Is XSLT worth it? [closed]

Asked 16 years, 3 months ago Modified 3 years, 4 months ago

Viewed 35k times

116

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 12 years ago.

A while ago, I started on a project where I designed a html-esque XML schema so that authors could write their content (educational course material) in a simplified format which would then be transformed into HTML via XSLT. I played around (struggled) with it for a while and got it to a very basic level but then was too annoyed by the limitations I was encountering (which may well have been limitations of my knowledge) and when I read a blog suggesting to ditch XSLT and just write your own XML-to-whatever parser in your language of choice, I eagerly jumped onto that and it's worked out brilliantly.

I'm still working on it to this day (*I'm actually supposed to be working on it right now, instead of playing on SO*), and

I am seeing more and more things which make me think that the decision to ditch XSLT was a good one.

I know that XSLT has its place, in that it is an accepted standard, and that if everyone is writing their own interpreters, 90% of them will end up on TheDailyWTF. But given that it is a functional style language instead of the procedural style which most programmers are familiar with, for someone embarking on a project such as my own, **would you recommend they go down the path that I did, or stick it out with XSLT**?

html **xml** **xslt** **xml-parsing**

Share

Improve this question

Follow

edited Aug 20, 2021 at 23:56

community wiki
6 revs, 3 users 64%
nickf

1 I think there is a severe disconnect between the subject of your question (which is argumentative) and the actual question that you ask (namely, whether SO readers actually use XSLT, or recommend using it). It's also unclear why you need this question to be anwered. – Martin v. Löwis Aug 13, 2009 at 7:42

3 @Martin, what would you suggest as a title? I don't NEED this question to be answered, but I think it's interesting, and also useful for someone who's trying to decide whether to

invest in XSLT or an alternative. – Benjol Aug 13, 2009 at 8:01

7 I think XSLT hast reached the plateau of productivity within the hype cycle (en.wikipedia.org/wiki/Hype_cycle).
– Dirk Vollmar Aug 13, 2009 at 8:30

I personally feel like my XML is not adding any value until I have run it through at least 1 or 2 transformations.
– user74754 Jun 20, 2012 at 0:26

@Martinv.Löwis, Agree with your assesment. Also this really comes down to enterprise concerns, meaning if the same guy does it all, and the method is start-up.... fine get it done fastest implementation style, your only screwing yourself in that case anyway. XSLT is fairly difficult until it clicks, requires domain specific knowledge, but in a large organization.... O my god you realize how wrong all the anti-XML people are. AND also, once you know XSLT, it's the best choice, it only seems otherwise when you don't know XSLT, so you factor in the learning investment. – J. M. Becker Dec 23, 2015 at 23:08

41 Answers

Sorted by: Highest score (default)

1 2 Next

93

So much negativity!

I've been using XSLT for a good few years now, and genuinely love it. The key thing you have to realise is that *it's not a programming language it's a templating language* (and in this respect I find it indescribably superior to asp.net /spit).

XML is the de facto data format of web development today, be it config files, raw data or in memory reprsentation. XSLT and XPath give you an *enormously* powerful and very efficient way to transform that data into any output format you might like, instantly giving you that MVC aspect of separating the presentation from the data.

Then there's the utility abilities: washing out namespaces, recognising disparate schema definitions, merging documents.

It *must* be better to deal with XSLT than developing your own in-house methods. At least XSLT is a standard and something you could hire for, and if it's ever really a problem for your team it's very nature would let you keep most of your team working with just XML.

A real world use case: I just wrote an app which handles in-memory XML docs throughout the system, and transforms to JSON, HTML, or XML as requested by the end user. I had a fairly random request to provide as Excel data. A former colleague had done something similar programatically but it required a module of a few class files and that the server had MS Office installed! Turns out Excel has an XSD: new functionality with minimum basecode impact in 3 hours.

Personally I think it's one of the cleanest things I've encountered in my career, and I believe all of it's apparent issues (debugging, string manipulation, programming structures) are down to a flawed understanding of the tool.

Obviously, I strongly believe it is "worth it".

