pageit Documentation

Release 0.2.3

The Metaist

December 11, 2016

Contents

pageit is a python tool for generating static website using Mako Templates similar to Jekyll and Hyde.

CHAPTER 1

Quick Links

- Code & Issues (GitHub)
- Documentation (Read the Docs)
- Package (PyPI)

CHAPTER 2

Getting Started

1. Install the package:

```
$ pip install pageit
```

2. Create a directory called *site* where you will put your html files:

```
$ mkdir site
$ cd site
```

3. Run `pageit` to generate the html files:

```
$ pageit
```

- Directories with names that end with `.mako` will be ignored. This is a good way to prevent your mako layouts from getting rendered.
- Files that end with `.mako` will be rendered using `mako`; the generated file will have the `.mako` extension removed.

CHAPTER 3

Contribute

If you want to play around with the latest code, start by cloning the repository, installing the dependencies, and building using Paver:

```
$ git clone git://github.com/metaist/pageit.git
$ pip install -r requirements.txt --use-mirrors
$ paver test
```

CHAPTER 4

License

Licensed under the MIT License.

CHAPTER 5

Contents

5.1 Installing pageit

5.1.1 Using pip

The fastest way to get pageit is using `pip`:

```
$ pip install pageit
```

5.1.2 From Source

You can also install pageit from source:

```
$ git clone git://github.com/metaist/pageit.git
$ pip install -r requirements.txt --use-mirrors
$ [sudo] paver install
```

5.2 Running pageit

pageit is designed for developers who need to quickly generate a standalone static website. There are several command-line options that may be useful for development.

5.2.1 Basic Example

The most common thing you want pageit to do is to process the files in the current directory:

```
$ pageit
```

Sometimes you want to run pageit on a different directory:

```
$ pageit some/other/path
```

Most often, you'll want to run pageit, have it automatically re-run when files change, and serve it out for testing:

```
$ pageit --watch --serve
```

5.2.2 Command-Line Options

All command-line options are optional and default to `False` or ignored unless inidicated otherwise.

`-n, --dry-run`
Do not perform destructive operations such as generating output or deleting files.

`-c, --clean`
Remove generated output.

See *`clean()`* for more details.

`-r, --render`
Render templates after cleaning (combine with *`-c`*).

`-w, --watch`
Watch the path using watchdog and re-run pageit when files change.

See *`watch()`* for more details.

`-s` `<PORT=80>`**`, --serve`** `<PORT=80>`
Serve the path using the SimpleHTTPServer. This is not recommended for production environments.

See *`serve()`* for more details.

`-f` `<PATH>`**`, --config`** `<PATH>`
Path to YAML configuration file (default: `pageit.yml`) containing a dictionary of environment values (key/value pairs). The environment values are passed to `mako` templates during rendering via the special `site` variable. The configuration file is first searched for in the current working directory and then in the `pageit` directory.

See *Special Mako Variables* in *`mako()`* for more details.

New in version 0.2.1.

`-e` `<ENV>`**`, --env`** `<ENV>`
Name of the configuration environment to load (default: `default`). The special `default` environment values are always loaded first and then extended with the values from the environment named this option.

See *Special Mako Variables* in *`mako()`* for more details.

New in version 0.2.1.

`--tmp` `<PATH>`
Directory in which to store generated `mako` templates. By default, generated templates are not stored.

`--ignore-mtime`
Render all the templates rather than only those that have changed (or whose dependencies have changed). For templates that have complicated inheritance rules, this flag may have to be set to get templates to render.

See *`mako_mtime()`* for more details.

`--noerr`
Do not alter the template output to be an HTML error page if an error arises during rendering. By default, template errors will be captured and inserted into the output file.

`--ext`
Extension for mako templates (default: `.mako`). Directory names ending with this string be ignored.

