



ХОЧУ ПОМОЧЬ ПРОЕКТУ

Модуль configparser в Python, парсер конфигураций

mango-office.ru

РЕКЛАМА



Виртуальная АТС Расширенная

2 000 ₽

Узнать больше

Статья / Модуль configparser в Python, парсер конфигураций

Парсер конфигурационных файлов

Модуль configparser реализует базовый язык конфигурации, у, аналогичную той, которая содержится в INI-файлах Microsoft Windows. Его можно использовать в программах на Python, которые могут быть легко настроены конечными пользователями.

Примеры использования: configparser интерпретирует и не записывает значения типа префиксов, используемые в расширенной версии синтаксиса INI реестра Windows.

Смотрите также:

- Модуль shlex, который поддерживает создания Unix-подобных мини-языков, которые могут использоваться в качестве альтернативного формата для файлов конфигурации приложения.
- Модуль json, реализующий подмножество синтаксиса JavaScript, которое также можно использовать для этих целей.

Примеры использования:

Возьмем очень простой файл конфигурации, который выглядит следующим образом:

```
[DEFAULT]
ServerAliveInterval = 45
Compression = yes
CompressionLevel = 9
ForwardX11 = yes

[bitbucket.org]
User = hg

[topsecret.server.com]
Port = 50022
ForwardX11 = no
```

По сути, файл INI состоит из разделов, каждый из которых содержит ключи со значениями. Классы модуля configparser могут читать и записывать такие файлы. Создадим вышеуказанный файл конфигурации программно.

```
>>> import configparser
>>> config = configparser.ConfigParser()
>>> config['DEFAULT'] = {'ServerAliveInterval': '45',
...                    'Compression': 'yes',
...                    'CompressionLevel': '9'}
>>> config['bitbucket.org'] = {}
>>> config['bitbucket.org']['User'] = 'hg'
>>> config['topsecret.server.com'] = {}
>>> topsecret = config['topsecret.server.com']
>>> topsecret['Port'] = '50022'      # mutates the parser
>>> topsecret['ForwardX11'] = 'no'  # same here
>>> config['DEFAULT']['ForwardX11'] = 'yes'
# сохранение конфигурации
>>> with open('example.ini', 'w') as configfile:
...     config.write(configfile)
... 
```

Вверх

Можно заметить, что синтаксический анализатор рассматривает конфигурацию как [словарь](#). Есть различия, но поведение очень близко к тому, что можно ожидать от словаря.

Теперь, прочитаем сохраненный файл конфигурации и изучим данные, которые он содержит.

РЕКЛАМА

ПН

Петербургская
Недвижимость

Квартира

2.9 млн руб.

от

```
>>> configparser.ConfigParser()
>>> config = configparser.ConfigParser()
>>> config.read('config.ini')
>>> config.get('server', 'host')
'localhost'
>>> config.get('server', 'port')
'8080'
>>> config.get('user', 'username')
'admin'
>>> config.get('user', 'password')
'password'
>>> config.get('secret', 'secret_key')
'secret.server.com'
>>> config.get('no_section', 'no_key')
'no'
>>> topsecret['Port']
'50022'
>>> for key in config['bitbucket.org']:
...     print(key)
user
compressionlevel
serveraliveinterval
compression
forwardx11
>>> config['bitbucket.org']['ForwardX11']
'yes'
```

Как можно увидеть из кода выше, API довольно прост. Единственное волшебство включает раздел DEFAULT, который предоставляет значения по умолчанию для всех остальных разделов. Обратите внимание, что ключи в разделах не чувствительны к регистру и хранятся в нижнем регистре.

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Структура файлов конфигураций для модуля configparser](#)
- [Типы данных поддерживаемые ConfigParser](#)
- [Поддержка в configparser методов словарей Python](#)
- [Интерполяция значений INI-файла парсером ConfigParser](#)
- [Значения ключей DEFAULT модуля configparser](#)
- [Настройка ConfigParser](#)
- [Тонкая настройка ConfigParser](#)
- [Класс ConfigParser\(\) модуля configparser](#)
- [Методы объекта ConfigParser](#)
- [Исключения модуля configparser](#)