Jeff Calow on new Web technologies in Domino 6

Interview by Christie Williams
with Laura Rutherford

Level: All
Works with: Domino 6
Updated: 02/04/2002

As an IBM senior technical staff member with a focus on Web technologies in Domino 6, Jeff Calow helps coordinate development across groups and keep everything in sync—and he gets to do some development work as well. In this interview, which has been updated for Pre-release 1 of Domino 6, he discusses the new Web technologies in Domino 6 and how to work with standard technologies like J2EE.

Tell us about the new Web technologies in Notes/Domino 6. What are they?
We've done a bunch of things in Notes/Domino 6 to enhance the Web development and deployment experience. Notes/Domino 6 includes everything that was available in R5, including the R5 servlet engine. Then, for the developer, we've improved both Domino Designer and the Web application server by adding many more features for more powerful, more beautiful, and better performing Web applications.

In Designer, we've added an HTML editor with color-coding and auto-complete for HTML tags. We've added WebDAV support to enhance your ability to share development work between Designer and third-party Web authoring tools. Designer also now supports design locking to ease team development. For those using Internet Explorer as their browser, we've added the ability to use the built-in Internet Explorer rich text control (also used in iNotes Web Access) instead of the applet, to enhance the rich text editing experience. And we're supporting layers, cascading style sheets (CSS), JavaScript libraries, shared HTML "snippets," and optionally generating DHTML/JavaScript for sections so they can show/hide without doing a round trip to the server. We've also added a new database property that can restrict the NSF from being opened from the Web, without having to mess with the ACLs.

On the server side, we've rewritten the HTTP server and then added the ability to plug the Domino HTTP server into third-party Web servers (including putting a firewall between the Web server and Domino). We've enhanced the Web site and virtual host/server administration and extended the DSAPI plug-in support to make it easier to write plug-ins to the Domino HTTP server.

The HTML generation engine has been enhanced to be more standards compliant (for example, with doctype header on returned pages and other small things), as well as providing the ability to have the pages generated in XHTML. XHTML generation is enabled by adding &outputformat=XHTML10 to the requesting Domino URL.

We've also added the ability to set regional display preferences (things like currency, date format, and so on) and time zone using a Web page that writes to a persistent cookie. The self-administration page is accessed using a ?OpenPreferences command on the Domino URL.

Finally, for J2EE developers, we're heavily testing the Domino Objects for Java with J2EE application servers and adding a custom tag library that makes it easy to access and update Domino data from a JSP page.

Jeff Calow

What do you get with the rewritten HTTP server?
You get HTTP 1.1 persistent connections and improved session handling. There's also better denial of service (DOS) attack handling with more administrative control over the number of path segments, max header size, URL length, and so on. You can also do IP filtering with wildcards by having access or deny lists based on IP address.

Tell us more about the third-party Web server plug-ins.
One area where we are leveraging our relationship with the WebSphere team is by reusing their third-party Web server plug-ins. This allows a third-party Web server like IIS or Apache to be the one "facing" the browsers and serving up static content (which is their speciality) and have all NSF requests be forwarded to the Domino Web server. The plug-ins use HTTP to communicate with the server, so the third-party HTTP server can sit in the DMZ with the plug-in communicating with the Domino server sitting inside the firewall. The architecture and coding is in place to take advantage of all the WebSphere plug-ins, and we plan to certify them over time. For "dot 0," we will certify with IIS, and we plan to certify with other Web server plug-ins in future maintenance releases.

What is WebDAV and how will Domino 6 support it?
WebDAV stands for Web Distributed Authoring and Version. Generally, it's an extension to HTTP that allows people to check-in and check-out things. It uses HTTP puts and HTTP gets, and includes some locking mechanisms so that you can say “Oh, I've got this thing checked out or not.” It’s a standard from IETF, which stands for the Internet Engineering Task Force. It's still fairly new, and it's going to be changing. Different products support it. Microsoft's very heavy into WebDAV. Macromedia Dreamweaver supports WebDAV as a source control system and for deployment. The importance of it for Notes/Domino 6 is in the context of third-party tool integration.

You want to be able to have someone with Dreamweaver interoperate easily with someone who is working in Designer. Obviously, Designer only knows about NSF files. So we need some way of getting stuff off the file system into these NSF files, using something that's standards-based so that the Dreamweaver people don't need Notes running too, for example.

