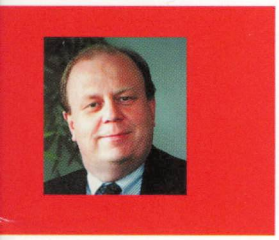
IBM

Developer Connection

JULY 1999

IBM SOLUTION DEVELOPER PROGRAM • TECHNICAL MAGAZINE • RELEASE 2 • VOLUME 3

Welcome to Solutions '99



The IBM* Solution Developer Program Technical Support team is pleased to welcome you to Solutions '99, the technical conference for solution developers that is sponsored by IBM, Lotus■ and Tivoli.★ Solutions '99 delivers answers to your toughest development challenges and helps you integrate the power of e-business in your solutions. If you are a commercial development professional who is building leading-edge, open solutions for networked environments, Solutions '99 should be an extremely beneficial experience.

The conference is packed with in-depth presentations, hands-on labs, insider case studies and plenty of interaction with e-business experts and peers. More than 100 sessions are scheduled that focus on Java●, XML, Linux, JavaBeans, IBM WebSphere*, MQSeries*, Web and Web application servers and much more! From the Solution Developer Program Technical Support area, the Technology, Tools, and Strategies track of the conference is featuring an elective session on Technical Support and tools to build your e-business solutions. This track provides insight into the latest Technical Support offerings available to solution developers.

The IBM Solution Developer Program provides support and services to commercial software developers worldwide who build e-business solutions using IBM technologies. The program can help you reach broader markets, control your cost of doing business and get products to market faster. Developers can take advantage of no-charge software and tools, hardware discounts and leases, cross-platform porting and testing facilities, global marketing opportunities, IBM Developer Support Online Web-based technical support, the IBM Developer Connection and more.

Developer Support Online provides technical support through an easy to access Web site that is available 24-hours-a-day, seven-days-a-week. The offering includes FAQs & HATs, sample code, information maps, white papers and technical articles, chat sessions and demos, Q&A forums and bulletin boards. This new offering can help get your products to market faster by providing technical support information in one repository.

The Developer Connection offers one-source access to more than 1,000 development tools on 14 platforms, plus IBM solutions for e-business. The content is available for download from the Web and on CDs. *The Developer Connection Technical Magazine* is part of the offering and contains technical articles written by leading-edge developers. Several articles in this special edition were written by conference presenters.

For more information about the IBM Solution Developer Program, visit our Web site at www.developer.ibm.com. You'll find the resources to help you take advantage of the ocean of global opportunities offered by e-business.

Solutions '99 is your chance to network with experts, collaborate with peers, gain hands-on experience and education and witness e-business solutions from the real world. I look forward to meeting and talking with you at Solutions '99 and encourage you to attend as many technical sessions as possible.

Roy Aho

Roy Aho

Manager, Solution Developer Marketing Technical Support

www.developer.ibm.com/devcon/

Contents

Welcome to Solutions '99 FRONT COVER

e-business has arrived: deal with it 2

Parallel Worlds: Why Java and XML will succeed 3

Using Java to bridge Domino R5 and WebShpere 7

Enterprise JavaBeans: an IBM unifying model 10

Deploying VisualAge for Java Enterprise Beans to the WebSphere Application Server 16

Business Intelligence with the AS/400 20

Java on OS/390 delivers real portability 21

IBM SanFrancisco at Solutions '99 24

A servlet architecture for IBM SanFrancisco 25

WebSphere Application Server FAQs 28

Directory 31

Power your solutions™

e-business has arrived – deal with it!

by Jean Swanson

e-business allows you to operate locally and sell globally. To become an e-business requires access to new technologies as well as managing your time and resources more effectively.

One of the best ways to "deal" with e-business is by utilizing the tools, information and technical support available through the Developer Connection and Developer Support Online. Both offerings are part of the IBM Solution Developer Program.

The Developer Connection puts the tools and information you need at your fingertips through our online catalog, our CD set and our technical magazine. This July Special Edition of the magazine contains articles by several of the presenters at Solutions '99. On the front lines of the e-business revolution, the authors share the latest developments in open, integrated technologies.

The lead article, "Parallel Worlds: Why Java and XML will succeed," is by Simon Phipps, Programme Manager, IBM Centre for Java Technology in Hursley, England. Simon, who is the chief XML and Java evangelist for the IBM Corporation, is hosting the power lunch on Wednesday, July 21. Don't miss this article and don't miss his power luncheon!

"Using Java to bridge Domino■ R5 and WebSphere," by Jim Hsu and Bill Lawton, discusses how to install WebSphere onto the Domino R5 server and how to use Java effectively in this environment. Together, WebSphere and Domino R5 open the door to exciting possibilities for e-business solutions that leverage the strengths of both technologies.

We've included two articles on Enterprise JavaBeans (EJBs): "Enterprise Java Beans: an IBM Unifying Model," by Mickey Nix and "Deploying VisualAge* for Java Enterprise Beans to the WebSphere Application Server," by Howard Borenstein. EJBs are logical bridges between application clients and data servers. This programming model is robust and flexible enough to allow developers to build new component-based frameworks, or to wrap legacy data and applications, and to deploy both solutions into a single Web application server with a single client interface.

You'll also find articles on a servlet architecture for IBM SanFrancisco, business intelligence on AS/400* and Java portability on OS/390* by authors that are speaking at technical sessions at Solutions '99. Check out their articles and take in their sessions at the conference.

e-business isn't about re-inventing your business. It's about streamlining your current business processes to improve operating efficiencies.

Whether you're just taking the first steps or are already engaged in e-business, taking full advantage of the opportunities e-business can present requires planning. It also requires ready access to new technology along with the tools and information you need to accelerate your development efforts.

New technologies...

New technologies bring excitement and an ocean of possibilities to application development. No matter what platform you are developing for, the Developer Connection can help you chart a course to navigate e-business successfully. Visit our Web site at www.developer.ibm.com/devcon/. We can help you "deal" with e-business.

Come see us...

Be sure to stop by the Developer Connection booth on the exhibit floor to visit with us. We always enjoy talking to you and getting your input on how we are doing. Let us know what we can do to make the Developer Connection program even better!

A new breed of business for a new way of life

- MARKS THE SPOT.
- BOOGIE WWW.CDWAREHOUSE.COM
- AMAZING WWW.MACYS.COM
- GRAMMY WWW.GRAMMY.COM
- WONDER WWW.NATIONALGEOGRAPHIC.COM
- CLIMBS WWW.REI.COM
- BEAR
- BULL WWW.SCHWAB.COM
- HOPE WWW.DISASTERRELIEF.COM
- SOLE WWW.SAFETYSHOES.COM

LOOK FOR THE IBM E-BUSINESS MARK AS YOU TRAVEL THE WEB.

Updated OS/2 Developer's Toolkit

THE IBM OS/2 DEVELOPER'S TOOLKIT, VERSION 4.5, IS AVAILABLE IN DEVELOPER CONNECTION RELEASE 2 VOLUME 3.

UPDATES TO THE TOOLKIT INCLUDE:

- DEBUGGING AND SERVICEABILITY ENHANCEMENTS, INCLUDING UPDATES TO SYSTEM TRACE AND PROCESS DUMP FACILITIES
- DESCRIPTIONS OF NEW AND REVISED APIS THAT ARE INCLUDED IN OS/2 WARP SERVER FOR E-BUSINESS TCP/IP TOOLKIT ENHANCEMENTS, INCLUDING NEW SAMPLE FILES AND A NEW 32-BIT VERSION OF THE RING-0 LIBRARY

THE TOOLKIT IS AVAILABLE AT THE ADVANCED LEVEL AND CAN BE DOWNLOADED FROM THE DEVELOPER CONNECTION WEB SITE AT WWW.DEVELOPER.IBM.COM/DEVCON/. THE TOOLKIT ALSO IS AVAILABLE AT THE ADVANCED LEVEL ON THE VERSION 3 CDS.

Parallel Worlds: *Why Java and XML* will succeed

by Simon Phipps

In the last few years, the focus in computing has gradually moved away from the raw technology and has most recently settled on the total cost of ownership (TCO) for a solution. *But what makes up the TCO?* That's hard to say, and everyone has a different answer. It usually depends on what they find easiest to fix! Most people agree that the TCO is not made simply from the sum of the prices of the parts that make the system. It comes from those initially, but a much greater cost in the end can arise from the cost of supporting the system in context. A popular approach to reducing TCO has been to try to centralize the administration of individual systems, the client desktop or both, but that's only part of the answer. It's good to keep the amount of travel to a minimum, but what actually causes the administration to be needed? The answer, of course, is change. But not on its own. Change in isolation would only necessitate work on the change itself. We all know that making a change in one part of a system results in support needs throughout the whole system.

The typical computer system often is headed toward "entropy death," where the cost exceeds the value, and where ordered simplicity has tended towards interconnected complexity. While a cure for the symptom may be central administration, the actual disease mandates avoidance of the complex network of dependencies in the first place. It is this avoidance that Java and XML start to address by helping eliminate the automatic co-dependency of systems, software and data.

The typical computer system is often headed toward "entropy death," where the cost exceeds the value.

A new world

The need for much of the support and administration comes from the web of dependencies woven by the software in our computers. To bring back the simplicity, we need to cut the dependencies. Where are they all? There are several categories:

- Software to platform
- Software to data
- Software to software
- Platform to platform

To cut the cord of these dependencies is not easy, but the new world of computing that has been developing over the last decade is finally coming to maturity and making it possible.

Let's first consider the computing model we have been living with. When computing was new, the choices were easy to make. I could pick any of the limited range of computers, write software to run on it and create file formats to store the data in. Trouble was, the software and data would only work on that kind of computer, so when a different kind was used I had to use different software, or if I used different software on the same system I could not use the same data and had to learn a new user interface.

Many of the problems were solved by two standardization steps: many people began to use the IBM PC initially with DOS and then Microsoft® Windows.® A degree of simplicity came back. But as time went on, it became clear that there was still plenty of scope for complexity to creep in! In particular, agreeing on the platform did not break the platform dependency of the software; it just meant it was all co-dependent. So when an update came along, everything might break! In addition, there was no standardization beyond the power of monopoly in the world of the data. Just as the software depended on a particular level of the platform, the data related to a particular level of a particular brand of the software. A complex web of dependency was woven, where a change at any point might lead to instability and perhaps failure in the whole web.

SOLUTIONS '99 PRESENTER

Co-dependency

A great enemy of computing is the creation of unintentional co-dependencies. As computer solutions are built, they all involve relationships between the software, the hardware, the platforms, the development tools and so on. Each is connected to every other by unseen connecting threads of co-dependency. Over time, the cost of owning any solution is proportional to the number of support dependencies between the parts. But by the unintentional creation of many co-dependencies, the cost rises in an exponential rather than a linear way. The result is that the addition of further co-dependent elements can increase the lifetime cost disproportionately. The point at which this begins to apply is the race point, and the condition beyond the race point is termed entropy death. The inevitability of entropy death is set well before the race point by the act of choosing a system philosophy prone to co-dependency, the unwitting reliance of one part of a system with another, possibly mediated by some other element. The most common unwitting co-dependency is between software and the operating system it predicates.

This is not to say that all co-dependencies can or should be avoided; it is inevitable there will be some. But in modern system specification and design they all should be identified and justified in the same way as any other cost driver, taking into account not only the direct cost but also the lifetime cost inherited by connection to the dependency network as shown in ***Figure 1***. In general, software needs to be insulated from the environment where it is used. In

CONTINUED ON PAGE 4

CONTINUED FROM PAGE 3

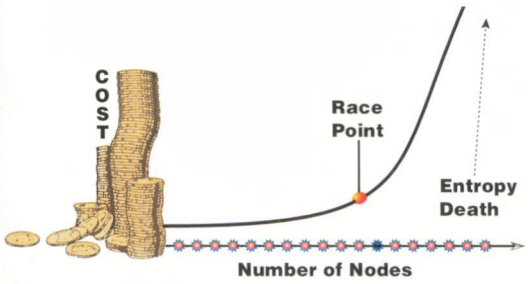


Figure 1. Cost and entropy death

some situations, use of native interfaces and binaries is unavoidable, but in these cases a platform-neutral 'wrapper' around the native code is almost always valuable.

For example, consider the apocryphal case of a company that has used the macro language of an office suite as the basis for an office automation system. One day, the company's IT group installs another piece of software and unknowingly updates one of the DLL files used by the office suite. They find that one of the macros no longer works. After a great deal of work, they manage to get the macro working again, but the new version needs an updated version of the spreadsheet program. To get that, they have to install a whole new level of the office suite, after which none of the macros work! They next crawl through all the macros, updating and fixing them. Among the other things the fixes mandate, they discover they need to use a new version of the database driver. Sadly, that needs the latest version of the database to work. So they upgrade the database and...well, you can guess the rest.

The new foundation

The problem is caused by the transmission of the impact of change from subsystem to subsystem. The integrated computing foundation currently in use in most systems can act as a transmission medium, allowing change in one place to have an impact elsewhere.

How can we escape this trap? The key to it all is to disconnect data from software from platforms, to use standards-based choices for all of these so that version-to-version variations of the implementation have the smallest effect possible. By doing this, we isolate changes from the transmission medium (the underlying platform) and prevent the impact of change from causing shock-waves of cost; we add the insulating layer mentioned earlier. *What would be an optimal base of standards?* The technology domains, shown in ***Figure 2***, such a foundation would have to cover are:

- The network protocols holding systems together and providing access
- The delivery model that brings the solution to the audience that needs it
- The programming model with which the solutions are created
- The data structuring model for the information the solutions consume
- The security model that allows the right audience access to the right data and solution

Much of the change in the computer industry over the last decade has involved the rediscovery of technology ideas and their establishment as standards within that model. The mappings, shown in ***Figure 3***, are:

- ***Network: TCP/IP***
  The use of TCP/IP has now become so widespread that it is no longer a topic of conversation.
- ***Delivery: Web model Stateless Client/Server***
  Stateless client/server computing is the chosen delivery mechanism of a growing number of business computer users. Rather than creating state-full clients, which need costly maintenance and support, state is instead maintained at the server and at most "loaned" to the client.
- ***Program: Java and JavaBeans***
  Only four years from release, Java has established itself as the standard for new software in a vast number of enterprises; its JavaBeans architecture allows component-based development to be used in earnest. This is not to say that all the code needs to be written in the Java language; it's the platform-neutral Java bytecode binary programs that win. Where these aren't feasible, at very least a wrapper of Java to insulate the rest of the solution from native code is essential.
- ***Data: XML and vocabularies***
  Apparently new to the scene, XML is actually simplified SGML – 80 percent of the function for 20 percent of the complexity. Uptake throughout the computer industry has been huge and it shows every sign of dominating data formatting in the future.
- ***Security: Public key***
  By removing the need to send full key information "in the clear," public key-based security systems are already dominant, especially on the Web.

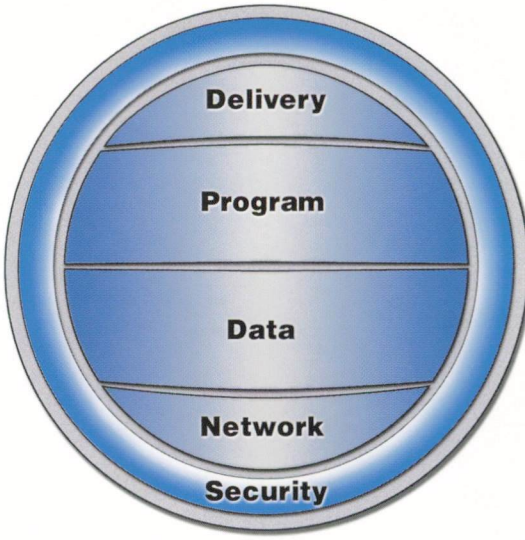


Figure 2. Technology domains

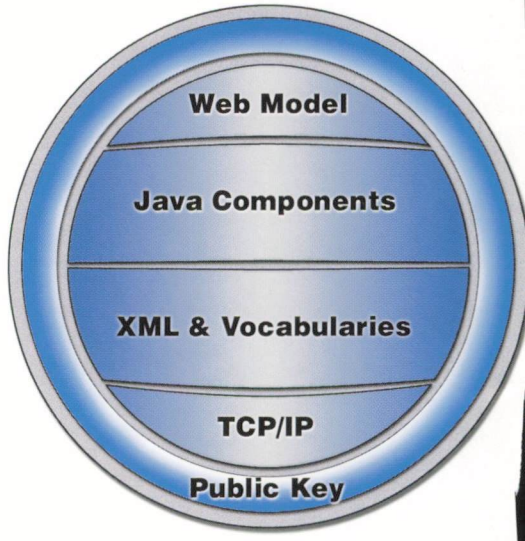


Figure 3. The new foundation

From technologies to audiences

Alongside the agreement of the standards for the new world of computing has been a shift in the requirements for business solutions. In the past, each solution would be built with the requesting customer only in mind. The focus was on who was using the solution and where they were, hence terms like intranet, extranet and Internet. But progress has meant that the focus now is much more on the modelling of all the data and the definition of the relationship of the user to the data. There has been an inversion in the approach to computing solutions, and the focus has switched from technologies and systems to information and audiences.

Today, defining a new solution involves

defining the relationship that an audience has to a body of information. In most cases, a given body of information has multiple audiences. Thus, for an online shop when customers are viewing information, generally access is given only to their particular data and it can be presented in a way to suit them. When customer service staff from the vendor view the same information, both the scope and the presentation differ. It is the transition to a solutions-and-audiences view that presents the greatest challenge in IT today. Users are able to proceed with confidence because all of the technologies in the new tradition are in fact mature and proven, so the transition is one of emphasis and strategy rather than a leap into unknown technology.

Parallel worlds

The fact that all five of the foundation technologies are well understood also offers another benefit. For many users, migration to the new world of e-business is something evolutionary rather than revolutionary. They are able to take the first steps without scrapping all the investment that they already have made. So this new world is in fact a parallel world rather than an alternate world.

