Application configuration files [closed]

Asked 16 years, 4 months ago Modified 9 years ago Viewed 23k times

42

Closed. This question is seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. It does not meet Stack Overflow guidelines. It is not currently accepting answers.

We don’t allow questions seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. You can edit the question so it can be answered with facts and citations.

Closed 8 years ago.

Improve this question

OK, so I don't want to start a holy-war here, but we're in the process of trying to consolidate the way we handle our application configuration files and we're struggling to make a decision on the best approach to take. At the moment, every application we distribute is using it's own ad-hoc configuration files, whether it's property files (ini style), XML or JSON (internal use only at the moment!).

Most of our code is Java at the moment, so we've been looking at Apache Commons Config, but we've found it to be quite verbose. We've also looked at XMLBeans, but it seems like a lot of faffing around. I also feel as though I'm being pushed towards XML as a format, but my clients and colleagues are apprehensive about trying something else. I can understand it from the client's perspective, everybody's heard of XML, but at the end of the day, shouldn't be using the right tool for the job?

What formats and libraries are people using in production systems these days, is anyone else trying to avoid the angle bracket tax?

Edit: *really needs to be a cross platform solution: Linux, Windows, Solaris etc. and the choice of library used to interface with configuration files is just as important as the choice of format.*

java xml json cross-platform configuration-files

Share

Improve this question

Follow

edited Dec 14, 2015 at 10:59

asked Aug 15, 2008 at 11:19

ninesided

23.2k ● 16 ● 87 ● 109

upvoted just for saying that XMLBeans "seems like a lot of faffing around" – Cheeso May 19, 2009 at 16:41

15 Answers

Sorted by: Highest score (default)

29

YAML, for the simple reason that it makes for very readable configuration files compared to XML.

XML:

```
<user id="babooey" on="cpu1">
    <firstname>Bob</firstname>
    <lastname>Abooey</lastname>
    <department>adv</department>
    <cell>555-1212</cell>
    <address password="xxxx">ahunter@example1.com</add
    <address password="xxxx">babooey@example2.com</add
</user>
```

YAML:

```
    babooey:
        computer : cpu1
        firstname: Bob
        lastname: Abooey
        cell: 555-1212
        addresses:
            - address: babooey@example1.com
              password: xxxx
            - address: babooey@example2.com
              password: xxxx
```

The examples were taken from this page: http://www.kuro5hin.org/story/2004/10/29/14225/062

Share Improve this answer

Follow

answered Aug 15, 2008 at 14:17

engtech
2,791 ● 3 ● 20 ● 24

7 Kuro5hin cited line and char count as reasons to use YAML over XML. I wonder why he forgot to cite "number of supporting tools and libraries"? YAML: 2, XML: 2,000,000.
– Cheeso May 19, 2009 at 16:48

4 Yaml has many more libraries than 2. yaml.org – engtech Jan 20, 2011 at 22:14

I like YAML, and about 12 languages with third-party libraries are listed at yaml.org. But the only language that ships with YAML support by default seems to be Ruby (since 1.9.2). Are there any others? Adding dependencies can be a hassle.
– nealmcb Mar 21, 2012 at 16:22

8 I'm not sure if it was @cheeso's intent, but I'd see having only 2 libraries to choose between (rather than 2M) as a benefit not a drawback (especially if they got those two right...)
– bacar May 15, 2012 at 14:31

1 @Cheeso If there's one well-written YAML library for the language I'm using, it really doesn't matter that there are 11 XML libraries for the language I'm using. Sure, yours goes up to eleven. But when was the last time you used two XML libraries in one program? (If the answer to that question is not "never", you may want to consider another line of work.)
– Parthian Shot Aug 8, 2014 at 13:47

14

First: This is a really big debate issue, not a quick Q+A.

My favourite right now is to simply include Lua, because

- I can permit things like width=height*(1+1/3)

- I can make custom functions available
- I can forbid anything else. (impossible in, for instance, Python (including pickles.))
- I'll probably want a scripting language somewhere else in the project anyway.

Another option, if there's a lot of data is to use sqlite3, because they're right to claim

- Small.
- Fast.
- Reliable.

Choose any three.

To which I would like to add:

- backups are a snap. (just copy the db file.)
- easier to switch to another db, ODBC, whatever. (than it is from fugly-file)

But again, this is a bigger issue. A "big" answer to this probably involves some kind of feature matrix or list of situations like:

Amount of data, or short runtime

- For large amounts of data, you might want efficient storage, like a db.

- For short runs (often), you might want something that you don't need to do a lot of parsing for, consider something that can be mmap:ed in directly.

What does the configuration relate to?

- Host:
    - I like YAML in /etc. Is that reimplemented in windows?
- User:
    - Do you permit users to edit config with text editor?
    - Should it be centrally manageable? Registry / gconf / remote db?
    - May the user have several different *profiles*?
- Project:
    - File(s) in project directory? (Version control usually follows this model...)

