Using Web 2.0 in Domino

Series: Web 2.0 for Lotus, WebSphere Portal and You

Listen to Pete Janzen from IBM Lotus interview various experts from the Lotus and WebSphere Portal development teams on Web 2.0 technologies. Hear how IBM has utilized these technologies in the various products and how you, the developer, can leverage them in your custom applications.

Abstract for Using Web 2.0 in Domino: Vinod Seraphin looks at how the Lotus Domino platform supports various Web 2.0 technologies. Hear how developers were using Ajax in Domino applications before the term Ajax was coined and how Domino’s support for Ajax has evolved. You'll also hear how Domino Web Access can be viewed as one of the earliest rich internet applications and how developers can use similar techniques in their custom applications. Vinod Seraphin is the lead architect for Domino Web Access.

Discussion:

JANSEN: Welcome to our developerWorks podcast series on Web 2.0 as it applies to various Lotus and WebSphere Portal offerings. My name is Pete Jansen, and I'm the product manager for Lotus Component Designer and Enterprise Integration for the Notes and Domino platform.

I am your host for this podcast series where I will interview a number of our technical thought leaders on topics related to Web 2.0. Our objective is to enlighten you on the various concepts and technologies that comprise Web 2.0, and how they apply to the products we develop.
In this podcast we will explore the topic of Web 2.0 for Domino. I am very pleased to welcome Vinod Seraphin who is a senior technical staff member at IBM and the chief architect for Domino Web access.

Vinod has been instrumental in leading the adoption of Web 2.0 technologies like Ajax and Domino even before the term Ajax was coined. Welcome to our podcast series, Vinod.

SERAPHIN: Hi Pete, thank you.

JANSEN: You've been involved with Domino for a long time, and I think that you can really share a lot of good insight on how we go ahead and have leveraged Web 2.0 in the past and, you know, how we're doing it now. And then maybe some thoughts on the future.

But what I'd like to start off with is, you know, looking back a little bit, since Domino has a long history of the Web application platform, could you please outline how Domino has evolved its support for building Web 2.0 style applications?

SERAPHIN: Yes, so I think Domino goes way back and it sort of evolved along with the Web. So if you look at the early releases of various standard templates we ship with Domino, they have a very Web 1.0-ish, meaning that you would make a URL, it would come back with a static page.

There wouldn't be that much interactivity with the page without reloading another page. And over time, that's, you know, Domino has evolved to provide more of those dynamic capabilities.
And I think those dynamic, you know, better, richer experiences for a user using a browser is part of this Web 2.0, is one of the core things that's part of the Web 2.0 umbrella.

And so if you wanted to outline how we evolved, how Domino evolved to allow more interactive applications, you know one of the first things was probably around the Domino 5 timeframe where Domino introduced a view applet. And the view applet was used so that when you're in a view it wouldn't reload that page for the view but rather would go get newer data segments based on how the user was interacting with the applet. So as they scrolled in the view, the applet would go out and retrieve that chunk of data.

And so I think that was really the original background for how the read view entries URL command first developed and along with that applet was also using read design and something else called read entries. And so those were some of the early sort of like Rest APIs that were supported by Domino 2 to do that.

In addition, Domino has always had a very rich set of design elements that could be used in lots of different ways by developers. And just the nature of a Web application is that there are often lots of different parts.

So there's the main page and there could be external script pages, and then you're doing various posts and targeting other URLs which can be other design elements in Domino. So those design elements also allowed the people to do more Rest type of calls.

JANSEN: Okay, Vinoid, thanks for giving us that history of how Domino has
evolved as a Web application platform. And you know, one of the things that you did mention was about how in our early templates that were being served up to the Web, it was definitely a 1.0 experience.

And I think that, you know, since you are the chief architect for Domino Web Access which is basically delivered via a template, that's a good way to see how our support for richer Internet applications has kind of evolved because I think you know DWA, as well as our the Domino Web Administrator both have kind of pushed the envelope as far as providing a richer user experience via the Web.

SERAPHIN: Yes, definitely. But let me just qualify that a little bit. So DWA is a little bit different than the traditional template in that we've put in some mechanisms to allow all our design elements to be stored in one place, that of saving the cost of replicating those design elements into every database.

But other than that, we still do us this normal Domino design elements to build our screens. So as far as how we've been leveraging, you know, Domino to build DWA let me just go back to when DWA first came out.

So, you know, we came out back in July of 2001. And at that time, you know, the term Ajax really wasn't around. At that time, those technologies that are today referred to as Ajax were really at that time known as DHTML. So dynamic HTML.

