Level: All
Works with: Lotus Domino Toolkit for WebSphere Studio
Updated: 01-May-2003

Martha Hoyt on the Lotus Domino Toolkit for WebSphere Studio

Interview by
Tara Hall

This month we talk with Martha Hoyt, senior product manager for the Lotus Domino Toolkit for WebSphere Studio, about creating J2EE applications that integrate Domino data, the Domino JSP custom tag libraries, and the Web rapid application development (RAD) components for WebSphere Studio.

What is the Lotus Domino Toolkit for WebSphere Studio?
The Lotus Domino Toolkit is a set of plug-ins for WebSphere Studio that allow you to easily connect to Domino applications and work with Domino data. It includes the Domino custom JSP tags that we shipped with Notes/Domino 6. With the toolkit installed in WebSphere Studio 5 or later, you can look at an existing Domino application residing on your server and work with forms, views, and agents within the WebSphere Studio environment. You can re-use those elements in a new JavaServer Page (JSP) or in existing transactional applications to give you a head start on building J2EE applications.

Why would a developer want or need the toolkit?
Many Notes/Domino customers have made a strategic decision to use J2EE as the platform for their business applications due to the open standards, scalability, flexibility, and open integration capabilities it provides. They are left wondering how their current Domino infrastructure and existing investments in deployed Notes/Domino applications will fit in their future IT infrastructure. The toolkit makes it really easy to continue to run and use Notes/Domino alongside WebSphere. It enables J2EE applications to incorporate key Notes/Domino functionality that simply doesn't exist on WebSphere Application Server (WAS), such as human-process workflow, email, calendar/scheduling, granular access control, and unstructured document storage and distribution. With the toolkit, the Notes/Domino applications continue to run unaltered, still usable by Notes and Web browser users, and fully integrated into other applications running on WAS. From an IT cost management perspective, this enables customers to get even higher returns on their Notes/Domino application investments because they don't have to rewrite these apps—they just reuse them.

We talked a little bit before we started about the Domino strategy and the WebSphere strategy. Can you describe the benefits that customers get from incorporating Domino data into J2EE applications?
The biggest benefit is the ability to re-use what you already have. Whenever you re-use something, you're minimizing your development costs, your data migration costs, your skill sets, and your existing application costs.

There are a lot of benefits for reuse and for not recreating code. You've already worked out the bugs. You've already worked out the user interface that you want. So, the basic strategy of putting a lot of emphasis on Domino and WebSphere integration is really about pulling in the information that you have accumulated over the years and

leaving it where it works best. But accessing it as you need it from a transactional or a purchasing application is really the biggest benefit.

We talk a lot about the benefits of the NSF as a data store. We know that a lot of customers have a lot of information there. So it makes sense to keep it there, if it's rich content, if it is image-intensive, or if it needs to be full-text indexed. Why put yourself through the stress—the re-architecture strain—when you can leave it hosted on Domino and access it from one or more J2EE applications using the toolkit?

Roughly, when can people expect general availability of the toolkit?
The Lotus Domino Toolkit for WebSphere Studio is going to ship as part of the Domino Designer applications directory in Notes/Domino Release 6.0.2.

Will customers be able to download the toolkit separately?
They can download it separately, but they'll need a technical support contract for technical support. So, they can download the code, but unless they've already bought a license for Domino Designer 6, they cannot get technical support for the toolkit.

Will this toolkit require Notes/Domino 6.0.2?
It will require Domino 6.0.2 because the JSP tags are only shipped as part of the fixed code stream. It also requires WebSphere 3.0.5 and WebSphere Studio 5.

How easy is it to install the toolkit?
Installation of the toolkit is simple because it's an Eclipse plug-in, so it copies the necessary files to your WebSphere Studio environment, and you're ready to go. Using it is pretty straightforward too because all it requires is your data directory from which it pulls applications. So, you give it a server name, and then you also give it a location for installing the WebSphere Studio plug-ins. These are the only two things that the installation needs. After the plug-ins from the toolkit are in the appropriate directory, the toolkit recognizes them. Then, you set up the Web perspective in WebSphere Studio to show the Domino view. This Domino view is different from the Domino view that our customers are aware of. A view and a perspective in WebSphere Studio is a UI that will help your toolkit. When you access your Domino application, it asks for a user name and password for authentication, but it doesn't care where your Notes ID is.

I've been reading through some of the feedback from the toolkit Beta forum. Some customers have said, "I'm a Domino developer, but I don't know what to do with these tags." Can you describe to me what features people can expect to get with these tags?
The features that the JSP tags provide are the same features that you get from the Domino Java Objects. They allow you to use the Domino Java capabilities more easily. Instead of worrying about the Java language paradigm, you can focus on which document you want to get. If you know the LotusScript object model, you can use the JSP tags in the same way.