So why, after all that, will Java and XML succeed? There are several reasons why:

- ***Proven technology:*** All five of the segments of the new foundation are based on the oldest, most proven ideas in the industry. TCP/IP, "dumb terminals," virtual machines, mark-up languages, public key systems – all have been proven through decades of use.
- ***User driven:*** In the final analysis, the move to the new foundation is driven by the needs and desires of the marketplace rather than by the fiat of any one vendor or even of a consortium. As the costs of computer technology become more of a focus item and those driven by the upgrade arms-race toward entropy death become more and more obvious, the demand for the new foundation becomes greater and greater.
- ***Vendor supported:*** All five technologies form the basis of almost all vendors' new solutions. Vendors choosing an alternate at any point increasingly discover the market questioning their choice and suspecting an attempt at proprietary lock-in.
- ***Platform neutral:*** All five technologies are independent of each other and of the platforms where they are used. Thus, they all

CONTINUED ON PAGE 6

XML

Keeping the information in the data!

WITH ITS ROOTS IN **SGML** (ITSELF THE DEVELOPMENT OF WORK CARRIED OUT BY **IBM** IN THE LATE '60S), **XML** IS VERY SIMPLY THE IDEA OF USING TAGS FOR FORMATTING ALL DATA, NOT JUST TEXT INTENDED TO BE DISPLAYED IN A WEB BROWSER. YOU MIGHT ALREADY INSERT PSEUDO-**HTML** IN YOUR E-MAIL, LIKE THIS:

<JOKE> WHY DID THE CHICKEN CROSS THE ROAD ?</JOKE>

<PUNCHLINE> TO GET TO THE OTHER SIDE </PUNCHLINE>

WHEN YOU DO THIS, YOU'RE ACTUALLY CREATING A FRAGMENT OF **XML**! YOU'VE DEFINED A SMALL VOCABULARY FROM WHICH THE TAGS YOU'VE CHOSEN ARE DRAWN (LET'S CALL IT **JOKEML**), YOU'VE MADE SURE IT IS WELL-FORMED (HAS CLOSING TAGS TO MATCH ALL THE OPENING TAGS) AND APART FROM A LINE TO SAY WHERE THE DEFINITIONS OF THE TAGS CAN BE FOUND (A **DOCUMENT TYPE DEFINITION** OR **DTD**) AND A LINE TO SAY IT'S **XML**, YOU'RE ALREADY AN EXPERT! ANYONE WHO UNDERSTANDS WHAT A JOKE IS WILL UNDERSTAND THIS DATA (ALTHOUGH THERE IS NO GUARANTEE THEY WILL FIND IT FUNNY), SO INSTEAD OF JUST BEING DATA IT'S TURNED BACK INTO INFORMATION.

AS MORE AND MORE VOCABULARIES ARE DEFINED, **XML** IS BEING USED FOR TAGGING DATA IN ALL SORTS OF APPLICATIONS. IT HAS THE GREAT BENEFIT OF BEING DIGESTIBLE BY COMPUTERS AS WELL AS HUMANS — AFTER ALL, SOMETHING LIKE:

<DL>

<DT> WHY DID THE CHICKEN CROSS THE ROAD?

<DD> TO GET TO THE OTHER SIDE

</DL>

MIGHT CLEARLY BE A JOKE TO HUMAN EYES BUT A COMPUTER WOULD NEVER GUESS, WHEREAS THE **XML** FRAGMENT WOULD BE ACCESSIBLE TO ANY PROGRAM THAT UNDERSTOOD **JOKEML**.

XML WILL BE USED IN THE FUTURE ANYWHERE THERE IS DATA — NOT JUST ON THE **WEB**, AND CERTAINLY NOT JUST IN **WEB** BROWSERS. BY PUTTING THE FRAMEWORK OF MEANING BACK INTO THE DATA, IT BRINGS LIFE TO THE DATA, TURNS THE **WORLD WIDE WEB** INTO A WORLDWIDE DATABASE AND MAKES COMMUNICATION BETWEEN BUSINESSES POSSIBLE WITHOUT ALL THE PAIN OF THE **EDI** PROCESS.

TO LEARN MUCH MORE ABOUT **XML** AND RELATED STANDARDS, TECHNOLOGIES AND TECHNIQUES, START AT **http://www.ibm.com/xml/**

CONTINUED FROM PAGE 5

can be implemented anywhere, insulating the systems that depend on them from co-dependency.

- ***Vendor neutral:*** All five technologies are beyond the control of any one vendor, so that investments are protected from both the risks of vendor lock-in and also from the design choices of any one vendor starting an upgrade race. The only possible exceptions to this are Java and public key, and it is worth taking time to consider why neither is a problem in this context.

Java: public property?

Can a technology developed and apparently controlled by a single vendor be considered open? It all depends on the attitudes and actions of the vendor. In the case of all five domains in the new computing foundation, control has passed from the originator to "the mind of the market." For example, although the core ideas of public key systems are owned by one company, the industry has been willing to base almost all encryption and digital signatures on that technology, because of a combination of the open power of the technology and the attitude of the owners of the core patents. In the same way, Java has become public property that is currently protected rather than compromised by the owner of the core technology, although a move to standards body control would be positive. What is more, that ownership is nowhere as firm as in the case of public key systems. If the whole industry chose to implement Java differently, there would be almost no recourse. But that does not happen because any company seen to violate the value of Java is shunned by the market. The fact that the base of standardization in Java is actually the binary format of bytecodes rather than the language is, of course, a big help. Thus, if we feel safe basing key parts of the computing infrastructure on public key systems, there is all the more reason to feel safe using Java.

Conclusion

The key issue that should occupy us is not "how can I cut the cost of administration and support" but rather "how can I reduce towards elimination the amount of admin-support that is needed?" To reflect this changed concept, and to progress from the notions that sometimes turn consideration of TCO into TCP (total cost of purchase), we should perhaps use a modified term to express the issue in hand – lifetime cost of ownership or LCO.

The core assertion of this article is that the primary decision factor for new computer systems should be the cost of owning the system over its entire life: that is, to base a decision on LCO – of software, network, client and server hardware, complete with development, deployment, administration, management of impact during life-cycle and migration to replacement systems at sundown. The core proposal of the article is that this factor can be controlled by minimizing the network of co-dependent complexity that these various elements create. To achieve this control, a change of system philosophy rather than an instant change of technology is proposed. By basing future developments on a firm foundation of standards, entropy death can be avoided. And this is the reason Java and XML can succeed, cool though the technologies themselves may be!

Simon Phipps is IBM Corporation's chief Java and XML evangelist. Having been part of the team that recommended Java to IBM in 1995, he has since spoken worldwide on the new world that is engulfing computing, powered by Web and Java technologies. His new focus is eXtensible Markup Language, and he has worldwide responsibility for XML evangelism for IBM. With over 20 years experience in the computer industry, Simon has worked on networking, data communications and operating systems for various companies in many contexts including the development of the earliest commercial collaborative conferencing software with IBM. He joined IBM in 1991. He holds a degree in electronic engineering and is a Chartered Engineer and Member of the British Computer Society.

DB2 UDB for AS/400

THE DELIVERY OF NEW FUNCTIONS THAT DEFINE DB2 UNIVERSAL DATABASE FOR AS/400 (DB2 UDB FOR AS/400) LATER THIS YEAR, MARKS ANOTHER STEP TOWARDS COMMON FUNCTION ACROSS THE DB2 UDB FAMILY. WHILE EACH VERSION OF DB2 UDB HAS ITS DIFFERENCES DUE TO THE STRENGTHS OF THE UNDERLYING HARDWARE PLATFORM AND OPERATING SYSTEM, MUCH OF THE DATABASES ARE IDENTICAL.

DB2 UDB FOR AS/400 BRINGS WITH IT AN ENORMOUS WEALTH OF NEW FUNCTION. THE MOST SIGNIFICANT OF THESE FUNCTIONS IS THE ABILITY TO STORE AND MANIPULATE NON-RELATIONAL DATA IN THE RELATIONAL DATABASE.

EXAMPLES OF NON-RELATIONAL DATA INCLUDE IMAGES, VIDEO AND SOUND AND ALSO INCLUDE MORE TRADITIONAL BUSINESS DATA SUCH AS SPREADSHEETS AND DOCUMENTS. THESE TYPES OF DATA ARE KNOWN AS BLOBS. THE BASIS IN STORING THESE TWO DISPARATE TYPES OF DATA IS TO MAINTAIN THE RELATIONSHIP THAT EXISTS BETWEEN THEM. PICTURES OF ITEMS SOLD CAN BE KEPT WITH THE DESCRIPTIONS AND SKU INFORMATION.

DB2 UDB FOR AS/400 ALLOWS THE NON-RELATIONAL DATA TO BE STORED IN THE DATABASE TABLE OR OUTSIDE IN A FILE SYSTEM WHILE MAINTAINING THE LINKAGE BETWEEN THE ROW OF DATA AND ITS ASSOCIATED DATA. MAINTAINING THIS RELATIONSHIP IS DONE BY SUPPORT CALLED DATALINKS.

OTHER NEW ASPECTS OF THE RELEASE INCLUDE USER-DEFINED DATA TYPES AND USER-DEFINED FUNCTIONS. THESE FEATURES ALLOW MORE OF THE BUSINESS RULES AND PROCESSES TO BE STORED WITH THE DATA.

THIS RELEASE ALSO MARKS MORE SUPPORT FROM IBM SOFTWARE PRODUCTS FOR THE AS/400. VISUAL WAREHOUSE RECENTLY WAS ENHANCED TO PROVIDE NATIVE AS/400 DATA MOVEMENT AGENTS. IBM ALSO RECENTLY ANNOUNCED THAT THE AS/400 WILL HAVE COMMON OLAP TECHNOLOGY WITH OTHER FAMILY MEMBERS.

AVAILABLE IMMEDIATELY IS ESSBASE/400 FROM SHOWCASE CORPORATION. THIS PRODUCT IS BEING MADE AVAILABLE DIRECTLY FROM IBM. DB2 OLAP PRODUCT IS EXPECTED TO BE AVAILABLE LATER THIS YEAR. BOTH OF THESE PRODUCTS ARE BASED ON HYPERION SOLUTIONS ESSBASE PRODUCTS.

Using Java to bridge
Domino R5 and WebSphere

by Jim Hsu and Bill Lawton

The recent release of WebSphere 2.02 and Domino R5 creates a winning combination for sophisticated enterprise-class Web sites. Domino provides a Web-enabled foundation for group collaboration and workflow. WebSphere is used to power Web applications that utilize servlets and Java Server Pages (JSP).

WebSphere Standard Edition is packaged with Domino R5, but WebSphere Advanced Edition also can be installed and used with Domino R5. WebSphere Advanced Edition provides Enterprise JavaBeans (EJB) support.

Together, WebSphere and Domino R5 open the door to exciting possibilities for e-business solutions that leverage the strengths of both technologies. This article discusses how to install WebSphere onto the Domino R5 server and how to use Java effectively in this environment.

Enabling WebSphere on Domino R5

WebSphere plugs into a variety of popular Web servers, including Apache Server, Netscape FastTrack and Enterprise Servers, IBM HTTP Server, Microsoft Internet Information Server (IIS) and, of course, Lotus Domino R5.

To enable WebSphere support on a Domino R5 server, simply select the Domino R5 Web server plug-in option during the WebSphere installation process. Make sure the JSP check box is selected during installation to ensure automatic configuration of Domino R5 for both servlet and JSP support. After installation, the server hostname field should be set in the Domino server configuration. ***Figure 1*** shows the handling of incoming Web requests by a Domino R5 server with WebSphere installed.

Any existing WebSphere application is easily moved to Domino R5. The Web server files for the application are copied into the Domino HTML directory at \Lotus\Domino\Data\domino\html. If the application uses SSL, check that Domino SSL support has been enabled. If not, create a keyring using the Domino R5 Certificate Authority database (cca50.ntf) and set the keyring field in the Domino server configuration.

Domino R5 and WebSphere also can be used in combination with IIS. The IIS Web server plug-in option should be selected during the WebSphere installation process. In this configuration, both Domino and WebSphere receive Web requests from IIS server as shown in ***Figure 2***.

Java objects in WebSphere and Domino

Now that we have a server running Domino and WebSphere, we can start to look at the possible benefits of interoperability. The key is to leverage the interactions between objects in both the WebSphere and Domino environments by using Java as a common language to bridge the two. Let's start with an examination of the Java-based features in WebSphere and Domino so we can understand the potential advantages of using them together.

WebSphere servlets are Java software components that execute on the server and follow the Servlet API. As such, they have easy access to HTTP parameters, session state and cookies. A big potential advantage to using servlets is that they stay loaded in memory until the server is shut down or restarted. This can provide a big performance boost over CGI-BIN programs and Domino agents, which are loaded each time they are invoked and unloaded as soon as they are finished executing.

SOLUTIONS '99 PRESENTER

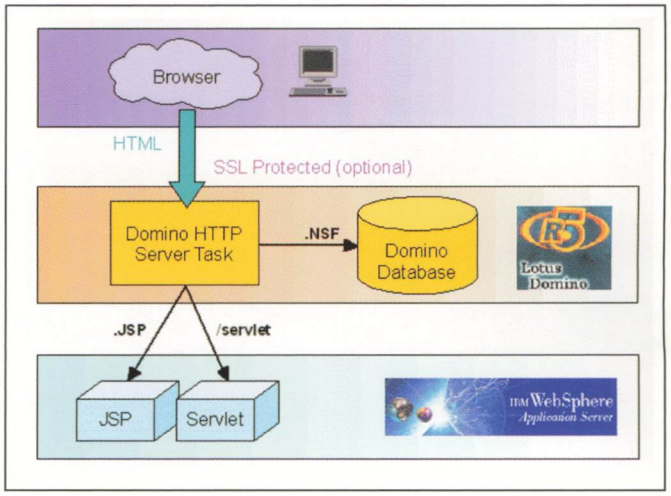


Figure 1: Delegation of Web requests on Domino R5 with WebSphere enablement

Together, WebSphere and Domino R5 open the door to exciting possibilities for e-business solutions that leverage the strengths of both technologies.

WebSphere also comes with JSP support, which allows a page creator to mix Java code "snippets" into HTML. JSPs provide scripting language support that executes on the server. JSPs are compiled into servlets at run time, so they generally enjoy the same performance benefits as servlets.

The Advanced Edition of WebSphere provides support for Enterprise JavaBeans. EJBs are reusable software components that can be transactional, persistent, distributed and secure. These properties are important for creating e-business Web applications.

In Domino, there are several ways to program in Java. Java applets can be embedded

CONTINUED ON PAGE 8

CONTINUED FROM PAGE 7

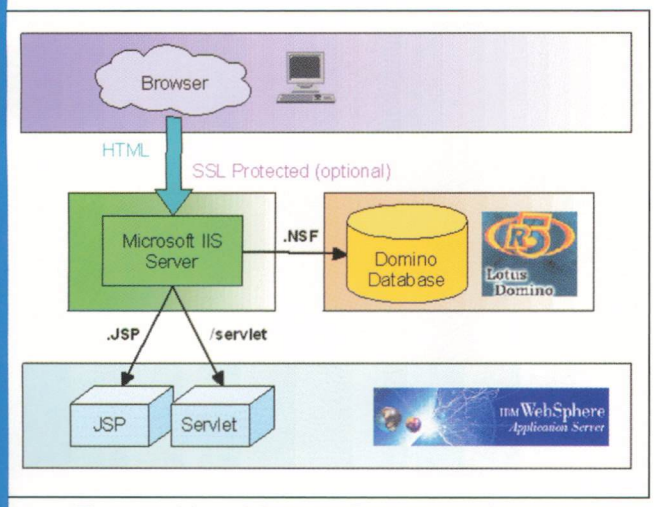


Figure 2. Use of Domino R5 and WebSphere in conjunction with IIS

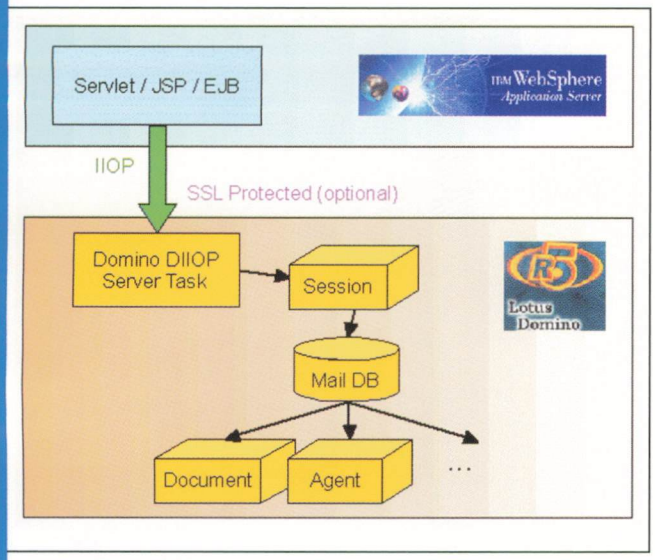


Figure 3. Accessing Domino objects from WebSphere

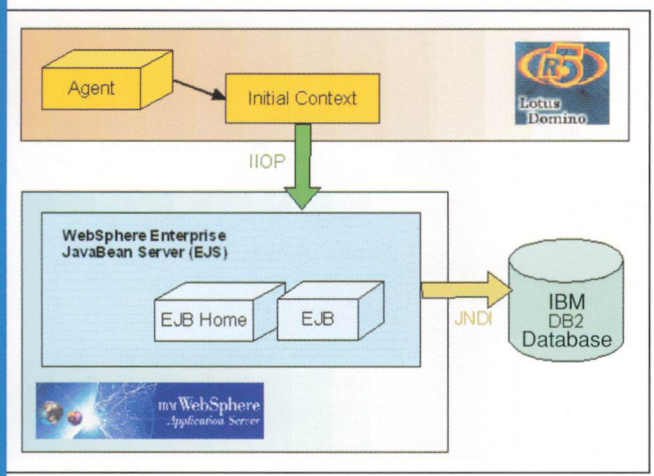


Figure 4. Accessing WebSphere EJBs from a Domino agent

into Domino Web pages. Also, a Java program running locally on the server can access Domino objects with server permissions through the Notes API. The Notes API also allows access through a remote IIOP interface. Finally, a Java agent can be used. Java agents are programs that reside in a Domino database and perform actions on behalf of a user. Like servlets, they can be triggered from a URL, but agents also can be scheduled to run every so often or when a server-side event occurs.

Additional details about using the built-in Java support in Domino can be found in the article "Writing Java Programs for Domino R5." This article was published in the June issue of *IBM Developer Connection Technical Magazine* and contains detailed sample code including versions of the AgentLauncher and DocumentCreator classes that are used in the next example.

