



# Hansel minutes

Hanselminutes is a weekly audio talk show with noted web developer and technologist Scott Hanselman and hosted by Carl Franklin. Scott discusses utilities and tools, gives practical how-to advice, and discusses ASP.NET or Windows issues and workarounds.

## Text transcript of show # 7

February 20, 2006

### XML Tools and Technologies

Scott talks with Carl about tools for using and manipulating XML from readers and writers to XPath readers, transformers, and more.

(Transcription services provided by [PWOP Productions](#))



#### Our Sponsors

AutomatedQA  
test, debug, deliver!

<http://www.automatedqa.com/>

**NET** DEVELOPER'S JOURNAL

<http://dotnet.con-sys.com/>

[peterblum.com](http://www.peterblum.com)

Start with better code. Finish with better sites.

<http://www.peterblum.com>



(Music)

**Laurence Ryan:** From [Hanselminutes.com](http://Hanselminutes.com), it's 'Hanselminutes' a weekly discussion with web developer and technologist Scott Hanselman and hosted by Carl Franklin. This is Laurence Ryan announcing Show #7 recorded Monday, February 20<sup>th</sup> 2006. Support for Hanselminutes is provided by Automated QA - Makers of 'Test Complete', providing automated testing of Windows, .NET framework, Java and web applications, online at [www.automatedqa.com](http://www.automatedqa.com) and by [peterblum.com](http://peterblum.com) - "Start with better controls, finish with better sites," online at [www.peterblum.com](http://www.peterblum.com). Support is also provided by .NET Developer's Journal - "The World's leading .NET developer magazine", online at [www.sys-con.com](http://www.sys-con.com). In this episode, Scott covers tools and technologies around XML.

**Carl Franklin:** Hi! This is Carl Franklin, you are listening to Hanselminutes. And Scott before we get going, we want to give a shout out to our friends over the Polymorphic Podcast, <http://polymorphicpodcast.com> who are just celebrating their one-year anniversary.

**Scott Hanselman:** Yeah, I really admire the stuff that Craig has been doing over there at Polymorphic Podcast, we even stole one of his JavaScript rollovers recently. We put it up on Hanselminutes and we are trying to get that tweaked. Craig has put together a really solid show for quite a while now and we really appreciate his support of Hanselminutes and we wanted to give him a shout out.

**Carl Franklin:** So the topic today is XML. It's a kind of a big topic you have sort of narrowed it down into a few choice sub topics. What are those going to be?

**Scott Hanselman:** XML is a big topic and for a lot of people they really are just angle bracket delimited files. Even in large companies, you will find people parsing XML in ways that maybe aren't as effective as they could be. And at my company we often deal with really large XML files, we'll deal with a file that might be dropped in with a list of bills that need to be paid that might be 1/2 gigabyte or a gigabyte in size with 10s or 100s or 1000 of records and there are many different ways to deal with XML and ever since MSXML was introduced around the VB5.0, VB6.0 days there was a class of solution that I call the select single node solution that people are still using today, right? This is; load XML up in a DOM and say, select nodes or select single node and poke out that value that you want. I mean we have all done that, I am sure you have done that.

**Carl Franklin:** Sure.

**Scott Hanselman:** When you load XML up into a DOM, you are using up a lot of memory; you are having to do a big complete thorough parsing of that XML document and often people are loading something into a DOM either just to extract one little bit of information or more often to translate that document into a different format. So, maybe if I have a 10-megabyte XML document it might get loaded up and take 30 Megs of memory -- and that could be a real problem.

One of the things that I like to avoid is, what I call caterpillar style XML management. So, this is when, if you imagine a caterpillar when the back legs of the caterpillar start walking up towards the front and then the back of the caterpillar gets really, really tall, its like you're loading up all this stuff into the XML document and it gets big, big, big, big, big and then it sits there for a second. And then you hold this thing in memory for a moment and then you start walking away and then you pull your information out of the XML document. And for a really effective XML transformation whether that be -- and I don't mean XSLT but I mean literally transformation of XML to another format using anything. You want to avoid that kind of caterpillar style development where you load something up, hold it for a second and then spit it back out again. And using XML readers and XML writers can help you do that.