Share Improve this answer Follow

answered Nov 19, 2008 at 10:27

community wiki
annakata

8 A small addition to your point about debugging. The latest versions of Visual Studio allow you to debug directly within XSL files. Setting breakpoints, inspecting etc. – Craig Bovis May 20, 2009 at 14:10

This is such a good answer, especially the refreshing story of the excel xsd! – Laguna Jan 10, 2012 at 14:23

1 @annakata, can you provide a link to a msdn article or some tutorial on how to do the excel thing? i think that might be something that i can use for my project too. thx! – Laguna Jan 10, 2012 at 14:35

8 JSON and JAML are superior **data** formats than XML. XML in its core is **markup** language; and it's very unfortunate that it's been so widely misused for structured data representation. – ulidtko Jul 31, 2013 at 13:20

3 @ulidtko, Working as a Systems Engineer, I've seen plenty of very inappropriate JSON as markup... I only expect more to come, and it makes XML look awesome in comparison. – J. M. Becker May 27, 2015 at 21:46

66

Advantages of XSLT:

- Domain-specific to XML, so for example no need to quote literal XML in the output.




- Supports XPath/XQuery, which can be a nice way to query DOMs, in the same way that regular expressions can be a nice way to query strings.
- Functional language.

Disadvantages of XSLT:

- Can be obscenely verbose - you don't have to quote literal XML, which effectively means you do have to quote code. And not in a pretty way. But then again, it's not much worse than your typical SSI.
- Doesn't do certain things which most programmers take for granted. For instance string manipulation can be a chore. This can lead to "unfortunate moments" when novices design code, then frantically search the web for hints how to implement functions they assumed would just be there and didn't give themselves time to write.
- Functional language.

One way to get procedural behaviour, by the way, is to chain multiple transforms together. After each step you have a brand new DOM to work on which reflects the changes in that step. Some XSL processors have extensions to effectively do this in one transform, but I forget the details.

So, if your code is mostly output and not much logic, XSLT can be a very neat way to express it. If there is a lot of logic, but mostly of forms which are built in to XSLT (select all elements which look like blah, and for each one

output blah), it's likely to be quite a friendly environment. If you fancy thinking XML-ishly at all times, then give XSLT 2 a go.

Otherwise, I'd say that if your favourite programming language has a good DOM implementation supporting XPath and allowing you to build documents in a useful way, then there are few benefits to using XSLT. Bindings to libxml2 and gdome2 should do nicely, and there's no shame in sticking to general-purpose languages you know well.

Home-grown XML parsers are usually either incomplete (in which case you'll come unstuck some day) or else not much smaller than something you could have got off the shelf (in which case you're probably wasting your time), and give you any number of opportunities to introduce severe security issues around malicious input. Don't write one unless you know exactly what you gain by doing it. Which is not to say you can't write a parser for something simpler than XML as your input format, if you don't need everything that XML offers.

answered Sep 17, 2008 at 1:26

community wiki
Steve Jessop

3 XSLT isn't functional, it's declarative (like SQL). – jmah Oct 10, 2008 at 12:23

An XSL Template seems to me to have all the criteria of a pure function, what disqualifies it from being described as functional? Why is 'declaritive' an alternative? a = 1; is declaritive. – AnthonyWJones Oct 10, 2008 at 14:32

It's declarative like Prolog. en.wikipedia.org/wiki/Declarative_programming – Loki Astari Nov 5, 2008 at 0:08

8 I believe functional programming is a type of declarative programming. – Zifre May 8, 2009 at 23:24

1 While the point about XSLT 2.0 is a good one, even as of the time I am writing there is not widespread support for XSLT 2.0. – PeterAllenWebb May 15, 2014 at 15:36

28

I have to admit a bias here because I teach XSLT for a living. But, it might be worth covering off the areas that I see my students working in. They split into three groups generally: publishing, banking and web.

Many of the answers so far could be summarised as "it's no good for creating websites" or "it's nothing like language X". Many tech folks go through their careers with no exposure to functional/declarative languages. When I'm teaching, the experienced Java/VB/C/etc folk are the ones who have issues with the language (variables are variables in the sense of algebra not procedural programming for example). That's many of the people answering here - I've never gotten on with Java but I'm not going to bother to critique the language because of that.

In many circumstances it is an inappropriate tool for creating websites - a general purpose programming language may be better. I often need to take very large XML documents and present them on the web; XSLT makes that trivial. The students I see in this space tend to be processing data sets and presenting them on the web. XSLT is certainly not the only applicable tool in this space. However, many of them are using the DOM to do this and XSLT is certainly less painful.