5.3 Script API v 0.2.3

5.3.1 Cleaning & Rendering

It's easy to include *Pageit* in your own scripts:

```
from pageit.render import Pageit
Pageit(path='.').clean().run()
```

5.3.2 Watching for File Changes

Use *watch()* to call a callback function when files change:

```
from pageit.tools import watch
def my_func(path):
    print path, 'has changed'

with watch(path='.', callback=my_func) as watcher:
    watcher.loop()  # wait for CTRL+C
```

See also:

watchdog for the cross-platform directory-watching library

5.3.3 Basic HTTP Server

Use *serve()* to serve up files on a given port:

```
from pageit.tools import serve
serve(path='.', port=80)  # will wait until CTRL+C
```

See also:

Python's built-in SimpleHTTPServer for serving directories

5.3.4 Watching & Serving

If you want to watch and serve a path:

```
from pageit.tools import watch, serve
with watch(path, callback) as watcher:
    serve(path, port)  # wait for CTRL+C
```

5.3.5 Python Modules

pageit.render

pageit generator

class `pageit.render.`**`Pageit`**(*path='.', ext='.mako', dry_run=False, noerr=False, ignore_mtime=False, watcher=None, tmpl=None, site=None, log=None*)

Bases: `object`

Mako template renderer.

watcher pageit.tools.Watcher
underlying watcher for this path

Parameters

- **path** (*str, optional*) – path to traverse; default is current dir
- **ext** (*str, optional*) – extension to look for
- **dry_run** (*bool, optional*) – if True, print what would happen instead of rendering; default is False
- **noerr** (*bool, optional*) – if True, don't create error files; default is False
- **ignore_mtime** (*bool, optional*) – if True, do not consider template modification times when rendering; default is False.

  Note: pageit does not currently warn you if the output is *newer* than the template.

- **watcher** (*pageit.tools.Watcher, optional*) – underlying watcher for the given path

  Note: You may attach the watcher after this object has been constructed via the *watcher* attribute.

- **tmpl** (*mako.lookup.TemplateLookup, optioanl*) – mako template lookup object
- **site** (*pageit.namespace.DeepNamespace, optional*) –

  DeepNamespace passed to mako templates during rendering

- **log** (*logging.Logger, optional*) – system logger

Changed in version 0.2.1: Added the `site` parameter.

clean()
Deletes pageit output files.

Note: This function only deletes files for which there is a corresponding template. If the template was moved or deleted, the output will not be touched.

Returns *Pageit* – for method chaining

list()
Generates list of files to render / clean.

This function only lists files that end with the appropriate extension, will not enter directories that end with that extension.

Yields *str* – next file to process

mako(*path*, *dest=None*)
Render a mako template. This function injects two *DeepNamespace* variables into the `mako` template:

- `site`: environment information passed into the constructor
- `page`: information about the current template
  - `path`: relative path of this template

–`output`: relative path to the output of the template

–`dirname`: name of the directory containing the template

–`basedir`: relative path back to the root directory

Parameters

- **path** (*str*) – template path
- **dest** (*str, optional*) – output path; if not provided will be computed

Returns *Pageit* – for method chaining

Changed in version 0.2.2: Added more template information (output, dirname, basedir).

mako_deps (*path*)

Returns set of immediate dependency paths for a mako template.

Note: This function does not recursively compute dependencies.

Parameters **path** (*str*) – path to a mako template

Returns *set* – paths of dependencies

Examples

```
>>> Pageit().mako_deps('fake.mako')
set([])
```

mako_mtime (*path*, *levels=5*)

Returns the modification time of a mako template.

Note: This function considers a limited portion of the template's inheritance tree to determine the latest modification time.

Parameters

- **path** (*str*) – template path
- **levels** (*int*) – number of inheritance levels to traverse (default: 5)

Returns *int* – latest modification time; 0 if the file does not exist

Examples

```
>>> Pageit().mako_mtime('fake.mako')
0
```

```
>>> import os.path as osp
>>> path1 = osp.abspath('test/example1')
>>> path2 = osp.join(path1, 'subdir/test-page.html.mako')
>>> Pageit(path1).mako_mtime(path2) > 0
True
```

on_change(*path=None*)

React to a change in the directory.

Parameters **path** (*str*) – path that changed

Returns *Pageit* – for method chaining

Examples

```
>>> import os.path as osp
>>> runner = Pageit('test/example1')
>>> _ = runner.run()
>>> _ = runner.on_change('test/example1/index.html.mako')
>>> _ = runner.on_change(osp.abspath('test/example1/index.html'))
>>> _ = runner.clean()
>>> _ is runner
True
```

run()

Runs the renderer.