And dynamic HTML really talked to taking HTML and then treating each of those HTML elements as an object. And being a object then, those elements had properties and methods and you could do very interesting things with them once they're on the page.

So there was also a very powerful property called [inter] HTML that let's you blast other
HTML elements into an existing element on the page. So you could have, you know, bring in a whole load of new UI into a particular area of the screen by just using this [inter] HTML method.

So at the time the DWA, we're kicking off the DWA, you know, the state of the art at that time for rich Web experiences was really shove an applet on a page and that applet gives you, you know, that gives you the rich experience.

But the big problem with those applets at the time was that the applets were really very heavy. So it incurred a huge download cost when you got to the page. And then the applets really never fit in with the rest of the page. They usually, you know, would end up sticking out like a sore thumb.

So the whole concept of the dynamic HTML, so the interactive HTML, was very appealing and allowed for the possibility of building a much richer experience. And that was really sort of the basis of Domino Web Access was to build, to leverage that technology to build a cutting edge UI on a browser that would try to rival Windows applications.

And so towards that end really what we're doing is leveraging, you know, HTML, Java Script, ESS, XML and the various DOMs, the HTML DOM and the XML DOM.

And so earlier I mentioned read view entries, URL command. So DWA actually leverages that to a great deal to do its views. But rather than do a view applet, we did all that by manipulating HTML elements to bring in that first list of entries in a view, as many as it will fit on a page.

And then when you page down on the scroll bar, we go out and we retrieve the next
chunk of entries. Or if you arrow down will get the next entry, will check to see if the next entry is in our internal cache. If not, will go get the next chunk.

So several arrow downs won't result in a transaction. But then when we run out of stuff in the cache, will go out and get more. So again, much richer experiences so you take in a lot of the process, putting more processing on the client and doing these asynchronous and sometimes synchronous transactions to the server to get data.

Another example is like you're in a view and you delete something. So in DWA we don't load a whole new page with the new set of entries again -- rather, we'll post the transaction to a hidden I frame and that'll cause that entry to be deleted on the server.

And then once we detect that that's happened, we'll just remove that visually from the screen. So using various HTML, DHTML operations and then that's it. So those are some examples of how we do this type of rich experience.

JANSEN: I see. So, you, this is, you were just giving examples of how you've implemented this within DWA but really, any developer who's building a custom application can go ahead and use these same approaches to provide a richer end user experience, correct?

SERAPHIN: Yes. So part of this whole business experience is really just knowing these technologies and you know, JavaScript and HTML and CSS and, you know, you can use pass through HTML or a mixture of traditional Domino rendering along with coding to use the XML HTTP request object which is what facilitates those asynchronous transactions.

JANSEN: Okay.
SERAPHIN: So DWA early on actually used something called an XML island. And the XML island was introduced in '95. And underneath the covers it used XML HTTP request. Then later when Mozilla was evolving, they also supported the XML HTTP request, properties and methods. And when we first supported Mozilla we started coding to XML HTTP requests as well.

So another developer, other developers can do the same thing. And then in order to do the asynchronous transactions what you need is a way to do something on the Domino side where you make a request and you get back some data.

Now Domino provides some standard ones that we mentioned before like read view entries and read design. But you could also use an agent and have that agent return either XML or JSON or whatever you want to retrieve, an HTML fragment, within this XML HTTP request.

JANSEN: Earlier when you were just giving that reply, you made the correlation between DHTML versus Ajax and kind of inferred that DHTML was the predecessor to it. Are there differences between DHTML and Ajax? Or are they really one in the same?

SERAPHIN: In my view, they're really one in the same. The one thing that Ajax also...DHTML really doesn't reference the XML issue to the request object per se, although that was something that was around at that time and used by a lot of the HTML developers.

But Ajax sort of calls that out front and foremost and says, that's the real significant thing in order to facilitate not updating the page and making these transactions back to the server in the background.
But there are other techniques to do that as well like hidden I frames, and that's also considered part of Ajax and that was a very common thing used by DHTML developers back then.

JANSEN: So when would you say that you started, outside of DWA, when do you think you started to first see developers implementing Ajax or DHTML like functionality in their Domino Web applications roughly?

SERAPHIN: I mean definitely when we first released like 2001 and I recall giving various talks with LotusSphere 2001, 2002, and there were strange talks because the talks were more on just how do you do this stuff using HTML and JavaScript because they're not really necessarily tied to the Domino server.

And the Domino server allows you to serve up all these things that you need and then, and provides a wide variety of design elements where you can house these things.