The banking students I see use a DataPower box in general. This is an XML appliance and it's used to sit between services 'speaking' different XML dialects. Transformation from one XML language to another is almost trivial in XSLT and the number of students attending my courses on this are increasing.

The final set of students I see come from a publishing background (like me). These people tend to have immense documents in XML (believe me, publishing as an industry is getting very into XML - technical publishing has been there for years and trade publishing is getting there now). These documents need to be processing (DocBook to ePub comes to mind here).

Someone above commented that scripts tend to be below 60 lines or they become unwieldy. If it does become unwieldy, the odds are the coder hasn't really got the idea - XSLT is a very different mindset from many other languages. If you don't get the mindset it won't work.

It's certainly not a dying language (the amount of work I get tells me that). Right now, it's a bit 'stuck' until Microsoft finish their (very late) implementation of XSLT 2. But it's still there and seems to be going strong from my viewpoint.

Share Improve this answer Follow

answered Aug 13, 2009 at 17:14

community wiki
Nic Gibson

I am a Java developer and I also love XML and XSLT. I wish people would realize the power of these. – Nikolas Dec 23, 2019 at 23:50

24

We use XSLT extensively for things like documentation, and making some complex configuration settings user-serviceable.

For documentation, we use a lot of DocBook, which is an XML-based format. This lets us store and manage our documentation with all of our source code, since the files are plain text. With XSLT, we can easily build our own documentation formats, allowing us to both autogenerate the content in a generic way, and make the content more readable. For example, when we publish release notes, we can create XML that looks something like:

```
<ReleaseNotes>
    <FixedBugs>
        <Bug id="123" component="Admin">Error when cli
button</Bug>
        <Bug id="125" component="Core">Crash at startu
missing</Bug>
        <Bug id="127" component="Admin">Error when cli
button</Bug>
    </FixedBugs>
</ReleaseNotes>
```

And then using XSLT (which transforms the above to DocBook) we end up with nice release notes (PDF or HTML usually) where bug IDs are automatically linked to our bug tracker, bugs are grouped by component, and the format of everything is perfectly consistent. And the above XML can be generated automatically by querying our bug tracker for what has changed between versions.

The other place where we have found XSLT to be useful is actually in our core product. Sometimes when interfacing with third-party systems we need to somehow process data in a complex HTML page. Parsing HTML is ugly, so we feed the data through something like TagSoup (which generates proper SAX XML events, essentially letting us deal with the HTML as if it were properly written XML) and then we can run some XSLT against it, to turn the data into a "known stable" format that we can actually work with. By separating out that transformation into an XSLT file, that means that if and when the HTML format changes, the application itself does not need to be upgraded, instead the end-user can just edit the XSLT file

themselves, or we can e-mail them an updated XSLT file without the entire system needing to be upgraded.

I would say that for web projects, there are better ways to handle the view side than XSLT today, but as a technology there are definitely uses for XSLT. It's not the easiest language in the world to use, but it is definitely not dead, and from my perspective still has lots of good uses.

Share Improve this answer Follow

answered Aug 13, 2009 at 8:02

community wiki
Adam Batkin

Thanks, that's a nice answer with a concrete example. – Benjol Aug 13, 2009 at 8:16

And yet someone felt the need to vote it down without even leaving a comment as to what was wrong with my answer – Adam Batkin Aug 13, 2009 at 8:23

Probably because they didn't agree... – Benjol Aug 13, 2009 at 11:06

There was another program similar to TagSoup that also creates a proper XML tree from HTML... but I can't remember the name. Does anybody know it? – erjiang Aug 13, 2009 at 12:30

Tidy is a nice program for this purpose – Erlock Jan 26, 2010 at 13:13

19

XSLT is an example of a declarative programming language.

Other examples of declarative programming languages include regular expressions, Prolog, and SQL. All of these are highly expressive and compact, and usually very well designed and powerful for the task for which they are designed.

However, software developers generally hate such languages, because they are so different from more mainstream OO or procedural languages that they're hard to learn and debug. Their compact nature generally makes it very easy to do a lot of damage inadvertently.

So while XSLT is an efficient mechanism to merge data into presentation, it fails in the ease-of-use department. I believe that's why it hasn't really caught on.

