IHIH Documentation

Release 0.1

Romain Dartigues

October 19, 2013

CONTENTS

Overview

IHIH (I Hate INI hacks) is an attempt to provide simple configuration parsers (for Python) with a dictionary-like interface.
It try to be flexible and let you alter the syntax by sub-classing it.

CHAPTER

ONE

WHY?

Because I Hate INI (initialization) files. I don't need sections, i think `ConfigParser` is a pain to use...

And also because in my opinion configuration files should not be *executed* (ie: i feel bad having a Python file as a configuration system, sure it is *flexible*, but, you know... [if you don't, you probably don't need this]).

1.1 Source documentation

ihih - simple configuration parsers with dictionary-like interface

License: BSD 3-Clause

class `ihih.`**`IHIH`**(*filenames*, **args*, ***kwargs*)

Bases: `dict`

IHIH - simple configuration parser

One key/value pair per line.

`encoding` **= 'utf8'**

define the encoding

`__init__`(*filenames*, **args*, ***kwargs*)

attempt to parse a list of filenames

Parameters:

- *filenames* – if is a string, it is treated as a single file, otherwise it is treated as an iterable
- other parameters are passed to the `dict` constructor

`reload`(*force=False*)

call `parse()` on each configuration file

`parse`(*filename*, *force=False*)

parse a configuration file

Note: *filename* should be an absolute path.

`_unescape`(*value*, *quote=None*)

remove escape prefix on "known escape"

See `_escaped_chars`.

This method attempt to utf8 encode `unicode()` objects.

`_handle_fragment` (*fragment*, *quote=None*)
handle a fragment of a value

Provided to help on subclassing.

`_strip_comment` (*value*)
remove the comment on value

`_parse_value` (*value*, *data*)
parse the "value" part of a "key / value"

This function handle the quoted parts.

Parameters:

- *value* (`basestring()` instance): value to parse
- *data*: instance supporting += operator

`_cast_str` (*value*)
return a string representation of *value*

`__contains__` (*key*)
True if self contains *key*

Note: The *key* will be casted as `str()` (see: `_cast_str()`).

`__setitem__` (*key*, *value*)
set item *key* to *value*

Note: Both variables will be casted as `str()` (see: `_cast_str()`).

`__getitem__` (*key*)
return *key* value as internal type

You probably want to use one of the following: `get_str()`, `get_unicode()`, `get_float()`.

Note: The *key* will be casted as `str()` (see: `_cast_str()`).

`__delitem__` (*key*)
delete *key* from dict

Note: The *key* will be casted as `str()` (see: `_cast_str()`).

`get_str` (*key*, *default=None*)
return *key* value as `str()` or *default* if not found

Note: The *key* will be casted as `str()` (see: `_cast_str()`).

`get` (*key*, *default=None*)
alias to `get_str()`

`__weakref__`
list of weak references to the object (if defined)

get_unicode (*key*, *default=None*, *errors='strict'*)
return *key* value as `unicode()` or *default* if not found

The *errors* parameter is passed to `str.decode()`.

Note: The *key* will be casted as `str()` (see: `_cast_str()`).

get_float (*key*, *default=None*, *errors='strict'*)
return *key* value as `float()` or *default* if not found

If *errors* is "ignore", return *default* value instead of raising `TypeError` on failure.

Note: The *key* will be casted as `str()` (see: `_cast_str()`).

class `ihih.`**IHIHI** (**args*, ***kwargs*)
Bases: `ihih.IHIH`

IHIH Interpolate - `IHIH` with variable interpolation

_handle_fragment (*fragment*, *quote=None*)
search for variables in *fragment*

__getitem__ (*key*, *path=None*)
return *key* value as internal type with interpolated variables

For more informations, see: `__getitem__()`.

_recursive (*value*)
recursive variable handler

Default: empty string

You can overwrite this function when subclassing and chose to return a unexpended version of the variable, raise an error or make a single, non recursive, lookup.

1.2 Examples

1.2.1 Getting started

Attempt to load a system-wide configuration file, whose settings will be overwritten by a user preferences files.