**Carl Franklin:** All right. So XML reader and XML writer are classes in the framework that work in this space; what -- how do you use these things? They sound like streams to me. Are they streams?

**Scott Hanselman:** They sit on, they can sit on top of streams -- exactly. So streams are like Byte Arrays, this is an interesting thing that you brought up. The concept of streams is something that .NET has introduced to make the idea of Byte Arrays and arrays information a lot simpler. I was working with some guys at Corillian recently doing some pair programming and I realized that amongst .NET programmers, in my opinion, particularly in my case C# programmers, there are two types of folks; there are the folks that lean hard on the language and there are the folks that lean hard on the library. And I was doing some pair programming with a fellow who was an expert C++ programmer and he was doing some manipulation of some Byte Array data in its raw form and was writing C# as if it were C++ -- because you can write C++ in any language, you could be a very C++, a VB programmer, right, you probably come upon VB.Net code that's been written and you just look at it and the



variable names you say, there has been a C++ person in this code I can smell it.

**Carl Franklin:** S to Z.

**Scott Hanselman:** Exactly.

**Carl Franklin:** SZ underscore.\_.

**Scott Hanselman:** OPSZ SDR. So, this individual leaned on the language very hard, while I lean on the Base Class Library but have less deep knowledge about the right things to do as far as keywords within the language. So, between us, we were very well placed. He was poking around inside of a byte array, when in fact a stream reader -- kind of, a forward only view of a byte array within the .NET base class library would have been the more appropriate thing to do for him to manipulate this data the way that he wanted to. I learned something, he learned something, but it really underscored the fact that the base class library has been thought about at such a depth that it's so much more about the libraries and it is about the language for a lot of different problems.

**Carl Franklin:** It's been a standard theme on .NET Rocks! to about reinventing the wheel and he really should have cracked the help file.

**Scott Hanselman:** Well, its so difficult though because the base class library is so complicated and I think that system.xml is one of those places where you could just live there; there is so much going on. So, an XML Reader, like the XML TextReader and Validating Reader are forward only views on some XML while the DOM loads that XML up and holds it in memory. A lot like back in the day in VB5 and VB6 we had forward only record sets and static record sets. I am sure you know more about that than anybody.

**Carl Franklin:** Yeah sure, static being that never changes underneath whereas they also had dynamic key sets, which would change -- the data would change on you.

**Scott Hanselman:** Exactly. And when beginner/developer started using Visual Basic they always went for the static stuff because that seemed to be the most logical way to do things and one of the tips that was constantly passed around was, oh no use a forward only if you don't need to go back and forth over the data spin through it once, use the forward only stuff. The XML TextReader is one of those examples where if you are going to be running through an XML document and you really don't want to load it up into memory, if you're going to be doing

something a lot, if you are going to have a very fast XML access to data and you want the memory footprint to be very flat; you don't want that entire document to get loaded up into memory, the TextReader is the most efficient way to do things. It's forward only and its read only.

**Carl Franklin:** So it does some memory management where it loads data up in chunks and courses through it, is that how it works?

**Scott Hanselman:** Yeah, it's actually loading it as it goes; it's walking through that stream whatever you pass it as it goes and it never actually loads the entire document. It's just looking a few nodes ahead.

**Carl Franklin:** Okay.

**Scott Hanselman:** And as you have heard of SAX, the SAX-Style parsing is push style of parsing where you start parsing XML and the SAX engine starts to announce, hey! I found an element, hey! I found an attribute, hey! I found an end element, it's hollering at you all the time. XML TextReader and the XML Reader style of things is the reverse of that; it is a pull style where you are saying, While reader.read and then each -- this is an interesting thing; each little piece, each little part, each atom of an XML document including white space, if you've told the XML Reader to care about white space is a node. I was parsing some OFX recently its an Open Financial Exchange format and they have some text nodes within the document just floating around and I had to decide whether I cared about those or not whether the white space was in fact significant so as you go next, next, next and you say pulling through this reader going read, read, read, each node is a type; everything from the comments -- you'll actually see comments go by when you are using an XML TextReader. It is the absolutely most efficient use of memory but it is read-only and by itself the XML TextReader doesn't do any validation, it won't validate your XML against an XST which is why folks then use the XML Validating Reader. So this is a reader that is pretty fast compared to the DOM but it -- and it does automatic validation but its two to three times slower than the XML TextReader; so things get real slow, real fast. Now, comparatively the DOM the XML document that we are all familiar with, is another two or three times slower than the XML TextReader. So we can -- we are looking at potentially 6-9 maybe 10 times slower when you start using a DOM in the memory issues...