Share Improve this answer Follow

answered Aug 13, 2009 at 7:47

community wiki
Bill Karwin

2 XSLT is functional, but I think it's debatable whether it is declarative (there are ordering dependancies etc). However, I agree with your point that this (whether functional or declarative) is both a powerful thing, as well as a source of frustration for most OO/ procedural programmers. However, in XSLT's case I believe that, as a functional language, it lacks too many of the features that make most functional

languages usable. As a result you typically end up writing a lot *more* code, rather than it being compact. Have you tried splitting strings in XSLT (1.0), for example? – philsquared Aug 13, 2009 at 7:55

3 XSLT is not functional, by the way - it doesn't have functions as first-class values. Yes, there are hacks (FXSL), but they are just that, and you still don't get variable capture with them (so no lambdas). XSLT is pure (side-effect free), yes, but this doesn't have to mean "functional". – Pavel Minaev Aug 13, 2009 at 8:11

1 XSLT is horrible perversion of a declarative programming language and PLs in general. Even INTERCAL is more coherent and for some use-cases about as productive. Yes, certain tree transformations are straightforward in XSLT, but I found a combination of XPath and a (pseudo-) functional traditional language to be much easier to write and much much much easier to read and understand. – pmf Aug 13, 2009 at 11:16

23 @Jeff Atwood: As with regex, XSLT's beauty is in the concept, not in the syntax. To those that can't read them, regexes are a giant mishmash of meaningless letters and symbols. It's the *mindset* behind regex that makes them beautiful. – Tomalak Aug 13, 2009 at 12:27

6 @Jeff Atwood Why do you write such categorical statements about an area you obviously don't know? XSLT and XPath do have good RegEx capabilities and some of these have been used in answers to questions on SO. I have written more than one parsers using RegEx in XSLT, for the lexer. The most complicated parser was for XPath 2.0. Writing without reading first -- as in the Chukche joke :)
– Dimitre Novatchev Aug 16, 2009 at 16:46

12

I remember all the hype around XSLT when the standard was newly released. All the excitement around being able built an entire HTML UI with a 'simple' transform.

Let’s face it, it is hard to use, near impossible to debug, often unbearably slow. The end result is nearly always quirky and less than ideal.

I will sooner gnaw off my own leg than use an XSLT while there are better ways to do things. Still it has its places, its good for simple transform tasks.

Share Improve this answer Follow

answered Sep 17, 2008 at 1:03

community wiki
Crusty

1 unbearably slow?? Compared with what? – AnthonyWJones Oct 10, 2008 at 14:27

Compared to hand-writing the conversion in VB6 as I did. Orders of magnitude faster than doing XSLT (I was converting ADO Recordsets into HTML - back in 2002 or something :-) – endian Oct 23, 2008 at 7:34

3 It is a lot easier to debug using tools like Oxygen than you would expect. – Andy Dent Mar 22, 2009 at 8:00

I've used XSLT (and also XQuery) extensively for various things - to generate C++ code as part of build process, to

10

produce documentation from doc comments, and within an application that had to work with XML in general and XHTML in particular a lot. The code generator in particular was in excess of 10,000 lines of XSLT 2.0 code spread around about a dozen separate files (it did a lot of things - headers for clients, remoting proxies/stubs, COM wrappers, .NET wrappers, ORM - to name a few). I inherited it over another guy who didn't really understand the language well, and the older bits were consequently quite a mess. Newer stuff that we wrote was mostly kept sane and readable, however, and I do not recall any particular problems with achieving that. It was certainly not any harder than doing it for C++.

Speaking of versions, dealing with XSLT 2.0 definitely helps keep you sane, but 1.0 is still alright for simpler transforms. In its niche, it is an extremely handy tool, and the productivity you get from certain domain-specific features (most importantly, dynamic dispatch via template matching) is hard to match. Despite the perceived wordiness of XSLT's XML-based syntax, the same thing in LINQ to XML (even in VB with XML literals) was usually several times longer. Quite often, however, it gets undeserved flack because of unnecessary use of XML in some case in the first place.

To sum it up: it is an incredibly useful tool to have in one's toolbox, but it is a very specialized one, so it is good so long as you use it properly and for its intended purpose. I really wish there was a proper, native .NET implementation of XSLT 2.0.

