Ned Batchelder: eXtending Domino with XML

by
Tara
Hall

Level: Intermediate
Works with: All
Updated: 11/01/99

Inside this article:

Related links:
IBM alphaWorks

Lotus Developer Network

IBM developerWorks XML zone

Get the PDF:

[Editor's note: After reading Ned's interview, head over to the Developer Spotlight discussion forum to talk about XML and Domino.]

Ned Batchelder is currently the primary architect for XML (eXtensible Markup Language) in Domino. It's his job to investigate possibilities, find the best fits, and educate the rest of Iris about the technology. In this interview, he "eXplains" the work being done to support XML in Domino.

Why is XML support important for Domino?
There are a couple of reasons why XML support in Domino is important. First, XML is the *de facto* standard for interchange of data going forward. Fundamentally, XML is about standardizing the boring parts of data interchange, like character sets and tokenization. By using XML, developers can concentrate on the interesting parts of data interchange -- what is the data to be represented and how will we represent it? It has been said that XML is the ASCII of the 21st century, and I think that is accurate. Both XML and ASCII are foundation technologies that are not that interesting in and of themselves. It is the solutions that they enable that are interesting.

The second reason is that XML introduces a new mindset to the Web. Until now, data on the Web has been held captive in HTML pages. HTML today is about appearance, not data. You can't easily write a program to read HTML pages and extract data from them. It's done, but it isn't a robust way to build a system. If the page containing the data is redesigned, you need to re-write the program to extract the data. XML is all about describing data. It introduces a mindset of structuring data independent of its presentation. This is a powerful new model for Web development because it allows more flow of data to programs, rather than flow of pages to eyeballs. By separating the data from its presentation, you not only have more powerfully structured data, but you also enable multiple presentations.

Domino is perfectly suited to this new model because it is the model that Lotus and Iris have been building on since Notes V1 a decade ago. Domino stores data in structured documents, separate from their presentation.

When the Web revolution came a few years back, the excitement over the Web helped to educate customers about the possibilities of the Notes architecture. The Web and its attendant hype convinced people that distributed, programmable, hyperlinked groupware was a good thing, and even provided us with words to describe what we'd been doing all along -- intranets and extranets. Web technology had caught up to what Notes had been providing all along, and we rode the wave with the Domino Web server.

Now that XML has arrived, we expect to ride a new wave: structured data separated from presentation. Once again, the Web has caught up to what we've known all along: data is more powerful if it is clearly described and separated from its appearance.

"Once again, the Web has caught up to what we've known all along: data is more powerful if it is clearly described and separated from its appearance."

How will Domino support XML?
XML is a big topic and will appear in many ways throughout the Notes and Domino world, but let's talk in broad terms about a few ways in particular. First, Domino databases will treat XML data as a first class citizen. Domino databases have always been a heterogeneous storage mechanism that have been extended over the years to accommodate important data. In future releases of Domino, we will extend the database system to allow XML to fully participate in all of the features of Domino databases. That includes top-notch programmability, so all of the services you'd expect for XML data (for example, parsing and transforming) will be built into the database system.

The other big appearance of XML in the Domino world will be as a way of accessing information in Domino. We'll support XML as another "API" to Domino data. Developers will be able to request XML representations of objects in the Domino system: documents, design elements, databases, and so on. They will create new data by creating XML representations and pushing them to Domino. Our goal is to have all of the information in a database, including design elements, expressed in an XML representation. So far, for lack of a better name, we've been calling this XML representation of Domino data "DXL".

What are the benefits of XML support for Domino application developers?
As XML becomes more and more prevalent in system architectures, Domino applications will handle more and more XML data. The native XML support in the database system will allow XML data to be part of Domino applications with all of the benefits that application developers are accustomed to -- cross-platform, scalable infrastructure, security, programmability, views, searching, replication, Web access, and so on. Domino has always been a

powerfully heterogeneous platform, and XML support will simply build on those strengths.

One of the big attractions for developers will be that XML provides a data abstraction layer for applications. Developers will write an application based on data in a certain XML format and will easily add, extend, or replace data sources, and the application will still work. As XML becomes more and more the standard basis for exchanging information, developers will design applications by choosing an XML representation for the data and building on it. DXL will provide a foundation for importing and exporting data to and from an application.

The availability of Domino data in DXL will allow a broader class of application. Currently, access to all the details of design elements and rich text is strictly the domain of the Notes C API. The C API is extremely powerful, but very complex. Writing rich text or designing applications in the C API is a daunting task. DXL will be a much more approachable representation for this data and will allow many more developers to be creative in this area.

What is the Domino DTD?
The Domino DTD is the technical description of the structure of DXL. DTD is an acronym for Document Type Definition. The XML standard doesn't define any tags, unlike HTML, which has a fixed tag set with a limited number of defined tags. Instead, XML defines how to describe tagsets.