**Carl Franklin:** Now we are talking about .Net 2.0 here? Because I am really...



**Scott Hanselman:** We are talking about really any version of .NET at this point.

**Carl Franklin:** Okay, because I do remember you -- I do recall you talking about a version of an XML component that was much faster in 2.0.

**Scott Hanselman:** The XSLT transform in .NET 2.0 the XSL compiled transform is way faster than 1.0.

**Carl Franklin:** Okay, but the rest of these are typically about the same speed? Same performance?

**Scott Hanselman:** I believe that almost everything in .NET 2.0 has received a speed boost, but for the most part the relationship between them is the same, the DOM will almost always, I mean when I say 'will almost' just in case someone writes and says "no in this one instance it isn't" but the...

**Carl Franklin:** Right, to the best of your knowledge.

**Scott Hanselman:** ...XML TextReader is the fastest way to get some information. Now, you pay for that though because you are having to go reader.read and you are poking around at a very low level on you. You don't get the luxury of backing up with an XML TextReader.

**Carl Franklin:** And Scott, I just want to remind the listeners at this time that Hanselminutes would not be possible without the generous support from our sponsors. And one of those sponsors is Automated QA and they make a product called TestComplete, which is just a complete solution for testing in .NET. And it's great stuff, we love it and that's why they are a sponsor. And so, if you would please do us a favor and check out their site at [automatedqa.com](http://automatedqa.com) and also let them know that you heard about them on Hanselminutes. That will help believe it or not, keep this show on the air.

(Music)

**Carl Franklin:** I recently wrote some code where I used an XML TextReader to -- or an XML Reader rather -- you pass in a URL to an RSS feed and then you use that reader on a dataset, a Read XML -- pass it to reader and you basically have loaded RSS data into a Dataset, where you can mess around with it.

**Scott Hanselman:** Using the XML Reader as a parameter to other things is a really nice way of doing things. You can -- passing data from back and forth, you are really not passing the data as you are passing the cursor at a certain location. Passing a DOM around means, passing this chunk of memory or the handle of this chunk of memory around. Passing a reader at a particular point is a really nice way to plug things in and out of the .NET framework. I like doing what is called XML Reader/Writer pipelines; we'll have an interface like "I," "my" XML handler, have some interface that takes a Reader and a Writer. So, I would like to say I read out of the left hand and I will write into the right hand. So, if someone wants to pass me a Reader and I am going to do some big transformation -- and I don't mean an XSLT transformation but maybe I have got a gigabyte file and I want to turn all of my attributes into elements; loading that giant XML export file into the DOM is unreasonable and probably not possible depending on how much memory you have. But I can read that up with the TextReader and I can make fundamental changes to what's going on in my document, I can convert attributes to elements, I can strip out comments, I can clean things up, a really common question is, how do I get automatic indenting in my document. Someone is going to give me an XML document and it's got no white space but I like my documents to be tidy for maybe debugging purposes? I can read that information in with a Reader and then with an XML TextWriter, which is the inverse of the XML TextReader I can then say XML formatting is going to be indented and then just read through and write out and I will get free indenting.

**Carl Franklin:** And of course that still is nicely tight on the memory.

**Scott Hanselman:** They are -- exactly because you are not loading an entire document out into memory.

**Carl Franklin:** But you do have to cursor through the whole thing?

**Scott Hanselman:** You do.

**Carl Franklin:** Which you would anyway in that case.

**Scott Hanselman:** But you are doing this -- and you are dealing with this in very-very small chunks of information at a time. And of course the XML document that we are used to using, is using a Reader underlying it anyway; this is the neat thing about readers is, that these are really



the lowest level, the chunks of information. They hide the angle brackets from you.

