Brian Levine: Java applets

Interview by Susan Florio

[Editor's note: This article resides in "Iris Today", the technical Webzine located on the http://www.notes.net Web site produced by Iris Associates, the developers of Domino and Notes.]

Meet Brian Levine, one of the developers behind the Java applets in R5. If you've got questions about applets, he's got the answers. Read on to get the scoop on how these applets can help you design dynamic Domino Web applications!

What were the design goals for Java applets?
For R5 we wanted to create applets that corresponded to the major components of the Notes R5 client user interface. The reason we wanted to do this was to provide the same functionality that those components provided within the Notes client to users accessing Domino applications over the Web. These major components were the view, the outline (which is a new component in R5), the action bar, and the rich text field (which became the editor applet). We provide applets that correspond to each of these four components so that you can create Domino applications that behave the same in a Web browser as they do in the Notes client. As an additional goal, we wanted to make sure the applets would reach the broadest possible audience. We wanted them to work with any Java-enabled browser.

When you say "we", who are you referring to?
The entire Domino Applet team. This includes Rama Annavajhala and Becky Gibson who did some amazing work on the outline and editor applets, respectively. I primarily worked on the view applet and the action bar applet. If you want some additional information on all of the applets, you can download the **Lotusphere presentation** where Rama and I discussed the applets and gave a demo.

How did the team choose to create these particular applets?
They are the main components you see when you look at a Domino R5 database. You present the data in a database with a view. You can present the structure of a database with an outline; present and execute actions with an action bar; and present and create styled text with a rich text field. These are key components of the Notes user interface.

The Java applets weren't part of R4.6, but you could generate the functionality that the applets provide. Can you explain what the difference is between the ways things worked in R4.6 and the applets in R5?
In 4.6 you had the ability to generate HTML from Domino databases. However, HTML is limited in that you can't quite provide the same interactive end-user experience that you can with applets. This is because applets are real programmatic entities in and of themselves. For example, now in the view applet, you can do things like slide the column headers back and forth to resize the column. You couldn't do this using HTML. So in the case of the view applet, we allow you to use something that looks very close to the Notes view. If you access Domino databases over the Web, you get more of the Notes client experience than you ever did before. For those people who access Notes databases using the Notes client when they are at work, but use a Web browser at home, the applets can provide a much more familiar environment for them on the Web. For those people who have never used a Notes client, the applets provide a richer, more interactive environment than they would otherwise get.

Did your goals change as you went through the development cycle?
Yes. In the beginning, we planned to be a lot more conservative about the UI functionality that we would provide in some of the applets. As we started getting more feedback internally from template developers, we started adding features that we hadn't necessarily planned on adding.

For example, QuickPlace running in Netscape 4.x was the first client that used the editor applet and had the most effect on its design. One important feature that we added based on feedback from the **Quickplace development team** was support for displaying images in the editor applet. Other enhancements we added to the editor applet based on internal feedback include: the ability to understand ordered bullets (even though the editor applet doesn't provide a UI option to allow you to add them), support for preserving HTML that the editor applet doesn't understand rather than throwing it out (this allows tables to go through the applet and back into Notes), and better international support. The international support includes support for UNICODE characters, and an input field at the bottom of the editor applet that works with foreign language Input Method Editors (IME) and allows you to enter double-byte characters.

The Discussion, TeamRoom, Mail, Domino directory (formerly called the Public Address Book), and the Document Library templates all use applets. The internal feedback that this generated was very helpful. As a result of this feedback, we added support for numerous @commands to the View applet including foldering operations and a folder selection dialog box. So our goals changed in regards to the breadth of features we provided in the applets. We feel that this was a great benefit to both our end-users and third-party Domino developers.

Can you give us some in-depth information about each applet? Let's start with the view applet.
The view applet provides Web developers with functionality similar to that of a Notes view. Notes views are lists of documents in a Domino database. Depending on how they're designed, you can use views to select, sort, or categorize documents in different ways. Views can also show many types of information about the documents listed in them, such as an author's name or date of creation. A view may show all documents in a database, or only a selection of documents. Anyone who is familiar with a Notes view is familiar with functionality such as resortable columns, where you can click on the header of the column and re-sort the column, or resizable headers, where you can drag the edges of the column headers to change the column widths dynamically. You can now do these things on the Web with the view applet.

You can also allow users to expand any number of twisties, which allows users to drill down and see information in multiple categories at once. In an R4.6 HTML view, you could really only expand one twistie at a time, and then any other twistie on the screen would collapse. In the view applet, you have the same flexibility that you have in the Notes user interface, where you can expand or collapse to your heart's content.