Share Improve this answer Follow

answered Aug 13, 2009 at 8:10

community wiki
Pavel Minaev

9

I use XSLT (for lack of better alternative), but not for presentation, just for transformation:

1. I write short XSLT transformations to do mass edits on our maven pom.xml files.
2. I've written a pipeline of transformations to generate XML Schemas from XMI (UML Diagram). It worked for a while, but it finally got too complex and we had to take it out behind the barn.
3. I've used transformations to refactor XML Schemas.
4. I've worked around some limitations in XSLT by using it to generate an XSLT to do the real work. (Ever tried to write an XSLT that produces an output using namespaces that aren't known until runtime?)

I keep coming back to it because it does a better job round-tripping the XML it's processing than other approaches I've tried, which have seemed needlessly lossy or simply misunderstand XML. XSLT is unpleasant, but I find using Oxygen makes it bearable.

That said, I'm investigating using Clojure (a lisp) to perform transformations of XML, but I haven't gotten far

enough yet to know if that approach will bring me benefits.

Share Improve this answer Follow

answered Aug 13, 2009 at 8:01

community wiki
bendin

> XSLT has saved me from writting POM modifacations in hackish shell scripts. I've come to terms with XML, it's bad.... but its better than anything else for markup. XSLT, it's bad, but it's the BEST way to do transforms from XML to anything. XQuery is cool, but not the best way to fix that pile of XML nonsense, that has to be turned into an organized set of XML meaning. – J. M. Becker May 27, 2015 at 21:51

7

Personally I used XSLT in a totally different context. The computer game that I was working on at the time used tons of UI pages defined using XML. During a major refactor shortly after a release we wanted to change the structure of these XML documents. We made the game's input format follow a much better and schema aware structure.

XSLT seemed the perfect choice for this translation from old format -> New format. Within two weeks I had a working conversion from old to new for our hundreds of pages. I was also able to use it to extract lots of information on the layout of our UI pages. I created lists of which components were imbedded in which relatively

easily which I then used XSLT to write into our schema definitions.

Also, coming from a C++ background, it was a very fun and interesting language to master.

I think that as a tool to translate XML from one format to another it is fantastic. However, it is not the only way to define an algorithm that takes XML as an input and outputs *Something*. If your algorithm is sufficiently complex, the fact that the input is XML becomes irrelevant to your choice of tool - i.e roll your own in C++ / Python / whatever.

Specific to your example, I would imagine the best idea would be to create your own XML->XML convert that follows your business logic. Next, write a XSLT translator that just knows about formatting and does nothing clever. That might be a nice middle ground but it totally depends what you are doing. Having a XSLT translator on the output makes it easier to create alternative output formats - printable, for mobiles, etc.

answered Sep 17, 2008 at 0:51

community wiki
Tom Leys

Yes, I use it a lot. By using different xslt files, I can use the same XML source to create multiple polyglot (X)HTML

6

files (presenting the same data in different ways), a RSS feed, an Atom feed, a RDF descriptor file and fragment of a site map.

It's not a panacea. There are things it does well, and things it doesn't do well, and like all other aspects of programming, it's all about using the right tool for the right job. It's a tool that's well worth having in your toolbox but it should used only when it's appropriate to do so.

Share Improve this answer

Follow

edited Aug 13, 2009 at 8:00

community wiki
2 revs
Alohci

4

I would definitely reccomend to stick it out. Particularly if you are using visual studio which has built in editing, viewing and debugging tools for XSLT.

Yes, it is a pain while you are learning, but most of the pain is to do with familiarity. The pain does diminish as you learn the language.

W3schools has two articles that are of particular worth:
http://www.w3schools.com/xpath/xpath_functions.asp
http://www.w3schools.com/xsl/xsl_functions.asp

Share Improve this answer

Follow

answered Sep 17, 2008 at 1:43

community wiki
Nat

3

I have found XSLT to be quite difficult to work with.

I have had experience working on a system somewhat similar to the one you describe. My company noted that the data we were returning from "the middle tier" was in XML, and that the pages were to be rendered in HTML which might as well be XHTML, plus they'd heard that XSL was a standard for transforming between XML formats. So the "architects" (by which I mean people who think deep design thoughts but apparently never code) decided that our front tier would be implemented by writing XSLT scripts that transformed the data into the XHTML for display.