Accessing Domino objects from WebSphere

WebSphere servlets, JSPs and EJBs can access Domino objects using the remote IIOP interface to the Notes API (as illustrated in ***Figure 3***). In our first example, a JSP was used to trigger a Domino agent.

Because resources in Domino may be protected by Domino access controls, you can use a Domino user ID and password to obtain proper security permissions. This information is first gathered in a Web login page shown in ***Listing 1***. This is an HTML form that calls our JSP with two parameters, uid and pwd. The JSP extracts the two parameters and

```
<HTML><HEAD><TITLE>AgentFromJSP.html</TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">

<H1>AgentFromJSP Login Form</H1>
This form will submit data to a <B>JSP</B> for processing.
<P>Please log in using your userid and password.

<FORM METHOD="POST" ACTION="AgentFromJSP.jsp">
<FONT SIZE=2 FACE="Arial">User ID: </FONT>
<INPUT TYPE="TEXT" NAME="uid" SIZE=20 MAXLENGTH=20><BR>
<FONT SIZE=2 FACE="Arial">Password: </FONT>
<INPUT TYPE="PASSWORD" NAME="pwd" SIZE=20 MAXLENGTH=20><BR><BR>
<INPUT TYPE="SUBMIT" NAME="submit_btn" VALUE="Submit">
<INPUT TYPE="RESET" NAME="reset_btn" VALUE="Reset">
</FORM></BODY></HTML>
```

Listing 1. AgentFromJSP HTML Login Form

```
<%@ import = "java.io.*,java.util.*,lotus.domino.*" %>
<%
   String[] values;
   String uid = "";
   String pwd = "";

   values = request.getParameterValues("uid");
   if ((values != null) && (values[0] != null) && (values[0].length() > 0))
      uid = values[0];

   values = request.getParameterValues("pwd");
   if ((values != null) && (values[0] != null) && (values[0].length() > 0))
      pwd = values[0];

   out.println("Executing Agent on behalf of user " + uid);
   out.println("\n<P>");

   // Create AgentLauncher
   test.AgentLauncher agentLauncher = new test.AgentLauncher();
   agentLauncher.uid = uid;
   agentLauncher.pwd = pwd;

   Thread thread = new Thread((Runnable) agentLauncher);
   thread.start();
%>
```

Listing 2. AgentFromJSP.jsp File

uses them on the AgentLauncher object provided in ***Listing 2***.

AgentLauncher uses the login information to create a remote IIOP session with a Domino server. If the user permissions are correct, AgentLauncher obtains a DocumentCreator agent and executes it. The source code for AgentLauncher is given in ***Listing 3***.

To use AgentLauncher, be sure to customize the constants for your system and compile it. The resulting AgentLauncher.class file should be copied into the WebSphere\AppServer\web\classes directory so that it can be loaded by WebSphere. For WebSphere to run AgentLauncher, the remote session classes from the Domino NCSO.jar file should be added to the WebSphere bootstrap.properties classpath.

Accessing WebSphere EJBs from Domino

Our second example uses IIOP to access a WebSphere EJB from a Domino agent. This scenario is illustrated in ***Figure 4***.

Listing 4 gives the source code for a Domino agent that accesses the IncBean EJB sample that is included in WebSphere Advanced Edition. It

is essentially a modified version of the com.transarc.jmon.examples.Inc.IncClient class. When the agent is executed, a counter is incremented by five in the database just like in the Inc demo in the WebSphere EJB samples page. The samples page is installed locally on the server by WebSphere at http://servername/IBMWebAS/samples/ejs/index.html.

Because this agent uses outside classes like IncBean, it uses the "Imported Java" option with the following three files:

- EJBFromAgent.class (Customize and compile the source code from ***Listing 4***)
- Inc.jar (located in \WebSphere\AppServer\deployableEJBs)
- ejs.jar (located in \WebSphere\AppServer\lib)

e-business example scenario

The above samples demonstrate two-way interaction between WebSphere and Domino objects. Next, let's discuss an example e-business scenario that takes advantage of this interoperability by using servlets, EJBs and Domino agents together to automate the task of e-mail confirmation.

Many e-business Web applications require users to provide a valid e-mail address. This feature is especially important to applications that need to send information to customers. For example, a requirement for an online trading site might be that a prospectus must be sent to all mutual fund buyers. If the customer has a confirmed e-mail address, the prospectus can be sent by e-mail instead of regular mail, thereby cutting costs. Another example is the use of a confirmed e-mail address to allow users to submit bids on an online auction site.

To implement e-mail confirmation in WebSphere you might have an HTML form with a submit button that calls a servlet. The servlet processes the data that was entered and creates an EJB instance containing data for the user. Then, the servlet sends a confirmation message to the user. To send the e-mail using the Domino R5 mail server, the servlet could create a mail document and send it by calling the Notes API, much like the first example. This technique also could be used to subscribe to a mailing list from a Web page or to send confirmation e-mail when a user makes a purchase from an e-commerce Web site.

A convenient way for the user to activate the account is to reply to the confirmation e-mail. If this is the case, you need a Domino agent to handle the responses. You set the agent to run when new mail is received, so it is triggered when the user replies. The agent could verify the message and call an account management EJB to activate the user's account. This uses the technique described in the second example. Similar agents might handle account deactivation or mailing list unsubscribe requests.

Summary

The interaction of WebSphere and Domino objects has been made possible by open APIs that cross both environments using the IIOP protocol. WebSphere servlets, JSPs or EJBs can access Domino objects using the Notes API. Conversely, Domino agents can access EJBs running on WebSphere. The samples and scenarios in this article provide a good starting point from which to build e-business applications that take advantage of the full potential of WebSphere and Domino interoperability.

Jim Hsu *is a Software Engineer who joined IBM in May 1997. He is a member of the Software Group Technology Center in Austin, Texas. Jim received his bachelor's and master's degrees in electrical engineering and computer science from MIT. He is interested in technologies with immediate widespread impact and has worked on the development of Java programs that involve the Internet. You can contact him at jimhsu@us.ibm.com.*

Bill Lawton *is a Senior Development Manager. He currently manages the Software Group Technology Center. The SWGTC innovates in the areas of e-business framework middleware with a focus on Web-based Java software technologies. Bill is the author of a book on multimedia and holds several patents. You can contact him at lawton@us.ibm.com.*

```
package test;

import lotus.domino.*;

// Located in \WebSphere\AppServer\web\classes\test
public class AgentLauncher implements Runnable
{

  // Modify these constants prior to running
  final String SERVERNAME = "ncf5.austin.ibm.com";
  final String AGENTNAME = "DocumentCreator";
  final String DBNAME = "mail\\jhsu";

  // Allow execution from command prompt for testing
  public static void main(String argv[]) {

    AgentLauncher agentLauncher = new AgentLauncher();

    if (argv.length >= 1)
      agentLauncher.uid = argv[0];
    if(argv.length >= 2)
      agentLauncher.pwd = argv[1];

    Thread thread = new Thread((Runnable) agentLauncher);
    thread.start();
  }

  public void run() {
    try {

      // Start remote IIOP session
      Session session = NotesFactory.createSession(SERVERNAME, uid, pwd);

      // Get database
      Database database = session.getDatabase(null, DBNAME);

      // Call agent
      Agent agent = database.getAgent(AGENTNAME);
      agent.run();
    }
    catch (NotesException e) {
      System.out.println("Notes Error #" + e.id + " " + e.text);
      e.printStackTrace();
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Listing 3. AgentLauncher Source Code

```
import lotus.domino.*;
import java.io.*;
import java.util.*;
import javax.naming.*;
import com.transarc.jmon.examples.Inc.*;

public class EJBFromAgent extends AgentBase
{

  // Customize these constants for your system
  public final static String NAMING_URL = "iiop://ncf5.austin.ibm.com:9019";

  public void NotesMain() {
    try {

      String primaryKey = "The Count";
      Properties p = new Properties();
      p.put("java.naming.factory.initial",
        "com.ibm.jndi.CosNaming.CNInitialContextFactory");
      p.put("java.naming.provider.url", NAMING_URL);

      InitialContext ic = new InitialContext(p);
      java.lang.Object tmpObj = ic.lookup("IncBean");
      IncHome incHome = IncHomeHelper.narrow((org.omg.CORBA.Object)
tmpObj);
      Inc inc = null;

      try {
          inc = incHome.findByPrimaryKey(new IncKey(primaryKey));
      } catch (Exception e) {
          inc = incHome.create(new IncKey(primaryKey));
      }

      for (int i = 0; i < 5; i++)
        inc.increment();
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Listing 4. EJBFromAgent Source Code

Enterprise JavaBeans: an IBM unifying model

by Mickey Nix

Java components have become an important part of client-side programming, and the JavaBean specification has proved its value. With the emphasis now on server-side computing, demand has risen for a similar component model on the server. As customers and vendors engineer multi-tier applications, Enterprise JavaBeans (EJBs) are emerging as a common design point and provide a standard, open, cross-industry component model for developing middle-tier (application server) architecture. IBM recognizes the importance of Enterprise JavaBeans and is an active and significant partner in the promotion and definition of EJB and Java technology. In addition, IBM has refocused its internal software development efforts making EJBs a first-class element across the middleware, tools and applications in the IBM Application Framework for e-business.

EJBs are logical bridges between application clients and data servers. Today's Web application servers provide enterprise Java services and facilities that allow developers to exploit and integrate new and legacy applications with a common external interface. When defining a server-side component model, there are many factors that must be considered, but three dominate: data representation or persistence, object services and client-side access.

The EJB programming model is robust and flexible enough to allow developers to build new component-based frameworks or to wrap legacy data and applications, and to deploy both EJB solutions into a single Web Application Server with a single client interface. While it is beyond the scope of this paper to describe the EJB specification in detail (more information can be found at java.sun.com), this paper examines IBM products that are EJB-enabled today and speculates on what the future may hold.

EJB development environment

VisualAge for Java Enterprise Edition Version 2.0 with Rollout2 and Enterprise Update provides an EJB development environment for creating, testing and deploying *session* (transient) and *entity* (persistent) EJBs. An entity EJB provides an object view of persistent data. The EJB container masks the complexities of mapping this object view to a relational database. This masking is important because it allows a programming team to simplify their designs by separating objects and business logic from underlying database, transaction and security programming details; it is the implementation of the container that provides these complex mechanisms and services. Currently VisualAge for Java (VAJ) provides support for building session EJBs, bean-managed persistence (BMP) entity EJBs and container-managed persistence (CMP) EJBs for IBM Universal DB2* 5.2 databases.

The VisualAge for Java integrated development environment (IDE) also includes a minimal IBM WebSphere Application Server that can be used for testing and debugging CMP and other types of EJBs within the IDE itself. These EJBs can be exported as Java JAR files and then deployed and executed in WebSphere Application Server (WAS) 2.0 Advanced, which provides container support for Universal DB2 databases. Even though WAS 2.0 Advanced is IBM's initial release supporting EJBs, the offering meets Sun's EJB specifications and provides support for deploying entity EJBs. In addition, a beta version of Component Broker 2.0 provides a set of tools that allow developers to wrap existing EJBs as Component Broker managed objects. Wrapping the EJBs gives them access to even more robust run-time management services such as workload management and availability management over multiple servers in addition to persistence, transaction and security functions. The following EJB creation example steps through the process of creating, testing, and deploying a CMP bean using IBM's tool suite.

Enterprise JavaBeans (EJBs) are emerging as a common design point and provide a standard, open, cross-industry component model for developing middle-tier (application server) architecture.

SOLUTIONS '99 PRESENTER

With EJBs, object-oriented application development can be approached in the following ways:

- ***Top-down:*** Objects are directly derived from functional requirements – modeling and analysis define the classes and behaviors that must be implemented.
- ***Bottom-up:*** Objects are engineered to wrapper existing functions, applications and databases – which preserves investments in legacy systems and improves productivity through reuse of existing classes.
- ***Meet in the middle:*** Objects are created using a combination of the top-down and bottom-up approach.

While top-down development is considered the more ideal model for object-oriented development, the meet in the middle approach is used more often in customer and vendor application development where it is important to slowly integrate a new object-oriented framework and preserve existing development efforts.

In the following example, the bottom-up approach is used to reengineer a legacy database named SAMPLE that ships with Universal DB2. Specifically, the example creates a CMP bean to wrapper the SAMPLE table entitled EMPLOYEE. The EMPLOYEE table contains the following fields: EMPNO (a primary key representing an employee number), FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE, JOB, EDLEVEL, SEX, BIRTHDATE, SALARY and BONUS.

The CMP bean is created, tested and debugged within the VisualAge for Java EJB development environment and then deployed in the WAS. Following that example will be a brief discussion of how the object then can be deployed in a Component Broker run time at some point in the future. The exercise is meant to provide a quick-start to developers who want

to explore the VisualAge for Java EJB development environment for the first time. Sample code and a detailed set of instructions for working through the exercise can be found on the Developer Connection Web site and CDs in a Freelance* format.

Preparing the development environment

The following instructions assume that Windows NT▲ 4.0 with Serviced Pack 3 has been installed on a workstation. Begin the preparation of the development environment by installing the IBM WebSphere Application Server 2.0 Advanced product. This single installation program installs JDK 1.1.6, IBM Universal DB2 5.2, the IBM HTTP Server and the IBM WebSphere Application Server.

After the installation completes and after rebooting, complete the following steps to further configure the development environment:

1. Using the DB2 Control Center, break down the System tree until the DB2 element is visible. While the DB2 element is selected, choose Selected -> Configure from the menu to invoke the Configure Instance – DB2 window. From this window, specify the Java Development Kit 1.1 Installation Path.
2. From a Web browser, link to http://localhost:9527 to invoke the WebSphere Application Server Administration Tool. Log in and go to the Setup -> Java Engine. Ensure that X:/SQLLIB/Java/DB2Java.zip (where X is the drive on which Universal DB2 has been installed) appears in the Application Server Classpath.
3. Next, install VisualAge for Java 2.0, Enterprise Edition, the Rollout2 fix pack and the Enterprise Update – in that order. The Rollout2 and Enterprise Update patches can be found at the VisualAge for Java Web site at www.software.ibm.com/ad/vajava or on the VisualAge Developer Domain link at www.software.ibm.com/ad/r/vadd8.
4. After installing the Enterprise Update, go to the VisualAge for Java Workbench and select File -> Quick Start -> Features -> Add Feature -> IBM EJB Development Environment 1.1 to actually add the EJB development environment to the integrated development environment in VisualAge for Java.

Two additional steps to enable a JDBC connection between VAJ's database schema mapping utility and DB2 are needed later in the exercise:

1. Unzip X:/SQLLIB/Java/DB2Java.zip into a temporary subdirectory.
2. From the VisualAge for Java Workbench, import the temporary subdirectory into the workspace.

Adding and importing EJB groups and EJBs

To build an EJB, do the following as shown in ***Figure 1***:

1. Go to the VisualAge for Java Workbench and add a new project. In this example the project is named EJBSampleDB.
2. Switch to the EJB tab on the Workbench and select Add -> EJB Group to create a new EJB group named IBM_Sample_DB_EJBs and associate it with the EJBSampleDB project.
3. Right click on the newly defined group to invoke its pop-up menu and select Add -> EJB to add an EJB to the group.

This exercise focuses on the creation of an entity EJB with container-managed persistence (CMP) fields that encapsulates the fields of the EMPLOYEE table in the SAMPLE database. Name the bean Employee and ensure that its bean type is Entity Bean with container-managed persistence fields (CMP).

On the second page of the SmartGuide, select that the Create finder helper interface to support finder methods check box and press the Finish button. The following classes are generated by the Create EJB SmartGuide and appear in the Types pane of the EJBs development page.

- ***Employee (Remote Interface):*** Defines the methods implemented by the EJB
- ***EmployeeBean (EJB Implementation):*** Provides the server-side implementation of the EJB
- ***EmployeeHome (Home Interface):*** Defines the methods used to create or find the EJB
- ***EmployeeKey (Key Class):*** Used as a method parameter to create or find an entity EJB
- ***EmployeeBeanFinderHelper (Finder Helper):*** Helper class for special find methods for CMP entity EJBs

The generated classes and interfaces are independent of the EJB container and, with VisualAge for Java, all of these classes and interfaces can be managed as a single entity in the EJBs pane of the EJB development environment. For example, VisualAge for Java helps ensure that when a method is deleted from the EJB bean class, its corresponding method in the remote interface is deleted automatically. In addition, EJBs can be versioned as a single entity and EJB source code and associated packages are synchronized.

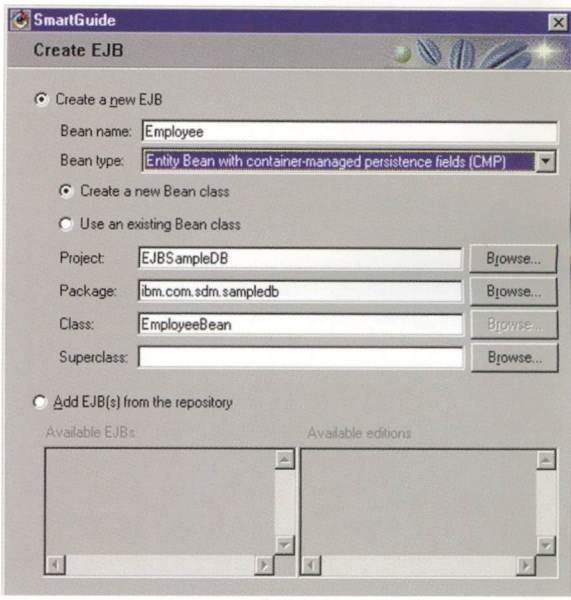


Figure 1. Building an EJB

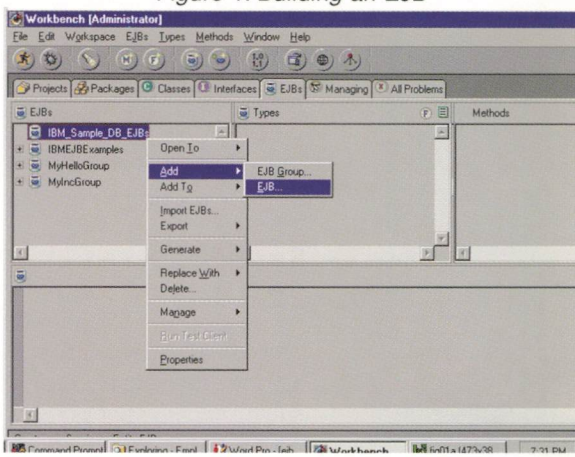


Figure 1a. Building an EJB

Adding, defining and mapping CMP bean fields

Once the initial EJB has been generated, a field named primaryKey of typejava.lang.String has been added. This field later is mapped to the EMPLOYEE primary key column named EMPNO. New data instance fields can be added to the EJB to reflect the columns in the EMPLOYEE table. For example, the following fields have to be added to the EJB because their corresponding columns in the EMPLOYEE table are tagged with the NOT NULL attribute:

CONTINUED ON PAGE 12

CONTINUED FROM PAGE 11

```
public String       firstName ;
public Character    midInit ;
public String       lastName ;
public short        edLevel ;
```

Of course, other fields such as:

```
public java.math.BigDecimal salary ;
public java.math.BigDecimal bonus ;
```

could be added, and are needed for the exercises to follow.

