Java 2 Enterprise Edition: The Fountain of Youth?

BY AL DEVITO AND MICHAEL ROSS MURPHY

This article examines how a software developer used J2EE to make the features of its mainframe software product, which was developed several years ago, available through a web browser.

A fountain of youth has been sought for centuries. To recapture one's youth is even now one of man's strongest desires and has carried over into many of his endeavors, including the programming of computers. As a developer of mainframe software, we were looking for a way to recapture the youth of our product so that it would continue to provide value in today's web world. Therefore, we were optimistic about discovering a fountain of programming youth when we learned that Java 2 Enterprise Edition (J2EE) had been announced.

At the time, we were struggling with various ways to capitalize on the growing market need for high performance access of mainframe data from a user's web browser, as our flagship product, tableBASE, provided turbo-charged performance to mainframe data, but within a 100 percent pure mainframe environment. The question was not only how to make these capabilities available through the browser but also how to get a product to market fast. We knew that our product could improve access to the data. It appeared as though J2EE could improve our ability to get to market within the window of opportunity.

J2EE enables mainframe sites to capitalize again on the investments in application development it made 20 and 30 years ago. Those investments continue to earn a good return as we embrace the 21st Century.

BACKGROUND

tableBASE is an in-memory table manager for IBM mainframes. It provides a comprehensive set of facilities for creating, maintaining and manipulating tables. tableBASE is used as an application development tool by Global 2000 organizations and is particularly popular in the financial community. It has a rich Application Programming Interface (API). Our primary challenge was to encapsulate this richness in reusable object-oriented terms and concepts. J2EE with its built-in support of Enterprise JavaBeans (EJB) provided this capability.

Because J2EE provided key application services such as transaction and resource management, we — as well as the eventual users of our product — could focus on the business logic and user interface. In addition to support of EJB technology, J2EE provides native support for Java servlets, JavaServer Pages and XML, all of which were vital for our move to the web. Most importantly, the J2EE platform is standards-based.

J2EE provides a multi-tiered distributed application model, the ability to re-use components, a unified security model and flexible transaction control. The multi-tiered application model was especially important because it meant that client applications need not be involved in dealing with back office systems and data sources; the middle-tier server handles these functions. This, in turn, enables a thin-client architecture, which we felt was necessary so that our clients could access our product from a variety of client types, such as web browsers, Java applications or hand-held devices.

The spectrum of facilities offered by J2EE gave us the confidence that we could build an Enterprise Information System (EIS) connector kit that would enable enterprise architects to develop their own EJB and/or CORBA-compliant applications using tableBASE.

REQUIREMENTS

At its core, our product requirement specification was very simple: provide high-speed connectivity from the mainframe (via a middle-tier server) to a client with a simple web browser, who would not need any special software installed on his desktop. Because our product is an application development tool, it would provide users the ability to develop their own custom applications using the Java Class and

Java Native Interface (JNI) adapter components we would build. Support of GUI customer applications as well as batch and script-oriented applications is required.

ALTERNATIVE SOLUTIONS

The J2EE Application Model divides enterprise applications into three fundamental parts: components, containers and connectors. Components are the key focus of application developers, our primary users. We would supply tableBASE persistence components in the form of EJBs; the users would develop their own business object components, which reflect their business rules, and the user components will integrate with the components we supply. To promote portability, application servers implement container services in a standard way. The design solution, therefore, was primarily a question of devising the most appropriate connector technology.

There are two basic kinds of connector technology: those that provide a "front-end" connector (presentation management such as browser support or terminal emulation) and those that provide a "back end" (a data source or repository). We looked at both kinds.

Front-End Connector Options

Four of the front-end connectors we examined — CICS Web Support, Host on Demand, CICS Transaction Gateway and 3270 Web Bridge Facility — required CICS. The latter three provide only character-based interaction to web clients; a graphical user interface (GUI) is neither available nor announced. These options were rejected, as we wanted to provide GUI web access. CICS Web Support is of most value in shops where the workload is almost entirely CICS. However, in our clients' and prospects' computing centers, CICS is just one of the many computing environments. This option was also rejected.

We also looked at Host Publisher. It had many nice features but, because it required a data source compatible with SQL, it was rejected since our product was designed to handle non-relational as well as relational data sources.

Back-End Connector Options

Three possibilities were reviewed: Enterprise Access Builder for Transactions (EAB), Java Data Base Connector (JDBC) and a Custom Java Native Interface (JNI) Connector Library. tableBASE does not support an SQL interface, which is required by both EAB and JDBC. Therefore, these options were discarded. We selected the JNI Connector Library combined with EJB technology (which encapsulated the functionality of our API) as the best path for us to take to meet the specified requirements.