Could you describe the work that you have done on the Domino DTD?
The structure of data in Domino databases dictates the design of DXL. We weren't going to change the way databases store information so that we could represent it in DXL. Keep in mind, DXL is separate from the work needed to add XML as a native data type in the database. So designing DXL amounts to understanding all the data in Domino databases and developing good ways of representing that information in XML. It isn't always obvious which is the best way, even for apparently simple data. There were many times during the design that we had to re-think previous decisions so that the language as a whole made sense. Since we aren't done yet, I expect that there will be a number of times more that it happens.

"Features of the design elements are unique to Domino, so only a Domino-specific language can describe them."

What do you see as the advantages or disadvantages of developing a server-specific markup language?
The advantage of creating our own markup language is that we can describe precisely what we want to describe. In the case of DXL, we need to accurately describe Domino data and design elements. Features of the design elements are unique to Domino, so only a Domino-specific language can describe them. When we want to talk about embedded views, for example, we don't want to pretend that they are something else, like a table, so that a more generic language can describe them. We don't want Domino to be a least-common denominator sort of product, and we don't want our XML expression of Domino to be either.

The disadvantage of our own markup language is that it might narrow the range of interchange. For example, only another Domino system will understand the DXL that comes from a Domino system. We rely on the malleability of XML to compensate for this limitation. For example, eXtensible Stylesheet Language (XSL) is a very good system for transforming XML from one form to another. If you're building a system that uses DXL, but you need some other XML language, you can write an XSL transform to morph the DXL into the XML that you need. We've been keeping this flexibility in mind as we design DXL to make sure that it is structured well for this sort of operation.

The only standard for describing pages is eXtensible Hypertext Markup Language (XHTML), but it falls far short of what we need to represent Domino pages precisely. Keep in mind, I'm not talking about rendering pages for display to a user, but about capturing all of the design elements of a page. Consider paragraph hide-when formulas, for instance. The Domino Web server does a fine job using HTML to render Domino pages. When it encounters a hide-when formula, it evaluates the formula to determine if the paragraph is hidden or shown. In either case, the formula is no longer needed for rendering in a Web browser. Domino doesn't have to represent the formula in the HTML output; either the paragraph is in the output or it is not.

However, if the goal is to represent the hide-when formula, not just the result of the formula, how can we do that in HTML? HTML has no facility for representing this, so we have a handful of options: 1) use HTML, but lose any information that can't be represented; 2) use HTML and find some nonstandard way to represent other information; or 3) design a language that describes exactly what we need. Option 1 is unacceptable: DXL must provide a way to read and write all interesting Domino information. Option 2 is interesting and has been used by others facing this sort of problem, but it leads to complicated files that are difficult to understand. We wanted DXL to be an approachable representation, so we chose option 3.

To date, the World Wide Web Consortium (W3C) has approved recommendations for XML 1.0, and XML Namespaces; however, working groups continue to draft specifications for XSL, XLink, XPointer, XML Schemas, and XML Query. How has this affected your work and what impact will these recommendations have once they are approved by the W3C?
The XSL specification is almost done, so it hardly affected our work. We're using XSL a great deal, but the recent changes have not been dramatic and haven't affected the architecture we're designing. Scott Boag at Lotus has been implementing LotusXSL alongside the successive drafts of the specification, so we always have a good implementation of the latest ideas.

"Lotus is a member of the World Wide Web Consortium, and we have played a direct role in the evolution of key XML standards."

XML Schemas is a replacement for DTDs as a way of describing the structure of XML. For practical purposes, a DTD describes DXL because that technology is available now. DTDs are a fairly primitive way to describe structure, but we did not let the limitation of DTDs limit DXL. We track the

XML Schemas work to keep in mind the possibilities that it will offer us, both in terms of structures and data types. Noah Mendelsohn at Lotus is one of the editors of the standard, and we work with him to make sure our needs are met.

XLink and XPointer are about connecting different pieces of XML together, and we're looking at how they can be used to express connections within Domino data. Those standards are still evolving, but we expect that the basics will remain the same.

XML Query is in a different category because the working group has only begun drafting the specification. We don't know what the standards will be. The best we can do is to help define the standard so that it does what we need it to do.

You mentioned XML Schemas as a replacement for DTDs. Why are DTDs being replaced, and why are Schemas more sophisticated than DTDs?
DTDs were introduced with XML 1.0. They provide a simple facility for describing the structure of an XML document. Schemas provide a number of enhancements over DTDs: schemas are themselves XML documents, while DTDs are not, so tools that process ordinary XML documents can also process schemas. The schemas specification also provides much richer facilities for describing the structure and content of XML documents. For example, a schema can indicate that particular XML content is an integer, a date, and so on. Schemas provide additional facilities for validation of documents that use combinations of XML vocabularies. For example, one can validate a purchase order document in which shipping addresses come from an XML vocabulary defined by the Postal Service, descriptions of books to be ordered, for example, are defined by the Library of Congress, and the remainder of the tags are defined by the e-Commerce software vendor.

