XML or own-format file?

Asked 16 years ago Modified 16 years ago Viewed 1k times

2

When is a good idea to save information in a XML file and when in a own-format file?

For XML (or other standard) I see:

- (+) Standard format.
- (-) It's tedious to hand modify.

For own-format files I see:

- (-) We need to build a own-parser (non-standard).
- (+) It can be easy to hand modify the files.

xml file

Share

Improve this question

Follow

edited Dec 10, 2008 at 23:46

asked Dec 8, 2008 at 11:34

FerranB

36.7k 18 69 86

By "own format" do you mean JSON or YAML or some totally non-standard format? – S.Lott Dec 8, 2008 at 11:36

1 Can you give some more information what the file format is good for? And who are the primary editors, humans or programs? Is it configuration? A narrative document? Data storage? – Torsten Marek Dec 8, 2008 at 13:04

12 Answers

Sorted by: Highest score (default)

12

Use XML when it's a good fit in various ways:

- Need to share between different applications which are all capable of handling XML
- Natural tree-like structure
- Primarily data easily represented as text (binary data is a bit of a kludge in text-based formats)
- Extensibility is important
- Performance isn't critical (parsing XML isn't exactly blazingly fast - although if performance is important and you go for XML, shop around for a fast parser, as there's a wide difference between fastest and slowest)
- Schema can be pre-defined and documents can be verified against it
- Simpler formats (e.g. name=value pairs) don't cut it

Basically if there's a pretty natural representation of your data model in XML, that may well be the easiest way of

handling it. If you'd end up having to mess around a lot to fit it in with XML, think about other formats. Note that there are plenty of other standard (or "somewhat standard" - e.g. supported by tools on multiple platforms) formats available beyond just XML.

Share Improve this answer

Follow

answered Dec 8, 2008 at 11:39

Jon Skeet

1.5m ● 889 ● 9.3k ● 9.3k

6

For XML I see:

- (+) Standard format.
- (-) It's tedious to hand modify.

  I only use XML when the API requires it.

For JSON/YAML I see:

- (+) Standard format.
- (+) It's easy to hand-modify.

  I use JSON/YAML for almost everything. Except when an interface requires something else.

For CSV I see:

- (+) Standard format.
- (+) It's easy to hand-modify.
- (-) It's a little murky when the column names are screwy or data isn't in simple first-nromal form.

I use CSV whenever possible.

For Language Serializers I see:

- (+) Standard format for the given language.
- (-) nearly impossible to hand-modify.

  I use serialized files once in a while to pass data among processes when I'm sure both sides are in the same language.

For own-format files I see:

- (-) We need to build a own-parser (non-standard).
- (+) It can be easy to hand modify the files.

  I avoid inventing my own file format. Haven't invented my own file format in years.

Share Improve this answer

Follow

edited Dec 8, 2008 at 16:49

answered Dec 8, 2008 at 16:17

S.Lott

391k ● 82 ● 517 ● 788

3

XML gives you the power of XSLT and Xpath, your own format does not.

Share Improve this answer

Follow

answered Dec 8, 2008 at 16:20

annakata

75.7k ● 18 ● 115 ● 180

2

For a discussion of pros and cons, see [before xml became a standard and given all its shortcomings what made xml so popular](#).

Share Improve this answer

Follow

edited May 23, 2017 at 11:55

Community Bot

1 ● 1

answered Dec 8, 2008 at 11:40

gimel

86.1k ● 10 ● 79 ● 104

1

Also remember you have all sort of nifty XML editors, that with the schemas will give you autocompletion, syntax checking and all sorts of modern editing perks, that other formats don't fully support

Share Improve this answer

Follow

answered Dec 8, 2008 at 11:43

Robert Gould

69.7k ● 61 ● 191 ● 275

1

> (-) It's tedious to hand modify.

I think that depends strongly on the XML/own format that you define. If you use e.g. a binary format (which might

be very efficient to do) it will nearly be impossible to manually edit the file.

I think that there are further aspects influencing the choice of a file format such as

- performance
- interoperability with other components
- capability to manually edit files (debugging)
- backward-compatibility issues
- etc

If you are going to use a textual format I would choose an XML-based solution in most of the cases.

Share Improve this answer

Follow

answered Dec 8, 2008 at 11:44

Dirk Vollmar

176k ● 53 ● 259 ● 316

1

My rule of thumb is: if I'm going to need to transform or validate it, or I'm going to need to share the data with application domains that I don't control, I consider XML first, and if I'm not, I don't.