To add a field, select EmployeeBean and right click. Select Add -> Field from the pop-up menu. In the Create Field SmartGuide, make sure that the Access with getter and setter methods check box is selected. After creating each of the fields, the new fields must be designated as Container Managed fields. First, press the Fields icon (circle with an F inside) in the upper-right corner of the Types pane to toggle to the Fields pane. Select each new field and right click to invoke its pop-up menu. In the pop-up menu, select Container Managed. This tells the VisualAge for Java code generation tool that these fields should be mapped automatically to an underlying data store such as DB2. Additional steps are necessary to provide the code generation tool with the information it needs to map the EJBs container-managed fields to the EMPLOYEE tables in the SAMPLE database:

1. Open the Schema Browser tool from the VisualAge for Java Workbench (use the "octopus-like" icon on the far right of the tool bar) and import the EMPLOYEE schema from the SAMPLE database. In this example the schema has been saved to the VisualAge for Java development environment with the name SampleDBSchema.
2. Using the Table Editor in the Schema Browser, associate a VapStringToCharacterConverter with the MIDINIT table column as shown in ***Figure 2***. This allows mapping of a java.lang.Character class to an SQL char type. By default all fields are associated with VapConverter, which passes datum "as is" without doing any conversion.

   Note: A better database design would use a Java primitive character rather than the class java.lang.Character – the approach used in this article reflects the associated lab's requirement to demonstrate the use of a VapConverter.
3. Use the VisualAge for Java Map Browser to associate the EJBs container-managed persistent fields with EMPLOYEE data columns, shown in ***Figure 3***. To create an EJB Group Map:
   a. Click on the icon to the left of the Schema Map icon on the tool bar (looks like a globe with longitude and latitude lines).
   b. On the Datastore_Maps menu in the Map Browser, select New EJB Group Map – the New Datastore Map dialog appears.
   c. Name the new map to be created (SampleDBMap for example), associate it with an EJB group (IBM_Sample_DB_EJBs) and associate it with an existing database schema (SampleDBSchema).
   d. On the Datastore_Maps menu in the Map Browser, ensure that SampleDBMap and Employee are selected from the Map Browser and then select Table_Maps -> New Table Map -> Add table map with no inheritance.
   e. Ensure the new table map is selected and then select Table_Maps -> Edit Property Maps to launch the Property Map Editor.
   f. On the Attributes pane of the Property Map Editor, map the container-managed fields of the EJB to their corresponding table columns in the EMPLOYEE table.

Because each of the fields was a Java primitive or String, each of the map types is specified as Simple. However, it is possible to map complex objects (classes with field types other than Java primitives and Strings) using VapAttributeComposer classes. The VisualAge for Java EJB Development Environment documentation details how to create these classes. Be aware that the documentation fails to mention one class that must be added to a composer:

```
public TargetClass singleton() {
    if (singleton == null) {
        singleton = new TargetClass() ;
    }
    return singleton ;
}
```

For more information, see the EJB Development Environment document that is included with the Enterprise Update.

Adding create, finder and business methods

Now that the EmployeeBean is wired for persistence, focus can be turned to adding create and finder methods to the bean's home interface and to adding business methods to the bean's remote interface. An EBJ's home interface performs two roles: it provides factory methods for creating instances of the EJB class and it functions as a type of controller or manager in that all SQL select queries against the underlying datastore are contained here. When the EJB is first generated, a default create method, ejbCreate( ), is provided. In this exercise, the developer will need to add at least one other create method with a parameter list that can be used to create and initialize a new Employee EJB. Add a new ejbCreate method to the EmployeeBean class by selecting Add -> Method from its pop-up menu. The new method should look something like:

```
public void ejbCreate(EmployeeKey key, String firstName,
        Character midInit, String lastName, short edLevel,
        java.math.BigDecimal salary, java.math.BigDecimal bonus)
        throws java.rmi.RemoteException, javax.ejb.CreateException
```

depending on how many columns from the EMPLOYEE database were included in the EJB fields definitions. The real trick is that each of the container-managed persistence EJB fields must be initialized in the create method. For example:

```
primaryKey = key.primaryKey ;
setFirstName(firstName) ;
setMidInit(midInit) ;
setLastName(lastName) ;
setEdLevel(edLevel) ;
setSalary(salary) ;
setBonus(bonus) ;
```

can be used in the new ejbCreate method above. After the new ejbCreate method has been defined, it must be promoted to the home interface. To do this, right click the new ejbCreate method in the Methods pane of the EJB Development Environment and then select Add to -> Home Interface.

Next, add one or more finder methods to your Employee EJB. These methods perform SQL queries and return either single Employee objects or java.util.Enumeration objects, which contain multiple EmployeeBean objects. The current EJB specification from Sun Microsystems does not provide for any other type of Java collection object in finder methods.

To add a finder method

1. Select the EmployeeHome interface and choose Add -> Method from its pop-up menu.
2. Define a finder method such as:

```
public Enumeration findByEdLevel (short edLevel)
    throws java.rmi.RemoteException, javax.ejb.FinderException
```

3. Select Employee in the EJBs pane, and select Generate -> Deployed Code from its pop-up menu.
4. In the Types pane, you should see many new classes (you can hide these classes from view by clicking on the icon in the upper-right corner of the Types pane that resembles a piece of paper) – select the EJSJDBCPersisterEmployeeBean.
5. Use the final static findByKeySQLString string of this class as a model to complete the next step. (Source code appears in the Source pane when the EJSJDBCPersisterBean is selected.)
6. In the Types pane, select the EmployeeBeanFinderHelper interface and add a statement similar to the following in the Source pane:

```
public static final String findByEdLevelQueryString = "SELECT
    T1.EMPNO, T1.FIRSTNME, T1.MIDINIT, T1.LASTNAME, T1.EDLEVEL,
    T1.SALARY, T1.BONUS FROM DB2ADMIN.EMPLOYEE  T1
    WHERE T1.EDLEVEL =  ?" ;
```

The order in which the column names appear in the statement above is critical; they must be specified in exact order as originally created by the VisualAge for Java tools. Use the findByKeySQLString referenced above to ensure correct ordering.

7. Save all changes.

Next focus on the EJB's business logic. To add new business methods, select the EmployeeBean class and then choose Add -> Method from its pop-up menu. For this exercise, add:

```
public double getTotalCompensation( ) throws java.rmi.RemoteException {
        return (salary + bonus) ;
}
```

After defining the new method, promote it to the bean's remote interface by selecting Add to -> Remote Interface from the new method's pop-up menu.

Setting descriptor properties and generating deployed classes

Now that the EJB is defined, attention must be turned to one additional task. An EJB provider must include a deployment descriptor for each EJB to describe the quality of services a bean requires from its container. A deployment descriptor is a serialized instance of a javax.ejb.deployment.EntityDescriptor or javax.ejb.deployment.SessionDescriptor object. In a nutshell, a deployment descriptor describes the basic level of services, such as environment, transaction, isolation and security functions, that an EJB requires from a container. There is no need to change any of the deployment descriptors in this exercise. The default properties associated with the Employee EJB can be viewed by right clicking the Employee EJB in the EJBs pane and then selecting the Properties menu item in the pop-up menu. The Properties menu is shown in *Figure 4*.

After fully defining the EJB and viewing the deployment descriptor properties, deployable code must be generated. To do this, right click the Employee bean in the EJBs pane and select Generate -> Deployed Code. The EJB utility Generate -> Deployed Code generates types in the form EJSxxx, where xxx is the type name – these types are specific to the WebSphere Enterprise Java Server (EJS) container.

Testing EJBs within the VisualAge for Java EJB Development Environment

Before exporting an EJB and deploying it into a container and EJS outside the VisualAge for Java environment, VisualAge for Java provides a self-contained Web application server (WAS) within its environment for testing and debugging. First ensure that the DB2 database servers are started (net start db2 and net start db2das00). Then return to the VisualAge for Java EJB development environment and right click on the IBM_Sample_DB_EJBs group from the EJBs pane. Select Add to -> Server Configuration from the pop-up menu to launch the EJB Server Configuration utility.

To start the servers, first view and modify any communication and database properties that may need changing (there is a Properties menu item in each server's pop-up menu). Start each of the servers in the following order, waiting for a "server is listening" message before starting the next server:

1. Location Service Daemon
2. Persistent Name Server
3. EJB Server (Server1)

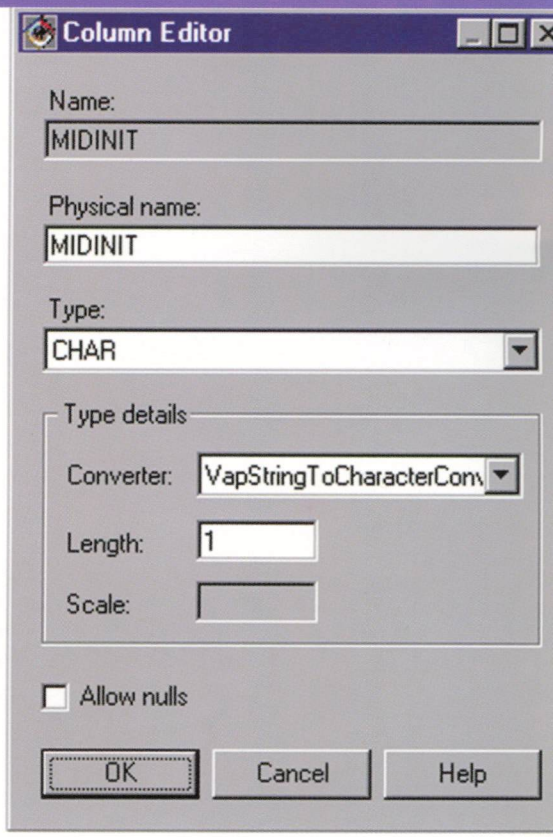


Figure 2. Associating MIDINIT with the converter

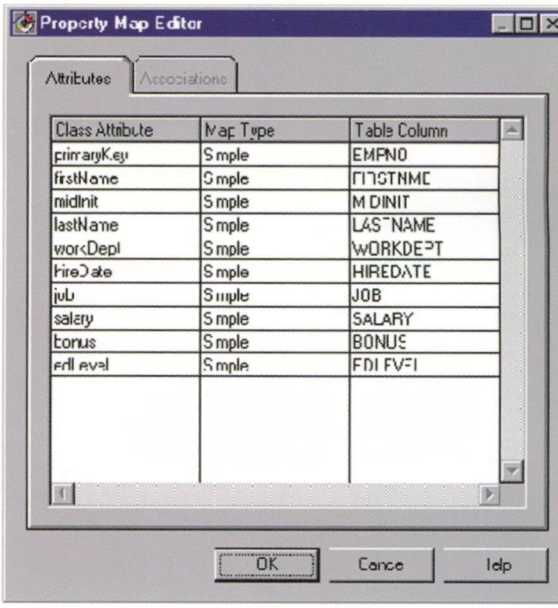


Figure 3. Associating fields with columns

When all three servers have successfully started and are listening, go to the EJB Server Configuration window and expand the EJB group on the right to expose the Employee EJB. Right click the Employee EJB and select Generate -> Test Client from the pop-up menu. Then, run the test client by selecting Run Test Client, which also is found in the EJB's pop-up menu. When the Employee test client appears, establish a connection to the EJS name server.

CONTINUED ON PAGE 14

CONTINUED FROM PAGE 13

```
package com.ibm.sdm.demos;
import java.rmi.*;
import java.util.*;
import javax.naming.*;
import javax.ejb.*;
import com.ibm.sdm.sampledb.* ;

public class EmployeeClient {
public static void main(String args[])
  {
    String primaryKey = "000340" ;
    try {
      Properties p = new Properties();
      p.put(Context.INITIAL_CONTEXT_FACTORY,
          "com.ibm.jndi.CosNaming.CNInitialContextFactory");
      p.put("java.naming.provider.url",
                          "iiop://ebafsrv.austin.ibm.com:9019");
      InitialContext ic = new InitialContext(p);
      System.out.println("initial context created") ;

      Object tmpObj = ic.lookup("Employee");  //: JNDI name
      EmployeeHome employeeHome =
        EmployeeHomeHelper.narrow((org.omg.CORBA.Object)tmpObj);
      Employee employee = null;
      employee = employeeHome.findByPrimaryKey
                                  (new EmployeeKey(primaryKey));
      System.out.println("Employee found") ;
      } catch(Exception exception) { exception.printStackTrace() ;
    }
  }
}
```

Listing 1. Testing the bean

When the connection is established, you are ready to test your new create, finder and business methods.

Figure 5 reflects a few methods and fields that were not added during the exercise above.

Deploying EJBs in WebSphere Application Server 2.0 Advanced

When the new EJB has been fully tested in the VisualAge for Java environment, the next step is to deploy the bean in a fully functional EJS environment such as WAS 2.0 Advanced. VisualAge for Java gives you a few choices for exporting the EJB code, and two of those are of interest to developers. One choice for exporting is to generate an EJB jar file for immediate deployment into a container provided by WAS and running in a WAS/Universal DB2/LDAP environment. The other option is to generate a generic EJB jar file that can be deployed in any other EJB container running in a Web server's EJS. In the second scenario, the other containers have to provide a set of tools for viewing or modifying the EJB's deployment description and for mapping the EJB's container-managed fields to an underlying datastore. For simplicity, let's export an EJB jar file specifically tailored for the WAS environment. To do this, right click the Employee bean in the EJBs pane of the EJB development environment in the Workbench and select Export -> EJS Jar from the pop-up menu. In the Export to an EJS jar file SmartGuide that appears, specify the location and name of the new jar file as: x/WebSphere/AppServer/DeployableEJBs/Employee.jar where X is the drive on which IBM WebSphere Application Server 2.0 Advanced has been installed.

Next, from within a Web browser, link to the WebSphere Application Server Administration pages found at http://localhost:9527. Log in and go to the Enterprise Java Services page as seen in ***Figure 6***.

From this page, link to three other pages of settings that need to be modified to deploy the new Employee EJB in WAS:

EJS global settings

- Enable EJS radio button is Yes.
- EJB Server Name is Server1.
- Protocol is CORBA.
- Host Name is the machine's host name.
- Use the default on the others.

EJS containers

- Select the defaultEntityContainer and add the information needed to make a JDBC connection to the SAMPLE database.
- Delete any empty containers (such as the defaultSessionContainer) as JNDI queries to LDAP tend to fail in WAS when empty containers are found.

EJB jar files

- Deploy the Employee.jar file into the defaultEntityContainer
- The developer is prompted to regenerate files (re-create all the existing WebSphere-specific files) or to redeploy existing files (use what has already been generated and simply copy files and resources to the appropriate subdirectory). In this example, choose to redeploy existing.
- If successful, stop and re-start the WebSphere Application Server. To do this, go to Start -> Settings -> Control Panel -> Services and

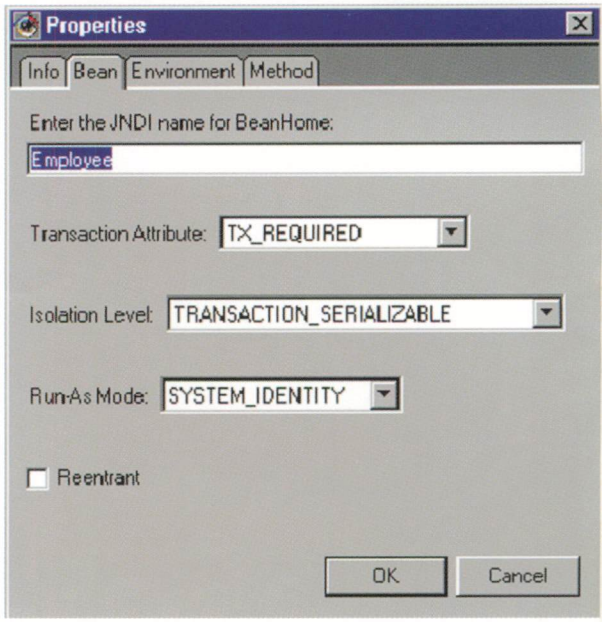


Figure 4. Viewing the beans properties

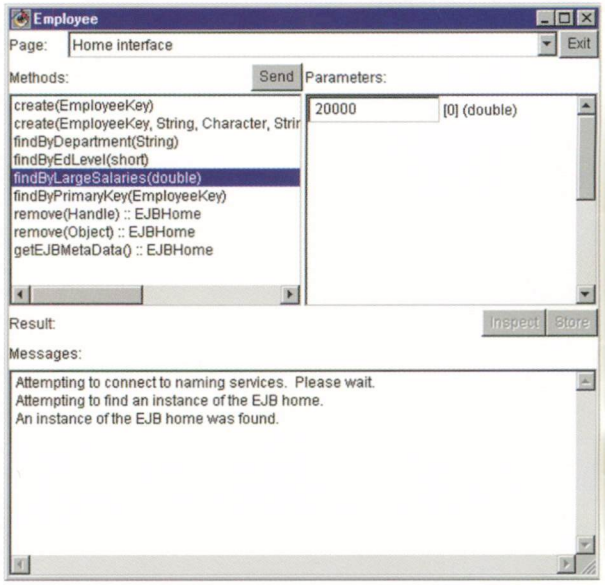


Figure 5. Adding additional methods and fields

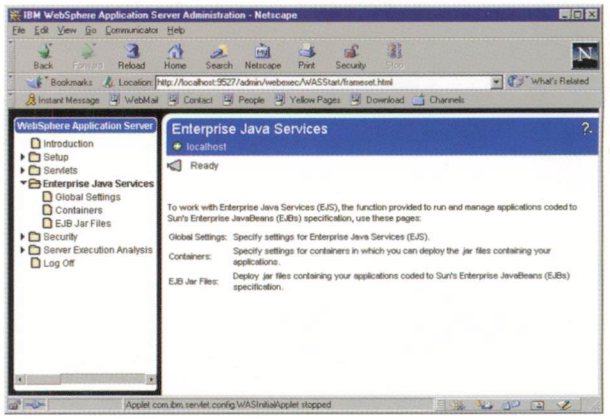


Figure 6. Enterprise Java Services page

locate the HTTP server and WebSphere Servlet services.