WebDAV does this. It sits inside the Web server—it's actually part of the Web engine in the Web server. It responds to a particular protocol and instead of storing the files for WebDAV in the file system, it stores them in an NSF file as a file resource, which you can see in Designer.

In addition, let's say that you've got creative people working on images and developers working on a Web application for the same Web site. Well, both of them could use a common NSF repository for their work while using their favorite tools to actually design and develop. The designers go in, check out the images, change the images, and so forth; and the developers are doing the same on their particular aspect of the project. The meeting place is all in the NSF. The mechanism for hooking in all of these third-party tools is going to be WebDAV.

We've been hearing a lot about J2EE. First off, what is it?
J2EE stands for Java 2 Enterprise Edition. It's a standards-based programming model that developers can use to get more control over the exact HTML that is presented to the browser. It includes servlets, JSPs, EJBs, and other technologies. The Domino Objects for Java can be used to access NSF data in Domino from all J2EE components. If you are developing JSPs, you can also use the Domino Custom Tag library.

There are currently many vendors that provide J2EE application servers like IBM's WebSphere Application Server, BEA WebLogic, Sybase EAServer, Macromedia JRun, Sun iPlanet, and others. In addition, there are various open source projects that provide pieces of J2EE, like Apache Tomcat for servlets and JSPs, and JBOSS for EJBs. The main difference between the open source projects and the commercial products is "industrial strength," administration, scalability, testing, customer support, "connectors" to back-end systems, and cost. Using J2EE doesn't lock you in to any single application server vendor, and you are free to trade off what aspects are important to you when choosing one.

There are many benefits to a standards-based programming model. Since it is a standard, there are a large number of books and training courses available. In addition, the skills learned are easily transferable from one server to another. It's also easier to find trained developers to work on projects.

So what does J2EE mean for Domino developers?
Current Domino Web developers can leverage their knowledge of DHTML and JavaScript and bring these to an environment where even more control is possible over the results. The Java programming required to build JSPs is very much the same as doing JavaScript. Doing servlets has about the same complexity as building Java agents. When writing in Java, developers can access Domino just as a data store or take a hybrid approach where the JSP/servlets are mixed in with Domino Web applications. I was talking with someone at Lotusphere who is building a J2EE application that uses Domino for part of the rendering of the application and servlets/JSPs for the rest, based on a J2EE application server and Domino R5.

That said, I think it's important to note that Domino developers are not *required* to move to J2EE. The Domino Web application server will continue to exist and be enhanced. Today, it is probably the fastest and easiest way to develop data-driven Web sites. Domino 6 Web Administrator is an example of the kind of rich application that can be developed using the Domino Web programming model. But you pay a price for this ease of development. Domino takes care of generating most of the HTML (embedded HTML aside) and most of the processing. You are, therefore, constrained by the Domino model and have less control over the *exact* HTML that is generated. Some Web designers who want complete control are willing to trade off application complexity to get that control. So it all comes down to a choice. And the choice can be made differently for different situations.

How does WebSphere fit into the picture?
IBM's WebSphere Application Server (WAS) is a premier J2EE application

server that can be used to host any J2EE compliant application. It is fully J2EE 1.2 compliant, supporting servlets, JSPs, EJBs, and all other J2EE standards. In addition, the WAS team tracks the evolving J2EE standards and will have a J2EE 1.3 compliant server out later this year. Also available from IBM are a set of J2EE development tools that are closely integrated with the application server. They make up the WebSphere Studio family, and one of the first tools available is WebSphere Studio Application Developer. It has everything you need to develop J2EE applications including a Java editor, a WYSIWYG HTML/JSP editor, local and remote debugging, deployment, and much more. It also includes the ability to debug JSP page execution.

WAS is integrated with Domino in multiple ways. The WAS servlet engine can plug in to the Domino HTTP server via DSAPI. WebSphere can use the Domino Directory for authentication via LDAP. We also have the ability to do single sign-on (SSO) between Domino Web applications and WebSphere. This has all been available since WebSphere 3.5 and Domino 5.0.5.

Domino R5 also shipped with a version of WAS 3.5 "in the box," and R5 is currently shipping with WAS 4.0, Developer Edition. The exact details of the Domino 6 bundling options with WebSphere have not yet been worked out. There are many options available to us, and we're exploring which ones make the most sense.