Returns *Pageit* – for method chaining

Examples

```
>>> runner = Pageit('test/example1', dry_run=True)
>>> runner.run().clean() is runner
True
```

```
>>> runner = Pageit('test/example1', ignore_mtime=True)
>>> runner.run().clean() is runner
True
```

```
>>> runner = Pageit('test/example1')
>>> runner.run().clean() is runner
True
```

pageit.render.**create_config**(*path='pageit.yml'*, *env='default'*, *log=None*)

Constructs a *DeepNamespace* for attributes to pass to mako templates.

The configuration file should be a list of keys mapped to key/value pairs. The load process first loads an environment called "default" and then extends those keys to include those in the given environment.

Parameters

- **path** (*str*) – YAML configuration file
- **env** (*str, optional*) – section to load
- **log** (*logging.Logger, optional*) – system logger

Returns

pageit.namespace.DeepNamespace –

DeepNamespace of environment attributes

Examples

```
>>> create_config('file.yml', 'local') == DeepNamespace()
True
```

```
>>> conf = create_config('test/example1/pageit.yml', 'test')
>>> conf.debug == True
True
```

```
>>> conf = create_config('test/example1/pageit.yml', 'fakeenv')
>>> 'debug' not in conf
True
```

New in version 0.2.1.

pageit.render.**create_logger**(*verbosity=1*, *log=None*)

Constructs a logger.

Parameters

- **verbosity** (*int*) – level of verbosity
- **log** (*logging.Logger*) – an existing logger to modify

Returns *logging.Logger* – logger configured based on verbosity.

Example

```
>>> log = create_logger()
>>> log.getEffectiveLevel() == logging.INFO
True
```

```
>>> log = create_logger(0, log)
>>> log.getEffectiveLevel() == logging.NOTSET
True
```

```
>>> log = create_logger(2, log)
>>> log.getEffectiveLevel() == logging.DEBUG
True
```

pageit.render.**create_lookup**(*path='.'*, *tmp=None*)

Constructs a mako TemplateLookup object.

Parameters

- **path** (*str*) – top-level path to search for mako templates
- **tmp** (*str, optional*) – directory to store generated modules

Returns *mako.lookup.TemplateLookup* – object to use for searching for templates

Example

```
>>> create_lookup() is not None
True
```

pageit.render.**main**()

Console entry point.

Constructs and dispatches the argument parser.

pageit.render.**render**(*args*)
Convenience method for *Pageit*.

Parameters **args** (*Namespace*) – argh arguments

Changed in version 0.2.1: Added configuration loading.

pageit.render.**strip_ext**(*path*, *ext*)
Remove an extension from a path, if present.

Parameters

- **path** (*str*) – path from which to strip extension
- **ext** (*str*) – extension to strip

Returns *str* – path with extension removed

Examples

```
>>> strip_ext('foo.bar', '.bar')
'foo'
>>> strip_ext('foo.bar', '.baz')
'foo.bar'
```

pageit.tools

Tools for changing, serving, and watching paths.

class pageit.tools.**Watcher**(*path=None*, *callback=None*, *log=None*)
Bases: watchdog.events.FileSystemEventHandler

Handler for file changes.

Parameters

- **path** (*str*) – path to watch
- **callback** (*callable*) – function to run when files change
- **log** (*logging.Logger, optional*) – logger to use

__enter__()
Enter a context.

Returns *Watcher* – this watcher

__exit__(*exc_type*, *exc_value*, *traceback*)
Exit a context.

loop()
Run until CTRL+C is pressed.

Returns *Watcher* – for method chaining

on_modified(*event=None*)
Handle a file modification.

Parameters **event** (*object*) – watchdog event object

Returns *Watcher* – for method chaining

start()
Start the underlying observer.

Returns *Watcher* – for method chaining

stop()
Stop the underlying observer.

Returns *Watcher* – for method chaining

pageit.tools.**pushd**(**args*, ***kwds*)
Change the current working directory for a context.

Parameters **path** (*str*) – temporary path to change to

Yields *str* – absolute path to the new path

Example

```
>>> cwd = os.getcwd()
>>> with pushd('..') as newpath:
...     os.getcwd() != cwd
True
```

pageit.tools.**serve**(*path*, *port=80*, *log=None*)
Serve a path on a given port.

