

BM Lotus Expeditor: The Microsoft .NET client alternative

Abstract: IBM Lotus® Expeditor software is an open standards based alternative to Microsoft® .NET® Framework client. In this podcast, we will discuss various options available to .NET client and Microsoft Visual Basic customers for migrating to Lotus Expeditor software.

Transcription:

DUTCHER: Welcome to this [IBM] Lotus® Expeditor Podcast.

Today our topic is Lotus Expeditor [software] as an Open Alternative to Microsoft® .NET [Framework]. I'm David Dutcher, Worldwide Technical Sales Support for Enterprise Access. And I'm joined by Angus McIntyre, Product Manager for Lotus Expeditor.

McINTYRE: Hi, David. Thanks for having me here today.

DUTCHER: Thanks. Hey, can you explain this podcast series, Angus, and what this podcast series is about, and why developers should care.

McINTYRE: This is the sixth and final in the series of podcasts on extending applications out to the edge of networks. It's put together to show IT organizations and developers that they can reuse both their skills and reuse the applications that they've created to extend these applications out to people who need information. It's a composite desktop, a rich client environment that they can use.

Other podcasts that we've already covered are an [1] Introduction and Overview to [Lotus] Expeditor, [2] Taking Portal Offline, [3] Business Process Automation with Offline Forms in Lotus Expeditor, [4] Taking WebSphere Offline with Lotus Expeditor, [5] Mobile Access with Lotus Expeditor, and now finally, [6] Expeditor as an Open Alternative to Microsoft .NET [Framework] and Silverlight™.

DUTCHER: Why would a Microsoft customer ever want to migrate towards an open client and away from the .NET migration strategy?

McINTYRE: David, analysts have said that the cost to convert [Microsoft] Visual

Basic® to Visual Basic .NET puts quite a speed bump in front of application developers and IT shops. Estimates that the code conversion would be over 40 to 60 percent of the original developer costs.

Also retraining of Visual Basic developer to be a VB .NET developer could cost in the neighborhood of \$40,000 and take up to six months.

So the whole notion you hear is they're at a decision point where they may want to go forward to .NET, but the skills aren't there, the forward migration isn't there.

And customers are looking at alternatives. Customers have said, we don't want to move in that direction, we want to take our existing applications and extend them out to give a more compelling user experience using open standards.

DUTCHER: So it's out with C# and in with Java™?

McINTYRE: Well, it's not about Java versus C#. It's really about how to extend application from the server to the client. Right, if the shop's fully Microsoft with C# and all of the Microsoft databases, am I about to convert them into the Java shop? No. The age old arguments are always still out there, but you're not really searching to convert them.

The age old arguments are, why Java over .NET? You know, consistency... it's avoiding vendor lock-in; stability to protect customer investments; skills availability. It's driven by a community and expert process.

Also that it's based on open standards... software investment protection. It's flexible. And portable deployments that are supported by a multitude of vendors, business partners.

It's a consistent programming model and execution environments on the client and on the server complete with security. Like those arguments are all out there for people that are want to make the switch, but it's really about extending a provider rich client computing platform.

DUTCHER: So is this really a rich client argument?

McINTYRE: Exactly. How does the software team extend existing server based applications out to provide a compelling and responsive end user experience. For Microsoft customers who have C#, SQL server, MS Message Queuing, Visual Studio and no intention of ever switching, be happy and stay where you are.

But for a Java EE development shop that is looking for a compelling alternative to a Microsoft .NET client, the Eclipse™ rich client platform programming model is one alternative that they really should explore. [Editor's note: Eclipse community is an open source community committed to the implementation of a universal development platform.]

DUTCHER: But why the Eclipse™ Rich Client Platform?

McINTYRE: Well, it's proven open source technology. Eclipse itself has been downloaded by over 60 million... over 60 million times. The Eclipse developer community has... it's 2.6 million strong.

This proven open source technology is a core subset of the popular Eclipse tools IDE platform. It's a client workbench with application plug-in framework. So it supports local native applications, Web applications, rich client applications and even Java applications. And it provides the support for building and executing client applications which that, with the rich compelling user interface, SWT, J Face, UI component.