To test the bean, run a minimal Java application as shown in ***Listing 1***. Before compiling and trying to execute the test client, ensure that X:/WebSphere/AppServer/lib/EJS.jar and X:/WebSphere/AppServer/DeployableEJBs/Employee.jar are in the classpath.
To compile the sample, use javac -d .EmployeeClient.java and to invoke the application, use java com.ibm.sdm.demos.EmployeeClient. The application should display the "Employee found" message.

Deploying EJBs in Component Broker 2.0 Beta

One of the most important reasons for building object-oriented applications with EJBs is the promise of Write Once, Run Anywhere. While the Sun EJB 1.0 specification is still vague about API contracts between container providers and EJS providers, it is possible to build EJBs that can be deployed into any containers that conform to the minimal specifications and that provide the required level and quality of services. As an example, IBM currently has a second EJS, Component Broker 2.0, in a limited beta. Component Broker 2.0 supports two programming models including its own Managed Object Framework (C++ and Java) and the EJB programming model. Component Broker ships with a comprehensive set of solutions for building and managing distributed objects:

Component Broker Connector

- Extends enterprise software capabilities
- Enables transparent client (Java, ActiveX,* and C++) access to multiple servers
- Manages throughput and response time
- Monitors availability and reroutes workflow
- Integrates new applications with existing applications and middleware
- Manages for efficiency, control, and availability

Component Broker Toolkit

- Absorbs complexity in development of distributed, object-oriented applications
- Automates design and code generation using a well-defined programming model
- Uses visual building processes and integrates with popular object-oriented analysis and design tools
- Includes distributed, object-aware debugger

When Component Broker 2.0 becomes generally available in the future, it should provide a robust set of services and tools to manage enterprise-wide, mission-critical distributed applications. Component Broker 2.0 Beta provides a suite of tools to wrap existing EJBs as Component Broker managed objects and to deploy them into the Component Broker runtime. These tools may include:

- ***jetace:*** Used to assist in creating and/or editing of an EJBs environment properties and deployment descriptor
- ***cbejb:*** Used to generate and build the C++ and Java code necessary to run the EJB in a Component Broker environment
- ***Object Builder utilities:*** Contains schema and table mapping utilities for binding to a datastore
- ***PAOtoEJB:*** Used to associate CICS* or IMS PAA services with an EJB server
- ***ejbbind:*** Used to bind an EJB's JNDI name to a Component Broker namespace

However, as mentioned earlier, Component Broker 2.0 is still in beta and there are a number of restrictions on deploying EJBs into a Component Broker environment today. As an example, the Employee EJB with container-managed persistence fields created above would not convert to the Component Broker managed object framework without a number of significant changes due to the following current Component Broker restrictions:

- The EJBMetaData interface is not implemented
- The following methods are not supported by Component Broker:
  - EJBObject.getPrimaryKey,
  - EJBHome.remove(Object) and
  - EJBHome.getEJBMeta Data (these methods require object-by-value support)
- Finder methods that return collections of objects are not supported
- The primary key class of an entity bean must be a primitive type or java.lang.String.

Of course, developers who want to deploy EJBs into a Component Broker run-time could develop EJBs with these limitations in mind for now. As Component Broker matures, many of these restrictions could be lifted.

Summary

So what is in the future for EJB integration with IBM software? While nobody can predict the future, a number of IBM software brands have stated directions that EJBs are a central component of their strategy.

IBM recognized the importance and power of EJBs in creating common, unifying programming across its software application programming interfaces. We can expect to see other IBM and non-IBM products EJB-enabled in the future. To track and keep up with the latest IBM software product plans, visit www. software.ibm.com and www.developer.ibm.com frequently.

Mickey Nix *is a Senior Programmer with the IBM Solution Developer Marketing organization in Austin, Texas. In his role as a network computing consultant, Mickey assists IBM's top independent software vendors in their efforts to migrate their mission critical applications to Web-enabled platforms using IBM's middleware and servers. For the last year, Mickey also has taken on the additional responsibility of IBM SanFrancisco Frameworks consultant.*

Editor's Note: The sample code included in this article can be downloaded from the Developer Connection Web site at www.developer.ibm.com/devcon/. The code also is available on the July CDs.

Deploying VisualAge for Java *Enterprise Beans to the* WebSphere Application Server

by Howard Borenstein

The IBM VisualAge for Java product provides a powerful and productive environment in which to develop enterprise beans. It includes tools to help create and organize enterprise beans, and it features an IBM WebSphere Application Server environment that allows developers to completely unit test the code. Once the beans have been created and tested in VisualAge for Java, they can be deployed to an EJB-compliant server.

This article shows how to take enterprise beans that are developed using VisualAge for Java, Version 2.0 with the Enterprise Update, and deploy them to the WebSphere 2.0 Advanced Application Server. The article assumes that WebSphere is installed on the Windows NT platform, but the information also applies to WebSphere installations on the AIX* and Solaris platforms, except for the starting and stopping of the WebSphere Servlet Service.

Exporting your beans

The first deployment step is to export the beans into a JAR file. There are two types of JAR files that developers can create: an EJB JAR file, and a deployed JAR file. An EJB JAR file contains the bean code (bean, home interface and remote interface), but it does not contain the deployed classes required to run the bean on an EJB server. When deploying an EJB JAR file to the WebSphere Application Server, it generates the necessary deployed classes.

The recommended way to export beans is to create a deployed JAR file. This JAR file contains the bean code and the deployed classes. When deploying this JAR file to the WebSphere Application Server, it recognizes that the deployed classes already exist and it does not have to regenerate them. If any entity beans are mapped with the VisualAge for Java Persistence Builder, then a deployed JAR file must be used to prevent WebSphere from re-creating the deployed classes.

On the EJB page in VisualAge for Java, select an EJB group or one or more enterprise beans, then select the Export menu on the item's pop-up menu. Then, choose whether to export the beans in an EJB JAR file or in a deployed JAR file, by selecting EJS JAR. The Export SmartGuide is shown in ***Figure 1***. The developer may export entity beans and session beans in the same JAR file; however the JAR file would need to be deployed into both an entity container and a session container.

Setting up the WebSphere Application Server

There are three files you must configure before using enterprise beans in the WebSphere server. The appropriate settings may be accessed through the WebSphere Administration UI or by editing the files. Direct editing of the files is the most efficient means of configuring the files.

- **properties\bootstrap.properties:**
  This file contains the setting java.classpath for setting the classpath of the server. If the enterprise beans use any support classes or call any Java code outside the bean, then you must add the location of the code (or the JAR file containing the code) to the java.classpath setting. The location of the IBM DB2 driver db2java.zip also must be specified on the classpath.
- **properties\server\servlet\debug.properties**
  This file enables the tracing capability. This is invaluable when first getting started with the WebSphere server. Trace output can be directed to either the WebSphere console or to a file. Tracing to a file is probably the most convenient way of directing output.

For enterprise beans, tracing can be enabled for the EJS trace group by specifying the following parameters:

```
# EJS Trace Group
trace.group.ejs.tracers=EJS_stdout EJS_stderr LSD_stdout
                        LSD_stderr PNS_stdout PNS_stderr
trace.group.ejs.state=on
trace.group.ejs.handlers=LogFile
```

When tracing is enabled, the output files are created in the logs directory. The files needed are websphere_trace.log and jvm_stderr.log. All of the tracing of the server is placed in websphere_trace.log. This is similar to the output seen in the Console window, shown in ***Figure 2***, when using the VisualAge for Java EJB server. This file is appended to each time the server is started, so delete the file periodically. The jvm_stderr.log file contains the output generated when deploying a JAR file into a container. This file is re-created each time the server is started, so ensure that the contents are checked after deploying a JAR file, but before restarting the server.

- **properties\ejs\ejs.properties**
  This file contains all of the configuration information for the EJB server. There is a line "containers=" that lists all of the containers in the server, and there also is a section for each container. Some things you need to configure include:
  - ***ejs.trace.enabled*** – Set this to true for more detailed tracing. It may not be needed, but it is helpful if you are having trouble getting things working.
  - ***nameservice.hostname*** – The default value is localhost. To access beans from a client on a remote machine, change this

The IBM VisualAge for Java product provides a powerful and productive environment in which to develop enterprise beans.

value to the name of the server machine.

Using the following parameters, specify the name of the database to use for any entity containers, as well as the username and password used to access the database:

```
defaultEntityContainer.dbUrl=jdbc:db2:sample
defaultEntityContainer.dbUser=howard
defaultEntityContainer.dbPassword=abc
```

If any containers do not have JAR files deployed, remove them from the containers= line. If there is an empty container, exceptions may be thrown when the EJB server is started. By default, the defaultSessionContainer is empty, so it should be removed for now.

Deploying your beans

Once you are finished setting up the WebSphere server, deploy the JAR files into the containers using the Administration UI. Copy the JAR files created in VisualAge for Java into the deployableEJBs directory. Make sure that any classes the bean uses are specified on the classpath as discussed above, then start the Web server. For example, if the Web server is Lotus Domino Go, then start the Lotus Domino Go Webserver service from the Windows NT Services panel.

Load http://machinename:9527 in your browser, then navigate to the Enterprise Java Services - EJB Jar Files page shown in ***Figure 3***. Select the JAR file from the list of JARs. In the beans panel, you should see a list of beans. If the error message "java.lang.NoClassDefFoundError" appears, then the server's classpath is missing some classes necessary for one of the .jar files in the deployable EJBs directory. The server will need to be restarted if any changes are made to the classpath.

If the JAR file contains entity beans, select an entity container and then select Deploy. If deploying a deployed JAR file, a dialog box should open. Select Redeploy existing. If deploying an EJB JAR file, it takes some time for the deployed code to be generated. In both cases, the last step of the deployment process is to create the database tables. A message box will indicate the success or failure of the deployment process.

At this point browse the jvm_stderr.log file for any errors reported during deployment or the creation of the database tables. If the tables already existed, you will see the following exception:

```
COM.ibm.db2.jdbc.app.DB2Exception: [IBM][CLI Driver][DB2/NT]
SQL0601N The name of the object to be created is identical to the
existing name "EJB.MYBEAN" of type "TABLE".
SQLSTATE=42710
```

This exception can be ignored. Check the ejs.properties file and ensure that the .jar file was added to the following .jarFiles line:

```
defaultEntityContainer.jarFiles=Inc.jar,mybeans.jar
```

The JAR file also should have been copied to the deployedEJBs directory.

If using a deployed JAR and the database tables already exist, you do not need to use the UI. Just manually add the JAR file to the .jarFiles entry. Similarly, deploy any JAR files with session beans to a session container. If the JAR was deployed to a container that was removed from the containers line because the container was empty, be sure to add it back before starting the server.

Starting the server

After deploying a JAR file into a container, stop and restart the server in order to use the enterprise beans. Stop the Web server and then stop the WebSphere Servlet Service from the Services panel, then restart the Web server. To ensure that everything is operating properly, check the websphere_trace.log file. If the ejs.trace.enabled property in ejs.properties did not change, then the last two lines in the file are:

```
[EJS_stdout,1] server          E Server is listening...
[EJS_stdout,1] Nameservice     D Local namespace:
```

If ejs.trace.enabled was changed to true, the last message should be a bindings message listing the beans in the container. This is similar to what is seen in the VisualAge for Java console when running the EJB server.

If Java exceptions are seen in the log, try to diagnose the cause and fix them. If a "No suitable driver" message is shown, the cause could be one of the following:

- The db2java.zip file is not on the classpath.
- A wrong username or password is specified in the ejs.properties file.
- Change the WebSphere Servlet Service to run as a user with database access. To change the configuration of the WebSphere Servlet Service, open the Windows NT Services window from the Control Panel. Click Startup and change the *Log in as value* to *This account*, then add the username and password used to access the database.

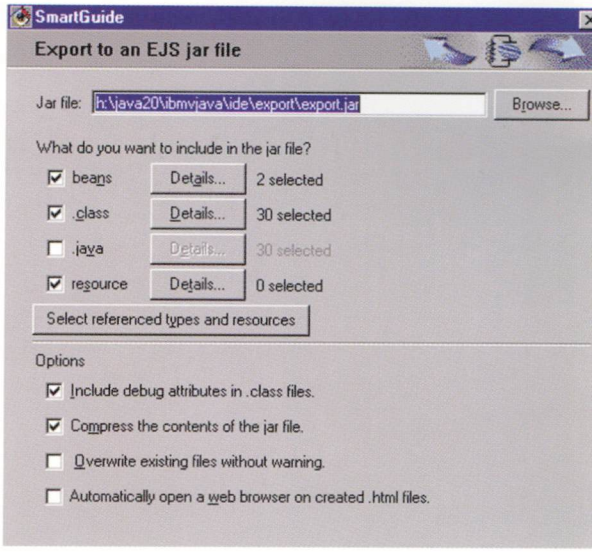


Figure 1. Export SmartGuide

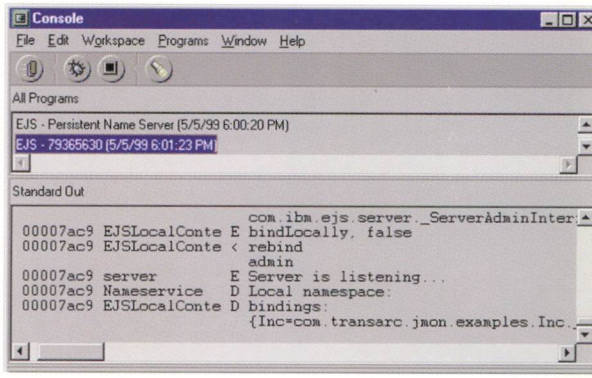


Figure 2. Console Window output

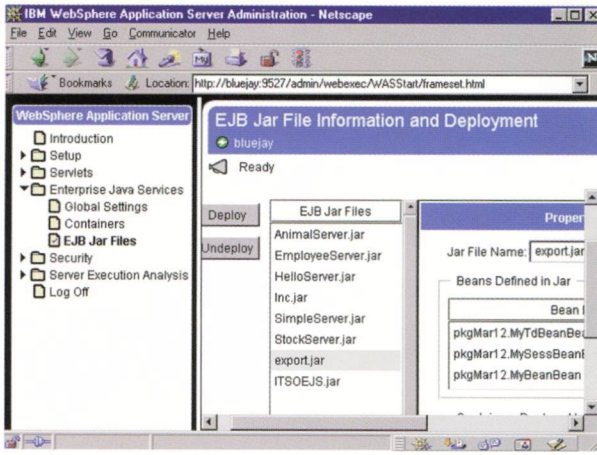


Figure 3. Enterprise Java Services — EJB Jar Files page

Testing your beans

Use the generated Test Client in VisualAge for Java or your own test client to test the beans as shown in ***Figure 4***. The default port of the Persistent Name Server in WebSphere is 9019, so change *iiop:///* to *iiop://machinename:9019* in

CONTINUED ON PAGE 18

CONTINUED FROM PAGE 17

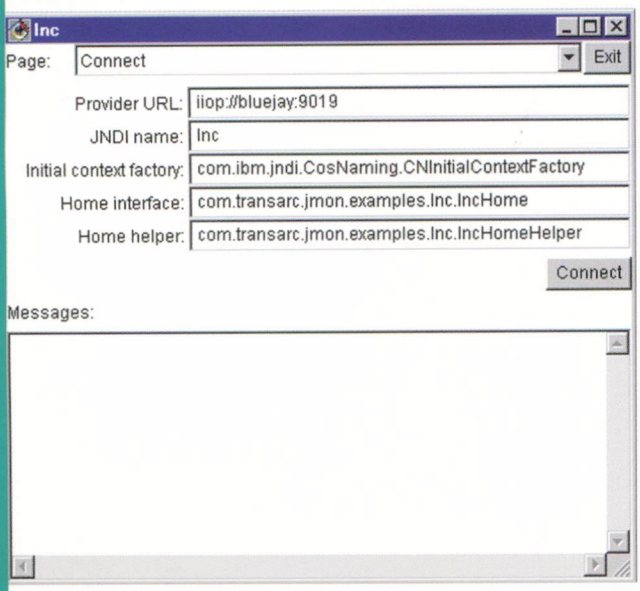


Figure 4. Test client generated

the VisualAge Test Client. If the enterprise beans are not working as expected, browse the websphere_trace.log file for any run-time exceptions.

Conclusion

To successfully deploy enterprise beans to the WebSphere Application Server, there are many configuration steps the developer must take to enable an EJB environment, as well as several files that need to be checked to see if the deployment and EJB server startup was successful. However, because VisualAge for Java contains the same WebSphere runtime as the WebSphere Application Server, developers can feel confident that the run-time behavior experienced is the same in the WebSphere environment as it was in the VisualAge for Java environment.

Howard Borenstein *is a Software Developer at the Toronto Lab, IBM Canada Ltd. He has been developing application development tools for eight years, and he currently works on the WebSphere tools in the VisualAge for Java product. He can be reached at borenst@ca.ibm.com.*

Implementing Enterprise JavaBeans in AIX 4.3.2

The Enterprise JavaBean (EJB) specification defines a model for component-based distributed computing. It defines a clear separation of business logic and persistent, transactional support and other middleware-related services. Solution developers can concentrate all their efforts on developing the business logic and leave the middleware-related tasks to the EJB container in which the EJBs are deployed.

An EJB can be transient (SessionBean), which means it is created by an application and lasts only as long as the application is running; or persistent (EntityBean), which means it is written out to permanent storage (usually a Relational Database) and can be accessed in the future.

