects` will search recursively for subdirectories with a `file_identifier` file in it.

```
from dirtools import Dir

d = Dir('/path/to/dir')
projects = d.find_projects('.project')
```

2.4 Compress the directory with gzip

Dirtools provides a helper to compress the whole directory (except excluded files/dirs) with gzip.

```
from dirtools import Dir

d = Dir('/path/to/dir')

# By default, dirtools will create a temporary file to store the archive for you
tmp_archive = d.compress_to()

# But you can specify a file
archive_path = '/home/thomas/mydir.tgz'
d.compress_to(archive_path)
```

Or if you want to do it manually:

```
import tarfile
from dirtools import Dir

d = Dir('/path/to/mydir', exclude_file='.gitignore')

with tarfile.open(fileobj=out, mode="w:gz")) as tar:
    tar.add(filename, arcname=arcname, exclude=d.is_excluded)
```

2.5 Track changes in directories

Dirtools provides an helper `DirState` to help tracking changes in a directory over time, without duplicating it or without having direct access to it.

```
from dirtools import Dir, DirState

d = Dir(path)
dir_state = DirState(d)

state_file = dir_state.to_json()

# Later... after some changes

dir_state = DirState.from_json(state_file)
dir_state2 = DirState(d)

changes = dir_state2 - dir_state
```

2.6 Helpers

All methods/properties exclude files and directories based on patterns in `exclude_file` and the `excludes` list.

2.6.1 Custom Walker

If you need to perform operations on files or directories, you can use `Dir.walk`, it works exactly like `os.walk`, except it will skip excluded files/directories on the fly.

```
from dirtools import Dir

d = Dir('/path/to/dir')
```

```
for root, dirs, files in self.walk():
    # do something
```

2.6.2 List all subdirectories of a directory

```
from dirtools import Dir

d = Dir('/path/to/dir')

dirs = d.subdirs()

myproject_dirs = d.subdirs('myproject_*')
```

2.6.3 List all files recursively

```
from dirtools import Dir

d = Dir('/path/to/dir')

files = d.files()

py_files = d.files('*.py')
```

CHAPTER 3

License (MIT)