And at that time, there were already other people who were building comparable view widgets like we had done and, you know, using them in their apps and also showing them as part of that, the business partner, there's that track showing, you know, Domino applications.

JANSEN: I see.

SERAPHIN: And we had lots of conversations in the hallway about, how did you get around this problem or that problem. And I think the Domino community has been doing this for quite a while as well.

JANSEN: Right. I think that's a great point. We've certainly seen in various community forums and things like that a lot of conversation about this over the past
seven, eight years. So I guess you can almost kind of tie this back to this shipment of R5. Would you agree to that?

SERAPHIN: Well, we first came out of 5.08 but in 5 it was more Java applets but I think 5.02 was when we first documented the read view entries call. So I would think that people started playing around with it at about that timeframe.

And that's sort of when we first started doing development for what eventually shipped in 5.08. I think that's probably the right area where people started thinking, hey, this is, you know.... And that's when IE was first out and there was a lot of technical articles about the Trident engine which is part of IE 4, 5.

JANSEN: I see. Okay. Well I think that's basically a lot of good information about how we are leveraging, we have leveraged Ajax in the past and particularly in DWA. Are there any things that you want to point out that are coming in Domino 8 with respect to Ajax? Or, Domino or DWA in particular?

SERAPHIN: We're just wrapping up 8.0 now and I think people will really be happy with the Domino Web Access experience in 8. So the UI now, it tries to be much more in sync with Hanover, particularly in the mail area.

We haven't really invested that much in the calendar area yet because DWA had a pretty good calendar experience compared to Notes. So now with Hanover, Notes has got a really, Hanover has got a really great calendar experience.

But the mail experience now, DWA exports a preview pane. So the preview pane is a huge productivity improvement. And I think people will really like that, especially the side preview which Notes never even had before and is being debuting with Hanover.
And in addition to that, so now with the side preview, we need to support something that's called a narrow view. So before, Notes always had views where things are very tabular and each column goes down and, down vertically and there's, just keeps stretching out to the right.

And that really doesn't play well when you want to get into a narrower space. So there were various view design attributes added in 8 to Designer to flag certain columns as ones that would stick around if it ran out of room and which ones would wrap down to a second line. And so DWA adds support for all of those within its view widget and so now renders things really nicely.

Other things in the Web 2.0 umbrella...let me get back to that. So also in DWA 8, you'll see other things that we've been doing.

One is like when you switch between different folders in the mail view, if you switch between inbox and all documents, something that mail users frequently do, in the past in DWA we would reload the page even though the page was really using the same design elements in order to render it, but some of the URL organs would be different sort of identifying what the title of the new view or folder was.

In 8 we no longer reload that page, rather we'll just update all the different areas of the screen. So it's much more Ajax, so it's now, so before our Ajax was really when you were scrolling in a view now even when you flip between folders it's the same page that's loaded and we're just updating the areas. So it yields much improved responsiveness and a much better experience.

Other newer Web 2.0 things we're adding are, and this is actually premiered in 7.02 but
may not be as readily apparent. It requires some ini settings to turn them on, but we actually support Atom and RSS feeds to your inbox.

So if you turn on those ini settings -- and you can look them up by doing some searches on the Web -- iNotes underscore WA feeds equals RSS, and comma Adam and you can do whatever combination you want.

In 8 will display a little feed indicator right next to the title of the inbox to just let you know that this, the feeds are enabled. And then when you select that it'll generate that feed URL and the browser will display the feed in the way that the browser displays it.

But you can then put that into an RSS reader if you wanted to and in effect get notified when you have new mail using a feed reader. So it's a different concept. The one caveat is that we do have secured feeds so you do need a feed reader that can manage passwords and such.

JANSEN: I see. And just to bring it back to a developer's perspective, are these features that they could go ahead and utilize within their own custom application to some degree, or?

SERAPHIN: Not really, but people have been doing feeds in Domino for a long time. And another way that you could generate a feed and what I've seen a lot of developers do is to create a custom view and within that view the column formulas can be used to emit out, you know, fragments of the XML stream that relate to that feed.

And then using other passthrough HTML before and after the embedded view on a normal page or a V template page, then in order to construct that feed. So that's one way to do it.
You know, DWA actually does it a different way, and I would love to make this type of formula available for other developers and we're working on that in the future hopefully.

But DWA has a really powerful formula that internal DWA formula uses called H view. And we then map it to this, we have this new thing called DWA tags in 7 and 8 which are all covered in the DWA customization articles that I've written and spoken about at LotusSphere. And so that maps to a tag called DWA column format view entries.