Application servers are a major part of the framework for developing e-business applications. To be effective with Enterprise JavaBeans, an application server should incorporate a multitude of functions. These include database access, transaction support, component functionality and, of course, Web functionality and capabilities.

The technology of Enterprise JavaBeans is not limited to the AIX platform. This technology defines a model that supports the "Write Once, Run Anywhere•" Support for the EJB model, as the means by which application services and application server integration are provided on the server tier. This means that Java is the language, and JavaBeans are the component model.

However, the new RS/6000* servers are the perfect solution to take advantage of the Enterprise JavaBean model by providing the server tier, which consists of application services, and one or more application servers because of their industry-leading decision support performance.

The IBM suites for e-business provide a Web application server. The WebSphere Application Server is available now for developers on the AIX platform. It allows developers to take advantage of the Enterprise JavaBean portability. WebSphere Application Server Advanced Edition and the Enterprise Edition include support for Enterprise JavaBeans. The Enterprise Edition includes relational database connectivity and support for Java Transaction Services. Enterprise JavaBeans for connecting to MQSeries, CICS and other transactional environments are expected future enhancements.

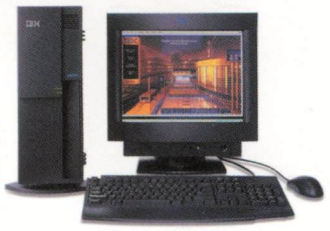
VisualAge for Java Version 2.1 Enterprise Edition provides the capability to develop Enterprise JavaBeans as well as other enterprise applications that can be deployed on the WebSphere Application Server. This support is part of an installable feature called IBM Enterprise Java Server (EJS) which contains:

- An implementation of the Java Web Server and Java Server Toolkit that provides Java Server Pages (JSP) and Java servlet support.
- An implementation of an Enterprise Java Server (EJS) and a set of EJB containers that allow EJBs to access relational databases.
- Support services, such as JNDI, CORBA CosNaming services, and administration and deployment tools to operate the server and manage EJBs, servlets, and JSPs.

The VisualAge Toolkit provides a powerful set of tools to create e-business applications that allow developers to rapidly develop distributed applications that run on a wide variety of client and server platforms.

Take Advantage - take your business to e-business.

Business Intelligence with the AS/400



by Mark Wulf

Business Intelligence may seem like an oxymoron. How many businesses do you know of that actually take advantage of the "intelligence" that they have available? Many of our customers don't understand the value of the intelligence they have in their businesses.

Business Intelligence (BI) is simply using information that a company has, to make better business decisions. As suppliers of technology and solutions to customers, IBM can play a major role in helping customer businesses become intelligent and creating more competitive offerings ourselves.

The AS/400, as well as all of the IBM servers, can play a key role in the addition of BI to your applications.

The AS/400 and its business partners have long been known for delivering solutions to its customers. The BI market is no exception to this strategy. IBM is actively working with its business partners to add BI as part of the solution they can offer their customers by pairing BI technology providers with vertical industry application providers.

The BI technology providers sell things like OLAP and data warehouse management tools or query and report writers that are the basis for many BI solutions. The vertical industry application providers have knowledge about what information their customers need to run their businesses. Working together, these two providers can create a BI solution that requires a very small investment to develop while creating great market advantage.

Many application providers are taking advantage of these new partnerships to sell more products to their installed base of customers while being able to compete more effectively for new customers.

Many of the AS/400 BI technology providers are using this strategy to provide more solutions and reach many more customers then they could by themselves.

Many AS/400 BI technology providers have been hard at work integrating their products with IBM Visual Warehouse.* This integration will allow these products to exchange meta data with other products. Definitions created in one product will be available in other products and there is no need to duplicate work or identify pieces of information multiple times. Using multiple products in a single implementation becomes much easier when the meta data can be defined in one product and used by the rest of the products.

New technology

Two new technology providers have recently made the decision to make their products available on the AS/400. The first is SPSS, Inc. SPSS is known for advanced data analysis products using statistical analysis and data mining technology. Their products are a great complement to the capabilities of IBM Intelligent Miner and supplement the ability of AS/400 to provide a wide spectrum of data analysis tools.

The second company is Viador. Viador provides products to create Web-based information portals. These portals can provide information to hundreds if not thousands of users through a browser interface. Each user experience is customizable and information can be provided through query tools or report writers that come with the product or come from almost any other source of business intelligence information. These portals are the latest technology in providing information to many users who might not have the skills or the desire to learn one of the traditional BI tools.

The AS/400 team has recently announced many new capabilities that make it an even better system to host BI or traditional applications. The most important announcement was the addition of Universal Database (UDB) enhancements to DB2 for AS/400. (See DB2 UDB on page 6.) With these enhancements comes an AS/400 database name change – from DB2 for AS/400 to DB2 Universal Database for AS/400.

A new feature: EVIs

One of the major new features that was shipped in V4R3 and is undergoing additional usage in V4R4 is Encoded Vector Indexes (EVIs). EVIs are a new type of index developed and patented by the IBM Thomas J. Watson Research Center. They are used much like traditional indexes are used today (to provide fast ways to find specific rows in a database table), but have efficiencies that traditional indexes do not.

SOLUTIONS '99 PRESENTER

These efficiencies come in part from the fact that EVIs are up to 16 times smaller than a traditional index. This means less I/O is required to get the EVI into memory and more EVIs will fit into memory. The size of the EVIs make it faster to scan for key values or to create other data structures used to resolve a query.

These data structures, which are actually bit maps, can then be combined together very simply to solve more complex queries. The effect is that multiple EVIs can be used in conjunction to solve a single query. The ability to use multiple EVIs reduces the database administrator's need to create the perfect single traditional index for each query.

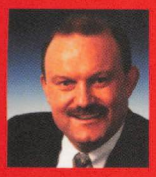
These new indexes are having major impact on the performance of customer implementations. Many customers are reporting up to 30 times better performance on their longest running queries.

Summary

Whether the AS/400 is delivering solutions, new technologies from our partners or delivering database functions that can have enormous impact on the performance of a BI implementation, the AS/400 is ready to be part of adding intelligence to any business.

Mark Wulf *is the Business Intelligence Segment Manager for AS/400 Partners in Development. His group is responsible for the recruiting and technical support of AS/400 Business Intelligence Solution providers. Mark also manages the AS/400 Teraplex Center. The AS/400 Teraplex Center was started to help customers and business partners implement multiterabyte data warehouses on the AS/400. Mark has been at IBM 10 years and worked in DB2/400 development prior to joining the Partners in Development organization. Mark's roles in DB2/400 development included lead developer for Distributed Data Management, Distributed Relational Database, Commitment Control and ODBC. His last role in development was as Chief Data Warehouse Architect for the AS/400.*

Java on OS/390 delivers real portability



by Mark Cathcart

Java delivers cross-platform portability like nothing before it! For S/390, both OS/390 and VM/ESA provide implementations of the Java Virtual Machine (JVM) that are compatible with the 100% Pure Java initiative. This article looks at portability and Java, gives an overview of the JVM, and outlines the plans and strategy for Java on OS/390.

Portability concerns

Just a few years ago, the computer industry model of portability came in two flavors, both of which had problems:

- Source code portability in UNIX® systems
- Reuse of business objects

Source code portability depended on the availability of much more than just the UNIX application programming interfaces (APIs). It depended on vendor products, run-time libraries and an identical implementation of tools and utilities that were not part of the core UNIX standard.

Reusable business objects usually were implemented through program language source code files. Thus, porting objects had the same problems as porting UNIX applications. True, for some object or component technology, portability depended on a definition. But the portability of the definition was restricted by the availability of tools to understand the definition. While part of the IT industry was trying to work around these problems, others were looking for a technology that would allow the distribution of executable, component-based technology over the Internet. In order for this to be a commercial success outside of the academic world, the technology needed to allow programmers to:

- Build/model business and technology in components
- Package those components as executable code
- Make the code available via networks for use on many different computer systems

For a short time, many people thought that the only solution was Windows everywhere. It became apparent, however, that Windows wasn't the answer either. The functional differences between "palmtop" and true server computing simply were too great.

The ability to package and execute a common, industry-standard program format has made the portability of Java programs an industry-leading technology.

Java to the rescue

Java provides a completely different approach to portability. It does not need or require language source code portability. Instead, Java provides a common execution environment – the Java Virtual Machine – across a number of different environments, processors and microprocessors and operating systems. The Java Virtual Machine is, in concept, a microprocessor implemented in software.

The definition of Java architecture is managed by Sun Microsystems, with input from others in the Java arena, one of these being IBM.

It is interesting to note that Java also provides support for component or object-based technology. This means that Java addresses the primary portability problems. The JVM runs a standard form of executable program, commonly called byte-code. This byte-code contains a binary class structure that the JVM uses to create and manage objects modeled in software. This combination of JVM and binary class byte-codes is packaged with an extensive set of defined underlying classes. The result is that Java programs can be moved at the binary, byte-code level, knowing that a 100% Pure Java environment will provide common run-time support.

Through support of reusable business objects or components known as Java Beans, Java also encourages the design and programming of applications in small functional classes, which implement only the business or technology process logic. These can be packaged and reused wherever it makes sense, irrespective of platform or technology. Thus, the same applications can run on workstations, departmental servers, midrange servers or enterprise servers.

Java basics

The ability to package and execute a common, industry-standard program format has made the portability of Java programs an industry-leading technology.

Figure 1 shows the Java Virtual Machine and its relationship to the operating system and hardware where you run Java. The diagram illustrates the process from source code to execution. As shown, the first step is to compile

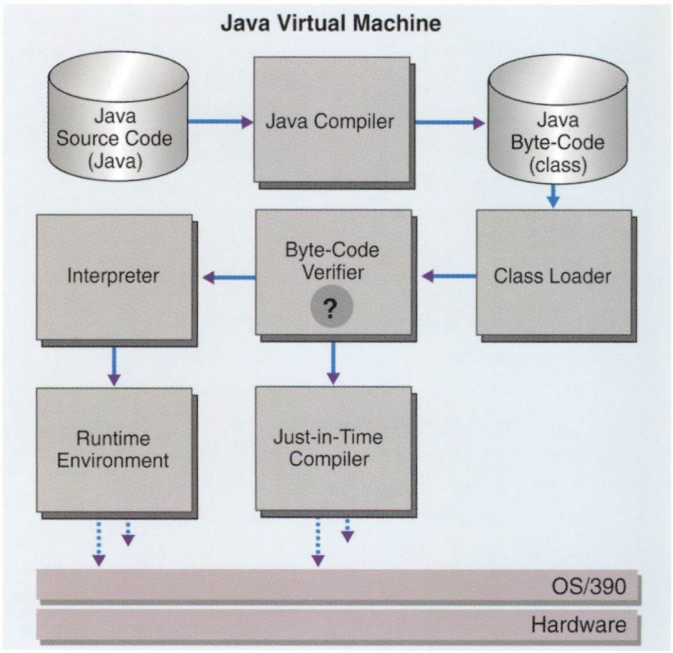


Figure 1. Relationship of Java Virtual Machine to the operating system and hardware

CONTINUED ON PAGE 22

CONTINUED FROM PAGE 21

the source code, filename.java. This compilation takes place inside the JVM, because the Java compiler is a Java program that has been compiled into byte-code.

The output from the compiler is a filename.class file. In order to run the .class file, the JVM is started and given the name of the class for the .class file, not the filename of the .class file. The initial class is loaded by the Class Loader.

As it processes a stream of byte-codes, it sends them to the Byte-Code Verifier.

The Byte-Code Verifier makes a number of checks on the structure and validity of the byte-code and passes it to an Interpreter or to a Just-In-Time (JIT) Compiler.

The Interpreter or JIT Compiler then executes the instruction, which is directly processed on the hardware processor or trapped by the operating system and indirectly executed.

S/390 and Java

In an OS/390* or VM/ESA* system, the JVM is implemented using the available UNIX interfaces. Java source code is stored in the S/390* UNIX file system and the JVM uses the UNIX services. This means that OS/390 and VM/ESA customers can benefit from the portability of Java.

The remainder of the article concentrates on the OS/390 implementation of the JVM. On OS/390, well-structured and well-written Java programs benefit from the fact that OS/390 runs each thread within a Java class as a fully pre-emptive, multi-tasked OS/390 TCB (Task Control Block). This means that multithreaded Java programs – especially those that are launched and then execute for extended periods of time – can benefit from extensive workload management and multi-processor, multi-threading capabilities in OS/390.

Why run Java on OS/390?

A question I often hear is: "Why run Java on OS/390?" Currently, many people are experimenting with Java, and a number of large, well-known projects are being completely written in Java. Because many of these projects require integration and collaboration with existing OS/390-based applications and databases, that, in itself, might be a good enough reason. There may be a more compelling one, however; it's called payback.

As organizations drive more and more for competitive advantage, they need to be able to exploit the best, most current skills and tools in the computer industry. The past year has seen a massive swing towards Java.

A recent Gartner Group study showed that almost one-third of all students in computer degree courses chose to use Java as their primary programming language. In addition, many leading edge Integrated Development Environments (IDEs) for Java – such as IBM VisualAge for Java – have arrived. Together, they create an impressive opportunity that would be foolish to pass up.

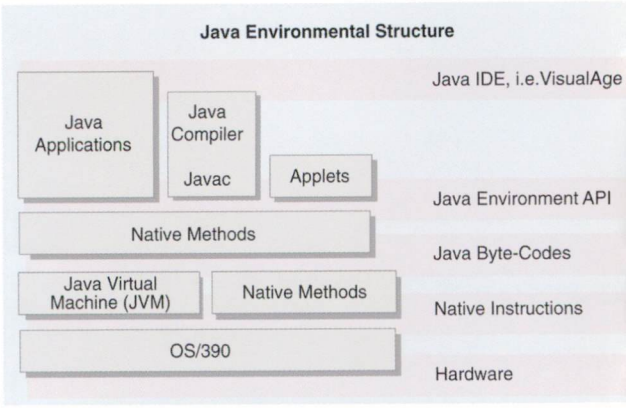


Figure 2. Relationship between Java components

Companies can put new staff straight to work using a programming language (and potentially an IDE) that they are completely at home with. Program development and local testing of "standard" Java byte-codes can be done entirely in a workstation environment and then deployed on the platform of choice. For example, byte-codes can be uploaded, tested and put into production on OS/390 without recompilation or changing a single line of code. From a business perspective, this is an almost irresistible combination.

There's no reason to wait

Figure 2 shows the relationship between the various components in Java and at what level each technology exists. Tools such as VisualAge for Java or OS/390 products themselves provide Java gateways or connectors that allow OS/390 Java programs to access back-end, "native" systems. Examples of such systems are CICS, IMS and DB2. The OS/390 edition of WebSphere Application Server also supports Java Servlets and JavaServer Pages (JSPs) with the Common Connector Framework to communicate with back-end systems.

In addition, offerings such as Host-On-Demand provide low technology solutions for interfacing to 3270 applications through an API. Java-based solutions also are available for ISPF, Tivoli Information Management, NetView and a number of S/390 services and subsystems. If you are not familiar with the Java programming language, you might be considering this as a career enhancing skill. Yet be aware that while it may be easier to program in Java than it is in C or C++, it could be even easier.

The good news is that you can use one of the existing skills you have – the ability to program in REXX! With some additional reading, you can take those programming skills and write in NetRexx*, which is compiled into Java byte-code. This gives you all the benefits of Java byte-code portability and system independence, using a language in which many OS/390 people have skills and which is much more readable and maintainable. In fact, after three years working in the object-oriented (OO) field, I've finally become hooked on OO programming through NetRexx.

VisualAge for Java Enterprise Edition, also is now available with a OS/390 Enterprise Toolkit. The toolkit provides a native code compiler that compiles Java byte-code into an OS/390 DLL that can be used without the JVM, a remote debugger, a performance profiler and JPORT, a porting utility. The Native code-compiler is a key tool as it allows Java to be used as a development language for environments where the Java Virtual Machine currently cannot be used. The first supported environment is CICS, where a developer now can write CICS transactions using the JCICS classes; next will be DB2 Stored Procedures and, at a later date, IMS* MPP programs.

Performance

I often am asked about the performance of Java programs. This concern usually stems from the notion that Java byte-code is "interpreted" rather than compiled in the traditional sense. But this really isn't true any more. Even if it were, a number of other changes have significantly changed the performance profile of Java. A complete makeover of the JDK in the 1.1.6 (December 1998 and April 1999 issues of the *Developer Connection Technical Magazine*) release has lead to significant improvements.

In general, the perception of Java performance stems from the amount of time it initially

takes to load the JVM. If you have a trivial program that only does something simple and then exits, the time spent loading the JVM sometimes exceeds the time to run the program.

For long-running, server-type applications, the performance of Java is broadly similar to COBOL. When comparing identical function, that might mean an additional 20 percent overhead, depending on the program. If using Java as a procedural scripting language, then performance has a different profile. It is less like COBOL and more like interpreted REXX, again depending on the program. In the future, additional planned changes should enhance and improve Java performance on OS/390.

What's next?

In April 1999, a new version of JDK 1.1.6 was released. This is known internally as 116p, and for the first time amongst other enhancements it uses the OS/390, industry-leading WorkLoad Manager(WLM) to manage, schedule and prioritize Java execution threads.

Next in plan is a 1.1.8 JDK, which is expected to achieve another 2X performance improvement and provide some significant function back-ported from Java 2. These include the Security, RMI over IIOP, Swing and other functionality.

An alpha implementation of Java 2 is planned for late 1999. We will be assessing the applicability to OS/390 of the Hotspot Compiler as it is developed by Sun. We continue to work on incremental performance improvements, while at the same time looking for major breakthroughs to make OS/390 the industry's leading Java business transaction server.

In addition to the base JDKs, we also are starting to ship specific OS/390 class-packages: JRIO, a set of record I/O classes that will provide access to Sequential, PDS and VSAM files on OS/390; RAWT, a set of Java classes that replace the standard AWT classes. They turn your Java AWT calls into a call from OS/390 to any Java client running the RAWT client classes, where the call is received and mapped into local/client AWT classes. In this way, you can write distributed AWT calls without having to worry about the network communication. RAWT does that. RAWT gives you the ability to put distributed management consoles or monitoring into your OS/390 Java applications; RACF* classes are the first step toward integrating Java security with native RACF security.