Missing files are silently ignored.

```
from ihih import IHIH

conf = IHIH(
   (
      '/etc/example.conf',
      os.path.join(os.path.expanduser('~'), '.example.conf')
   ),
   debug='1'
)

if conf.get_float('debug', errors='ignore'):
   print 'i am running in debug mode'
```

1.2.2 Reloading the conf

Assuming *conf* is a `IHIH` instance.

```
# reload on SIGHUP
import signal

signal.signal(signal.SIGHUP, lambda s, f: conf.reload())
```

1.2.3 Configuration format

By default, `IHIH` parse files using the following rules:

- the key is before the first = character
- the value is everything after the first = character
- the value might be empty
- key and value have their leading and trailing spaces stripped
- values can be quoted (between `'` or `"`)
- quoted values have their quotes automatically removed (ie: `"my value"` becomes `my value`)
- single quotes are considered as a character
- lines not matching the key / separator / value are ignored
- comments (beginning with a `#` or `//`) are ignored and deleted from the value except if they are escaped or quoted
- specials characters (`\'"#/`) can be escaped by prefixing them with a backslash (`\`) to not be treated specially
- other (non-special) characters preceded by the escape character are not treated specially and the escape character is preserved

By default, `IHIHI` parse files accordingly the following rules:

- same-same than `IHIH`
- add dollar (`$`) in the special character list
- every word prefixed by a non-escaped dollar and not embraced by single-quotes (`'`) is considered as a variable
- strings beginning with `${` and ending with `}` are also variables, this let you define variables containing non-word characters such as dots hyphens, or spaces
- variables interpolation is done when using the variable, this let you define (or change) the variable content later
- when a variable is not found, it resolve as an empty string
- variable recursion resolve to an empty string

Which mean that it could parse, to a certain extent (see *Single-line only*), subset of:

- shell script
- Postfix main.cf
- Python
- INI (will ignore the sections)

That could be convenient if you have to share a configuration file between scripts, given you pay attention to respect both formats.

Examples of configuration files

Parsing a shell script:

```
# as in shell
FOO="bar"
FOOBAR=foo-$FOO    # resolve as: foo-bar
FOOBAR="foo-$FOO" # resolve as: foo-bar
FOOBAR='foo-$FOO' # resolve as: foo-$FOO
BAR=${FOO}         # resolve as: bar
ABC="a" 'b' c      # resolve as: a b c
C=hello # world    # resolve as: hello
D=hello \# world   # resolve as: hello # world

# different
DATE=$(date)       # resolve as: $(date)
```

Parsing a main.cf:

```
smtpd_banner = $myhostname ESMTP
myhostname = foo.example.net
```

Parsing some Python:

```
# same
a = 'AA'
b = "BB"

# notably different
c = 'A' "B"     # resolve as: A B
d = c           # resolve as: c
```

Parsing an INI file:

```
; section is ignored
[uwsgi]
http-socket = :9090
processes = 4

; different, resolve as: localhost:9000
URL = localhost${http-socket}
```

1.3 Warnings

1.3.1 Still in beta

This library is still β, expect its internal API to change over time.

Please let me know if you use it, your features requests, bugs, etc.

1.3.2 Not extensively tested

Some tests exists in the test/ directory, but it's still missing much.

Note: I only tested it over Python 2.6.

1.3.3 Default item getter return internal type

You probably want to favor `ihih.IHIH.get()` over `ihih.IHIH.__getitem__()` as the latter return the internal type which might not be suitable for your needs.

1.3.4 Automatic type conversion

This is a key / value, file-based, configuration system; so it forces everything as a string.

Just be aware of that.

1.3.5 File opening failure

Missing configuration files will be silently ignored, *but*, if a configuration file is not readable (permissions errors) or not a file (dead link or directory), it *will* raise an exception, as the user should be notified of this error.

1.4 Known bugs / limitations

If you find some bugs, you are welcome to report them :^)

Please see also the *warnings*.

1.4.1 Partial unicode handling

Unicode is only partially supported, for example it is *not* supported to pre-populate the configuration object with `unicode()`; see not a true dict.

It also assumes all files use the same encoding (default to UTF8, or at least ASCII7).

1.4.2 Not a true dict

The configuration objects do not behave like a true `dict`, especially:

No type conversion on some methods

Type conversion is not supported, at least, on:

- pre-population / initialization (ie: `IHIHI((), {'a': 'b'}))`
- functions: `pop`, `popitem`, `setdefault`, `update`

```
# this will not work as expected (yet)
conf = IHIHI('file.conf', {'pi': 3.14, 'lang': u'', u'': 'Chinese'})

# as a workaround, use this method
conf = IHIHI('file.conf')
conf['pi'] = 3.14
conf['lang'] = u''
conf[u''] = 'Chinese'

# now the defaults has been set, reparse
conf.reload(force=True)
```

```
# or you can alternatively, carefully specify (utf8) strings on the init
conf = IHIHI('file.conf', {'pi': '3.14', 'lang': u''.encode('utf8'),
       u''.encode('utf8'): 'Chinese'})

# now you can
conf['test'] = u'$pi, $lang, $!'

print conf.get_unicode('test') # resolve as: 3.14, , Chinese!
```

1.4.3 Single-line only

It does not, yet, support line-continuation; that mean your configuration value must fit on one line.

CHAPTER

TWO

INDICES AND TABLES

- *genindex*

PYTHON MODULE INDEX