What about using Apache Tomcat?
Apache Tomcat is the reference implementation for the servlet and JSP specifications. There are two main versions: Version 3.3.x, which implements the Servlet 2.2 and JSP 1.1 spec, and version 4.x, which implements the latest specs (Servlet 2.3 and JSP 1.2). Tomcat contains only servlets and JSPs and is, therefore, not fully J2EE compliant. Configuration and administration is done via the file system and XML configuration files. There is no support for clustering. There appears to have been a DSAPI plug-in for Domino in one of the earlier releases of Tomcat, but it has not yet been ported to 4.x. The **Web site** says that there are plans to do a Web services and Web-based administration tool "at some point," but I haven't seen any code related to this yet. They do have a Web application that can be driven by scripts to install/start/shutdown other Web applications.

For people willing to trade off manual configuration and no real support for zero cost, Tomcat may be an alternative application server worth looking at.

An important note for anyone thinking of writing an automatic configuration file generator—the XML configuration file has changed dramatically from 3.2 to 3.3 and then to 4.0. (Since 4.0 is a complete rewrite of the servlet engine, it's not unexpected that the configuration file would change.) Hopefully, things will settle down for 4.0, but you never know.

OK, that's the hosting side of J2EE; what about developing applications?
As I mentioned before, there are lots of third-party tools out there that make it easy to develop J2EE applications. There are tools from IBM, Borland, Sun, WebGain, Macromedia, and others. I personally use my programmer's editor to do all of my development; and for debugging, I use IBM WebSphere Studio Application Developer or System.out.println(). App Developer allows me to debug both JSP and Java code at the same time, which makes it easy to work at all levels of the application.

In the short term, these tools all offer nuts-and-bolts programming with some wizards. For the medium or longer term—at Lotusphere, Al Zollar announced a RAD J2EE tool that we are currently working on. We're all really excited about creating this new RAD environment, but it's too early to share any details other than: "It will be built on top of the Open Source Eclipse framework and will be tightly integrated with the other WebSphere Studio tool offerings." One of our primary goals is to make it easy for existing Domino developers to develop J2EE applications, without having to do a ton of coding.

Having tools is good, but you still need to know how to go about building the applications. There are a ton of books out there talking about servlet and JSP development. One of the developers I work with showed me a comprehensive book that was at least three inches thick. In addition to this, there are several **IBM Redbooks** available on doing J2EE development with Domino, and these will be updated for Domino 6.

For those developers doing JSP development, the Domino Custom Tag library also provides an easy on-ramp to getting at Domino data, while still allowing full control over the layout and presentation.

Before we talk about the Domino Custom Tag library, can you fill us in on what happens with JSP and servlet applications that were developed in Beta 4?
There is no wasted work. These applications can be exported from the NSF and run in any standard application server. The easiest way to do this is *before* you upgrade to Notes/Domino 6 Pre-release 1. You can use the nwar utility to extract your applications to the file system and then package them up for the application server. On the server where the NSF is, use:

```
nwar -e<nsfname> -d<dest dir>
```

For example:

```
nwar -emywebapp.nsf -de:\myroot\mywebapp
```

This will take the application out of mywebapp.nsf and put it in e:\myroot\mywebapp, preserving the directory structure. (If you want a list of all the arguments, use nwar -?.)

If you forget to do this, you won't lose your JSP's or anything imported with nwar. All your JSPs are still in the NSF as file resources. You can see them in Designer under Shared Resources/Files. But you'll have to export them manually using the Export action. Alternatively, you could also use WebDAV to extract the files. (This will work in both Beta 4 and Pre-release 1.) Since servlets developed with Designer were stored in a different format (like Java agents), they're not exposed via WebDAV or nwar. So you'll need to export them from the Beta 4 version of Designer.

What are the Domino Custom Tags?
In the interest of trying to support template developers who are just HTML savvy but who aren't necessarily Java programming savvy, JSPs have provided this capability called custom tags. They're tags that look like HTML tags, or really more like XML tags, and you embed them in the midst of your HTML. The processing for that runs on the server side, and whatever the tag happens to do, the server generates the HTML and sends it down to the client.

For example, our tag library will loop over a view, so you can go through all of the entries in the view, and have the body and the detail line to show what you want to show for each detail entry in the view. We've got all sorts of different tags that do things. We've got security tags. We've got tags for doing forms, so similar to the way you build a Domino form today, you can build the same kind of form in HTML. One of the key features of JSPs is the ability to support custom tags. If you get some other custom tags from another third party, you can use some within your JSP as well. So JSP supports any number of custom tag libraries to be used on the same page.