And when you program to this framework, you can take an application as it runs on a [Microsoft] Windows® XP system, it looks like a Windows XP application. When you run it on a [Microsoft] Windows 2000 system, it looks like Windows 2000. When you run it under Linux system, it looks like a native Linux widget.

Ideally, it's a platform-agnostic programming model that will let you explore other platforms that you may decide to target, which is ideal not only for end customers that are making that decision, but [also] for business partners that want to cover the most platforms with their application sets and solutions that they deliver.

DUTCHER: Now, wait a minute, Angus. Isn't Eclipse an IDE that's been downloaded by over 2.6 million developers?

McINTYRE: Yes, but there's a project off of Eclipse that's called the Rich Client Platform. And it's an application integration platform for end user application. Really RCP equals Eclipse minus that development environment.

Eclipse started out as an organization for application development tool integration. Let's say a modeling tool. A class signature change within a modeling tool would cause a class signature change within an editor for the source code for that API.

As one tool changed the class signature, it would automatically be reflected in the other. So we've taken the IDEs [Integrated Development Environments], the compiler out and we've left the framework there as an integration platform for end user applications.

The way we use it is as follows. As a cell within a spreadsheet -- let's say an [Microsoft] Excel spreadsheet -- changes, that change in that cell can be reflected into a 5250/3270 emulation screen. It's that type of end-user application and application integration on the clients that we're talking about.

DUTCHER: Well, just how popular is the programming model? And what is the uptake?

McINTYRE: Many companies are building on it... and many others are planning to build on it. Companies such as IBM, We built Lotus Expeditor, Lotus Sametime® and Lotus Notes® 8 on this platform. If you go out to the Eclipse website [eclipse.org], SAS, NASA, the US Army, all of these companies have endorsed the Eclipse platform. When you go to the site you'll see why they have endorsed this platform as well.

The other things in terms of how Lotus is using it: portal, forms, mobility connectivity. These things are building upon the rich client platform with collaborative and offline capabilities that are not within the open source but are a value add inside the product.

DUTCHER: And how does Lotus Expeditor extend this rich client platform model?

McINTYRE: Sure. Expeditor takes the rich client platform programming model and augments it. So we've created a single integrated desktop that supports both Web and native client application. It's based on open standards, OSGI™, Eclipse, Java, XML, extends existing applications with APIs.

We support both online and offline user experience through our middleware with a local data store and transactional synchronization with back end server based applications.

We also have the capability of doing no touch, low touch and server managed provisioning and maintaining of the software and of applications that are running on top of the environment. Finally, we support both Linux... Windows and Linux.

The way that we augment the rich client platform is in three main areas. The first area is at the interactional layer -- this is at the glass or at the screen or at the PDA face, the plastic that people look at.

And we've taken the rich client desktop and we've integrated in Web capabilities with an embedded servlet JSP model so that Web pages can be rendered locally. We have the ability to embed [Microsoft] ActiveX® controls. We talked about Visual Basic; SWT and even a portlet container to take portal applications in.

We're based on the RCP model, and then when you can have any type of browser based plug-ins run in this interaction model.

From an access point of view, we take all of the WebSphere known APIs, JDBC for data, along with the relational data store, a SOAP client, as well as Web services security.

And then we have transactional messaging. What this really does is connect the client to back end data transactions and applications. We're really extending the same middleware that developers know on the server out through to the client.

And then finally we round this up with management. We use the OSGI service platform to do provisioning, maintenance, update and deleting of the software and applications on to the client. And then we add security elements in terms of single sign on, desktop lock down, et cetera.

The other things that Lotus Expeditor brings in terms of additional value, extends the programming model APIs to the client so developers are familiar offlining, business data replication, centralized control management, security of the client software, a credential store, single sign on, desktop lock down.

This is the ability to take a single instance of an application and the middleware and lock down that desktop so that an end user can't add any other plug-in or they can't even browse the Web to URLs that aren't sanctioned by the central IT organization.

It's very flexible. You can allow end users to plug in anything you want, but it is, can be managed down to an extremely fine-grained management.