Complexity

- Are there only a few flat values? Consider YAML.
- Is the data nested, or dependent in some way? (This is where it gets interesting.)
- Might it be a desirable feature to permit some form of scripting?

- Templates can be viewed as a kind of configuration files..

Share Improve this answer Follow

answered Aug 15, 2008 at 15:19
Anders Eurenius
4,226 ● 2 ● 25 ● 20

11

XML XML XML XML. We're talking *config files here*. There is no "angle bracket tax" if you're not serializing objects in a performance-intense situation.

Config files must be human readable and human understandable, in addition to machine readable. XML is a good compromise between the two.

If your shop has people that are afraid of that new-fangled XML technology, I feel bad for you.

Share Improve this answer Follow

edited Jun 22, 2009 at 11:41

answered Aug 15, 2008 at 13:45
user1228

1 Not sure if sarcasm, but given that the answer is 9 years old, I guess its not. :) – Roman K Nov 20, 2017 at 7:35

1 It was a different times back then... – THIS USER NEEDS HELP Mar 28, 2018 at 21:21

10

Without starting a new holy war, the sentiments of the 'angle bracket tax' post is one area where I *majorly disagree* with Jeff. There's nothing wrong with XML, it's reasonably human readable (as much as YAML or JSON or INI files are) but remember its intent is to be read by machines. Most language/framework combos come with an XML parser of some sort for free which makes XML a pretty good choice.

Also, if you're using a good IDE like Visual Studio, and if the XML comes with a schema, you can give the schema to VS and magically you get intellisense (you can get one for NHibernate for example).

Ulimately you need to think about how often you're going to be touching these files once in production, probably not that often.

This still says it all for me about XML and why it's still a valid choice for config files (from Tim Bray):

"If you want to provide general-purpose data that the receiver might want to do unforeseen weird and crazy things with, or if you want to be really paranoid and picky about i18n, or if what you're sending is more like a document than a struct, or if the order of the data matters, or if the data is potentially long-lived (as in, more than seconds) XML is the way to go. It also seems to me that the combination of XML and XPath hits a sweet spot for data formats that need to be extensible; that is to say, it's pretty easy to write XML-processing code that won't fail in

the presence of changes to the message format that don’t touch the piece you care about."

Share Improve this answer Follow

answered Aug 15, 2008 at 12:24

Kev
120k ● 53 ● 305 ● 390

5

@Guy

But application config isn't always just key/value pairs. Look at something like the tomcat configuration for what ports it listens on. Here's an example:

```
    <Connector port="80" maxHttpHeaderSize="8192"
           maxThreads="150" minSpareThreads="25" maxSp
           enableLookups="false" redirectPort="8443" a
           connectionTimeout="20000" disableUploadTime


    <Connector port="8009"
           enableLookups="false" redirectPort="8443" p
```

You can have any number of connectors. Define more in the file and more connectors exist. Don't define any more and no more exist. There's no good way (imho) to do that with plain old key/value pairs.

If your app's config is simple, then something simple like an INI file that's read into a dictionary is probably fine. But for something more complex like server configuration, an INI file would be a huge pain to maintain, and something more structural like XML or YAML would be better. It all depends on the problem set.

Share Improve this answer Follow

answered Aug 15, 2008 at 20:26

Herms
38.7k ● 13 ● 79 ● 104

2

We are using ini style config files. We use the Nini library to manage them. Nini makes it very easy to use. Nini was orignally for .NET but it has been ported to other platforms using Mono.

Share Improve this answer Follow

answered Aug 15, 2008 at 12:41

Clint Davis
451 ● 1 ● 12 ● 29

1 ini isn't defined by any standard, and has some limitations for being only a key/value store – CharlesB Mar 12, 2013 at 9:12

2

XML, JSON, INI.
They all have their strengths and weaknesses.
In an application context, I feel that the abstraction layer is the important thing.
If you can choose a way to structure the data that is a good middle ground between human readability and how you want to access/abstract the data in code, you're golden.

We mostly use XML where I work, and I cant really believe that a configuration file loaded into a cache as objects when first read or after it has been written to, and then abstracted away from the rest of the program, really is that much of a hit on neither CPU nor disk space.
And it is pretty readable too, as long as you structure the file right.

And all languages on all platforms supports XML through some pretty common libraries.

Share Improve this answer Follow

answered Aug 15, 2008 at 14:26

Lars Mæhlum
6,102 ● 3 ● 29 ● 32

1

@Herms

What I really meant was to stick to the recommended way software should store configuration values for any given platform.

What you often get then is also the recommended ways these should/can be modified. Like a configuration menu in a program or a configuration panel in a "system prefs" application (for system services softwares ie). Not letting the end users modify them directly via RegEdit or NotePad...

Why?

1. The end users (=customers) are used to their platforms
2. System for backups can better save "safe setups" etc

@ninesided

About " *choice of library* ", try to link in (static link) any selected library to lower the risk of getting into a version-conflict-war on end users machines.

