python 模块介绍-ConfigParser 配置文件解析器

目录

项目简介 1

简介： 2

配置文件格式 3

读取配置文件 3

访问配置设置 5

修改配置 10

保存配置 12

选项查找路径 13

变量替换 15

项目简介

Python 中文库 https://bitbucket.org/xurongzhong/python-chinese-library 主要基于个人的使用经验，收集一些重要的外部和内部模块的中文教程和实例。发起人是 ouyangchongwu#gmail.com，xurongzhong@gmail.com 。

欢迎大家加入分享经验。联系方法：xurongzhong#gmail.com，微博：http://weibo.com/cizhenshi，python 及测试开发 qq 群 1：113938272，群 2:6089740。

文件下载：

1， https://bitbucket.org/xurongzhong/python-chinese-library/downloads。 推荐

2， hg clone 克隆所有文件 hg clone https://bitbucket.org/xurongzhong/python-chinese-library。

3， https://bitbucket.org/xurongzhong/python-chinese-library/src 浏览文件，右键点击文件，选另存为下载。

Bug 提交：https://bitbucket.org/xurongzhong/python-chinese-library/issuest。

版本管理

版本	修订发布时	修订人	备注

号	间		
V1.0	2013-11-28	ouyangchongwu@gmail.com	初始版本，由《The Python Standard Library by Example 2011》生成。

参考资料：

官方网址： http://docs.python.org/2/library/configparser.html

简介：

功能：配置文件解析器。读写类 windows 的 ini 文件。

月下载量：

Python 版本：*Python 1.5 以上，在在 3.0 中已经重命名为 configparser。*

当前版本：

下载地址：

平台：跨平台

相关模块：

config 强大的，好用的层次性配置模块，月下载 4k 左右。

configs 人性化的配置模块，月下载 2k 左右。

config-enhance 提供可重用的 ConfigParser 文件。月下载 1k 左右。

config_resolver 自动查找配置文件。月下载 1k 左右。

Configglue 连接 OptionParser 和 ConfigParser。月下载 5k 左右。

configobj 配置文件读写验证。月下载 15k 左右。推荐。

使用 ConfigParser 模块可管理应用程序的用户可配置文件。配置文件的内容可分组，支持整型，浮点值和布尔值。

配置文件格式

ConfigParser 使用的文件格式和旧版本的 Microsoft Windows 类似。它由域组成，每个名字域可包含选项（name 和 value 组成）。

域名包含在方括号中，因此域名不能包含方括号。一行包含一个选项，分隔符是分好或者等号，分隔符首尾的空格会在文件解析时忽略。使用的示例文件 simple.ini 如下：

```
[bug_tracker]

url = http://localhost:8080/bugs/
username = dhellmann
password = SECRET
```

读取配置文件

SafeConfigParser 类的 read 方法可以读取配置文件。

```
from ConfigParser import SafeConfigParser

parser = SafeConfigParser()
parser.read('simple.ini')

print parser.get('bug_tracker', 'url')
```

执行结果会显示域'bug_tracker'的选项'url'：

```
# python ConfigParser_read.py

http://localhost:8080/bugs/
```

read 方法可接受的文件名列表，依次扫描，如果文件存在就打开和读取。

```
from ConfigParser import SafeConfigParser
import glob

parser = SafeConfigParser()

candidates = ['does_not_exist.ini', 'also-does-not-exist.ini',
              'simple.ini', 'multisection.ini',
              ]

found = parser.read(candidates)

missing = set(candidates) - set(found)

print 'Found config files:', sorted(found)
```

```
print 'Missing files     :', sorted(missing)
```

read 返回可以加载的文件列表。

```
# python ConfigParser_read_many.py

Found config files: ['simple.ini']

Missing files     : ['also-does-not-exist.ini', 'does_not_exist.ini', 'multisection.ini']
```