JNI is an interface facility for the Java language platform that allows Java methods to use code that is native to the run-time environment. It is really a two-way interface, as it supports the invocation of Java methods by non-Java programs as well. JNI could provide the data source connection to our product. A C/C++ program library would act as an adapter to convert Java method references to standard IBM calls in the OS/390 environment. Each C/C++ library element would match an interface definition within an EJB business object.

SELECTED SOLUTION

Figure 1 depicts the web architecture we selected. A "thin-client" workstation invokes interactive web pages to request and specify tableBASE resources from an application server, which, in the initial release, will be WebSphere. The application server uses a set of Java Servlets and Java Server Pages (JSPs) to process user requests contained on standard HTML web pages. The JSPs enable separation of HTML and Java code.

The application server compiles and executes the JSPs as servlets, which activate EJB business objects and helper classes to manage the tableBASE data source and present it to the client. The "metadata" that defines the properties of the tableBASE data to EJB entity beans is stored in XML document format, enabling easy integration with other data sources.

The EJBs communicate with our product's data structures using a custom connector interface library, which is a C/C++ library that implements a set of tableBASE calls in the OS/390 environment that exactly match the interface definitions within the EJB business objects.

Because all user interactions are managed by Java Servlets and EJBs activated by the application server, clients obtain the full benefit of business rules and security policies that protect enterprise information. Secure Socket Layer (SSL Version 3.0) encryption and authentication services are fully supported in the J2EE environment.

Only the JNI adapter C++ library (along with tableBASE itself, of course) needs to be deployed in the mainframe host environment. The JNI library must be installed under the Hierarchical File System (HFS) of UNIX System Services (USS). The Java Virtual Machine (JVM) is also separately pre-installed under USS.

In the figures presented, both the application server and the JNI adapter C++ library are shown running on the same host. The application server can be deployed on the same mainframe host or another middle-tier server, such as a Windows NT server host. In this case, the user would use a Common Object Request Broker (CORBA) configuration to perform the functional separation. Figure 2 shows how the product would fit in a typical OS/390 environment.

While the J2EE specification offers several interesting facilities, not all the features became available for the OS/390 environment until the first quarter of 2001. This required separate packaging and "backporting" certain J2EE components as part of the application Java classes, instead of being referenced in the JVM.

OBSERVATIONS

This project used IBM products primarily. Perhaps, the most used product was the Visual Age Integrated Development Environment (IDE), Release 3.0, Enterprise Edition. While the learning curve for the IDE was rather steep, selecting it turned out to be a wise decision as we found it to be a powerful, easy to use, fully integrated development environment. The IDE provided all the facilities we needed. In addition, IBM's support was top-notch. We were especially grateful for the active and responsive newsgroups focused on Visual Age for Java, WebSphere and XML. When we had questions, we were able to get answers quickly from fellow newsgroup subscribers.

Since our initial target market was the large IBM mainframe user, we opted to use WebSphere 2.0.2, Enterprise Edition, for two reasons. It was the only application server that provided native OS/390 support and it was integrated with Visual Age. We found WebSphere to be a powerful application server and its integration with Visual Age was virtually seamless.

We did have some trepidation about using USS, as our experiences with Open Edition

were not very rewarding. However, we found USS to be a smooth implementation of the UNIX standard.

Our primary difficulties stemmed from our being relatively "early adopters," as the J2EE specification had not been completely implemented for OS/390. For example, the Enterprise Edition of the SDK, Release 1.2, does not include EJB technology, while JDK 1.1.8 supports only the EJB 1.0 specification. Fortunately, our project used "bean-managed persistence" features for our EJB entity beans, since "container-managed persistence" is optional for server vendors even with the EJB 1.1 specification that forms part of J2EE. Moreover, there were other costs of being an early adopter. Collection classes had limitations. There were incompatibilities in different versions of Swing, the Java Graphical User Interface (GUI) tool kit. Some JNI features were not supported in JVM 1.2.

Finally, the fact that OS/390 uses EBCDIC (except within Java where it uses Unicode) caused some difficulties, most of which we were able to overcome with IBM's tools and support for migration and translation between codesets.

FOUNTAIN OF PROGRAMMING YOUTH DISCOVERED?

Is J2EE a fountain of programming youth? We would answer, "yes" to the question. With J2EE we were able to rejuvenate a 20th Century product that was built for a centralized mainframe computing environment so that it brings value to its users in a 21st Century world of distributed, heterogeneous computing.

With the approach described here, mature mainframe data sources can be readily accessed over the web. Furthermore, these mainframe data sources can be combined with object databases, relational databases and embedded database solutions by using the tableBASE classes as components in custom Java and CORBA applications. The Java Remote Method Invocation (RMI) objects can be invoked with Java Remote Method Protocol (JRMP) or Internet Inter-Orb Protocol (IIOP) interchangeable without source program modification.

