

Re: ConfigParser shootout, preliminary entry

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2004-10/3884.html>

From: Skip Montanaro (skip_at_pobox.com)

Date: 10/23/04

Date: Sat, 23 Oct 2004 14:06:24 -0500

To: David Wilson <dw@botanicus.net>

David> Skip Montanaro wrote:

>> ** Both attribute-style and dictionary-style access supported*

David> Can someone please enlighten me as to why this is a good idea?

I find attribute-style access much cleaner, but in some situations you have config bits with non-identifier names.

David> The abuse of `__getitem__` or `__getattr__` (IMHO) should only occur

David> inside transparent object proxies and suchlike -- I see no

David> relation between (key, value) mapping and attributes.

There's no abuse here as far as I can tell. I'm using both the way they were intended. That I have attributes under the covers is irrelevant.

David> I also have trouble with the `__getitem__` idea, although I can see

David> at least that this might make sense if you consider your

David> configuration some sort of dictionary or array.

Which is how I think of them.

>> *I wrote it more as an exercise and to satisfy myself that to deal*

>> *with more general config structures than Windows INI files supports*

>> *you don't have to fall into the XML tarpit. I suspect it's not more*

>> *featureful than Michael Chermiside's example implementation though*

>> *it's somewhat shorter.*

David> IMHO you are blurring the lines between complex data storage and

David> simple INI files. Microsoft's history here was to move to the

David> Windows registry – a concept that could be easily represented in

David> the modern age as an XML database.

Here's where I'm coming from. In the past year I started working at a company that uses XML files as its config file format. INI files wouldn't do, because the information they want to store has more hierarchy than they will accomodate. XML is a disaster because the APIs for it suck for this

sort of thing and because although the file storage is ASCII, you often can't read the contents without a lot of study. Good luck getting people to successfully edit such things by hand. You're thus left having to write a custom GUI editor for each application that uses the damn stuff. Glade or no glade that's just a PITA as far as I'm concerned. I have better things to do with my time.

David> Again, I'll raise the point that this work is being done under
David> the false banner of 'ConfigParser' -- this is the only reason I
David> have come out of idleness, it has nothing to do with
David> ConfigParser.

I see no particular reason to limit configuration file format to the one Microsoft blessed, what 15–20 years ago? If all you want is ConfigParser but don't like its API, write a little shim class to provide you with a simpler API. I don't know what that API is. Me, I want something more than INI files and something easier to work with than XML.

David> It is a new concept entirely. Could we reserve the name
David> 'ConfigParser' for configuration parsers that at least in
David> function or form resemble the original, like any sane developers
David> should.

I suppose. I see no particular reason to limit ideas about what is read or written or how it's manipulated at this point. Nor do I understand why the name matter so much at this point. Nonetheless, I changed the name of my class from "ConfigParser" to "Configuration".

David> If you agree with me on the above, then again I think you should
David> first hammer out exactly what you *need* your new configuration
David> system to accomplish, and then perhaps agree on an unambiguous
David> name for it.

I wrote my Configuration class (note the new name!) based upon my needs (which are based on my recent experience):

- * I prefer attribute–style or dict–style access
- * I need more than one level of sections
- * YAML and XML are too complex
- * I want it to be readable and editable by humans

Note that I don't use Windows. I could care less about the format of actual Bill Gates–style INI files. I imagine for people who do need that stuff the current module and class should suffice, perhaps with some tweaks. Its format is too limiting for me though.

David> In the meantime, AFAICS the only implementation that deserves the
David> name 'ConfigParser' so far is my own -- and I only wrote it to

David> prove this point.

That's fine. I renamed mine. You can have the name "ConfigParser". I'd offer to help you trademark the name but I think it's been in common use too long for that. <wink>

David> As per a recent popularly read weblog entry written by A.M.K.,
David> the standard library is aged and disorganised. The thought of a
David> replacement ConfigParser[2] module that embodies an entirely
David> different configuration storage concept doesn't sound like it is
David> going to help this situation much.

Why not? Like I said, there is a ConfigParser module that does the job now. Keep it and move on to something better. We lived for years with only the getopt module as a way to parse command lines. Now people are starting to use Greg Ward's optparse more and more. The two modules will probably live in harmony in the standard library for years to come. You have backward compatibility and more modern thinking about command line parsing.

David> I think the way forward here is to define the goals of a new
David> configuration framework that supports silliness like magical
David> attribute access and multiple backends (with a first backend
David> being an XML DB or pickle or similar), then additional 'mix in'
David> features such as schemas, then after we have a featurelist, open
David> the shootout again.

You call it magic. I call it Python.

David> Perhaps a PEP or SIG could be opened on the topic, and
David> development could commence in a sane fashion.

You seem to think we're all loonies here.

Skip