使用 codecs 模块可以配合处理 unicode，配置文件

```
[bug_tracker]
# test
url = http://localhost:8080/bugs/ # test
username = dhellmann
password = ßéç®é†
```

ConfigParser_unicode.py 处理 unicode：

```
from ConfigParser import SafeConfigParser
import codecs

parser = SafeConfigParser()

# Open the file with the correct encoding
with codecs.open('unicode.ini', 'r', encoding='utf-8') as f:
    parser.readfp(f)

password = parser.get('bug_tracker', 'password')

print 'Password:', password.encode('utf-8')
print 'Type     :', type(password)
print 'repr()   :', repr(password)
```

执行结果：

```
# python ConfigParser_unicode.py

Password: ßéç®é†

Type     : <type 'unicode'>

repr()   : u'\xdf\xe9\xe7\xae\xe9\u2020'
```

实际上，代码指定编码 utf-8 的情况下，不需要 codecs 也是可以正常输出的：

```
#!/usr/bin/env python
# encoding: utf-8
```

```
from ConfigParser import SafeConfigParser
import codecs

parser = SafeConfigParser()

parser.read('unicode.ini')

password = parser.get('bug_tracker', 'password')

print 'Password:', password
print 'Type     :', type(password)
print 'repr()   :', repr(password)
```

执行结果：

```
# python ConfigParser_unicode2.py

Password: ßéç®é†

Type     : <type 'str'>

repr()   : '\xc3\x9f\xc3\xa9\xc3\xa7\xc2\xae\xc3\xa9\xe2\x80\xa0'
```

访问配置设置

SafeConfigParser 可以查看配置文件，比如列出域和选项并获取他们的值。配置文件‘multisection.ini’如下：

```
[bug_tracker]
url = http://localhost:8080/bugs/
username = dhellmann
password = SECRET
[wiki]
url = http://localhost:8080/wiki/
username = dhellmann
password = SECRET
```

section 方法返回域名列表，options 返回选项名列表，格式都是字符串。Items 返回名字-值元组的列表。

```
from ConfigParser import SafeConfigParser

parser = SafeConfigParser()
parser.read('multisection.ini')

for section_name in parser.sections():
```

```
    print 'Section:', section_name
    print '  Options:', parser.options(section_name)
    for name, value in parser.items(section_name):
        print '  %s = %s' % (name, value)
    print
```

执行结果：

```
# python ConfigParser_structure.py
Section: bug_tracker
  Options: ['url', 'username', 'password']
  url = http://localhost:8080/bugs/
  username = dhellmann
  password = SECRET

Section: wiki
  Options: ['url', 'username', 'password']
  url = http://localhost:8080/wiki/
  username = dhellmann
  password = SECRET
```

has_section 方法可以测试域名是否存在：

```
from ConfigParser import SafeConfigParser

parser = SafeConfigParser()
parser.read('multisection.ini')

for candidate in [ 'wiki', 'bug_tracker', 'dvcs' ]:
    print '%-12s: %s' % (candidate, parser.has_section(candidate))
```

执行结果：

```
# python ConfigParser_has_section.py
wiki         : True
```

```
bug_tracker : True

dvcs        : False
```

类似的 has_option 可以判断是否选项：

```
from ConfigParser import SafeConfigParser

parser = SafeConfigParser()
parser.read('multisection.ini')

SECTIONS = [ 'wiki', 'none' ]
OPTIONS = [ 'username', 'password', 'url', 'description' ]

for section in SECTIONS:
    has_section = parser.has_section(section)
    print '%s section exists: %s' % (section, has_section)
    for candidate in OPTIONS:
        has_option = parser.has_option(section, candidate)
        print '%s.%-12s  : %s' % (section,
                                  candidate,
                                  has_option,
                                  )
    print
```

执行结果：

```
# python ConfigParser_has_option.py

wiki section exists: True

wiki.username       : True

wiki.password       : True

wiki.url            : True

wiki.description    : False



none section exists: False

none.username       : False

none.password       : False
```

```
none.url          : False

none.description  : False
```