Furthermore, the tableBASE connector classes and EJBs can be deployed so that data updates run as transactions under the Transaction Manager. This allows a "two-phase commit" framework for referential integrity when multiple data stores are used or when failover capability is required. Accordingly, Java Messaging Service, a "message-oriented framework," is also supported along with IBM MQSeries to enable tableBASE data source updates to be message- and event-driven.

J2EE enables mainframe sites to capitalize again on the investments in application development it made 20 and 30 years ago. Those investments continue to earn a good return as we embrace the 21st Century.

FIGURE 1: THE WEB ARCHITECTURE SELECTED

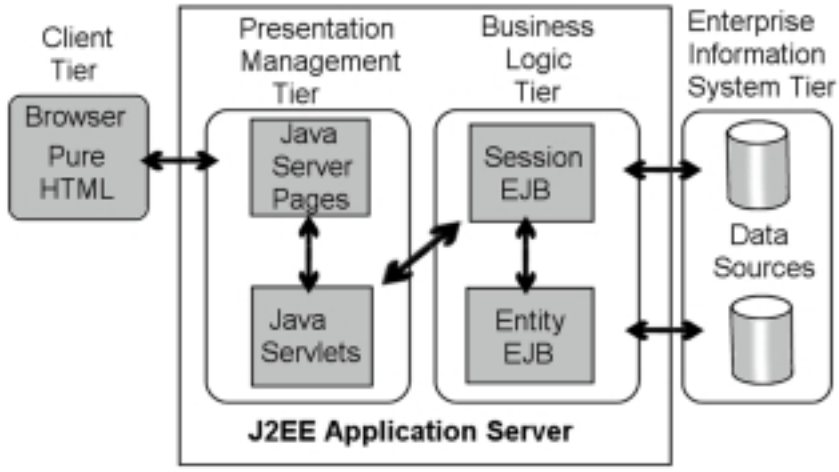
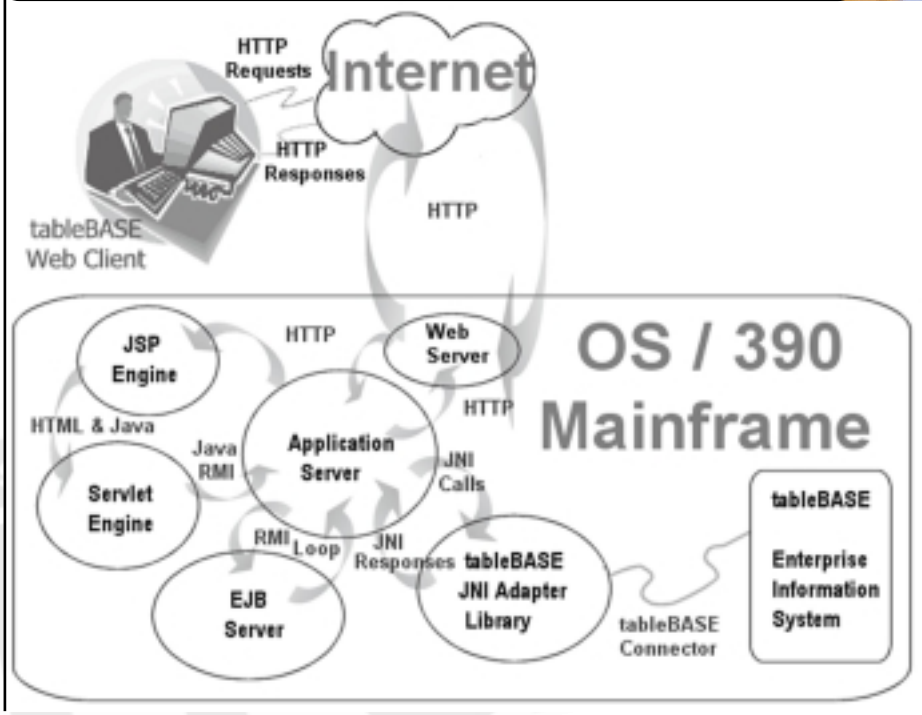


FIGURE 2: HOW THE PRODUCT WILL FIT INTO A TYPICAL OS/390 ENVIRONMENT



Al DeVito is Chief Operating Officer of Data Kinetics Ltd., an Ottawa-based software solutions company. Al is a veteran of the software industry, having developed his first software product more than 30 years ago. He has served in various capacities with leading North American software companies.

Michael Ross Murphy is an Enterprise Architect at Data Kinetics, Ltd., and is the project manager and lead architect for the tableBEANS product. He has previously managed design, development and implementation of large-scale enterprise Java solutions. Michael has more than 20 years of experience in enterprise application development, database architecture and network operations.