Expeditor really provides five major pieces of functionality. [1] With the Web capability, you can browse virtually any Web application, including browser plug-ins, such as Flash, Flex, et cetera. [2] It can be rich... a rich client platform.... optimized to a user experience exploiting the underlying native platform look and feel. [3] From a composition perspective, you can also have a seamless integration, having composite applications integrated on the client desktop. [4] Semi-connected using our transactional middleware to take the client offline, do some transactional messaging and then synchronize when reconnected. [5] And finally, the client is managed, the ability to deploy updates and delete applications, middleware and the platform from the client desktop.

DUTCHER: Angus, can you take us down a level and point out the similarities in the programming model from server to client?

McINTYRE: Sure. It's all based on the OSGI Release 4 and the Eclipse Core Extension point framework, Eclipse 3.2 Rich Client Platform.

On top of this, we have elements for management, like an enterprise management agent that essentially allows you from a central location to inventory all of the bundles that have been deployed through to the client and the ability to update those in terms of a scheduled update while the user is sleeping -- like, you can do all end users that start with A at one o'clock in the morning, B at two o'clock in the morning, et cetera.

It also is augmented with the access layer. A Web container, local servlet JSP container so that if enough information is there to render a Web page locally, it will be rendered on the client. A portlet container so you can take portlets offline and use them in the environment.

Complete capabilities for JNDI in terms of getting that directly through to back end systems. A services layer for both data through JDBC, transactional through JMS and even Web services through a SOAP client integrated on the client.

There's also a property broker in this access layer, too. And what the property broker allows you to do is have an event and data framework so that as an advance within one application happens or as data changes within one application, you can notify another application to take that data change and reflect it within that application.

Expeditor is also network aware so that when the client reattaches to the server, the synchronization of the data and the offline messaging can really forward through to the back end system without even the end user knowing that the server's there.

We augment the Eclipse RCP with a bit more security in terms of a JAAS framework, a local key store or credential store so that users can authenticate themselves locally while they're offline. [Editor's note: JAAS is Java Authentication and Authorization Service.]

And then finally, an accounts paradigm so that multiple users can use the same workstation and have multiple accesses to back end application.

On top of this we run an extension through the interaction layer, a portlet container in terms of rendering these portlets offline. And then an application launcher which really handles the life cycle of the applications inside the composite desktop.

And finally, as I mentioned before, a restricted workbench capability to lock down the ability for end users to add new bundles or even to browse URLs that you don't want to browse... them to browse... while they're working on business problems for their eight hours a day.

On the device side, we structure it for the device. You know, browser plug-ins and the rich client offered through ERCP. The middleware layer remains mostly the same in terms of the Web container direct resources, as well as data transactions and Web services on the client.

The enterprise management layer is there, which is important when you deploy out to thousands of end clients. The synchronization framework is also there as well, to synchronize objects, to synchronize data from the client to the server.

Finally, when you back it up with both a client to server end-to-end management, you can manage an Expeditor client from an Expeditor server. Tens of thousands of clients can be provisioned per hour using Expeditor.

DUTCHER: Angus, you talked earlier about three ways of migrating Visual Basic applications to Expeditor. Extension of server applications replace VB. VB embedded within Expeditor and how VB can be converted into Expeditor code. Can you outline the conversion process for our listeners?

McINTYRE: Sure. You know, if you go to Diamond Edge®, they're an IBM business partner and they have a Visual Basic to Expeditor migration tool. And the value proposition that they really have is they're an alternative to the Microsoft Update Wizard for moving from Visual Basic to Visual Basic .NET.

The Diamond Edge tool automates roughly about 90 percent of the code. So 90 percent of the work required to take a Visual Basic application and convert into an Expeditor based application is done through the tool.

Moving to Expeditor opens up considerable flexibility on the client. You can move to Linux, et cetera. Then they have a services based offering where they can do roughly a thousand lines of code per day and they will go in and manually convert the rest of the code for you.

And they automatically convert VB programs Expeditor and that supports a wide variety of Visual Basic versions, 3, 4, 5 and 6. Supports Java version 1.4 or higher, and it just

basically goes in and converts the code, the UI code to SWP and J Face which is the elements of RCP.