One exciting aspect of the view and outline applets is that they read in their data in XML. We made enhancements to Domino to allow it to generate certain information in XML, which is then parsed by the view and outline applets. The structured nature of this data lent itself quite well to representation in XML. Although this XML is used only as part of the private protocol between Domino and the applets, this sets us up nicely for even more advanced uses for XML in Domino in the future.

The applet also allows for vertical scrolling within a view and that means you can scroll over the contents of a database without having to repeatedly choose "Next Page" as you did in the HTML view. The big win with this is that you can scroll directly to a given portion of your view by dragging the scroll bar.

One of the most important features of the view applet is multiple document selection. Now you can select multiple documents in the view and perform actions on those documents from the view. Previously, you had to navigate to a single document and execute that action at the document level.

The other benefit is that we cache some of the data. Previously, every time you went to a page in the HTML view, you were bringing information in from the server. With the view applet, we cache some of that data so if you go back to a place in the view where you've already been, the browser doesn't have to read the data in from the server again.

Does this help performance?
Yes, because when you scroll, the view applet doesn't read every scrolled entry from the beginning to the end. It knows where you scrolled to and reads that data. It helps performance quite a bit that the applet doesn't have to read in information that you just want to scroll past anyway. This is similar to the way the Notes view works. It too doesn't

read in any of the data and it in fact, caches a little less data than the view applet does. There are more features in the view applet, but I think we've hit on the main ones.

The R5 Beta Feedback forum on Notes.net contains some specific concerns that users have about the view applet. Can you address those concerns here?
The view applet doesn't do everything that the Notes view does. We knew going into it that we couldn't do everything that the Notes view does, so we didn't intend that. We could only do so much in the R5 timeframe and we picked what we considered to be the important things.

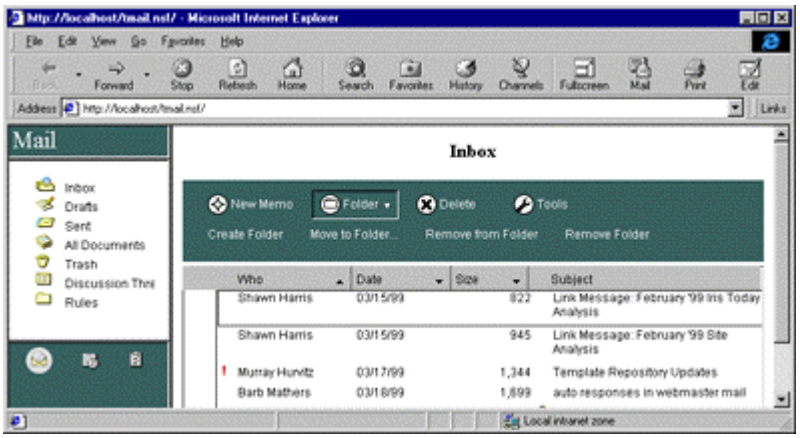
Another thing that concerns Notes developers is that they don't have quite the same programmatic access to the applets that they have in the Notes client components. Although you can do some significant programming against the Domino back-end classes on the Web, we haven't provided similar access to the front-end (UI) objects that the applets provide. This is due to time constraints. It's quite a different development effort to package a programmatic entity with a well-defined, well-tested, public API, than it is to provide a UI widget that users can use interactively. For R5, it was not a goal to provide a public API. However, this is first on our list for post-R5. This is true for all of the applets. Some won't ever have a public API, such as the action bar applet, because what would you do with an action bar applet? It really just calls the API of other things. But, we've had feedback that users really want public APIs for the view and editor applets. Although we haven't specifically received feedback on it, we will probably also provide a public API for the outline applet.

I should also point out that we have posted a preliminary, unsupported version of the API for the **view** and **editor** applets in the Beta Feedback forum for those developers who want to get their feet wet programming the applets. Developers should be aware that these APIs will most likely change in a future release.

Can you explain what the outline applet is?
The outline applet is almost identical in functionality to the Notes outline design element. New in R5, an outline presents the structure of your database or of your Web site. The outline also has many features, including the ability to control the icons displayed at each level of the hierarchy, and the ability to control whether or not twisties display as you expand each level of hierarchy. You have very fine control over mouse effects, meaning that you can control what happens as the mouse moves over each outline entry. For example, you can specify for the entry to turn a different color, highlight, or display with a bevel effect. You have control over what launches when you click on one of the entries. You can choose to launch a specific page or form in the database, or specify that the outline entry perform an action when clicked. All of this new functionality is part of the outline applet. When you design an outline, you choose "Web access: Using Java Applet" in the outline's infobox. [Editor's Note: For more information on outlines, see "**Domino Designer R5: The Outline Designer**."]