**Carl Franklin:** Now the XML document is different from the Dom, right?

**Scott Hanselman:** No, the XML document is the DOM.

**Carl Franklin:** Is the DOM.

**Scott Hanselman:** The DOM is an implementation, the W3C said, there is this thing called the document object model that we want to view of some XML and the XML information set. The XML document is .NET's implementation of the DOM.

**Carl Franklin:** Okay, all right.

**Scott Hanselman:** Now, the thing I keep saying that we haven't talked about is, the XML information set. The idea that if you look at the angle brackets and the text and try to do like a DIF -- like doing a DIF with a tool like Beyond Compare on an XML document doesn't really help you because the XML document might be different from -- bytes per bytes might be different, but what it is trying to express, the information set within that document is the same. So, if I had a document that had like books and the XML prefix, the shortcut before the element might be like "A:books" where A is the Namespace for Amazon. And I have another document where they just chose B as the Namespace prefix. The documents might express the exact same XML information set but if I did a DIF it would come out as radically different. So, when you are doing XML DIF's you need to ask yourself am I doing a low level -- is this the same bytes or is this the same information. Another example would be empty elements, if I have got a start Foo and then an end Fu or I've got a single Foo that's pre closed like <foo/> those express the same thing, an empty element, except one is short circuited closed and one is not; but they are expressing the same kind of a thing. The DOM tries to hide that and give you a pretty clear abstraction on top of that, so you get the same information set. So, once you have these kinds of underlying pieces and you become familiar with it I suggest that everyone really try to deal with XML TextReader and that validating reader and think about what you can do. You don't have to live with that really low level all the time; you can run through to a point within an XML Reader, find something you recognize and then maybe use the node, use an XML node reader or use XML serialization, XML serialization is one of my favorite things.

**Carl Franklin:** So what's that?

**Scott Hanselman:** That's when you can mark up a Class, like I could say public class book and then I can put attributes on it, .NET attributes that say, when I serialize a book, when I save a book as XML I want it to have these elements in these places. So, it's the automatic kind of marshalling of data from the angle bracket world into .NET CLR type world. And in .NET 1.1 you can say get to a point within an XML Reader and say Read in your XML and then de-serialize a document and I've got an example of reading and saving XML fragments at [shrinkster.com/c1](http://shrinkster.com/c1) and in .NET 2.0 you can say like read sub-tree and you can, at a certain point within a document, kind of, break off that reader and take a chunk out of it. So, you get the best of both worlds, you don't have to build an XML serialization tree over the entire document but you don't have to keep poking around at the low level with the XML Reader.

**Carl Franklin:** Cool.

**Scott Hanselman:** So, you can walk forward, grab a piece and then keep moving. But, one of the really exciting things about the XML Reader is that it is an abstract base class. The XML TextReader and the validating reader are examples, concrete instances of that, but we can build our own XML Readers using the same model. Dare Obasanjo and Howard Hao wrote a XPath reader; XPath is that kind of tight language that you can say things like, give me all the books that have an author equal to this and you go like '\\books [author] and you build a whole expression.

**Carl Franklin:** Sort of like SQL for XML, query language.

**Scott Hanselman:** It's a query language for XML nodes. But typically you have to load these things up into an XPath document, a faster more optimized XML document that's optimized for XPath and for XSLT, and you have to then query from there. So, how do I get the convenience of an XPath query without all the memory usage that I would get from an XPath document or an XML document? And Dare Obasanjo have got an XPath reader. So at [shrinkster.com/c8w](http://shrinkster.com/c8w) Dare and Howard's example is up there and I have actually linked to another a really smart XML guy name Oleg Tkachenko and Oleg Tkachenko's blog is at [shrinkster.com/c8v](http://shrinkster.com/c8v) he is an XML pundit. Another great guy is Daniel Cazzulino at [www.shrinkster.com/c8u](http://www.shrinkster.com/c8u). These are folks that I really keep an eye on when it comes to what's really interesting and going on within the XML



space. The XPath reader that Dare and Howard have got has a subset of XPath, it lets you do those XPaths that would make sense within a forward only context. So, you can ask questions of the XPath reader and actually build up an XPath collection and you could say, well, here are three different queries that I want you to watch Mr. XPath reader, and I could say, rather than 'reader.read' I would say 'reader.read until match' and then ask it which match it found. So, I would get the benefits of XPath and the speed and the convenience of an XML TextReader.