所有域名和选项名是字符串，值类型可以为字符串，是整型，浮点值和布尔值。先创建文件 types.ini:

```
[ints]
positive = 1
negative = -5
[floats]
positive = 0.2
negative = -3.14
[booleans]
number_true = 1
number_false = 0
yn_true = yes
yn_false = no
tf_true = true
tf_false = false
onoff_true = on
onoff_false = false
```

get 方法总是返回字符串，类似的有 getint，getfloat，getboolean 分别返回整型，浮点值和布尔值。

```
from ConfigParser import SafeConfigParser

parser = SafeConfigParser()
parser.read('types.ini')

print 'Integers:'
for name in parser.options('ints'):
    string_value = parser.get('ints', name)
    value = parser.getint('ints', name)
    print '  %-12s : %-7r -> %d' % (name, string_value, value)

print '\nFloats:'
for name in parser.options('floats'):
    string_value = parser.get('floats', name)
    value = parser.getfloat('floats', name)
    print '  %-12s : %-7r -> %0.2f' % (name, string_value, value)
```

```
print '\nBooleans:'
for name in parser.options('booleans'):
    string_value = parser.get('booleans', name)
    value = parser.getboolean('booleans', name)
    print '  %-12s : %-7r -> %s' % (name, string_value, value)
```

执行结果：

```
# python ConfigParser_value_types.py
Integers:
  positive     : '1'     -> 1
  negative     : '-5'    -> -5

Floats:
  positive     : '0.2'   -> 0.20
  negative     : '-3.14' -> -3.14

Booleans:
  number_true  : '1'     -> True
  number_false : '0'     -> False
  yn_true      : 'yes'   -> True
  yn_false     : 'no'    -> False
  tf_true      : 'true'  -> True
  tf_false     : 'false' -> False
  onoff_true   : 'on'    -> True
  onoff_false  : 'false' -> False
```

注意这里的布尔类型和 python 有点差异，no 和 false 认为是 False。

SafeConfigParser 中的参数 allow_no_value 设置为 true 的话，允许选项没有值，默认为 None。配置文件 allow_no_value.ini 如下：

```
[flags]

turn_feature_on
```

ConfigParser_allow_no_value.py 如下：

```
import ConfigParser

# Require values
try:
    parser = ConfigParser.SafeConfigParser()
    parser.read('allow_no_value.ini')
except ConfigParser.ParsingError, err:
    print 'Could not parse:', err
```

```
# Allow stand-alone option names
print '\nTrying again with allow_no_value=True'
parser = ConfigParser.SafeConfigParser(allow_no_value=True)
parser.read('allow_no_value.ini')
for flag in [ 'turn_feature_on', 'turn_other_feature_on' ]:
    print
    print flag
    exists = parser.has_option('flags', flag)
    print '  has_option:', exists
    if exists:
        print '         get:', parser.get('flags', flag)
```

执行结果：

```
# python ConfigParser_allow_no_value.py
Could not parse: File contains parsing errors: allow_no_value.ini
        [line  2]: 'turn_feature_on\n'

Trying again with allow_no_value=True

turn_feature_on
  has_option: True
         get: None

turn_other_feature_on
  has_option: False
```

修改配置

SafeConfigParser 的 add_section 方法可以添加域，set 方法可以添加或修改选项，注意所有选项要以字符串的形式体现，哪怕是其他类型。

```
import ConfigParser

parser = ConfigParser.SafeConfigParser()

parser.add_section('bug_tracker')
parser.set('bug_tracker', 'url', 'http://localhost:8080/bugs')
parser.set('bug_tracker', 'username', 'dhellmann')
parser.set('bug_tracker', 'password', 'secret')

for section in parser.sections():
```

```
    print section
    for name, value in parser.items(section):
        print '  %s = %r' % (name, value)
```