This function will change the working directory to the path and host it on the port specified. If *path* is not supplied, it returns immediately.

Parameters

- **path** (*str*) – path to host
- **port** (*int, optional*) – port on which to host; default is 80.
- **log** (*logging.Logger, optional*) – logger to use

pageit.tools.**watch**(**args*, ***kwds*)
Watch a directory and call the given callback when it changes.

This is a convenience method for *Watcher*.

Parameters

- **path** (*str*) – path to watch
- **callback** (*callable*) – callable to call when path changes; will be passed the path to the file that changed
- **log** (*logging.Logger, optional*) – logger to use

Yields *Watcher* – file system event handler for this watch

pageit.namespace

Flexible objects and dictionaries.

Namespace objects provides simple ways to bunch together key/values while providing both dot- and array-notation setters and getters.

DeepNamespace act in a similar manner, except that they apply theselves recursively.

class pageit.namespace.**DeepNamespace**(**args*, ***kwds*)

Bases: *pageit.namespace.Namespace*

A recursive namespace.

Similar to a normal *Namespace*, except that setting an attribute to a dictionary, converts it into a *DeepNamespace*.

Parameters

- ***args** – dictionaries or objects to merge
- ****kwds** – converted into a dictionary

Note: Variable arguments take precedence over keyword arguments.

Examples

```
>>> ns = DeepNamespace({"a": {"b": 1}})
>>> ns.a.b == 1
True
```

```
>>> ns = DeepNamespace(x=DeepNamespace(y=1))
>>> ns.x.y == 1
True
```

New in version 0.2.2.

__getattr__(*name*)

Returns the attribute value (dot notation).

This lets you safely reference attributes that don't exist in a chainable way. You can test for existince using `len()`.

Note: Since this method is only called when an attribute does not exist, by definition this method will always return an empty *DeepNamespace*.

However, it also has the side effect of **creating** that attribute in the namespace so that you can assign arbitrary values.

Parameters **name** (*str*) – attribute name (ignored)

Returns *DeepNamespace* – this method is only called when an attribute does not exist

Example

```
>>> ns = DeepNamespace(a=1)
>>> ns.b.c is not None
True
>>> len(ns.b.c) == 0
True
```

```
>>> ns.b = 2
>>> ns.b == 2
True
```

__setitem__(*name*, *val*)
Sets the value of an attribute (array notation).

If `val` is a dictionary or an object with attributes, it will be recursively converted into a *DeepNamespace*.

Parameters

- **name** (*str*) – attribute name
- **val** – attribute value

Example

```
>>> ns = DeepNamespace()
>>> ns['a'] = {"b": {"c": 2}}
>>> ns.a.b.c == 2
True
```

```
>>> ns.x.y.z = 'Works'
>>> ns.x.y.z == 'Works'
True
```

```
>>> ns.q = Namespace(a=1)
>>> isinstance(ns.q, DeepNamespace)
True
>>> ns.q.a == 1
True
```

class pageit.namespace.**Namespace**(**args*, ***kwds*)
Bases: _abcoll.MutableMapping

A simple namespace.

Access attributes of this object with dot or array notation.

Parameters

- ***args** – dictionaries or objects to merge
- ****kwds** – converted into a dictionary

Note: Variable arguments take precedence over keyword arguments.

Examples

```
>>> ns = Namespace(a=1, b=2)
>>> (ns.a == 1 and ns['b'] == 2)
True
```

```
>>> Namespace(a=1, b=2) == Namespace({'a': 1, 'b': 2})
True
```

```
>>> Namespace(None, a=1) == Namespace(a=1)
True
```

```
>>> class Foo(object):
...     pass
>>> x = Foo()
>>> x.a = 1
>>> Namespace(x) == Namespace(a=1)
True
```

```
>>> x = None
>>> try:
...     x = Namespace([1,2,3])
... except AssertionError:
...     pass
>>> x is None
True
```

__add__(*other*)

Add another object to this object.

Parameters **other** (*Namespace, dict, object*) – object to add

Example

```
>>> Namespace(a=1) + {'b': 2} == Namespace(a=1, b=2)
True
```

__contains__(*name*)

Returns True if name is in the Namespace.