**Carl Franklin:** That's shiny to quote you.

**Scott Hanselman:** It is shiny. Another really shiny one, it was written by a fantastically brilliant woman name Helena Kupkova and she actually wrote a thing called Fast XML at [shrinkster.com/c8z](http://shrinkster.com/c8z), an XML processor and parser that is 8, 10, 20 times faster than the ones that came from Microsoft. She did it as her, I think her Master's thesis or her PHD thesis, and she has written an XMLBookmarkReader; remember how I said that XML Readers are forward only? Well, she has got one at [shrinkster.com/c8y](http://shrinkster.com/c8y) that would let you mark a point within an XML Reader, continue to move forward and play around, and then say, return and remove bookmark and actually hop back to where you were and then keep going.

**Carl Franklin:** Awesome.

**Scott Hanselman:** So it's kind of like TiVo for XML Readers. I don't need to overuse the TiVo reference too much but I absolutely love this thing because...

**Carl Franklin:** Yeah that's great.

**Scott Hanselman:** ...a lot of times when you are doing an XML pipeline like this XML reader-writer pipeline where you are passing in readers and writers, you can be concerned that your implementation is going to misuse the Reader. Someone might read one step too far or not far enough and kind of mess things up. With the bookmark reader, your outer pipeline can return back to a known point and continue on. So, even someone who wrote a malformed or well, poorly behaved chunk of that pipeline can have themselves saved by this bookmarking reader.

**Carl Franklin:** Wow that's fabulous man.

**Scott Hanselman:** Well, some other cool things that you can do with XML Readers are, you can - and this will blow your mind, I hope it does. You can make XML Reader implementations over

things that aren't themselves XML. So, I'll pause for effect there and you can just drink that in.

**Carl Franklin:** Yes, so like a comma separated value file or an INI file or something like that?

**Scott Hanselman:** Oh, absolutely you have done your research. At [shrinkster.com/c91](http://shrinkster.com/c91) they have got an XML Reader that reads CSV files. So, what its doing is -- remember that the XML Reader, when you say reader.read it sets a number of variables and properties like nodeType saying, what kind of node am I on? Am I on an element? Am I on an attribute? Am I on an end element? So, if you wanted to provide a view over a CSV document you could decide what does that mean to you, is it the comma, that is the XML element or is that the quote within a CSV document? You get to decide and as long as you return the right answers so that the consumer thinks its XML, the fact that you are underlying not reading XML doesn't matter, because the XML Reader doesn't care about angle brackets, it cares about this information set. So, the Xml Csv Reader makes CSV files look like XML.

**Carl Franklin:** That's cool.

**Scott Hanselman:** So what really is cool about that is, you could then run an XSLT transform on a CSV file. You can actually write an XSLT in XML style sheet transformation over the top of an Xml Csv Reader; so literally going directly from comma separated values to HTML or XML or something else without having to do intermediate steps. Because, typically you would want to just read that CSV file on yourself, maybe save it to a Dataset or put it in a temporary file and then transform from there.

**Carl Franklin:** Have you had to use this yourself at work?

**Scott Hanselman:** Oh yeah. Anytime I can do a streaming transformation over any kind of thing rather than loading it up in the memory, I love doing that kind of stuff. The ability to make an XML Reader over the top of something that isn't XML, I think is a really powerful thing. The INI reader is pretty cool; the XML INI reader by Ralf Westphal at [shrinkster.com/c92](http://shrinkster.com/c92) makes regular rolled INI files like any files from back in the day look like XML. But what's interesting about that article is it gives you the tools to do this yourself with maybe flat files you have now. If you are a company that has got flat files that you wish for XML or you know that you are going to move them to XML, you could start now by making that information set available and then when you



eventually do change them to XML, your consumers wouldn't actually care.