How do the custom tags work with development tools?

Most J2EE development tools that support JSP development have support for custom tag libraries. Since we followed the standards, you can get the full support of the tool with our tag library. We have tested our custom tag library with WebSphere Studio Application Developer (because we use it internally). I'd like to hear about it on Notes.net if anyone has any trouble using the tag library with any particular tool. (And you should probably also notify the tool vendor.)

Can you give us more details about how the tag library works and what you can do with it?
The primary goal of the tag library was to make it easier to access Domino data. What is it that makes up Domino data? You've got documents and then you've got collections of documents that are views. The primary set of tags is giving you access to views and documents. With the access to documents, you want to be able to both read the data and to build user interfaces that give you an ability to update the data without having to write a lot of code to do the updating. If you look at the back-end classes, all the code that you need in order to take the data from a form, pull it out, slam it into the document, save the document, and do validation—all of that is a fair amount of code.

So one of the largest chunks of our tag library is the ability to build forms. There's the Domino form tag, and that, combined with input tags and other tags, let you basically do things in a way that is similar to what you do in Designer. You drop fields onto the form and add buttons, and the tags let you build a fully functional form with no Java coding at all.

You write something that looks like a straight HTML form. And what you get on the other side is something that supports full updating and viewing of the data; it supports both display mode and edit mode. We're using Notes as a loose model, so the results are similar to Notes. We're not trying to make it exactly compatible but we are using it as a model, because people are already oriented towards doing applications in that way. We wanted to make sure that the skills would transfer, knowing that the target audience is people who had done LotusScript and worked with a Notes-type of development.

And keeping in mind that we don't want to generate a lot of HTML, there's not a lot of magic HTML generated. There are a couple of tags that do some magic—to do paging for example—but most of the tags basically turn into either one HTML tag or no HTML tag. They turn into just data that gets displayed on the screen.

What else can you use besides the form tag?
One of my design goals when I go and design something is that I always want to have a graded path so that people can have a lot of magic and lose some control, and then a little bit less magic, without having to jump all the way down to the Java classes.

So the document tag allows you access to Domino data in a formatted way, but it doesn't do any input and validation processing. As part of the form tag, we also have validation tags that do validation and all sorts of things like that. From the next step down, if someone wants to do some more of their own processing but wants help getting at the Domino data, we have a document tag that's completely non-visual, lets you get at items, and lets you do set items as well. You have a get-item, set-item, type of tag in the context of a document, but you still don't have to worry about navigating to the document and all that.

And then for the next level down, each one of the tags exposes the corresponding back-end class. For example, take a form tag. You can get at the document that is represented by that form and do some more advanced manipulations. So if the tags don't do exactly what you want and you want to have a little bit of tweaking—but not so much that you want to throw out the tags completely—we give you an easy path into that.

And that philosophy goes all the way down. For example, if you don’t like our form tag and want to do all of that yourself, we still have the session tag to set up the session, shut down the session, and make sure that everything gets cleaned up properly. So we can do a lot of automated stuff for you—navigating to the right document, navigating to the views, looking up the views, getting the scenario message if you don't find the view—all that code that you'd have to write, we can do for you.

Then you can just do your own view processing in the midst of it. And you can trade off magic functionality for really heavy-duty control. In our sample applications, there are certain places where we've gone beyond what the tags can do. We've plugged in our own little Java back-end class code, because we wanted to do something that was very particular to that application, that didn't have broad applicability to the tag audience. This was done in the context of an application that was fully made with tags as well, so we got the rapid application development, except for the tiny little places where we wanted to go to the bit level of control.

Any other interesting tags?
We have a tag for doing full-text search; similar to views, you get a list of documents. Another area of tags is access control. So obviously, we take advantage of the reader and author fields, and you can have sections of your page that are conditional, based on what the browser user's access control to the current database is.

And the last thing that we've got is a utility tag library. It's not really related to Domino data at all. It's just things like ifs and conditional stuff, some browser capabilities, and the formatting tag—things that are not specific to Domino data but are things you might want to do. Heavy-duty JSP developers might have their own favorite utility tag library, and they can use theirs instead. But we have the utility tags in case you need them.

What about security?
The custom tags are written using the Domino Object for Java and thus participates in the full Domino security—readers fields, author fields, access control, roles—all of that is exposed. The tags allow both authenticated and anonymous access and can plug in with single sign-on (SSO) on WebSphere. SSO is not currently supported by any other application server, but based on some feedback from Lotusphere, it's something we're going to look at—but we can't make commitments at this time.