Share Improve this answer Follow

edited Aug 15, 2008 at 14:09

answered Aug 15, 2008 at 13:58

epatel
46k ● 17 ● 111 ● 144

The argument for this way would be much stronger if we could find some libraries which offer a unified API to various config file formats. – hippietrail Nov 5, 2012 at 8:03

1

If your configuration file is write-once, read-only-at-bootup, and your data is a bunch of name value pairs, your best choice is the one your developer can get working first.

If your data is a bit more complicated, with nesting etc, you are probably better off with YAML, XML, or SQLite.

If you need nested data and/or the ability to query the configuration data after bootup, use XML or SQLite. Both have pretty good query languages (XPATH and SQL) for structured/nested data.

If your configuration data is highly normalized (e.g. 5th normal form) you are better off with SQLite because SQL is better for dealing with highly normalized data.

If you are planning to write to the configuration data set during program operation, then you are better off going with SQLite. For example, if you are downloading configuration data from another computer, or if you are basing future program execution decisions on data collected in previous program execution. SQLite implements a very robust data storage engine that is extremely difficult to corrupt when you have power outages or programs that are hung in an inconsistent state due to errors. Corruptible data leads to high field support costs, and SQLite will do much better than any home-grown solution or even popular libraries around XML or YAML.

Check out my page for more information on SQLite.

Share Improve this answer

Follow

edited Mar 12, 2013 at 9:13

CharlesB

90k ● 29 ● 200 ● 227

answered May 19, 2009 at 16:40

Jay Godse

0

As far as I know, the Windows registry is no longer the preferred way of storing configuration if you are using .NET - most applications now make use of System.Configuration [1, 2]. Since this is also XML based it seems to be that everything is moving in the direction of using XML for configuration.

If you want to stay cross-platform I would say that using some sort of a text file would be the best route to go. As for the formatting of said file, you might want to take into account if a human is going to be manipulating it or not. XML seems to be a bit more friendly to manual manipulation than INI files due to the visible structure of the file.

As for the angle bracket tax - I don't worry about it too often as the XML libraries take care of abstracting it. The only time it might be a consideration is if you have very little storage space to work with and every byte counts.

[1] System.Configuration Namespace - http://msdn.microsoft.com/en-us/library/system.configuration.aspx

[2] Using Application Configuration Files in .NET - http://www.developer.com/net/net/article.php/3396111

Share Improve this answer Follow

answered Aug 15, 2008 at 12:04

rjzii
14.5k ● 12 ● 81 ● 122

0

We are using properties files, simply because Java supports them natively. A couple of months ago I saw that SpringSource Application Platform uses JSON to configure their server and it looks very interesting. I compared various configuration notations and came to the conclusion that XML seems to be the best fit at the moment. It has nice tools support and is rather platform independent.

Share Improve this answer Follow

answered Aug 15, 2008 at 12:32

dlinsin
19.6k ● 13 ● 43 ● 53

0

Re: epatel's comment

I think the original question was asking about application configuration that an admin would be doing, not just storing user preferences. The suggestions you gave seem more for user prefs than application config, and aren't usually something that the user would ever deal with directly (the app should provide the configuration options in the UI, and then update the files). I really hope you'd never make the user have to view/edit the Registry. :)

As for the actual question, I'd say XML is probably OK, as plenty of people will be used to using that for configuration. As long as you organize the configuration values in an easy to use manner then the "angle bracket tax" shouldn't be too bad.

Share Improve this answer

Follow

answered Aug 15, 2008 at 13:18

Herms

38.7k ● 13 ● 79 ● 104

0

Maybe a bit of a tangent here but my opinion is that the config file should be read into a key value dictionary/hash table when the app first starts up and always accessed via this object from then on for speed. Typically the key/value table starts off as string to string but helper functions in the object do things such DateTime GetConfigDate(string key) etc...

Share Improve this answer

Follow

answered Aug 15, 2008 at 15:23

Guy

0

I think the only important thing is to choose a format that you prefer and can navigate quickly. XML and JSON are both fine formats for configs and are widely supported--technical implementation isn't at the crux of the issue, methinks. It's 100% about what makes the task of config files easier for you.

I have started using JSON, because I work quite a bit with it as a data transport format, and the serializers make it easy to load into any development framework. I find JSON easier to read than XML, which makes handling multiple services, each using a config file that is modified quite frequently, that much easer for me!

Share Improve this answer

Follow

answered Nov 13, 2012 at 17:54

AnthonyAlmighty
135 ● 7

-3

What platform are you working on? I'd recommend trying to use the preferred/common method for it.

1. MacOSX - plists
2. Win32 - Registry (or are there a new one here, long since I developed on it)
3. Linux/Unix - ~/.apprc (name-value perhaps)

Share Improve this answer

Follow

answered Aug 15, 2008 at 11:27

epatel
46k ● 17 ● 111 ● 144

The question now specifies "cross platform". – hippietrail Nov 5, 2012 at 8:05