yamldirs Documentation

Release 1.1.8

Oct 06, 2021

Contents

Create directories and files (including content) from yaml spec.

This module was created to rapidly create, and clean up, directory trees for testing purposes.

Installation:

```
pip install yamldirs
```

New in version 1.1.7: a (very basic) `yamldirs` command has been added:

```
yamldirs dirname
```

prints the yaml for the directory tree rooted at `dirname` (the formatting, while technically correct, probably needs some human editing).

To extract the directory tree defined by `file.yaml`:

```
yamldirs file.yaml
```

the extension needs to be exactly `.yaml`.

CHAPTER 1

Usage

The YAML record syntax is:

```
fieldname: content
fieldname2: |
    multi
    line
    content
nested:
    record: content
```

`yamldirs` interprets a (possibly nested) yaml record structure and creates on-disk file structures that mirrors the yaml structure.

The most common usage scenario for testing will typically look like this:

```
from yamldirs import create_files

def test_relative_imports():
    files = """
        foodir:
            - __init__.py
            - a.py: |
                from . import b
            - b.py: |
                from . import c
            - c.py
    """
    with create_files(files) as workdir:
        # workdir is now created inside the os's temp folder, containing
        # 4 files, of which two are empty and two contain import
        # statements. Current directory is workdir.

    # `workdir` is automatically removed after the with statement.
```

If you don't want the workdir to disappear (typically the case if a test fails and you want to inspect the directory tree)

you'll need to change the with-statement to:

```
with create_files(files, cleanup=False) as workdir:
    ...
```

`yamldirs` can of course be used outside of testing scenarios too:

```
from yamldirs import Filemaker

Filemaker('path/to/parent/directory', """
    foo.txt: |
        hello
    bar.txt: |
        world
""")
```

CHAPTER 2

Syntax

If you're new to yaml and receive a `yaml.parser` error you don't understand, it might be useful to run your yaml through an online validater (e.g. https://codebeautify.org/yaml-validator).

The yaml syntax to create a single file:

```
foo.txt
```

Files with contents uses the YAML record (associative array) syntax with the field name (left of colon+space) is the file name, and the value is the file contents. Eg. a single file containing the text *hello world*:

```
foo.txt: hello world
```

for more text it is better to use a continuation line (`|` to keep line breaks and > to convert single newlines to spaces):

```
foo.txt: |
    Lorem ipsum dolor sit amet, vis no altera doctus sanctus,
    oratio euismod suscipiantur ne vix, no duo inimicus
    adversarium. Et amet errem vis. Aeterno accusamus sed ei,
    id eos inermis epicurei. Quo enim sonet iudico ea, usu
    et possit euismod.
```

To create empty files you can do:

```
foo.txt: ""
bar.txt: ""
```

but as a convenience you can also use yaml list syntax:

```
- foo.txt
- bar.txt
```

For even more convenience, files with content can be created using lists of records with only one field each:

```
- foo.txt: |
    hello
- bar.txt: |
    world
```

Note: This is equivalent to this json: `[{"foo.txt": "hello"}, {"bar.txt": "world"}]`

This is especially useful when you have a mix of empty and non-empty filess:

```
mymodule:
    - __init__.py
    - mymodule.py: |
        print "hello world"
```

directory with two (empty) files (YAML record field with list value):

```
foo:
    - bar
    - baz
```

an empty directory must use YAML's inline list syntax:

```
foo: []
```

nested directories with files:

```
foo:
    - a.txt: |
        contents of the file named a.txt
    - bar:
        - b.txt: |
            contents of the file named b.txt
```

It's worth noting that you cannot mix record and list syntax in the same nesting level:

```
# wrong
dir1:               # top-level record
    - file1         # first level is a list..
    - file2         # .. file1 and file2 are here empty files
    dir2:           # <== ERROR: You cannot define a mapping item when in a sequence
        - file3
        - file4
```

the solution is to make `dir2` a list item:

```
dir1:
    - file1
    - file2
    - dir2:    # <== Correct.
        - file3
        - file4
```

the corresponding json is:

```
>>> print json.dumps(yaml.load("""
... dir1:
...   - file1
...   - file2
...   - dir2:
...     - file3
...     - file4
... """), indent=4)
{
    "dir1": [
        "file1",
        "file2",
        {
            "dir2": [
                "file3",
                "file4"
            ]
        }
    ]
}
```

or make the first level (`b, c, d` below) record fields:

```
a:
    b: b
    c: c
    d:
        e: e
```

corresponding json:

```
>>> print json.dumps(yaml.load("""
... a:
...   b: b
...   c: c
...   d:
...     e: e
... """), indent=4)
{
    "a": {
        "c": "c",
        "b": "b",
        "d": {
            "e": "e"
        }
    }
}
```

Note: (Json) YAML is a superset of json, so you can also use json syntax if that is more convenient.

CHAPTER 3

Extending yamldirs

To extend `yamldirs` to work with other storage backends, you'll need to inherit from `yamldirs.filemaker.FilemakerBase` and override the following methods:

```
class Filemaker(FilemakerBase):
    def goto_directory(self, dirname):
        os.chdir(dirname)

    def makedir(self, dirname, content):
        cwd = os.getcwd()
        os.mkdir(dirname)
        os.chdir(dirname)
        self.make_list(content)
        os.chdir(cwd)

    def make_file(self, filename, content):
        with open(filename, 'w') as fp:
            fp.write(content)

    def make_empty_file(self, fname):
        open(fname, 'w').close()
```

CHAPTER 4

API documentation

CHAPTER 5

Indices and tables

- genindex
- modindex
- search