What does the action bar applet provide?
The big story here is that we didn't have a good way in HTML to represent the subactions, which are new in R5. Since R4.x, databases could include an action bar, which is a row of buttons you can click as shortcuts to perform common tasks in the database. Now in R5, actions can also have subactions that you access by clicking one of the action buttons on the action bar. A drop-down menu containing subactions appears. These actions and subactions are in most of the templates in R5. In the mail template, for example, if you click on the Folder action, you'll see the Create Folder, Move To Folder, and Remove From Folder subactions. On the Web, in HTML, there was no good way to represent that drop-down list of subactions. This spurred us on to create an applet that would better reflect the Notes R5 user interface. Unfortunately, we didn't quite get there. It turns out that there is a limitation in the 1.0.2 version of the Java Developer's Kit (JDK) and drop-down menus are not supported. So, we had to modify the way we present subactions. Instead of presenting them in a drop-down menu, we present them on a second line below the actions.

[Editor's Note: The following screen shows the action bar applet in action.]



In the HTML action bar you see all of those subactions going off the edge of your pane and you have to use the browser scrollbar to move back and forth to get to the different actions. Since the action bar applet makes the action bar itself scrollable, you don't have to worry about that. Again, this is all about providing the same user experience on the Web as you have in the Notes client. The look-and-feel, and the way that you interact with Domino applications on the Web should be as close as possible to the way that you interact with Domino applications when you use the Notes client.

What is the editor applet?
The editor applet is a rich text editor. Now, you can create and edit text on the Web with attributes such as font style (bold, italics), text color, paragraph formatting, bulleted lists, and so on. To enable the editor applet, you select "Web Access: Display using Java Applet" in the rich text field's infobox. Domino sends HTML to the editor applet, which reads the HTML and displays it. If you have a rich text field on a form or a page, Domino translates it from rich text to HTML on-the-fly and sends it to the applet. You can then interact with that text through the editor applet, send the text back, and it either converts back to rich text or remains as HTML depending on the properties of the rich text field.

The main reason we use HTML is that it is the language of the Web. We wanted users to be able to view the data in read and edit mode and have it look the same. In addition, since the editor applet outputs its contents in HTML, we don't need to translate that data to allow users to view it in a browser when they aren't editing the data.

We also designed it this way because we already had access to the mail applet from Lotus eSuite. In fact, we based the editor on that original code. In addition, there was already code on the server-side to convert rich text to HTML and vice versa, so we just made use of that code. The conversion of rich text fields to HTML has been supported by Domino for quite a while. It was much easier to convert rich text to HTML than it would have been to write a brand new editor from scratch that edits Notes rich text format on the Web.

Are there any limitations within the applets?
In all of the applets there are subtle variations in what features we support as compared to each counterpart component in the Notes client. Of course, we'll continue to enhance the features supported in the applets on an ongoing basis.

In the editor applet, for example, we don't support everything that you can possibly do in HTML. The editor applet is not a full-fledged HTML editor. It does not support frames, tables, or dynamic HTML. It is really just meant to be a basic editor, like WordPad. Users may be chagrined to learn about some of the limitations because they may want to put HTML within tables, or some other HTML feature. Another feature that is not in the editor applet, but is highly requested in the Beta Feedback forum is external cut, copy, and paste. This is because JDK 1.0.2 doesn't support this feature. It is on the top of our list post-R5 in a JDK 1.1.x version of the applet.

For each applet, it is important to realize that for every feature we add, the size of the applet grows and therefore, the time to download the applet over the Web increases. We are very sensitive to this download time, so we've made a constant trade-off between features that we provide and the size of the applet.

Will you add more features post-R5, or are they pretty much complete given that you want to keep the size down?
We are certainly not done with features. We will add features because there are things that we didn't do for R5 that we know we want to do post-R5, and because we know that the feature set in Notes will increase and we want to match that in the applets. We will constantly have to be sensitive to the size issue when we add these features. It may never be the case that these applets will have 100 percent of the features we would like them to have because of the size issue.

Post-R5, we are also looking at various techniques for reducing the size of the applets independent of the feature set. For example, there are certain things we can take advantage of if we move to JDK 1.2. For each function that we use in the JDK, that's one less function we have to implement in the applet and download to the browser. Unfortunately, no browser today supports JDK 1.2. When they do, we would love to move to JDK 1.2. We are also looking at various Java plug-in technologies that allow you to download on-the-fly a specific JDK to the browser. If a browser doesn't support the JDK that we want our applets to use, we can auto-download a JDK at the user's discretion to provide that support. We're also looking at ways to auto-download and install the applets themselves so that users don't need to download them every time. This functionality will be directed primarily at intranets where an administrator typically has access to the end-user's desktop.