**Carl Franklin:** Sweet.

**Scott Hanselman:** Another great one is by Aaron Skonnard actually at [shrinkster.com/c90](http://shrinkster.com/c90) who has got a file system navigator. Basically, he has taken the interface of XPath node iterator and he has made it over the file system. So, you could run XPath queries over your hard drive. Again, all of this is about implementing interfaces or making classes that derive from abstract base classes, that as long as you follow the contract, as long as you answer the questions and produce the right information, the fact that you are not reading underlying XML is an implementation detail.

(Music)

**Carl Franklin:** Okay Scott that's very cool; what else are we going to talk about today?

**Scott Hanselman:** Well, so we have been talking about -- I am extending some of the interfaces that Microsoft gives, like the XML Reader but there are other extension points within kind of, the XML Ethos that can be exploited. The XML MVP group, a bunch of MVP's have gotten together and they have added things to System.XML, the XML Namespace, that they wish Microsoft would have done; maybe Microsoft didn't have time or it was an esoteric thing that they didn't feel was necessary. And up at [xmlmvp.org](http://xmlmvp.org) they have added a couple of classes that you can just download their BST license there is one for .NET 2.0 and there is one for .NET 1.1 and an interesting one is an XSL Reader. So, this is a transformation, we know often times we will do XSLT transformations where you will take a DOM and apply a style sheet to it, a tree transformation language and typically that gets transformed into another DOM. They have got an XSL Reader that let's you read the results of that transformation as if it were an XML reader.

**Carl Franklin:** Nice.

**Scott Hanselman:** Then what's significant about this is, if you were stringing lots of things together, making a pipeline of readers into readers into writers into readers then you could take something like this and you could say, I am going to take this CSV file and read it with the CSV reader, do an XSL transform with some XSLT and the results will go straight into an XSL reader. And this is important because in .NET 2.0 the XSL compiled transform, you remember how

we said that they have a new class in .NET that has compiled transforms? Well, because of the way it's doing that compilation they've thrown away the ability to transform into an XML Reader. So, this team has basically put that functionality back. And they are making it available. There is other great features that they have added to XML that you might want to take a look at it [xmlmvp.org](http://xmlmvp.org). Another thing that they have added is what's called EXSLT and you can learn about that at [exslt.org](http://exslt.org). These are kind of formalized extensions to XSLT. When you are writing this XSLT transformation language and you hit a wall, Microsoft lets you throw a script in there, you know a lot of people have with MSXML and back in the COM days and the VB days, you would throw Java script or VB script inside of your XSLT transformation. And when you did the transform they would fire up the scripting language and you would have a little opportunities to poke around in there. But they have actually formalized this XSLT series of libraries [exslt.org](http://exslt.org) and they have built support for these libraries into .NET and the MVPXML library. What's cool about that is that before, when you wrote Java script and VB Script you were pretty much stuck with whatever XSLT transformer you were using, you are pretty much stuck on Microsoft. But now EXSLT, these libraries are described in a way that is more about the interface and less about the implementation. So, they have got a .NET implementation of EXSLT, which means I could actually run it on different implementations of an XSL transformer, I could take an XSLT that maybe a Linux user was using against a Java XML library and transforming on a Linux machine but he used these libraries, these methods that are available via EXSLT and they'd still work on .NET. So, it's almost like saying cross platform XSLT. XSLT at its pure essence is something that if you just write pure XSLT for the most part if they follow the spec you can use any number of transforming engines. But as soon as you start extending it and putting in script of your own, you are pretty much stuck and have basically written Microsoft specific XSLT. This EXSLT is becoming enough of a standard now that they are a lot of choices for people who are writing these style sheets.

**Carl Franklin:** Smokin'.

**Scott Hanselman:** It is pretty sweet.

**Carl Franklin:** Yeah. You have been writing a lot about XML this week. What are some posts that you want to point us to?