OS/390's strategy is to ship new and updated JDKs when appropriate. If there is major new Java function, then an OS/390 implementation will be available. If there is no major new function, then we will ship maintenance releases. Not every maintenance release produced will be shipped, but quarterly or twice yearly updates might be reasonable to expect.

We have seen the emergence of the Enterprise JavaBeans specification as the server infrastructure for Java components, providing enterprise-class facilities such as transaction, naming and directory services. The ability to utilize and exploit this type of technology through Java is paramount for an enterprise-class solution and will be a natural extension of the OS/390 JVM and CORBA support in Component Broker. In addition, Enterprise JavaBeans will be available on OS/390 through the CICS Transaction Server for lean, mean transaction beans, and also through vendor products such as Bluestones' Sapphire/Web. Other exploiters of Enterprise JavaBeans will be the IBM WebSphere Application Server and the Software AG Bolero Business Application Server.

Summary

The opportunity for your business to benefit from Java is substantial. Its component model encourages applications to be designed and built to represent real business and technology processes. And the Java Virtual Machine delivers on the ability to reuse these components throughout your organization, irrespective of platform. With the planned and future enhancements in Java performance, plus application infrastructure support for Enterprise JavaBeans, OS/390 will provide a scalable, reliable, secure foundation for your enterprise-class Java applications.

Additional information

To learn more about Java and OS/390, a detailed presentation on this topic, a short OS/390 programming workshop and sample programs are on my Web site. Our first Java redbook, *Integrating Java for OS/390 with Existing Applications and Data*, is available in printed format and from the Web as a PDF. The IBM publication number is SG24-5142. The second, *e-business Application Solutions on OS/390 Using Java Vol.1 SG24-5342*, is available from the Web now as a PDF. It is planned to be available in printed format in June 1999.

For more information, see:

- www.ibm.com/s390/corner for Cathcart's Corner for presentations, white papers and so on.
- www.ibm.com/s390/java for Official Java OS/390 web site.
- www.ibm.com/redbooks for redbooks on Java, VisualAge for Java and so on.
- www.developer.ibm.com/devcon/

Mark Cathcart *is Principal Consultant to IBM System/390 Division in Europe, Middle East and Africa for new software technology and software strategy. His current assignment is object-oriented and component-based software and software development including Java, Enterprise Java and Component Broker and XML. Other areas he is responsible for include e-business and the Internet, Client/Server and Open Systems.*

Mark is a "certified" Consultant IT Specialist and a member of the IBM S/390 Software Design Council. In 1999, he was elected to the position of Technical Staff Member in the IBM UK Technical Consultancy Group. Mark is also a Knight of the order of VM and joined IBM Systems Engineering in 1987 to become a 'Poacher turned Game-keeper' after spending 13 years as an IBM customer. This included four years with responsibility for the VM systems at a leading New York bank. Mark likes to play "devils advocate" at IBM and has a unique insider's perspective.

IBM SanFrancisco at Solutions '99

by Verlyn Johnson

IBM SanFrancisco is an industry leading initiative for developing object-oriented server-based applications using pre-built, pre-tested application business components written in Java. It also is key to IBM's e-business strategy, which helps businesses exploit the network economy by transforming their business processes with Internet technologies. It will allow application developers to combine capabilities from products such as IBM's Net.Commerce and MQSeries with capabilities from existing and newly developed component-based applications.

Currently in its third release, IBM SanFrancisco is attracting significant industry interest, with more than 1,100 Independent Software Vendors and customers with licenses to SanFrancisco. A number of SanFrancisco applications became available in 1998, while more than 100 Independent Software Vendors (ISVs) have projected delivery of new applications based on SanFrancisco during 1999.

ISVs success stories

Two sessions are planned for Solutions '99 where ISVs will relate how they have successfully increased productivity while developing and deploying applications based on SanFrancisco.

Provider Solutions will describe how they built the ProviderFirst OutcomesManager System. It is a SanFrancisco-based solution and set of reusable components that allow home health agencies to meet U.S. government data collection and analysis requirements for Medicare patients. The session will share Provider Solutions' experience using SanFrancisco, VisualAge for Java and WebSphere to build the application. Topics covered include the transition from requirements to models, implementation using SanFrancisco, use of JavaBeans and servlets, and the use and extension of SanFrancisco components.

FrontEnds will describe how they have used SanFrancisco to achieve a five-fold increase in programmer productivity. The application they have developed is a billing, scheduling and workflow system for a healthcare network in Texas. The session will focus on the development advantages of using SanFrancisco and FrontEnds' experiences in using the product. This session also will include a short overview on the common business objects, application business components and infrastructure for distributed object applications that is delivered with IBM SanFrancisco.

Product expertise and directions

Several presentations will describe techniques that have proven effective in building SanFrancisco-based applications and how SanFrancisco components can be used in conjunction with other IBM products. A separate session will provide additional details about how IBM plans to transition SanFrancisco to run on Enterprise Java Servers.

The session, *Performance Tips for Java Server Objects and SanFancisco Schema Mapping*, will review the SanFrancisco Schema Mapping tool and how it can be used to efficiently map business objects to relational databases (RDBs). It also will cover tips and techniques on how to achieve the best performance for SanFrancisco Java applications.

Another session, *Building Thin Client Applications Using SanFrancisco Business Objects and WebSphere*, will show how to enable SanFrancisco applications for e-business. It will explain how a three-tiered model can be applied to both fat and thin client front-ends. The session will show how to use a JavaBeans-based approach together with the WebSphere Application Server to allow business applications to run over the Internet or intranet. The session will show how to utilize JavaBeans, XML, WebSphere and VisualAge for Java to increase development productivity.

A hands-on session will be conducted to provide experience in using MQSeries and SanFrancisco together. The session will review how to:

- Setup an MQSeries test environment
- Write a Java client/server application using MQSeries
- Connect external applications with SanFrancisco applications by using MQSeries

SOLUTIONS '99 PRESENTER

The future transition of SanFrancisco to run on WebSphere EJB servers will be reviewed in the session, *SanFrancisco Migration to Enterprise JavaBeans and WebSphere*. This session will describe the plans for accomplishing the transition, including migration of existing applications and business objects. It also will indicate what developers can do today to position their SanFrancisco applications and business objects for migration.

Demonstrations and SanFrancisco Specialist Certification

Several demonstrations, based on the technical sessions, will be shown in the exhibit hall. They will include an ISV application's use of SanFrancisco, prototypes showing how to use SanFrancisco alone and with other IBM products to build e-business applications.

IBM will be offering SanFrancisco Specialist Certification testing at no charge during Solutions '99. Generally this is a $120 (U.S. dollars) charge. For more information, read about the test and the requirements on our Web site: www.software.ibm.com/ad/sanfrancisco/cert_appdev.html.

Summary

IBM SanFrancisco is being used today by Independent Software Vendors and customers to achieve large gains in productivity as they deliver Java applications. The sessions and demonstrations at Solutions '99 will tell how they are doing this and share valuable techniques that have been learned. The sessions also will discuss the future of SanFrancisco and how the product will evolve.

Additional information on IBM SanFrancisco is available at: www.software.ibm.com/ad/sanfrancisco.

A servlet architecture for IBM SanFrancisco

by Richard Cook

This article builds upon the "Creating a Servlet Gateway to an IBM SanFrancisco Logical Network" article by Mickey Nix, published in the December 1998 issue of the *Developer Connection Technical Magazine*. It is based on the NCF Examples code provided on the IBM SanFrancisco 1.3.0 CD. It uses the architecture outlined and implements a reusable implementation that gives you a framework for creating applet or application front ends that connect to SanFrancisco via servlets. The servlet acts as the SanFrancisco client on behalf of the applet or applications. By doing this, the applet does not need to download any of the SanFrancisco classes and can be as thin as possible. With only a few classes, the applet or application can access an IBM SanFrancisco logical network.

The article briefly examines the servlet architecture and how it communicates with SanFrancisco. Then, it discusses how to develop code using the servlet architecture and how to deploy the code.

Servlet architecture

The architecture consists of two servlets: The SFProcess servlet and the MainServlet. The SFProcess servlet is responsible for calling Global.initialize(), which should be called when the servlet engine is started. This helps ensure that everything is properly initialized before any servlet requests are received by the engine. After calling Global.initialize(), the SFProcess servlet suspends its distributed process in preparation for a servlet request.

Servlet requests are handled by the MainServlet. A Java client (applet or application) creates a ServletRequest object, which contains the URL of the MainServlet and the data for a command that the servlet executes on behalf of the client. When the ServletRequest is executed, a connection is opened to the Web server and the data is sent to the servlet using Java serialization.

When the ServletRequest is received, the MainServlet deserializes the data from the client and goes to the SFProcess servlet for a Work Area. Then the MainServlet dynamically loads the command class requested by the client and hands off the data to the command. The command executes any SanFrancisco code, beginning and committing transactions, accessing persistent objects and so on. Next, it passes any data back to the MainServlet, which completes the ServletRequest by serializing any data and passing it back to the client as shown in ***Figure 1***.

The IBM SanFrancisco Servlet Architecture is a means to standardize thin-client access to SanFrancisco applications.

Walk-thru

Let's take a look at some sample client code to see how it communicates with the MainServlet. First, the client creates a hashtable object that contains the particular data needed to perform the task at hand. Let's say we are going to create a new Customer object and need to pass the data from the applet to the MainServlet. Data is stored in the hashtable using String objects as the keys.

```
Hashtable h = new Hashtable(3);
h.put("firstName", "Lou");
h.put("lastName", "Gerstner");
h.put("mailingAddress", 123 IBM Lane, Somers NY 11111");
```

It is important to note that any data you put into the hashtable should be a Java primitive, string, or object that implements the serializable interface. The servlet architecture uses Java serialization to pass the data to the MainServlet.

Next, create a new ServletRequestDataContainer and give it the hashtable. In addition, there are two attributes the client must set in all instances of ServletRequestDataContainer.

- **CommandToken:** This is a string attribute that is the fully qualified class name of the class that the MainServlet will load and execute on behalf of the client. This is covered in more detail later in this article.
- **TransactionCode:** This attribute can be one of three things:

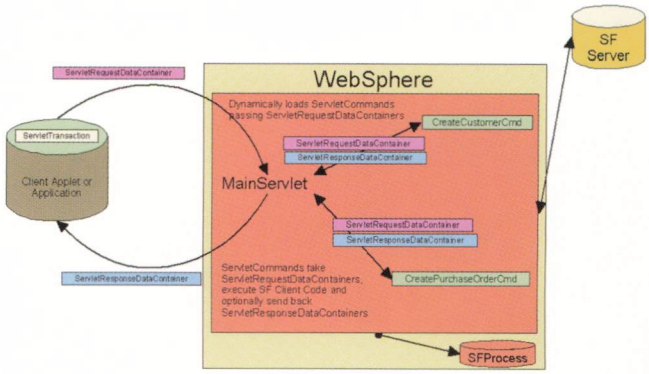


Figure 1. SanFrancisco servlet architecture

– ***TransactionCode.TRANSIENT_TX:*** This ServletTransaction begins and ends a SanFrancisco Transaction. There is no need for the MainServlet to keep track of the particular SanFrancisco Work Area that is used to perform SanFrancisco Client work on behalf of the applet. When the servlet request is completed, the work area is put back on the stack of available work areas it keeps on hand for requests.

– ***TransactionCode.PERSISTENT_TX:*** This signifies that the client is expecting this ServletTransaction to execute a command that may begin a SanFrancisco Transaction, access objects, and then want to keep a lock (Optimistic, Pessimistic, and so on.) across ServletRequests. When the command is finished executing the MainServlet suspends the Work Area and stores it in a

CONTINUED ON PAGE 26

CONTINUED FROM PAGE 25

hashtable using the ID as its key. The MainServlet then gives the ID number to the Client in a response object. More on this later.

- ***An int.*** The client has previously issued a ServletRequest that locked objects by starting a SanFrancisco Transaction, accessing objects, and then not committing the transaction before finishing the servlet transaction. The MainServlet then passes an ID number back to the client which the client should then use in a subsequent ServletTransaction (via another ServletRequestDataContainer) to finish the SanFrancisco transaction. That ID number should be used to set the TransactionCode.

Then, the client applet creates a ServletTransaction object using a URL that points to the MainServlet and the ServletRequestDataContainer, which contains the hashtable of data as shown in ***Listing 1***.

An attribute of the ServletTransaction that the client should set is the ExpectResponse boolean. If the client is expecting the MainServlet to send back a ServletResponseDataContainer, then this should be set to true. The default is set to false.

The client applet executes the request as shown in ***Listing 2***.

Calling doTransaction() causes the connection to the host Web server to be opened, an input stream to be opened and the ServletRequestDataContainer to be serialized and sent to the servlet.

The servlet receives the request and unpacks the ServletRequestDataContainer. Depending on the request's transaction code, it picks up a transient work area, creates a new persistent work area for long running transactions or retrieves an existing work area.

The servlet then looks at the request's command token and attempts to dynamically load the command class using this token and creates a new instance of it. It then calls the initialize method of the command object passing in the Servlet RequestData Container and the servlet itself to allow the command to prep itself for the command's actions. Next, the servlet calls the command's doTransaction method that contains the business logic of the transaction.

The commands executed by the servlet should implement the ServletCommand interface. Let's look at a sample command as shown in ***Listing 3***.

The doCommand method is where the code that the ServletCommand executes on behalf of the client should reside. Usually a transaction begins and persistent SanFrancisco objects are accessed in this method.

In doCommand(), the client data is retrieved from the hashtable and then passed to createCustomer() on the CustomerFactory class. This command returns data in a ServletResponse DataContainer, again using a hashtable and string keys. The ServletRequest was set to expect a response so the MainServlet takes this response object, serializes it and sends it back to the client.

The ServletResponse DataContainer contains an exception field that can be used to pass exceptions back to the client. In doCommand(), any SFExceptions generated should be caught and then passed back to the client as an exception object that does not extend from SFException.

Additional notes

The Servlet RequestData Container contains fields that store the

```
...
    URL host = new URL("http://www.theMan.com/servlet/MainServlet");

    ServletRequestDataContainer  requestData = new ServletRequestDataContainer();

requestData.setCommandToken("com.ibm.servlet.example.CreateCustomerCmd");
    requestData.setSessionCode(TransactionCode.TRANSIENT_TX);
    requestData.setData(h); //The hashtable containing data

    ServletTransaction st = new ServletTransaction(host,requestData);
    st.setExpectResponse(true);
```

Listing 1. Hashtable data

```
...

    ServletResponseDataContainer responseData = null;

    try
    {
        ServletResponseDataContainer responseData = st.doTransaction();
    }
    catch(IOException e)
    {
         e.printStackTrace();
    }

    if(responseData.getException() == null)
    {
         //do work
    }
    else
    {
        //handle exception
    }
```

Listing 2. Execution of request

```
public class CreateCustomerCmd implements ServletCommand
{
    private   Hashtable data ;

    public void initialize(ServletRequestDataContainer srdc, GenericServlet servlet)
    {
            data = srdc.getData();
            //you can do any initialize code here.
            //you can store a reference to the servlet here if you like so that you can
            //write to the servlet's log if necessary
    }

    public ServletResponseDataContainer doCommand() throws SFException
    {
            ServletResponseDataContainer response = new ServletResponseDataContainer();
            try
            {
                    String firstName = (String)data.get("firstName");
                    String lastName = (String)data.get("lastName");
                    String mailingAddress = (String)data.get("mailingAddress");
                    Global.factory().begin();
                    Customer c = CustomerFactory.createCustomer(..., firstName,
                                                                lastName,mailingAddress);
                    Hashtable h = new Hashtable(3);
                    h.put("firstName", c.getFirstName());
                    h.put("lastName", c.getLastName());
                    h.put("mailingAddress", c.getMailingAddress());
                    response.setData(h);
                    return response;  // this command happens to return an object.  The
                            // ServletRequest for the this particular command was set to
                            // expect a response so it will properly receive this response object
                                                                                    }
            catch(SFException e)
            {
                    Exception e1 = new Exception(e.getMessage());
                    response.setException(e1);
                    return response;
             }
    }
}
```

Listing 3. Sample command

user name and password for a request. Servlet Commands can use this data and utilize the methods in the SFProcess servlet to authenticate and authorize a user. See the documentation for the SFProcess servlet that is included on the SanFrancisco CD.

The Servlet architecture can handle long running transactions. It does this by storing work areas in a hashtable. The attributes of this hashtable can be fine tuned depending on the load the servlet may need to handle.

"WorkAreas Capacity" and "WorkAreas LoadFactor" are the two parameters that can be passed on to the hashtable constructor in the MainServlet's init method.

If using long running transactions, the Servlet's WatchDog thread should be started. It monitors the work areas that are stored in the hashtable. If a work area ages beyond a certain threshold, the WatchDog thread resumes the work area, rolls back any transaction that it may have been in and removes the work area from the hashtable. This effectively retires the work area, so resources or persistent objects are not locked by a dead client that never returns.

"StartWatchDog" should be "true" for the thread to be kicked off.
"WatchDogCheckMillis" tells the WatchDog thread how often to check for old Work Areas.
"WatchDogTTLMillis" sets the age threshold for the work areas. Look at the MainServlet's init method for examples.

Deployment

The SFProcess and MainServlet servlets can be deployed on any Web server that supports Java servlets. These servlets were developed and tested using WebSphere 2.0. Whichever servlet engine is used, it is important that the SFProcess servlet is loaded upon startup of the engine. This causes the servlet's init method to be called, which in turn calls Global.initialize(). The MainServlet should be loaded on its first service request.

The following steps for deploying the servlet are for WebSphere 2.0 and the IBM HTTP Web Server that comes with WebSphere 2.0. ***Note:*** The RMI patch for WebSphere 2.0 must be installed.