In terms of designing applications using the Domino applets, how does a developer actually go about it?
For all the applets there is either a combo box or a check box in the Properties box of the design element that you must select to use the applet for that design element on the Web. All you have to do is choose Enable Java Applet. You don't have to write any code. If you want to modify the way the applet appears, you do it through Domino Designer. For example, if you want different actions to appear in the action bar, you add or remove the action in Designer. If you want columns to appear a certain way, you design them that way in Designer, the same way you would design them for the Notes client. This fits into the larger R5 goal for Designer, which is that you can design your application once, and have it run in both the Notes client and on the Web. Today, the extent of the flexibility you have as far as presentation is whatever you can do in Designer. For example, we don't support subclassing the applets in the Java sense. So, you can't write your own version of the view applet and plug it in. We don't support that, just as we don't provide a public API. We are looking at that for a future release.

Do designers have to use these applets?
No. It's completely up to the designer whether or not to enable the applets. There are trade-offs involved in using either HTML or the applets. There are reasons that you may want to use the HTML counterparts. I guess the biggest concern a developer would have is the time it takes to download the applet. The interactive performance of the applets is very, very good, but it does take some initial download time to get the applet to your browser. The applets really shine in intranets or extranets where most people are connected to servers over high-speed connections. However, if you provide content over the Internet and people dial-up to get the information, you may want to think about whether the applets are right for your application. The action bar applet is about 10K, so there isn't much of an issue there. The editor applet on the other hand, depending on which browser you use, could be about 250K (it's only about 120K if you use a level 4 browser, such as Internet Explorer 4 or Netscape 4). Although that's not too bad when you consider the feature set, it is significant if you have a slow connection to the server.

There are also certain features that you can use if you use HTML that aren't part of an applet. For example, the applets are very limited in the number of fonts that they can display, due to JDK 1.0.2 limitations. In HTML, you can display any font installed on your system. If using a unique font, or many different fonts is important to you, you might want to use the HTML view. It is your choice.

Can you tell us about your personal experience developing the applets?
Developing in Java for browsers has been a real eye-opener. I've been programming in Java for quite a while. Mostly I worked on stand-alone Java applications. Quite a bit of the research and design we did for R5, over and above coding the applets themselves, was researching and working around various idiosyncrasies in the different browsers across all the OS platforms. It turns out that the Java support in each version of each browser is different. Java in Netscape 3.x doesn't work the same way as in Netscape 4.x, which doesn't work the same way as in Internet Explorer 4. In addition, each version can have subtle variations if you run it on a different operating system. It was challenging because this "write once, run anywhere" promise in Java hasn't been realized yet in browsers, and it's not because of Java the language, but because of the implementation of Java on each browser. So I was surprised to

find that out, and hopefully as time goes on, this will be less and less of a problem and we can really realize the "write once, run anywhere" paradigm, which only serves to help our users.

How do the applets help us competitively in the market?
The applets allow you to design a complex object, like a view, within the Designer environment without writing any (or at least very little) code. Then by selecting one option that says you want to present the object as a Java applet, the view you designed runs on the Web. No other product allows someone with little or no programming experience to easily design a Web application where end-users can interact with and manipulate data. This is a direct result of the way that Domino R5 presents and stores information. The presentation of the information is separate from the data itself, which is exactly where the whole industry is going with XML, and why we're able to do this. Notes has always been about presenting information to users the way they want it and letting them do something with it, and now we can bring that same flexibility to a browser.

Will you add more applets in the future?
Yes, we are looking at adding more applets. One applet that we know we need to add is a calendar applet. Today we present the calendar view as HTML on the Web, but we are looking at adding an applet that makes the Web calendar view look more like the Notes client calendar view. Another applet that users often request is a date picker applet. We are looking to our users for feedback on what other applets that they would like to see. Also, if other Notes components appear post-R5, we'll look at doing applet versions of those. Post-R5, we would also like to make it possible to write your own applets and plug them in where our applets appear now. As long as you adhere to a certain API, your applet would show up where ours would.

BIOGRAPHY

Brian joined Iris Associates about a year and a half ago and is currently working on the Domino applets and Java support in the Notes client. He is a Long Islander who, after more than 20 years in the Boston area, has successfully lost his accent. Brian first started programming in the 7th grade learning Basic and FORTRAN on a PDP-8 and has been hooked on programming ever since. He has worked in the computer industry for the past 17 years, with positions at Computervision, First Floor Software, and Sun Microsystems where he became a self-avowed Java bigot. In his spare time Brian likes to spend as much time as possible with his wife and 2 1/2-year-old son.