The choice turned out to be disastrous. XSLT, it turns out, is a pain to write. And so all of our pages were difficult to write and to maintain. We would have done much better to have used JSP (this was in Java) or some similar approach that used one kind of markup (angle brackets) for the output format (the HTML) and another kind of markup (like <%...%>) for the meta-data. The most confusing thing about XSLT is that it is written in XML, and it translates from XML to XML... it is quite difficult to keep all 3 different XML documents straight in one's mind.

Your situation is slightly different: instead of authoring each page in XSLT as I did, you only need to write ONE

bit of code in XSLT (the code to convert from templates to display). But it sounds like you may have run into the same kind of difficulty that I did. I would say that trying to interpret a simple XML-based DSL (domain specific language) like you are doing is NOT one of the strong points of XSLT. (Although it CAN do the job... after all, it IS Turing complete!)

However, if what you had was simpler: you have data in one XML format and wanted to make simple alterations to it -- not a full page-description DSL, but some simple straightforward modifications, then XSLT is an excellent tool for that purpose. It's declarative (not procedural) nature is actually an advantage for that purpose.

-- Michael Chermside

answered Sep 17, 2008 at 1:09

community wiki
mcherm

3

XSLT is difficult to work with, but once you conquer it you will have a very thorough understanding of the DOM and schema. If you also XPath, then you on your way to learning functional programming and this will expose to new techniques and ways about solving problems. In some cases, successive transformation is more powerful than procedural solutions.

Share Improve this answer Follow

answered Oct 10, 2008 at 12:05

community wiki
David Robbins

heh - you get a pretty good appreciation of xpath and the DOM when writing your own custom transformation code. There were a lot of things, including but not limited to: resizing images, generating javascript (from XML), reading from the filesystem to generate navigation, etc. – nickf Oct 11, 2008 at 14:29

3

I use XSLT extensively, for a custom MVC style front-end. The model is "serialized" to xml (not via xml serializaiton), and then converted to html via xslt. The advantage over ASP.NET lie in the natural integration with XPath, and the more rigorous well-formedness requirements (it's much easier to reason about document structure in xslt than in most other languages).

Unfortunately, the language contains several limitations (for example, the ability to transform the output of another transform) which mean that it's occasionally frustrating to work with.

Nevertheless, the easily achievable, strongly enforced separation of concerns which it grants aren't something I see another technology providing right now - so for document transforms it's still something I'd recommend.

Share Improve this answer Follow

answered Aug 13, 2009 at 8:21

community wiki
Eamon Nerbonne

3

I used XML, XSD and XSLT on an integration project between very dis-similar DB systems sometime in 2004. I had to learn XSD and XSLT from scratch but it wasn't hard. The great thing about these tools was that it enabled me to write data independent C++ code, relying on XSD and XSLT to validate/verify and then transform the XML documents. Change the data format, change the XSD and XSLT documents not the C++ code which employed the Xerces libraries.

For interest: the main XSD was 150KB and the average size of the XSLT was < 5KB IIRC.

The other great benefit is that the XSD is a specification document that the XSLT is based on. The two work in harmony. And specs are rare in software development these days.

Although I did not have too much trouble learning the declarative nature XSD and XSLT I did find that other C/C++ programmers had great trouble in adjusting to the declarative way. When they saw that was it, ah procedural they muttered, now that I understand! And they proceeded (pun?) to write procedural XSLT! The

thing is you have to learn XPath and understand the axes of XML. Reminds me of old-time C programmers adjusting to employing OO when writing C++.

I used these tools as they enabled me to write a small C++ code base that was isolated from all but the most fundamental of data structure modifications and these latter were DB structure changes. Even though I prefer C++ to any other language I'll use what I consider to be useful to benefit the long term viability of a software project.

Share Improve this answer Follow

answered Aug 13, 2009 at 8:55

community wiki
Sam

3

I used to think XSLT was a great idea. I mean it *is* a great idea.

Where it fails is the execution.

The problem I discovered over time was that programming languages in XML are just a bad idea. It makes the whole thing impenetrable. Specifically I think XSLT is very hard learn, code and understand. The XML on top of the functional aspects just makes the whole thing too confusing. I have tried to learn it about 5 times in my career, and it just doesn't stick.

OK, you could 'tool' it -- I think that was partly the point of it's design -- but that's the second failing: all the XSLT tools on the market are, quite simply ... crap!