执行结果：

```
# python ConfigParser_populate.py
bug_tracker
  url = 'http://localhost:8080/bugs'
  username = 'dhellmann'
  password = 'secret'
```

同样 SafeConfigParser 的 remove_section 方法可以删除域，remove_option 可以删除选项。删除域会删除其下面的所有选项：

```
from ConfigParser import SafeConfigParser

parser = SafeConfigParser()
parser.read('multisection.ini')

print 'Read values:\n'
for section in parser.sections():
    print section
    for name, value in parser.items(section):
        print '  %s = %r' % (name, value)

parser.remove_option('bug_tracker', 'password')
parser.remove_section('wiki')

print '\nModified values:\n'
for section in parser.sections():
    print section
    for name, value in parser.items(section):
        print '  %s = %r' % (name, value)
```

执行结果：

```
#python ConfigParser_remove.py
Read values:

bug_tracker
  url = 'http://localhost:8080/bugs/'
  username = 'dhellmann'
  password = 'SECRET'
wiki
  url = 'http://localhost:8080/wiki/'
  username = 'dhellmann'
```

```
  password = 'SECRET'

Modified values:

bug_tracker
  url = 'http://localhost:8080/bugs/'
  username = 'dhellmann'
```

保存配置

SafeConfigParser 的 write 方法可以把配置存为文件：

```
import ConfigParser
import sys

parser = ConfigParser.SafeConfigParser()

parser.add_section('bug_tracker')
parser.set('bug_tracker', 'url', 'http://localhost:8080/bugs')
parser.set('bug_tracker', 'username', 'dhellmann')
parser.set('bug_tracker', 'password', 'secret')

parser.write('test.ini')
```

执行结果：

```
# python ConfigParser_write.py
[bug_tracker]
url = http://localhost:8080/bugs
username = dhellmann
password = secret
```

存为实际的文件：

```
import ConfigParser
import sys

parser = ConfigParser.SafeConfigParser()

parser.add_section('bug_tracker')
parser.set('bug_tracker', 'url', 'http://localhost:8080/bugs')
parser.set('bug_tracker', 'username', 'dhellmann')
parser.set('bug_tracker', 'password', 'secret')

f = open('test.ini','w')
```

```
parser.write(f)
```

配置文件可以用 ConfigParser 读取，但是不会加载注释。

选项查找路径

SafeConfigParser 的多级查找流程如下：

选项查找之前先检查域名，如果域名不存在且不是特殊值 DEFAULT 报 NoSectionErroris 异常。

1. 如果选项在 vars 字典中，返回对应值。
2. 如果选项在指定的域中，返回选项的值。
3. 如果选项在 DEFAULT 域中，返回对应的值。
4. 如果选项在传递给构造函数的 defaults 字典中，返回对应的值。如果还是没有找到，报异常 NoOptionError。

建立如下配置文件 with-defaults.ini：

```
[DEFAULT]
file-only = value from DEFAULT section
init-and-file = value from DEFAULT section
from-section = value from DEFAULT section
from-vars = value from DEFAULT section
[sect]
section-only = value from section in file
from-section = value from section in file
from-vars = value from section in file
```

代码如下：

```
import ConfigParser

# Define the names of the options
option_names =  [
    'from-default',
    'from-section', 'section-only',
    'file-only', 'init-only', 'init-and-file',
    'from-vars',
    ]

# Initialize the parser with some defaults
parser = ConfigParser.SafeConfigParser(
    defaults={'from-default':'value from defaults passed to init',
```

```
                'init-only':'value from defaults passed to init',
                'init-and-file':'value from defaults passed to init',
                'from-section':'value from defaults passed to init',
                'from-vars':'value from defaults passed to init',
                })

print 'Defaults before loading file:'
defaults = parser.defaults()
for name in option_names:
    if name in defaults:
        print '  %-15s = %r' % (name, defaults[name])

# Load the configuration file
parser.read('with-defaults.ini')

print '\nDefaults after loading file:'
defaults = parser.defaults()
for name in option_names:
    if name in defaults:
        print '  %-15s = %r' % (name, defaults[name])

# Define some local overrides
vars = {'from-vars':'value from vars'}

# Show the values of all the options
print '\nOption lookup:'
for name in option_names:
    value = parser.get('sect', name, vars=vars)
    print '  %-15s = %r' % (name, value)

# Show error messages for options that do not exist
print '\nError cases:'
try:
    print 'No such option :', parser.get('sect', 'no-option')
except ConfigParser.NoOptionError, err:
    print str(err)

try:
    print 'No such section:', parser.get('no-sect', 'no-option')
except ConfigParser.NoSectionError, err:
    print str(err)
```