In addition to that functionality—that's just a layer on top of our object models—we have a few tags that combine several activities into one. For example, one of the tags that you use a lot when you're dragging a new view onto a JSP generates a view loop which does several things: It accesses the database; it accesses the view; and it pulls

out a collection of documents for you—all within that one view loop tag. It combines functionality that simplifies development.

There are a number of other utility or combination tags that do more than get this document or get that view. They can make a document a category or make a document, offset it, and set it. The best way to get familiar with the JSP tags is to read through what each tag does. We have thorough documentation for the tags with the kit.

The Domino custom JSP tags originally came out with Notes/Domino 6.0, but the toolkit is shipping with Notes/Domino 6.0.2. Have any new tags been introduced since 6.0?
No, there are no additional tags in the tag library. The toolkit provides the convenience to the developer, making it really easy to use the tags in his or her JSP.

Are there plans to introduce new tags?
We are in discussion right now to add new tag functionality. The right way to do it is to add tags at the lowest level, exposing them first in our Java API, and then in our JSP tags. Then we make sure that the new JSP tags align with the new version of the Lotus Domino Toolkit for WebSphere Studio. We don't want to add something at the JSP level without adding it down lower because that's not the model that we've always followed.

This object model is basically the LotusScript object model. What sort of experience do developers need to use the Lotus Domino Toolkit for WebSphere Studio? Do they need to know Java? Will it help to have a familiarity with the LotusScript language to actually build JSPs?
No, not really because a JSP is made up of many parts—a presentation layer, HTML code, logic. Understanding the LotusScript language constructs and how you build an agent won't necessarily help you to understand what to do with a JavaServer Page. JSP tags are similar to HTML tags—with an open and close tag syntax—it makes it very easy to understand what's going on. If you can learn HTML coding, you can learn how to use the JSP tag library. It has different rules, and it's a different technology, but with a little help from a JSP reference book, any experienced HTML coder can create a JSP. The Domino custom JSP tags are directly correlated to the LotusScript and Java object models, so knowing that will definitely help.

WebSphere Studio provides a lot of documentation about writing JSPs. And there's a JSP debugger in WebSphere Studio to help you, as well as lots of other resources to get you up to speed. There are also a lot of books and online references available on JSP development because it is an open standard.

In terms of the documentation, is it strictly reference? Or are there any tutorials to go along with the toolkit?
There's one sample application that we take you through in the documentation about how to set it up, and then, what to do with it. It's not super-advanced, so a beginner can use it. To get started, I suggest importing a relatively simple view and form that you created in Domino Designer, then looking at the JSP source. This is an excellent way to begin to learn how to create the view or form using the JSP tag library.

Are there any plans to offer JSP tag libraries for other Lotus products?
The Domino JSP tag library is the first one available to date. With popular uptake and good feedback from our communities, it certainly makes a stronger case for us to create more libraries.

So if there's enough demand out there, we might see a Sametime JSP tag library that would allow people to actually build Sametime features, like awareness and instant messaging, into their J2EE applications.
I think that's reasonable. We want our development community to be comfortable adding features as they need them. JSP tags are the current technology that allows you to do that today regardless of the Web services that we have coming. This is a technical here-and-now story. I think we will see more in the future, but I can't really comment about what other products sets are doing. But certainly everybody looks very positively at the JSP tags.

Is there a benefit in porting some of your Domino applications to J2EE applications?
It's difficult to give a hard and fast answer to that question. You really have to look at the strategic plans and workload requirements for a particular application. When we talk to people about building integrated applications, we tell them that you really need to spend time thinking about each application that you're looking at because you're not going to integrate every application with WebSphere. It's expensive, not from a time perspective necessarily, but any time you put two different systems together, there's a cost. So you want to make sure you're moving or integrating the right applications.

If a customer came to you asking about migration, what advice would you give him or her to determine whether or not it's the right thing to do?
You're accessing an existing database on a server, but you're putting a new face on it—a new Web face that

happens to be J2EE-based. As far as criteria, it really depends on your long-term plans for the application. So look carefully and think about it. See what the cohorts at your organization think about the application. It is a planning activity. It's hard to provide specific criteria, except that it's going to live forever in Domino because that's where it belongs. But, that's a very strong, independent answer.

The toolkit requires WebSphere Studio 5, but after the JSPs are created, can I use any J2EE application server to host the JSPs?
The toolkit was built specifically for WebSphere Studio, but you can use the tags with any J2EE server. So the benefit you get from the toolkit is the fact that it hides some of the complexity of connecting to Domino. It generates your remote session connection to Domino and adds the JSP tag libraries to your project for you. WebSphere Studio was created on the Eclipse framework, which is why the plug-ins are so easy to build and so easy to use.