So you know, Diamond Edge has support for over 70 controls, common controls such as progress bar, image [of lists], data, database grid, data grids, et cetera. And it migrates DAO or RDO or ADO database access capabilities, the Java database access using JDBC and the Java implementation of the ADO. So it's something that people that want to look into whole scale conversion of those should really take a look at.

[Editor's note --Disclaimers: Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.]

DUTCHER: Where is this being used?

McINTYRE: If you take a look at where customers are migrating from Visual Basic into a Java environment, then it's really as I mentioned before, it's the people that have Java EE on the server. So it extending of Java EE applications that are running in line of business out to Expeditor for a rich client experience.

We're finding this extremely popular in insurance, particularly among customers who are looking at a way of getting out of the Visual Basic Version 6 going out of support and are looking at extending their existing programming model out to a rich client experience.

DUTCHER: How is Expeditor being used within IBM as an integration platform for client computing?

McINTYRE: Because we have this integration platform on the client, it is being used throughout all of IBM Software division to provide a compelling end-user experience and integration platform.

Take IBM [Lotus] Notes® 8 software -- In this case, Expeditor is being used to integrate

other applications into the composite desktop. So we're extending SOA to the Notes environment through Expeditor integration underneath the covers.

Host integration transformation service – 5250/3270. Those can also be part of the composite application on the composite desktop... IBM once again “eating its own cooking” to integrate its products together.

IBM system servers use Expeditor as a rich console for both System i™ and System p™ to give a compelling user experience into the consoles for these systems in terms of electronic forms for people on the move using Expeditor as the composite desktop to have forms and forms being taken offline and then transactionally synchronize the back end system.

WebSphere Commerce, compelling end user experience for Commerce people. For users of WebSphere Commerce to have access to storefronts, et cetera. [Lotus] Sametime uses Expeditor under the covers in terms of its Unified Communication and Collaboration.

The ability for Web services to be integrated in [Lotus] Sametime is brought forward by Expeditor being used underneath the covers in Sametime. It's the portion of Expeditor that's included in Sametime. Sametime can also be used in conjunction with Expeditor as part of the composite desktop line of business application.

And finally from a WebSphere and a WebSphere Portal standpoint -- extending application beyond the data center, beyond the browser, out to a rich compelling responsive user interface, an open alternative to Microsoft .NET.

DUTCHER: What type of development tools are recommended for use with Lotus Expeditor?

McINTYRE: David, you can use virtually any Eclipse 3.2 IDE. The one that IBM offers in this space is Rational Application Developer 7. We've been tested also with other visual building software such as Instantiation®'s Window Builder™ Pro. They've done a really good job in terms of creating a visual builder that creates SWT applications.

New in 6.1, portlets created from WebSphere Portlet Factory can also run in the Expeditor client environment. This augments the portal capability already in the product -- the capability with the Rational Application Developer tools to create JSR 168 portlets. Expeditor then provides the ability to extend these JSR 168 Portlets to run inside the Expeditor client.

You have to remember that Expeditor is aimed at IT shops that have Java and Eclipse skills. And those aren't hard to find, since there are over 2.6 million Eclipse developers, and Eclipse [platform] itself has been downloaded over 60 million times.

DUTCHER: Angus, can we take a few minutes and talk about Microsoft Silverlight?

McINTYRE: Sure. Silverlight is in beta right now, whereas Expeditor is... it's available and has been available for... since 2004. When you compare the programming model, their CLR is much like a virtual machine, right. So if you have CLR in scripting, we have a virtual machine and JavaScript capabilities on it. So it's really very similar in programming models.

As I mentioned before, we're here about extending Java EE applications out to rich client platforms. The APIs and the programming model is centered around that. If you're clearly centered around Microsoft and Windows server capability and browser type based capability, you might want to take a look at Silverlight.

DUTCHER: Angus, where can our listeners go to get more information?

McINTYRE: You can visit ibm.com/lotus/mobile. Or contact your Lotus sales representative. And take a look at the other podcasts in this series.

DUTCHER: Thanks, Angus. Another great podcast session. And thanks to those of you listening to this podcast today. As Angus mentioned, please remember to visit us at ibm.com/lotus/mobile. We hope you'll join us for all the podcasts in this Lotus Expeditor Series. Thanks and bye for now.

[END OF SEGMENT]