执行结果：

```
# python ConfigParser_defaults.py
Defaults before loading file:
  from-default    = 'value from defaults passed to init'
  from-section    = 'value from defaults passed to init'
  init-only       = 'value from defaults passed to init'
  init-and-file   = 'value from defaults passed to init'
  from-vars       = 'value from defaults passed to init'

Defaults after loading file:
  from-default    = 'value from defaults passed to init'
  from-section    = 'value from DEFAULT section'
  file-only       = 'value from DEFAULT section'
  init-only       = 'value from defaults passed to init'
  init-and-file   = 'value from DEFAULT section'
  from-vars       = 'value from DEFAULT section'

Option lookup:
  from-default    = 'value from defaults passed to init'
  from-section    = 'value from section in file'
  section-only    = 'value from section in file'
  file-only       = 'value from DEFAULT section'
  init-only       = 'value from defaults passed to init'
  init-and-file   = 'value from DEFAULT section'
  from-vars       = 'value from vars'

Error cases:
No such option : No option 'no-option' in section: 'sect'
No such section: No section: 'no-sect'
```

变量替换

配置的值可以包含变量，格式：%(name)s，类型为字符串。建立配置文件interpolation.ini：

```
[bug_tracker]
protocol = http
server = localhost
port = 8080
url = %(protocol)s://%(server)s:%(port)s/bugs/
username = dhellmann
password = SECRET
```

代码：

```
from ConfigParser import SafeConfigParser
```

```
parser = SafeConfigParser()
parser.read('interpolation.ini')

print 'Original value       :', parser.get('bug_tracker', 'url')

parser.set('bug_tracker', 'port', '9090')
print 'Altered port value   :', parser.get('bug_tracker', 'url')

print 'Without interpolation:', parser.get('bug_tracker', 'url',
                                          raw=True)
```

执行结果：

```
# python ConfigParser_interpolation.py
Original value       : http://localhost:8080/bugs/
Altered port value   : http://localhost:9090/bugs/
Without interpolation: %(protocol)s://%(server)s:%(port)s/bugs/
```

变量替换也可以配合使用默认值，建立配置文件：

```
[DEFAULT]
url = %(protocol)s://%(server)s:%(port)s/bugs/
protocol = http
server = bugs.example.com
port = 80
[bug_tracker]
server= localhost
port = 8080
username = dhellmann
password = SECRET
```

代码：

```
from ConfigParser import SafeConfigParser

parser = SafeConfigParser()
parser.read('interpolation_defaults.ini')

print 'URL:', parser.get('bug_tracker', 'url')
```

执行结果：

```
# python ConfigParser_interpolation_defaults.py
URL: http://localhost:8080/bugs/
```

如果变量有死循环，会报 InterpolationDepthError 异常：

```
import ConfigParser

parser = ConfigParser.SafeConfigParser()

parser.add_section('sect')
parser.set('sect', 'opt', '%(opt)s')

try:
    print parser.get('sect', 'opt')
except ConfigParser.InterpolationDepthError, err:
    print 'ERROR:', err
```

执行结果：

```
# python ConfigParser_interpolation_error.py

ERROR: Bad value substitution:

        section: [bug_tracker]

        option : url

        key    : server

        rawval : :%(port)s/bugs
```