You can use the Domino custom tags with any J2EE server and in any development environment. The tags themselves are not exclusive to WebSphere Studio, but the toolkit is. If you want to build a JSP using the Domino custom tags, you don't have to use WebSphere Studio. You can install the Domino custom tags and use them in any J2EE application development environment with any J2EE server. However, using WebSphere Studio and the toolkit is much easier than using the custom tag library with another JSP development tool.

Speaking of the Eclipse framework, there are probably customers out there who are not familiar with that technology. Can you explain it for them?
Eclipse is to the open source client as Apache is to the open source server. People understand what Apache is. Eclipse is a client, a development tool, and an open source project. Eclipse has many big vendors involved, including IBM, and some big tools vendors as well. As an open source project, it provides a client UI with different components that plug into each other. So when you download the open source, you get a basic application development tool and a basic client interface.

IBM has taken that framework and added onto it. Because WebSphere Studio is built on the Eclipse framework and because Eclipse is open source, any tools vendor can plug into WebSphere Studio. Lotus is acting as a tools vendor saying, "We want to be part of this Studio product." Eclipse has a very well-defined way of adding your plug-ins to that framework. These are the benefits of it being open source—it was a very standard way of saying, "If you write it this way, it will plug in this way, and you can use it this way."

Are there any plans to support other application development tools, for instance, Dreamweaver?
Not at this time. We are focusing on WebSphere Studio. You can use the Domino custom tag library in Dreamweaver, but our toolkit is only for WebSphere Studio. If you're using Dreamweaver, you can write the same J2EE/Domino integrated application, but it's much, much harder to do without the convenience of the toolkit.

Can you tell me a little bit about what sort of server configuration is required to ensure that your J2EE server can access the Domino data?
The toolkit can work either with the Domino and WebSphere servers, co-located—running on the same machine—which is under the licensing terms of how they have to be. To run the toolkit locally, all you need is the Notes.JAR. This is the file name of the Java code used by the JSP tags. Domino and WebSphere communicate via the Notes.JAR file, so you don't have to set up anything.

If you want to access data on a remote Domino server, you need to load the DIIOP task. That's how you get to the Domino server and the objects and the latest Java files that you need remotely via DIIOP and CORBA. You also need the two tag library definition files (TLD files) and the NCSO.JAR file running remotely.

Do you have recommendations for configuring both servers on the same machine?
There's a good deal of documentation and information available. You can use the Web container and JSP engine of WebSphere with Domino 6, and we give you a license to do that—basically mandating that you have to have them on the same server.

I can run both servers on my laptop, but for production, would you want to do that? Probably not. I would point people to requirements that shipped with Domino 6. There's a lot of documentation about that.

After customers create the JSPs, is there any way to export them into a Domino database?
No, Domino doesn't run JSPs. The JSP needs to be run in a WebSphere container. Or, if you're building them with another J2EE server, they need to run on that server and to call back to Domino for objects and documents. So there's no way to export them back to Domino.

Who is your primary target audience: Domino developers who want to build J2EE applications or J2EE developers wanting to incorporate Domino data into their J2EE applications?
I'd say primarily Domino developers who understand the object models and who want to learn about J2EE and the benefits of JSPs which, technically speaking, they compile at runtime into servlets and which have some performance benefits over our traditional load and unload agents. Also, our whole next gen strategy is based on J2EE, so it's a good time to learn about it. This is a great way of getting your feet wet if you already know the object model.

We do also believe that the Domino JSP custom tags have benefits to the J2EE developer community as well. But they need to understand that Notes data storage is document-based. There's a little more learning curve around objects. It's not rocket science. There's a lot of great documentation, but I think we're primarily focused on the Domino developer community and helping them get started in J2EE, and this is a good way of doing so.

If you're an experienced Domino developer, how difficult is it to ramp up on J2EE?
There are a couple of things that you have to keep in mind: First is that the architecture of a Domino application and the architecture of a J2EE application are different. Domino and building a Domino application have always been easy to start because you can build your presentation, your logic, and your data all at once. You can update at will, and you can move things around. You can add fields and forms. And you can do all of this very quickly.

J2EE assumes that you're building your presentation—what you see in the application and the logic in the application—and how it accesses data separately, even as far as separate people building those parts of your application. You have to change your mindset a little. There's a learning curve. You have to understand that when you built a Domino form and you added a field, you were creating a data model for the NSF. Many Domino application developers think that's just how it's done. That's the ease of it, and that's wonderful.