Share Improve this answer Follow

answered Aug 13, 2009 at 9:52

community wiki
Jack Ukleja

1 It seems to me that you've just described *your* problems with XSLT, not any problems with the language itself. I'd have to say that you are probably using the wrong tools :)
– Nic Gibson Aug 13, 2009 at 17:00

2

The XSLT specification defines XSLT as "a language for transforming XML documents into other XML documents". If you are trying to do any thing but the most basic data processing within XSLT there are probably better solutions.

Also worth noting that the data processing capabilities of XSLT can be extended in .NET using custom extension functions:

- MSDN Documentation
- CSharpFriends: Tutorial

Share Improve this answer Follow

edited Sep 17, 2008 at 0:53

community wiki
2 revs
spoon16

1 Extending standard language with non-standard extensions is the worst thing one can do. What you end up with is neither XSLT nor CLR code. – Piotr Dobrogost Nov 11, 2009 at 15:15

Fair, that doesn't mean it's not useful sometimes – Eric Schoonover Nov 11, 2009 at 23:05

2

I maintain an online documentation system for my company. The writers create the documentation in SGML (an xml like language). The SGML is then combined with XSLT and transformed into HTML.

This allows us to easily make changes to the documentation layout without doing any coding. Its just a matter of changing the XSLT.

This works well for us. In our case, its a read only document. The user isn't interacting with the documentation.

Also, by using XSLT, you are working closer to your problem domain (HTML). I always consider that to be good idea.

Lastly, if your current system WORKS, leave it alone. I would never suggest trashing your existing code. If I was

starting from scratch, I would use XSLT, but in your case, I would use what you have.

Share Improve this answer Follow

answered Sep 17, 2008 at 1:05

community wiki
Mashed Potato

2

It comes down to what you need it for. Its main strength is the easy maintainability of the transform, and writing your own parser generally obliterates that. With that said, sometimes a system is small and simple and really doesn't need a "fancy" solution. As long as your code-based builder is replaceable without having to change other code, no big deal.

As for the ugliness of XSL, yes it's ugly. Yes, it takes some getting used to. But once you get the hang of it (shouldn't take long IMO), it's actually smooth sailing. Compiled transforms run quite quickly in my experience, and you can certainly debug into them.

Share Improve this answer Follow

answered Sep 17, 2008 at 1:32

community wiki
SpongeJim

2

I still believe that XSLT can be useful but it is an ugly language and can lead to an awful unreadable, unmaintainable mess. Partly because XML is not human readable enough to make up a "language" and partly because XSLT is stuck somewhere between being declarative and procedural. Having said that, and I think a comparison can be drawn with regular expressions, it has it's uses when it comes to simple well defined problems.

Using the alternative approach and parsing XML in code can be equally nasty and you really want to employ some kind of XML marshalling/binding technology (such as JiBX in Java) that will convert your XML straight to an object.

Share Improve this answer Follow

edited Sep 17, 2008 at 1:56

community wiki
2 revs
Tom Martin

2

If you can use XSLT in a declarative style (although I don't entirely agree that it is declarative language) then I think it is useful and expressive.

I've written web apps that use an OO language (C# in my case) to handle the data/ processing layer, but output XML rather than HTML. This can then be consumed directly by clients as a data API, or rendered as HTML by

XSLTs. Because the C# was outputting XML that was structurally compatible with this use it was all very smooth, and the presentation logic was kept declarative. It was easier to follow and change than sending the tags from C#.

However, as you require more processing logic at the XSLT level it gets convoluted and verbose - even if you "get" the functional style.

Of course, these days I'd probably have written those web apps using a RESTful interface - and I think data "languages" such as JSON are gaining traction in areas that XML has traditionally been transformed by XSLT. But for now XSLT is still an important, and useful, technology.

Share Improve this answer Follow

answered Aug 13, 2009 at 8:24

community wiki
philsquared

1

I have spent a lot of time in XSLT and found that while it is a useful tool in some situations, it is definitely not a fix all. It works very well for B2B purposes when it is used for data translation for machine-readable XML input/output. I don't think you are on the wrong track in your statement of its limitations. One of the things that frustrated me the most were the nuances in the implementations of XSLT.

Perhaps you should look at some of the other markup languages available. I believe Jeff did an article about this very topic concerning Stack Overflow.

[Is HTML a Humane Markup Language?]

I would take a look at what he wrote. You can probably find a software package that does what you want "out of the box", or at least very close instead of writing your own stuff from the ground up.