And the idea we have is that you can specify, pick out columns by programmatic name and use those values in any, and then specify a pattern string.

So you specify a pattern string and you say within that pattern substitute the subject column here, the value of the subject column here, the value of the date column here. And then we provide some attributes to tweak getting the date in various formats that are required for RSS or for Atom and things like that.

And so we use this type of custom formula to do it. People could look at our SRSS and S Atom forms in the DWA forms file if you want to take a look at that and let us know what you think of it. So I'm trying to push for future evolution of all Domino Web apps to leverage some of the types of capabilities that were added for building DWA.

JANSEN: Excellent. You just mentioned that some of the things that you brought added to DWA in the 7.02 release. I'm aware of a couple other things that we did in Domino 7.02 that definitely provides some Web 2.0 capabilities.

And one certainly is the fact that we shipped a blog template with Domino 7.02. We have a very active blogging community for Notes and Domino out there. And I know
many people take advantage of this blog template.

SERAPHIN: That's very much a part of Web 2.0. So a part of Web 2.0 is really that there's this whole community aspect to the Web whereas previously with Web 1.0 it was just people going out and just digesting things that corporate sites would make available to them.

And with Web 2.0 the key is really that it's a participatory Web so that people can go enter their own musings within their blogs but not only that, others can go read the blogs and add their own comments to them as well.

So the whole commenting aspect of a blog is also a critical part of that. And then wikis are also another part of the Web 2.0. But, yes, the blog template we ship with Web 2.0.

And also so the other bit is if you're going to go off and read these things in different places about blogs, you may, you know, want to subscribe to blogs of certain people that you are curious, you're interested in their sayings. And so feeds are the way that you do that.

And so one of the other things we shipped with 7.02 is a standard feed template. And what it does is it allows a mechanism for taking any Domino database and generating an RSS feed from it.

So you in effect create different feed documents and each document can go point to a particular database and you can go browse what views are there and which view you want to make available as a feed and which column entries you want or, and that's the best way to do it.
But it even has the capability of pointing to specific fields in a document that happen to be in that database. That's not going to perform as well, but you can do that as well.

So that's a very, I think people will find that to be a very powerful tool as well, to feed enable various Domino databases without having to go write custom agents or those custom views being added to all sorts of different databases.

JANSEN: You mentioned something a little while ago about, you know, some of the talks you have given at LotusSphere and I know that last year, I believe, you did a session on, you know, how to basically, you know, modernize your Domino Web application.

So you may have developed a Web application in the past, and I believe you shared some thoughts about, ideas on how to go ahead and modernize that Domino Web application using some of the things that you've already talked about in today's podcast.

SERAPHIN: And so in that talk, one of the things...the purpose of the talk was really to just sort of.... It was almost a two part talk and the first part was really to try to introduce a newer technology that we're adding which is extending read view entries to support JSON.

So even DWA up until now has been using, retrieving all the view data in XML format. But among the Web 2.0 community, JSON is a really popular format. And the whole advantage of JSON is that it's bringing down data in a format that's easily digestible by the JavaScript engine without requiring any additional code.

So with XML you need various code to go parse that returned XML and pull out the nuggets of data that you want. With JSON it just becomes, you just do an eval and it
becomes part of the JavaScript object and you can just access the things using normal JavaScript access notations. And so that's really powerful.

And in our experimentation we also found that it had various performance benefits as well on certain browsers. So that was the first part.

And the second part was really looking at the Ajax libraries. So, and there's one particular one that IBM is endorsing, DOJO. And it's just taking a look at DOJO and seeing how you can use those, that library to build a much richer experience.

And I think it was really going to, if you look at the traditional Domino model was the frameset. And you know, the frameset really isn't a popular in...or visible frames that aren't that popular in the Web anymore.

But rather, you lay out panels and you may put a splitter and have another panel. And these DOJO widgets and the HTML markup that they support allow you easy ways of doing things like that.

And so the presentation went into talking about the class libraries and particularly DOJO and what type of added value it brings. And the added value is really that there's a lot of programming now that a JavaScript developer has to do to build these rich experiences.

But the good thing is that a lot of people have written a lot of the supporting code and packaged them up into nice functions and nice class libraries that take care of things like worrying about what to do with different browser releases or different types of browsers.

And so those can be leveraged so that each developer doesn't have to overcome all those same hurdles about how do you get this to work on Safari versus Mozilla or
Firefox or IE.

And so we showed a nice demo about how you might use DOJO in order to build a much richer view and also get assistance for inputs for input screens like a date picker next to a date field. Or have a type down combo box for a combo box.