1. Install the SFProcess classes in a directory.
2. Install any ServletCommand classes in a directory. Probably the same directory as the SanFrancisco application's classes.
3. Run the WS administration applet.
   a) In Setup:Java Engine add the directories for SanFrancisco, the ServletCommands you have developed and your SanFrancisco application's classes to the Application Server Classpath. Your servlet commands classes will probably be in the same directory as the SanFrancisco application's classes. Add the directory for the SFServlet classes to the Reloadable Servlet Classpath. It is important to note that the SanFrancisco directory should be the first path in the Application Server Classpath.
   b) Save these changes.
   c) Go to the Servlets:Configuration screen and configure the SFProcess Servlet and the MainServlet. In the Name field of the servlet configuration screen, name the servlets SFProcess and MainServlet appropriately. Set the SFProcess Servlet to load at startup. The Load at Startup indicator might not display correctly, but it does load the servlet at startup when set. The Save button might appear to be disabled (gray) when the Load at Startup option is changed from No to Yes. Type something in the Description field to enable the save button and click Save to save the servlet's configuration.
   d) Make sure the SanFrancisco Server is up and running, and then restart WebSphere. Shut down the WS service and the IBM HTTP service. Next, restart the IBM HTTP service. WebSphere starts up along with it and loads the SFProcess Servlet, which calls Global.initialize(). The MainServlet is loaded when the first Web server request is received.
   e) Verify that the SFProcess servlet initialized properly by going to the Server Execution Analysis:Monitors:Loaded Servlets screen in the WS administration applet.
   f) Make sure the applet classes and HTML page are in the Web server's directory structure so that it can send the applet classes to clients.
   g) Start testing the applet.

Summary

The IBM SanFrancisco Servlet Architecture is a means to standardize thin-client access to SanFrancisco applications. Since it is based on the general method of moving data between the thin client and the application, it can be reused by SanFrancisco systems that need access through applets or applications through a servlet. The architecture takes care of the data passing mechanisms as well as managing Work Areas for the active clients. With a standard architecture, you can speed up the development time of your SanFrancisco applications with Web access.

Richard Cook *is a Technology Consultant with the IBM Solution Developer Marketing organization in Waltham, Massachusetts. In the past, he has worked with custom client/server applications and has worked with Java since late 1995 when he joined a Java development tools company. Since 1998 he has been working with IBM SanFrancisco helping vendors develop distributed object business applications. He can be contacted at richcook@us.ibm.com*

Editor's Note: The sample code included in this article can be downloaded from the Developer Connection Web site at www.developer.ibm.com/devcon/. The code also is available on the July CDs.

WebSphere Application Server

The following are FAQs pertaining to the IBM WebSphere Application Server that are of general interest. The questions have been edited for length and clarity but are typical questions asked by developers. We plan to include FAQs on other topics of interest to solution developers in future issues.

Q: What is WebSphere Application Server?

A: WebSphere Application Server is a complete Java-servlet-based Web server environment. It consists of a run-time environment and an integrated tools environment and runs with most existing HTTP servers.

Q: What is IBM WebSphere Performance Pack?

A: WebSphere Performance Pack is composed of three major components:

- An enterprise file system
- A caching proxy server
- A load balancing server monitoring tool

These features provide a combination of caching, filtering, file content management and replication, and load balancing into a single Internet-Hosting infrastructure.

Q: What is WebSphere Studio?

A: The IBM WebSphere Studio is a tool environment for building Web applications that leverage the WebSphere Application Server and extend Web sites beyond publishing and Information transactions. WebSphere studio combines a Web development workbench and servlet generation wizards with IBM's VisualAge for Java development environment and Web site development tools from NetObjects, such as ScriptBuilder, Fusion and BeanBuilder.

Q: When would a customer use the IBM WebSphere Application Server family, and when would it be better to use Lotus Domino?

A: Lotus Domino and IBM WebSphere Application Server meet different customer needs.

Domino is for customers who are focusing on collaborative/workflow solutions and want to build intranet/extranet Web applications that integrate business processes with existing IT systems.

WebSphere is for organizations building transactional Web applications using Java servlets and Enterprise JavaBeans. WebSphere provides a platform to upgrade from a publishing-based Web presence to a transactional selling environment.

Q: Does IBM WebSphere Application Server 2.0 include the Apache server?

A: No. IBM WebSphere Application Server 2.0 includes the IBM HTTP server. However, it supports Apache 1.3.2.

Q: What is the IBM HTTP server that comes with WebSphere Application Server 2.0?

A: The IBM HTTP server is an HTTP server-based on technology from the Apache HTTP server, with additional SSL-based security and performance features.

Q: How does WebSphere software connect to the AS/400 database?

A: You can connect to the AS/400 database using either CGI programs or Net.Data.* With AS/400* 4.3, you can use JDBC from Java CGIs or servlets as well as AS/400 Java toolkit classes for record level access.

Q: Can we use servlets and JSPs on the AS/400 platform today?

A: Yes, OS/400 version 4.3 supports the WebSphere Application Server, which provides the capability to run Java servlets as well as JavaServer Pages (JSPs).

Q: Are WebSphere products compatible with HTML editors such as Microsoft FrontPage and Macromedia Dreamweaver?

A: WebSphere products are compatible with many HTML editor including the two mentioned above. WebSphere creates Java server pages, which are cross-platform, instead of active server pages, which are not cross-platform.

Q: What type of education is available for WebSphere?

A: IBM WebSphere has a Web-based tutorial located at www.software.ibm.com/web-servers/appserv/tutorial.html

Q: I am not able to load a DLL library with WebSphere. What should I do?

A: You need to get access to the jcicseci.dll file through the PATH statement rather than through the CLASSPATH or anything in the JVM.PROPERTIES file. Make sure the location of jcicseci.dll is in your PATH statement.

Q: How do I get servlets to reload without stopping and restarting the Web server?

A: In order to have servlets reload so that you can see changes you've made, put the servlet in the WebSphere/servlets directory. The /servlets directory must not be in the system classpath, that is, not in the classpath specified in jvm.properties.

Also, in the servlets.properties file, there is a servlets.classpath directive that can be used to extend the reloadable classpath. You can add directories to allow reloadable servlets to be placed in directories other than the /servlets directory. Directories specified in this directive are used as a root directory (just like the /servlets directory) so you must maintain the package structure under these directories as well.

Make sure that all of the classes your servlet needs are using the same directory in the servlets.classpath. There is currently a bug with this directive that makes servlet classes loadable only from the directory structure from which the servlet's class file is loaded. This is because each directory in the servlets.classpath is loaded in a different classloader. Therefore, servlets loaded from a reloadable directory other than the servlets directory do not have access to classes in other reloadable directories (such as the /servlets directory).

If you add an entry to the servlets.classpath property and put your servlet in that directory, WebSphere automatically reloads your servlet code if it changes. The only time that WebSphere does not automatically reload is if the servlet is loaded out of the system classpath.

Q: How can the IBM WebSphere Performance Pack help me?

A: IBM WebSphere Performance Pack is a combination of software that can be applied to a Web server to help reduce server congestion from Web traffic, increase content availability and improve Web server performance. It uses caching functionality from IBM Web traffic express (WTE), load balancing from IBM eNetwork* dispatcher (eND) and the enterprise file system that can be placed anywhere in the network from the Transarc Andrew file system (AFS).

To learn more about the performance pack, visit www.software.ibm.com/webservers/perfpack/index.html.

Q: How do I use the generic object servlet in WebSphere?

A: GenericObjectServlet is a CORBA object server embedded within a servlet. It supports any set of objects specified in the applicationserver_root/web/classes/ObjectImpl.properties file. Entries within this file have the form:

```
interface-name=implementation-name
```

All names are fully qualified, that is, they specify the package, too.

To enable GenericObjectServlet to support the greeter object for HelloWorld, use a text editor to add the following interface-implementation entry to the ObjectImpl.properties file:

```
getting.started.Greeter = getting.started._GreeterImpl
```

The left side is the fully qualified name of the greeter interface; the right side is the fully qualified name of the greeter implementation.

Note: Class com.ibm.jbroker.GenericObjectServlet is in the WebSphere application server CORBA support ibmjbrt.jar file. To allow the Web server to successfully activate GenericObjectServlet, the ncf.jvm.classpath property in the jvm.properties file must include the ibmjbrt.jar file and the directory containing the ObjectImpl.properties file. The jvm.properties file is in directory:

- <as_root>/properties/server/IBMWebAS/ servletservice
- where <as_root> = drive and directory in which WebSphere has been installed.

Starting an object servlet is done automatically when the client calls method com.ibm.jbroker.Utility.getIHome() to obtain a reference to the servlet's IHome. See object servlet solutions for a complete description of WebSphere Application Server CORBA support's object servlet features.

Q: Why are CORBA server-to-client callbacks, as implemented in WebSphere, critical?

A: Callbacks are crucial for client applets, because unsigned applets cannot listen for incoming TCP/IP connections. HTTP is a connectionless protocol and, as such, callbacks are not available when the client and server ORB are communicating using IIOP over HTTP.

Q: What features make the CORBA support, found in WebSphere Application Server, a valuable addition to WebSphere?

A: The following features make the CORBA support in WebSphere an extremely powerful addition:

- Object interfaces can be defined directly in Java. WebSphere application server CORBA support includes a Java-to-"distributed Java" compiler (similar to the Java RMI compiler, RMIC) that works directly with your Java interface.class files to create CORBA compliant distributed versions of them.
- Object interfaces also can be defined in CORBA IDL. WebSphere application server CORBA support includes an IDL-to-"distributed Java" compiler that generates CORBA-compliant distributed object bindings for interfaces written in IDL.
- CORBA name service.
- Easy server-side object creation. WebSphere application server CORBA support include IHome and EJBHome classes: distributed object factories that, when added to an object server, define the set of objects it will support and allow clients to create new remote objects.
- Automatic object server activation.
- Extensible, generic object server. The GenericObjectServlet class, provided in WebSphere as part of the CORBA support, implements a CORBA object server that is ready for use, right out of the box. The developer needs to specify the set of objects the server supports and then configure the client to use GenericObjectServlet.

CONTINUED ON PAGE 30

CONTINUED FROM PAGE 29

- Pass objects by value.
- Use Java exceptions in a CORBA environment.
- Server-to-client callbacks. Servers may sometimes need to notify clients of some event or change in state. The WebSphere application server CORBA supports server-to-client callbacks when a direct IIOP connection is available between a client and server. Callbacks allow a server to invoke methods on a client without establishing a new connection between the two.

Q: I enabled user profiles, but all that is stored in the database are the USERIDs. Where is the rest of the information?

A: When you set up user profile support in WebSphere, you define a username and password for accessing the JDBC database. The user must have insert and update permissions on the table where the profiles are stored.

The WebSphere API addUser method adds the record (an SQL insert) when it is called. As data is added to the record with other API calls (the various setXXX calls), an SQL update is performed. Therefore, the WebSphere JDBC user needs insert and update permissions to the table.

Q: How easy would it be to move a Windows NT-based Java servlet application to AS/400?

A: Java servlets written using the WebSphere application server for Windows NT should run on the AS/400 WebSphere Application Server with little or no change required to the Java code.

Q: Does WebSphere software handle multi-membered physical files smoothly?

A: Yes, the HTTP server serves pages from either members' source physical files or ASCII stream files from the IFS file system.

Q: Under OS/400 4.3, will WebSphere work with both the current HTTP server and the Apache server?

A: The WebSphere application server currently works only with the AS/400 HTTP server. The Apache server, currently in beta, does not support WebSphere. Development is continuing to evaluate this requirement.

Software and hardware components needed to run Enterprise JavaBeans on AIX

THE FOLLOWING ARE THE HARDWARE AND SOFTWARE REQUIREMENTS NEED TO RUN ENTERPRISE JAVABEANS ON AIX:

SOFTWARE LICENSE PROGRAM PRODUCTS

THE IBM VISUALAGE FOR JAVA, ENTERPRISE EDITION FOR AIX VERSION 2.0 IS DESIGNED FOR BUILDING JAVA APPLICATIONS, APPLETS, SERVLETS AND JAVA BEAN COMPONENTS. IT CONNECTS CLIENTS TO EXISTING SERVER DATA, TRANSACTIONS, AND APPLICATIONS. IT IS SUPPORTED BY AIX VERSIONS 4.2.1 AND 4.3.X.

OTHER SOFTWARE REQUIREMENTS AND PREREQUISITES

- TCP/IP COMMUNICATION IS SET UP AND CONFIGURED PROPERLY
- NETSCAPE NAVIGATOR 4.04 OR HIGHER IS INSTALLED (JDK 1.1 IS ENABLED)
- FOR APPLICATIONS CONTAINING SWING COMPONENTS, INSTALL JDK 1.1.2 OR HIGHER

RS/6000 HARDWARE MODELS AND FEATURES

THE IBM RS/6000 FAMILY HAS A LARGE SERIES OF WORKSTATION SERVERS AVAILABLE TO MEET THE REQUIREMENTS OF COMPANIES, BUDGETS, AND SYSTEM CONFIGURATIONS.

THE ENTERPRISE SERVER MODEL F50 IS RECOMMENDED FOR DEVELOPERS GETTING STARTED IN EITHER PORTING OR ENHANCING APPLICATIONS USING VISUALAGE FOR JAVA ENTERPRISE EDITION FOR AIX VERSION 2.0. THE F50 SUBSTANTIAL PERFORMANCE IS A LIGHTNING-FAST 332 MHZ PROCESSOR SERVER.

THE F50 SERVER CAN HELP GET YOUR CRITICAL APPLICATIONS UP AND RUNNING FAST. THE F50 MICROPROCESSOR IS A POWERPC604E WITH MINIMUM MEMORY OF 128 MB AND MAXIMUM MEMORY OF 3 GB.

THE INTERNAL DISK MINIMUM IS 4.5 GB AND THE MAXIMUM IS 172.8 GB.

THE RECOMMENDED DISPLAY RESOLUTION IS 1024 X 768 USING A RS6K GXT120 GRAPHICS ADAPTER.

DISK SPACE REQUIREMENTS:

- 325 MB FOR FULL PRODUCT WHEN INSTALLED AS A STANDALONE CONFIGURATION OR 300 MB FOR THE FULL PRODUCT WHEN INSTALLED AS A CLIENT-ONLY CONFIGURATION.
- 192 MB FOR PAGING SPACE
- 200 MB MINIMUM FOR EACH USER

DIRECTORY

Order numbers in the following countries:

Austria	0660 8705	Mexico (D.F.)	387-5900
Canada	800-561-5293	Mexico (Interior)	01-800-006-3900
Germany	0 130 812177	United States	800-6DEVCON(633-8266)

Latin and South America order numbers:

Argentina	0-800-IBM YA (426 92)	El Salvador	02-98 5011
Bolivia	02-35 1840	Guatemala	02-31 5859
Brazil	0800-111426 r.1351	Honduras	32-2319
Chile	(2) 200-6868	Panama	02-639 977
Colombia (Nacional)	9800-17555	Paraguay	444-094
Columbia (Bogota)	616-7555	Peru	349-0040
Costa Rica	223-6222	Uruguay	0-800-A-IBM (0-800-2-426)
Dominican Rep.	566-5161	Venezuela	800-DE-IBM (800-33-426)
Ecuador	(02) 565-090		

Asia/Pacific order numbers:

The Developer Connection can be ordered in Asia/Pacific countries from IBM in Australia (61 is the country code) and Japan. Please insure that you dial the international access code applicable to your country before the listed phone or fax number.

Australia:	Phone	+61 2-9354 7684
	Fax	+61 2-9354 7766
	E-mail	apsdp@au1.ibm.com
Japan:	Fax	+81 3-5200 6310
	E-mail	os2pid@jp.ibm.com

Europe and other countries not listed:

The Developer Connection can be ordered directly from IBM in Denmark (45 is the country code). Please insure that you dial the international access code applicable to your country before dialing the appropriate phone or fax number. Operators speaking the following languages are available:

Danish	+45 48 101300	German	+45 48 101000
Dutch	+45 48 101400	Italian	+45 48 101600
English	+45 48 101500	Norwegian	+45 48 101250
French	+45 48 101200	Spanish	+45 48 101100
Finnish	+45 48 101650	Swedish	+45 48 101150
		Fax	+45 48 142207

Note: To subscribe to the Developer Connection in South Africa, contact Claudia Wissler, IBM South Africa, at 27-113029111, ext. 6960.

Electronic Support for the Developer Connection Release 2 Program

Electronic support is provided through the Internet and OS/2 BBS. Obtain technical support or use the forums to exchange messages, ideas or comments with the Developer Connection team or other subscribers. Note: Refer to the specific product information for the operating system technical support methods.

Internet users may address their questions or comments to devcon@us.ibm.com. The DEVCON CFORUM is on the OS/2 BBS under TalkLink, which is a feature under the IBMLink Commercial Services. For Talklink access, U.S. customers can call 1-800-547-1283; customers outside of the U.S. should contact their local IBM Marketing Representative. Note: SDP commercial members can access the DEVCON CFORUM via the Web from the SDP page without specifically signing up for IBMLINK.

IBM Solution Developer Program

The new, electronic Solution Developer Program makes it easier and more productive for commercial software developers to work with IBM. Members receive, at no charge, a wealth of information about IBM products and technology that can be searched 24 hours a day, seven days a week, on the Internet at http://www.developer.ibm.com/.

This program brings together the best features of IBM's existing developer support programs into a single worldwide program, including multiple "specialty" Partners in Development extensions covering all IBM operating systems (OS/2, AIX, OS/400 and S/390) and many software products like DB2.

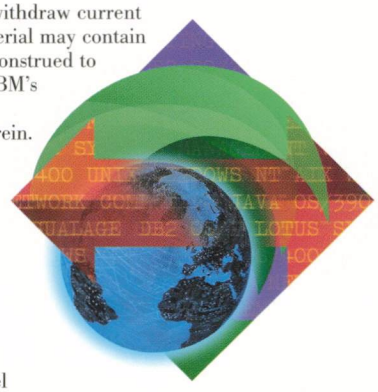
If you are not a member of our IBM Solution Developer Program, why not join now – registration is Free! It only takes a few minutes to register on the Internet, and in return, you'll receive a MEMBERSHIP ID and PASSWORD to access our services. Welcome aboard!

IBM Solution Developer Program Hotline:

United States/Canada	1-800-627-8363
Worldwide	1-770-835-9902
Worldwide Fax	1-770-835-9444

The Global Software Solutions Guide:

The Global Software Solutions Guide is IBM's comprehensive, online source for over 30,000 partner solutions. The guide is a worldwide resource for applications, tools and services. Visit the Guide today at www.software.ibm.com/solutions/isv/.



Staff

Publisher	Lorene Schlegel
Editors	Jean Swanson, Phyllis Aken, Angela Ausman, Tracey Marcelo
Technical Advisors	Ed Bardy, Amy Conner, Karen Foley, Steve Southworth
Graphic Design	Stil Point Images, Michael Rainer