Share Improve this answer Follow

answered Sep 17, 2008 at 0:56

community wiki
Jason Jackson

1

I'm currently tasked with scraping data from a public site (yeah, i know). Thankfully it conforms to xhtml so I'm able to use xslt to gather the data I need. The resulting solution is readable, clean and easy to change if need occurs. Perfect!

Share Improve this answer Follow

answered Oct 23, 2008 at 7:22

community wiki
Goran

1

I've used XSLT before. The group of 6 .xslt files (refactored out of one large one) was about 2750 lines long before I rewrote it in C#. The C# code is currently 4000 lines containing lots of logic; I don't even want to think about what that would have taken to write in XSLT.

The point where I gave up is when I realized not having XPATH 2.0 was significantly hurting my progress.

Share Improve this answer Follow

answered Aug 13, 2009 at 7:47

community wiki
Sam Harwell

2 Yes, XSLT is not all bad and has some really cool use cases, but microsoft not embracing XSLT 2.0 is a pain. – Daren Thomas Aug 13, 2009 at 7:50

1 Tell us what was the size of C# code just after you rewrote these 2750 lines of XSLT. Giving the current size tells us nothing. – Piotr Dobrogost Nov 11, 2009 at 15:06

1

To answer your three questions:

1. I've used XSLT once some years ago.
2. I do believe XSLT could be the right solution in certain circumstances. (Never say never)
3. I tend to agree with your assesment that it is mostly useful for 'simple' transformations. But I think as long

as you understand XSLT well, there is a case to be made for using it for bigger tasks like publishing a website as XML transformed into HTML.

I believe the reason many developers dislike XSLT is because they do not understand the fundamentally different paradigm it is based on. But with the recent interest in functional programming we might see XSLT making a comeback...

Share Improve this answer Follow

answered Aug 13, 2009 at 7:53

community wiki
Dawie Strauss

1

One place where xslt really shines is in generating reports. I've found that a 2 step process, with the first step exporting the report data as an xml file, and the second step generating the visual report from the xml using xslt. This allows for nice visual reports while still keeping the raw data around as a validation mechanism if needs be.

Share Improve this answer Follow

answered Aug 13, 2009 at 8:17

community wiki
Jack Ryan

1

At a previous company we did a lot with XML and XSLT. Both XML and XSLT big.

Yes there is a learning curve, but then you have a powerful tool to handle XML. And you can even use XSLT on XSLT (which can sometimes be useful).

Performance is also an issue (with very large XML) but you can tackle that by using smart XSLT and do some preprocessing with the (generated) XML.

Anybody with knowledge of XSLT can change the apearance of the finished product because it is not compiled.

Share Improve this answer Follow

answered Aug 13, 2009 at 14:32

community wiki
Toon Krijthe

1

I personally like XSLT, and you may want to give the simplified syntax a look (no explicit templates, just a regular old HTML file with a few XSLT tags to spit values into it), but it just isn't for everyone.

Maybe you just want to offer your authors a simple Wiki or Markdown interface. There are libraries for that, too, and if XSLT isn't working for you, maybe XML isn't working for them either.

Share Improve this answer Follow

edited Oct 8, 2009 at 18:55

community wiki
3 revs
brianary

0

XSLT is not the end-all be-all of xml transformation. However, it's very difficult to judge based on the information given if it would have been the best solution to your problem or if there are other more efficient and maintainable approaches. You say the authors could enter their content in a simplified format - what format? Text boxes? What kind of html were you converting it to? To judge whether XSLT is the right tool for the job, it would help to know the features of this transformation in more detail.

Share Improve this answer Follow

answered Sep 17, 2008 at 0:55

community wiki
Shmoggle

the authors write XML in a text editor. basically it is a simplified HTML - some things have been removed to force consistent styling, things like a <video /> tag have been added as a shorthand for more complex HTML. Other elements are used to generate menus/bibliography/glossaries, etc. – nickf Sep 17, 2008 at 1:15

0

I enjoy using XSLT only for changing the tree structure of XML documents. I find it cumbersome to do anything related to text processing and relegate that to a custom script that I may run before or after applying an XSLT to an XML document.

XSLT 2.0 included a lot more string functions, but I think it's not a good fit for the language, and there's not many implementations of XSLT 2.0.

Share Improve this answer Follow

answered Sep 17, 2008 at 1:04

community wiki
Ben