Parameters **name** (*str*) – name of the attribute

Returns *bool* – True if the name is in the namespace; False otherwise

Examples

```
>>> 'a' in Namespace(a=1)
True
>>> 'b' not in Namespace(a=1)
True
```

__delattr__(*name*)

Deletes an attribute (dot notation).

Parameters **name** (*str*) – name of the attribute to delete

Example

```
>>> ns = Namespace(a=1)
>>> del ns.a
>>> 'a' not in ns
True
```

__delitem__(*name*)

Deletes an attribute (array notation).

Parameters **name** (*str*) – name of the attribute to delete

Example

```
>>> ns = Namespace(a=1)
>>> del ns['a']
>>> 'a' not in ns
True
```

__eq__(*other*)
Returns True if the items are equal.

Parameters **other** (*Namespace*) – object of comparison

Example

```
>>> Namespace(a=1) == Namespace({'a': 1})
True
```

__getattr__(*name*)
Returns the attribute value (dot notation).

Note: Since this method is only called when an attribute does not exist, by definition this method will always return `None`.

Parameters **name** (*str*) – attribute name (ignored)

Returns *None* – this method is only called when an attribute does not exist

Example

```
>>> ns = Namespace(a=1)
>>> ns.b is None
True
```

```
>>> ns.b = 2
>>> ns.b == 2
True
```

__getitem__(*name*)
Returns the attribute value (array notation).

Parameters **name** (*str*) – attribute name

Returns value of the attribute or None if it does not exist

Example

```
>>> ns = Namespace(a=1)
>>> ns['a'] == 1
True
>>> ns['b'] is None
True
```

__hash__()
Returns the hash of this object.

```
>>> hash(Namespace(a=1)) == hash('Namespace(a=1)')
True
```

__iter__()
Returns an iterator.

Example

```
>>> [attr for attr in Namespace(a=1)] == ['a']
True
```

__len__()
Returns the number of attributes set.

Example

```
>>> len(Namespace(a=1, b=2)) == 2
True
```

__radd__(*other*)
Add this object to another object.

Parameters **other** (*dict, object*) – object to which to add

Example

```
>>> {'a': 1} + Namespace(b=2) == Namespace(a=1, b=2)
True
```

__repr__()
Returns a string representation of the object.

Example

```
>>> repr(Namespace(a=1))
'Namespace(a=1)'
```

Changed in version 0.2.2: Use the name of the class instead of a hard-coded string.

__setattr__(*name*, *val*)
Sets the value of an attribute (dot notation).

Parameters

- **name** (*str*) – attribute name
- **val** – attribute value

Example

```
>>> ns = Namespace(a=1)
>>> ns.b = 2
>>> ns.b == 2
True
```

New in version 0.2.2.

__setitem__(*name*, *val*)
Sets the value of an attribute (array notation).

Parameters

- **name** (*str*) – attribute name
- **val** – attribute value

Example

```
>>> ns = Namespace(a=1)
>>> ns['b'] = 2
>>> ns.b == 2
True
```

pageit.namespace.**extend**(**items*)
Extend a dictionary with a set of dictionaries.

Parameters ***items** – dictionaries to extend; the first argument will be modified

Returns *dict* – the first dictionary extended with values from the other dictionaries

Examples

```
>>> extend({}, {'a': 1}, {'a': None}) == {'a': 1}
True
>>> extend({'a': 1}, {'b': 2}, {'a': 4}) == {'a': 4, 'b': 2}
True
>>> extend({'a': {'b': 3}}, {'a': {'c': 2}}) == {'a': {'b': 3, 'c': 2}}
True
>>> extend({'a': {'b': 3}}, {'a': {'b': 2}}) == {'a': {'b': 2}}
True
```

pageit.namespace.**getattrs**(*obj*, **names*)
Returns multiple attributes of an object.

Parameters

- **obj** (*object*) – object
- ***names** – variable list names of attributes

Returns *tuple* – attribute values

Example

```
>>> x = Namespace(a=1, b=2, c=3)
>>> a, c, d = getattrs(x, 'a', 'c', 'd')
>>> a == x.a and c == x.c and d is None
True
```

p

Symbols

C

D

E

G

L

M

N

O

P

R

S

W