Are there going to be any example applications using the custom tags?
We are going to provide two extensive sample applications that were written using JSPs and servlets and the custom tags. These applications will not be supported, but they are good examples of how to build moderately complex applications using the J2EE programming model. The applications are a Web discussion application and document review and feedback application. There is also a little bit of documentation that describes how these applications were built and how to install them into WebSphere . Our current plan is to post the applications in the Notes.net Sandbox shortly after Pre-release 1 is posted.

Can you use any other application servers with the tags?
Since we built the tags using the JSP 1.1 standard, without any container-specific code, they should work in any application server that supports the standard. Given the large number of application servers out there, we will probably do the majority of our testing on WebSphere, but we don't expect problems on the other application servers. I've talked to some beta customers that have been successful running the tags in Tomcat and BEA as well. Issues with running the tags in other servers should be reported through the normal channels.

How will I know how to set up my application server to use these tags?
We'll provide documentation that describes how to set up various popular application servers to use the custom tags. We haven't determined the exact list of servers yet. For those servers that we don't cover, we'll have a set of generic instructions that can be mapped to a specific server using that server's documentation.

Now that there are new choices when you go to develop an application, how do you choose between doing a Domino application and J2EE (JSP and servlets) application?
In my mind, it comes down to how much time you want to invest in writing the application and how much control you want over the resulting HTML. Domino Web applications excel at getting data-driven Web sites up and running fast, but you need to live within the boundaries of what Domino generates. For a large number of applications, this is a good trade off. Notes forms and views can provide a lot of functionality with very little coding. Some of the places where Domino excels are hierarchical and categorized views, built-in search, rich text editing, built-in security—I could keep going but you get the idea.

If you are currently writing Web applications using Web agents or R5 servlets, you probably want to move to JSPs and servlets on a servlet container that supports the more current standards. If you are writing an enterprise-grade Web application that will need to support hundred's of thousands of simultaneous users, you probably want to consider a commercial J2EE application server.

In any case, I think it's worth it for a current Domino developer to at least familiarize themselves with JSPs and servlets, so that if the need arises, they can pick the best technology to meet the requirements.

Can you give us some examples of how customers are planning to develop applications to take advantage of what's offered in Domino 6?
I talked with a bunch of customers at Lotusphere and in the **Notes/Domino 6 Pre-release feedback forum** on Notes.net, and there is a large range of applications that people are writing. Most people are still writing Notes Web applications and some are kicking the tires of JSP.

One customer I talked to was using the custom tag library to build VoiceML pages for use with IBM's Voice server. Another was exposing their consultant knowledgebase using JSPs and the tags. A third was using both the custom tags and the Domino Web server to build a hybrid knowledge management application. Just today, I was talking to the architect of an internal IBM application that uses a Notes client application to do content authoring with approval workflow, extracts the content with the XML toolkit, transforms it to HTML using XSLT, and then presents it to the end users with a JSP and servlet-based Web application. Any attachments on the content remained in an NSF database.

Any comments on the future of Domino?
Domino is not going away and is not "turning" into WebSphere. There will be a Notes/Domino 7. It will continue to raise the bar as an e-mail messaging platform, and we also have a bunch of enhancement ideas to improve the capabilities of Domino Web applications. Our future strategy is to make the collaborative capabilities of Domino and other Lotus products available via Web services and then move towards "componentizing" these services so that they can be mixed and matched in solutions. (We're calling this a move towards contextual collaboration). We also plan to write new Lotus applications based on these services. The strategic road map is in place, and we are working on the plans that will deliver on this vision.

ABOUT JEFF CALOW
Jeff is an IBM senior technical staff member with a focus on Web technologies. Jeff

has been on the Messaging and Collaboration team for a year, and prior to that, spent his 12 year career working on application development tools. He has worked on PowerHouse 4GL from Cognos, a stillborn development tool from Sybase, and PowerBuilder from Sybase/Powersoft. Most recently, Jeff helped architect and build a personalization server using J2EE server technologies at an Internet startup. Jeff's professional interests include programming language design, distributed computing, and software engineering practices. When not working, he helps to raise a family and enjoys meditating, watching movies, and reading science books. His main topics of interest are philosophy (both eastern and western), cognitive science, evolution, and psychology. He holds a BS in Computer Science and Physics from Carleton University in Ottawa.