And in the future, I would hope that we’d…and some of the things we’ll talk about, we can talk about it later what we’re going to be thinking about doing.

JANSEN: So I think you certainly touched on a number of technologies here that a developer wants to go ahead and build these kind of rich Internet applications for Domino would want to learn. Obviously it sounds like, you know, Java Script, XML are two of them. Maybe you could just kind of recap what skills a developer would want to develop.

SERAPHIN: Yes, no, definitely JavaScript is key. XML also key. And also HTML, a really good understanding of HTML, different elements and then also what methods and properties are available for each of those elements. But the good thing is there’s also good reference material out on the Web on this as well.

Also, and then so the methods and properties sort of relate to the HTML DOM. So really the HTML DOM. And then other technologies are things just to understand what JSON is.

XXLT transforms are also very popular. So that’s taking XML and using this markup language that defines a style sheet and the style sheet sort of gets applied onto the XML and it results in some other third textural thing and that third textural thing may be HTML mark up, it might be XML markup, it might be JavaScript and so that’s a powerful thing
because what happens is that Java Script is an interpretive language.

And so it being an interpretive language it's going to be slower when you go to a different JavaScript calls. But if you can call into a single Java...method that then has implemented all the stuff natively, then that's going to be inherently faster. So people use XXLT transforms to sort of transform the read in XML into something else in a much faster performant manner.

JANSEN: So before we end the podcast I was wondering if you could kind of share some thoughts about where we are going with Web 2.0 for Domino in the future.

SERAPHIN: Sure. So let me first plug in some stuff that I'm, that we're doing in DWA. So the first thing is I wanted to sort of mention that in 8.01 we are planning on releasing a new low speed UI for DWA.

And initially it's going to be constrained to having just mail and contacts. But really it's leveraging various Ajax techniques to the max.

So here everything is asynchronous. People will find it very responsive and it's really focused on a very low download in order to get your initial inbox viewed.

So somewhere along the mark of like 50 to 70K and that's it of download in order to get it displayed. And so that was one thing I wanted to mention. And also no opening new windows, everything in context.

DIV dialogues are another thing that's a new trend for Ajax applications. And Sparkle will do a lot of that. And even DWA 8.0 starts to have a couple of dialogues implemented as DIV dialogues. And you'll see more of that in future releases of DWA as
Now going back to other Web 2.0 things and how are we going to leverage that for Domino application developers. And I think a lot of this I can't say are definitely committed, but definitely things that we're investigating and exploring and hopefully will be available in one of the future releases.

So the first was what I alluded to earlier. You know, I think it's critical that we have a JavaScript class library that we ship right with Domino and we improve our various fields so that they can leverage these class libraries. And so we'll be exploring ways of how to best do that by improving designer and the Domino Web engine.

In addition, the Domino Web engine renders lots of rich content within Notes rich text fields. And you can definite do a better job on some things. And so we'll be looking at trying to improve that rendering and giving better, more Ajax experiences.

And back to feeds, we introduced that feed template which allows you to set up a different database through these feeds. That's using agents in order to generate the feeds underneath the covers. We're also looking at adding just easier feed generation right within any Domino database.

So maybe various view properties that help you identify which fields or columns map to particular entries within a feed like the date or the subject or the author and then allow the feeds to be generated much more easily in that manner.

Other things that I think are trends that are happening out on the Web and we're trying to figure out how to better adopt them are, you know, some people call 2007 the year of the widget.
So you know Google gadgets are incredibly popular and [confabulator] was...so Yahoo's
got their own widgets, and so people are building HTML fragments that are usable and
useful within a Web page.

And I think we’re trying to figure out how we can bring those and make them leveragable
for other application, for other applications that are served up by Domino whether it be
DWA or other ones.

And then models to aggregate pages together and try to get more symmetry with the
Notes client. But those are just areas that we’re investigating, and hopefully we’ll make
some progress on coming up with more consistent ways of doing that.

JANSEN: Well, thank you very much. I think you shared a lot of really good
insight here in today’s podcast about how Domino has had a history of support for Web
2.0 technologies and continues to evolve to support that theme.

So in closing I just want to thank [VINOID SERAPHIN], the chief architect for Domino
Web access for coming and sharing his insight on how Web 2.0 applies to Lotus
Domino.

Please remember that this is one podcast in a series on how various Lotus and
WebSphere Portal offerings leverage Web 2.0 and that the other podcasts are available
on developerWorks. Thanks for listening. Have a good day.

[END OF SEGMENT]