13.10 读取配置文件¶

问题¶

怎样读取普通.ini格式的配置文件？

解决方案¶

`configparser` 模块能被用来读取配置文件。例如，假设你有如下的配置文件：

```
; config.ini
; Sample configuration file

[installation]
library=%(prefix)s/lib
include=%(prefix)s/include
bin=%(prefix)s/bin
prefix=/usr/local

# Setting related to debug configuration
[debug]
log_errors=true
show_warnings=False

[server]
port: 8080
nworkers: 32
pid-file=/tmp/spam.pid
root=/www/root
signature:
   =================================
   Brought to you by the Python Cookbook
   =================================
```

下面是一个读取和提取其中值的例子：

```
>>> from configparser import ConfigParser
>>> cfg = ConfigParser()
>>> cfg.read('config.ini')
['config.ini']
>>> cfg.sections()
['installation', 'debug', 'server']
>>> cfg.get('installation','library')
'/usr/local/lib'
>>> cfg.getboolean('debug','log_errors')

True
>>> cfg.getint('server','port')
8080
>>> cfg.getint('server','nworkers')
32
>>> print(cfg.get('server','signature'))

\=================================
Brought to you by the Python Cookbook
\=================================
```

```
>>>
```

如果有需要，你还能修改配置并使用 `cfg.write()` 方法将其写回到文件中。例如：

```
>>> cfg.set('server','port','9000')
>>> cfg.set('debug','log_errors','False')
>>> import sys
>>> cfg.write(sys.stdout)
[installation]
library = %(prefix)s/lib
include = %(prefix)s/include
bin = %(prefix)s/bin
prefix = /usr/local

[debug]
log_errors = False
show_warnings = False

[server]
port = 9000
nworkers = 32
pid-file = /tmp/spam.pid
root = /www/root
signature =
        =================================
        Brought to you by the Python Cookbook
        =================================
>>>
```

讨论¶

配置文件作为一种可读性很好的格式，非常适用于存储程序中的配置数据。 在每个配置文件中，配置数据会被分组（比如例子中的“installation”、 “debug” 和 “server”）。 每个分组在其中指定对应的各个变量值。

对于可实现同样功能的配置文件和Python源文件是有很大的不同的。 首先，配置文件的语法要更自由些，下面的赋值语句是等效的：

```
prefix=/usr/local
prefix: /usr/local
```

配置文件中的名字是不区分大小写的。例如：

```
>>> cfg.get('installation','PREFIX')
'/usr/local'
>>> cfg.get('installation','prefix')
'/usr/local'
>>>
```

在解析值的时候， `getboolean()` 方法查找任何可行的值。例如下面都是等价的：

```
log_errors = true
log_errors = TRUE
log_errors = Yes
log_errors = 1
```

或许配置文件和Python代码最大的不同在于，它并不是从上而下的顺序执行。 文件是安装一个整体被读取的。如果碰到了变量替换，它实际上已经被替换完成了。 例如，在下面这个配置中， `prefix` 变量在使用它的变量之前或之后定义都是可以的：

```
[installation]
library=%(prefix)s/lib
include=%(prefix)s/include
bin=%(prefix)s/bin
prefix=/usr/local
```

`ConfigParser` 有个容易被忽视的特性是它能一次读取多个配置文件然后合并成一个配置。 例如，假设一个用户像下面这样构造了他们的配置文件：

```
; ~/.config.ini
[installation]
prefix=/Users/beazley/test

[debug]
log_errors=False
```

读取这个文件，它就能跟之前的配置合并起来。如：

```
>>> # Previously read configuration
>>> cfg.get('installation', 'prefix')
'/usr/local'

>>> # Merge in user-specific configuration
>>> import os
>>> cfg.read(os.path.expanduser('~/.config.ini'))
['/Users/beazley/.config.ini']

>>> cfg.get('installation', 'prefix')
'/Users/beazley/test'
>>> cfg.get('installation', 'library')
'/Users/beazley/test/lib'
>>> cfg.getboolean('debug', 'log_errors')
False
>>>
```

仔细观察下 `prefix` 变量是怎样覆盖其他相关变量的，比如 `library` 的设定值。 产生这种结果的原因是变量的改写采取的是后发制人策略，以最后一个为准。 你可以像下面这样做试验：

```
>>> cfg.get('installation','library')
'/Users/beazley/test/lib'
>>> cfg.set('installation','prefix','/tmp/dir')
>>> cfg.get('installation','library')
'/tmp/dir/lib'
>>>
```

最后还有很重要一点要注意的是Python并不能支持.ini文件在其他程序（比如windows应用程序）中的所有特性。 确保你已经参阅了configparser文档中的语法详情以及支持特性。