Edit:

I forgot about text in general, and Unicode in particular: If a significant portion of my data is text (especially marked-up text), and if I need to support Unicode (which any

application working with blocks of text generally does), that moves XML up the list in a hurry.

Share Improve this answer

Follow

answered Dec 8, 2008 at 19:15

Robert Rossney
96.6k • 24 • 148 • 218

0

The ease of editing isn't a major issue, as pointed out above: there are lot of good (and free for some) XML editors around.

Another potential issue is verbosity, although the answer for large files is to gzip them: in lot of languages, it is nearly transparent.

XML is good in a number of ways: the standard is well defined (you don't have to think how to define charset, how to escape stuff, how to handle special cases (multi-lines, binary, etc.)); it has lot of tools (editors, parsers, XPath, etc.); it is great to exchange data with other tools.

If your needs are very simple, manipulating only Ascii, self-sufficient (only this app will use this format), maybe you can go with another format. But before defining your own, you might take a look at existing text-based formats, like Json, Yaml, even Lua (was a data description language at the origin) or for very simple needs, Windows' ini format or Java's properties.

Share Improve this answer

Follow

answered Dec 8, 2008 at 12:15

PhiLho
41.1k • 6 • 99 • 136

0

By order I use :

- property file if the data can be represented as key/value
- CSV if the data can be represented as a table
- XML if complex structure

For cons of XML in my opinion can be the performance of the parser, and the size of XML file when data is important can be a handicap (XML files of several MB are hard to open in many editors)

Share Improve this answer

Follow

answered Dec 8, 2008 at 16:28

Vinze
2,539 • 3 • 22 • 23

0

As annakata stated, you can use XSLT and XPATH if you choose the XML route. I've found that with some clever use of XSLT you can create "self-documenting" configuration files.

By creating an .xsl file and adding a declaration such as this to the XML file, a user can simply double-click the XML file and view the results of the transformation in their browser (I know IE and Firefox both support this)

```
<?xml-stylesheet type="text/xsl" href="config-document
```

Just thought that might be helpful.

Share Improve this answer

Follow

edited May 23, 2017 at 12:04

Community Bot

1 • 1

answered Dec 8, 2008 at 19:20

Rich

896 • 6 • 15

0

XML is usually my first choice. Part of it is because it is the standard configuration file format for my platform choice (.NET). I have found that, almost exclusively, a well-defined XML file is better than a customized format. I will shy away from CSV and flat files, as well, unless they are a requirement of the project.

My reasons for XML as my choice (note that some are platform specific):

- **Standard implementation for my platform.** Plenty of tooling available to work with XML, XSD, XSLT.
- **Schema enforcement (XSD).** Allows me to enforce the file structure. Very helpful when the format is consumed by others, as well.
- **Navigation (XPath, Linq to Xml).** Easy to extract and write nodes and their attributes. Less risk in

writing this type of code over customer readers and writers.

- **Transformable (XSLT).** Can convert the file to other presentational views with little effort.
- **Interoperable.** The structure of XML is a natural fit for describing objects. Objects serialized into XML is easily portable and can survive across application boundaries.
- **Easily Editable.** A well-defined XML is easy to read and easy to edit. A simple text editor is enough to get started, and there are many XML editing tools available with a variety of features and price points.

I don't understand the perception that XML would be any less easy to modify by hand than a custom format. XML might be more verbose than a format that you come up with, but it provides contextual relevance to the data it contains. If you can look at (well-formed) XHTML, it is not much different when you look at XML.

answered Dec 8, 2008 at 19:34

Joseph Ferris

12.7k ● 3 ● 48 ● 73

0

It really depends on your data.

See ESR's The Art of Unix Programming: Ch. 5 Textuality - Data File Metaformats. This quote about sums it up:

> XML can be a simplifying choice or a complicating one. There is a lot of hype surrounding it, but don't become a fashion victim by either adopting or rejecting it uncritically. Choose carefully and bear the KISS principle in mind.

XML certainly has its uses, and it is **wonderful** for expressing complicated hierarchical datasets, but it's overkill if all you need to do is store half a dozen `key:value` pairs, and inappropriate for row-based tabular data.

Share Improve this answer

Follow

answered Dec 8, 2008 at 20:05

Adam Jaskiewicz

11k ● 3 ● 36 ● 37