**Scott Hanselman:** I did some XSLT stuff a while back at [shrinkster.com/bf0](http://shrinkster.com/bf0) talking about how --



when you do XSLT, you have really got to just test-test-test and we actually got to interact with some of the guys at the XML Perf Team at Microsoft and they said that even though you can write XSLT and most often everything that you write is portable. If you are really serious about performance, you need to tweak it for the quirks of the particular parser that you are using. You get some performance testing on the COM based MSXXL 4, MSXML 6 we use .NET 1.0, .NET 2.0 and learned some techniques that can make XML transformations more powerful and more efficient.

**Carl Franklin:** Okay.

**Scott Hanselman:** Another one is some fragment writing, a lot of times people want to write out an XML document from an object but they don't want all the Namespaces; maybe they want some backward compatibility, they don't really want an XML document, they want an XML fragment. We have got an XML fragment writer up at [shrinkster.com/c1k](http://shrinkster.com/c1k) and then as I mentioned before, reading those fragments back in with the serializer at [shrinkster.com/c11](http://shrinkster.com/c11).

**Carl Franklin:** Okay, what are some XML sites that you visit regularly?

**Scott Hanselman:** Well, blogs that I read of folks that are into XML deeply; like I had mentioned Oleg Tkachenko at [shrinkster.com/c8v](http://shrinkster.com/c8v) is amazing; I just love reading his stuff, Daniel Cazzulino at [www.shrinkster.com/c8u/](http://www.shrinkster.com/c8u/) has got a whole series on XML performance. I love reading stuff by Dare Obasanjo, I love reading stuff by Aaron Skonnard, anything up on MSDN about XML they really have some brilliant stuff. Anytime, you can read an article by Helena Kupkova whose blog I have been unable find but she is absolutely brilliant and reading the code inside of the XML Bookmark reader is pretty amazing. And actually a link that I forgot to put up but I think you could probably find it up on my blog is a thing called an SGML reader. You know how HTML is not well formed, there is no rule that says that you absolutely have to have a closing tag, when you open a tag like the BR Tags real common tag that people don't close. You can't open up a regular old HTML document as an XML file, as an XML document because if its not well formed then it's not XML. Well, Chris Lovett wrote a thing called the SGML reader and you can Google for this, it will be the first hit. The SGML Reader is an XML Reader. So, he has derived an XML Reader and what he does is he actually auto close his tags so if he sees that there is a tag that has been opened and then he gets farther along into the document notices that

if I read the next element that this is going to be an invalid document he will go back and automatically close elements that need to be closed. So that a poorly formed document like whether it would be OFX in my case the financial exchange format or whether it's HTML, you will get valid XML. So if you have some HTML you want to clean up or you need to parse some XML that's not necessarily well formed, the SGML Reader is a really great way to do that. This is a lot of information in a really short period of time and XML is one of those things that you can get really deeply into and there are people who spend their entire career focused on XML. But the thing that I want to get across to folks is that A> If you don't want to see angle brackets, you don't have to. There are so many things that allow you to look at XML without ever seeing an angle bracket. It's really about the information set not the brackets.

**Carl Franklin:** And that said, it's always a good idea to have at least one or two really hot shot XML guys on your staff in your development team.

**Scott Hanselman:** And on your instant messenger.

**Carl Franklin:** Yeah. And it's probably good if it was you.

**Scott Hanselman:** One of the things that my boss says is that, every problem in computer science can be solved by one additional layer of abstraction.

**Carl Franklin:** Yeah, that's a good idea.

**Scott Hanselman:** And dealing with XML having these nice clear interfaces and then having just a little layer of abstraction around them gives you a lot of flexibility whether you are parsing RSS or pulling information out of a CSV file, those layers of abstraction and a nice clean library like the .NET based class library and System.XML make life a lot easier.

**Carl Franklin:** And that's a show. Everything else you want to know about XML, thank you Scott.

**Scott Hanselman:** Thank you Carl.

**Carl Franklin:** Thank you listeners and we will see you next week on Hanselminutes.

(Music)