But in the J2EE world, you create a JSP, and there's the data model which exists in your database. But you can pick only certain parts of it, and you don't have control over the data. The database administrator has control over the data store. So, you may have to change your understanding about accessing a data store that you don't have control over and that you can only pick pieces of.

Architecturally, it's a lot different. Writing logic, you have more options for what you can use—servlets, EJBs (Enterprise Java Beans), JSPs. Of course, in Domino, we have agents, forms, the formula language, simple actions, and things like that. There are differences to understand, some of which the toolkit masks. But I think that the J2EE architecture is comprehensive and allows you to go farther. It opens a door that allows you to build bigger, more robust, more comprehensive applications. Again, it depends on the kind of applications you want to build.

With these JSP tags, can developers expect to have access to all of the Domino design elements—everything from a shared field to a form?
Right now, I can only concretely say that there will probably be a subset of design elements that they will have access to. We are exploring things like shared field, replication, and selection formulas related to a JSP tag or to a J2EE-type construct. Some design elements translate well, like forms and views.

If you have existing Domino applications, are there certain ones that are better candidates than others to use if you're going to hook into a J2EE application?
You can always pick pieces of an application. If it's an application like a document library where you have a lot of documents, that's an easy application to access. If you have discussion forums that you want to search through, that's pretty easy to hook in as well. The basic templates that we provide are very easy to integrate because they're known entities. With more complicated applications, you probably want to, again, sort through which pieces of the application you want to incorporate, what data you are trying to get to from the J2EE application, and what agents you want to run in the application. So the complex applications are the ones that you really have to look at more closely to figure out which data you want to get from them.

It's difficult to say which applications are more well suited, except to say that you should start with something simple, and then move up to picking pieces from the more complex applications.

Lotus is planning Web rapid application development (RAD) components for WebSphere Studio. Will the functionality of the Web RAD components be somewhat similar to the toolkit's?
Not really because the toolkit integrates Domino functionality into a JSP. You drag and drop forms and views on a JSP. The Web RAD components do not pertain to Domino specifically, but make overall Web UI development easier. The Web RAD components really simplify building a J2EE Web application. The toolkit is about how we use Domino features in a J2EE application.

Presumably, you're going after a different audience with the Web RAD components. Who is that audience?
We're not targeting the Domino community, per se. We're targeting the skills of the Domino community: Developers who know some HTML, who have some scripting skills, but who have never written a JSP or know how to break up the presentation from the logic and from the data. How do they get started in J2EE? With the Web RAD components, all they have to do is drag these elements onto a page, build a Web page, and start with what the user looks at in an application, rather than starting with the data and adding some logic to it. That's what a traditional J2EE developer does. The skills that we're looking at would be more typical of a Visual Studio or VB developer. They often have similarities to Domino developers. It's more of the scripting-HTML-Web applications-corporate-business analysts-type developers.

Someone coming at it from the end user perspective, rather than from the data model, has data that he wants to access. How does he make it available to the user? That is a different tactic and an important distinction because the WebSphere Studio tools today do a very good job of working with the Java programmer who understands the data model. We are adding features that will make it a super creative process and a quick prototyping environment. So we're taking some of the expertise that we've built with Domino Designer and applying that expertise to the Web RAD components for WebSphere Studio.

Is there anything else to say about the Web RAD components?
We will be doing a managed Beta of the Web RAD components in Q3 2003. We'll do our best to make sure that LDD has that information, so that we can make sure that the community that you target knows about it because that's where we'll be getting feedback. We'll be requesting feedback and asking a lot of people to really kick the tires.

Are the toolkit and the other next gen products intended to target those customers who are hybrid or to encourage those who are Domino-only to start looking at other technologies?
Both, really. They're also intended for Lotus Business Partners looking to expand their businesses into new markets, particularly J2EE. I think the next gen products are really targeting our whole Domino community and reaching out to build new customers relationships too. But, certainly, it's always easier to work with customers who already have a trust relationship with us. They know we put out quality products, so they stick with Lotus and IBM because they know what they're getting.

In the next gen application development space, is there anything else that customers can look forward to?
You'll see rolling application development features in all of our products coming out over time. At this point, we're not coming out with next gen application development at once. The best thing is just to be aware of the press releases that we're doing for upcoming products. After Lotus Workplace Messaging comes out, you'll hear a story about application development. The plan is to keep people in application development over time, and in a rolling fashion. So just stay tuned.

ABOUT MARTHA HOYT
Martha Hoyt is a senior product manager with IBM's Lotus Software division and the author of the white paper, "**The IBM Lotus Technical Strategy and Domino Developer's Roadmap**."