What role have Lotus and IBM played in the W3C working group drafts of XML and XML-related specifications?
Lotus is a member of the World Wide Web Consortium, and we have played a direct role in the evolution of key XML standards. I mentioned earlier that Scott Boag has contributed to the development of the XSL specification and has also written an XSL Transform implementation that is widely used not only at IBM and Lotus, but also by hundreds, if not thousands, of customers and business partners who have downloaded it from IBM's **alphaWorks Web site**. Lotus is also playing a key role in the development of the new XML Schemas specification, which will provide a significant enhancement to the capabilities now available through XML DTDs. In fact, Noah Mendelsohn of Lotus and Ashok Malhotra of IBM have served as co-editors of the schemas specification. In all of these activities, we work closely with IBM and with other companies involved in the development of XML.

You mentioned e-Commerce earlier. There is much discussion that XML will change e-Commerce, but how will XML change knowledge management (KM)?
Let me first say a little about e-Commerce. There's a lot of hype around XML and e-Commerce, and much of it is warranted. The Web has done a great job connecting everybody. No matter which business you're in, you've got a Web site, and so do your business partners. Your customers and business partners can find you on the Web, and you can find them. But once you find each other, all you can do is read Web sites. How do you exchange information? Until now, there has not been a good infrastructure for transferring structured data across the Web, no matter how well-connected we are. XML will change that. With XML, we can build systems that exchange data with one another, including orders, catalogs, inventories, and

"Until now, there has not been a good infrastructure for transferring structured data across the Web, no matter how well-connected we are. XML will change that."

so on.

As for knowledge management, XML will have a huge impact on the core KM technology, such as search and classification services. Having documents that are well-structured and that have consistent metadata makes searching more precise and allows automated routines to accurately classify documents in topic catalogs. These systems will benefit immediately by having standard DTDs and metadata schemas with which to develop systems. Even if the source documents are not XML documents, XML unifies the disparate representations in real-world systems. For example, we will support a standard version of XML-RDF (Resource Description Framework) for catalog import/export.

I expect XML to be a given in the KM world, since its philosophy so closely matches the problems that KM is trying to solve.

There are many XML tools in development or available for download. Lotus offers the LotusXSL processor that you mentioned earlier, and many tools are available from IBM alphaWorks Web site. Are there plans in Lotus or Iris to offer more tools, such as DTD development tools or XML search engines?
It goes without saying that foundation services, like XML parsers, DOM implementations, and the LotusXSL transformation engine, will be built in to Domino with great integration. For example, it will be very simple to parse items or file attachments as XML. We want Domino to be a great place to build XML-centric applications.

In addition, we're evaluating all available tools to see what makes the most sense for us to adopt. Domino is a very broad platform, and we need to be sensitive to that. If a tool is only useful to 5% of the developers, then we won't include it in the box. For future releases, we want to extend Domino Designer, so that useful tools can be integrated by developers, rather than having to be built in by us. We can't provide all the XML tools people will want to use. XML has been growing at a tremendous pace, and developers conceive and develop tools just as quickly. We want Designer to be a good host for any of those tools being developed.

How is IBM using XML, and how are you coordinating your efforts with IBM?
IBM views XML as the fundamental technology supporting interoperability of data formats and document exchange among applications. IBM has hundreds of people working on XML and participates actively in most of the significant XML standards efforts. Domino and Notes are at the center of collaborative and document management applications in the IBM product family, so the XML support in Domino is even more important. Beyond IBM itself, many Lotus customers are building applications that integrate Domino with data from IBM databases, MQ applications, and so on. Iris, Lotus, and IBM work closely on coordinating our work on XML.

What can users expect in terms of support and tools in the upcoming releases of Domino R5.x?
In 5.0.2, we will document and support the ?ReadViewEntries URL syntax, so that developers can use the Domino Web server to retrieve XML data from views. In 5.0.3, we'll add one more URL syntax, ?ReadNote, which will open a note and return its contents in XML format. These are just the tip of the iceberg when it comes to data access in XML, but we limit how much new code we include in a quarterly maintenance release.

"I expect XML to be a given in the KM world, since its philosophy so closely matches the problems that

In 5.0.3, you'll also see the first appearance of XML parsing and transformation engines in the product. We'll integrate these into the Java backend classes so that developers can make use of these foundation

KM is trying to solve."

services in a way that's really natural for Domino applications.

What can users expect in future versions of Domino?
In future feature releases, you'll see the full expression of the Domino/XML vision that I talked about earlier. We think that XML is a good fit for the Domino world, and that our XML support is going to enable a wide range of new applications. The industry is excited about XML because of the solutions that people can build with XML, and Domino is well-positioned to be at the center of it all.

BIOGRAPHY

Ned Batchelder is an Architect at Iris Associates. He has worked on Domino and related technologies at Lotus and Iris for six years. His projects have included the **Domino Designer**, the **Domino Web Server**, and **NotesPeek**. More challenging than any of that has been helping to raise three sons and finding ways to relax.

ABOUT THE AUTHOR

Tara Hall is a User Assistance writer for Lotus where she has worked for over a year. She is part of the Web Applications and Streaming Media teams and writes online help, programming guides, and release notes. She also is a member of the Notes UA Web team.