;login:

THE MAGAZINE OF USENIX & SAGE

December 2000 • volume 25 • number 8

inside:

PROGRAMMING:

USING JAVA

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

using java

Enterprise Java Beans: An Overview

by Prithvi Rao

Prithvi Rao is the co-founder of KiwiLabs, which specializes in software engineering methodology and Java/CORBA training. He has also worked on the development of the MACH OS and a real-time version of MACH. He is an adjunct faculty at Carnegie Mellon and teaches in the Heinz School of Public Policy and Management.

<prithvi+@ux4.sp.cs.cmu.edu>

It goes without saying that distributed applications are here to stay. One reason for this is that hardware and software advances have made it possible for computers to talk to one another across a network relatively more easily than perhaps ten years ago. Whereas hardware has permitted connectivity of a heterogeneous arrangement of machines on a network, writing the software has been a more arduous endeavor.

The enabling capability that facilitates the running of distributed applications is the Remote Procedure Call (RPC), a network service whereby a remote client can make a procedure call that may be executed on a remote server.

Although RPC was a major breakthrough in permitting networks of machines to run distributed applications (filesystems are one example), the Application Programmer's Interface (API) – was not simple to master, affecting the ability to write robust and extensible applications in reasonable time.

In more recent times, RPC has been adopted as the transport mechanism for the Component Object Model (COM) and Distributed COM (DCOM), as well as the Common Request Broker Architecture (CORBA) and Remote Method Invocation (RMI).

In this article I present a brief overview of the various object technologies and then a summary of Enterprise Java Beans (EJB). In future articles I will provide more detailed examples on how to write an EJB application.

The Enterprise Environment

The term "enterprise" is ubiquitous in today's computing jargon. It typically means that the environment in question has to contend with and accommodate the changing needs of the organization deploying it.

For instance, in an Enterprise Environment it is typical to have to deal with:

- legacy systems
- decentralized IT decision making
- commercial products
- home-grown applications
- new acquisitions.

The main thrust of an enterprise environment is the pressure to extend, integrate, and evolve functionality over time.

Most if not all of these capabilities are driven by the need to get software to market on time or ahead of competitors, and also to do it at lower cost and higher efficiency and with more stability.

The Distributed Component Object Model (DCOM)

The first technology to facilitate the deployment of an "enterprise" solution was DCOM from Microsoft. More specifically, DCOM permitted the development of small pieces of code called "components" that could be connected to other components of the same type using well-known APIs. The windows "registry" was the central repository for information regarding DCOM "servers," and client applications were able to query the registry for the location of the server and so avail itself of its services. This technology

Another way of looking at this is that CORBA is the rest of the world's answer to DCOM.

was originally restricted to Windows-based platforms, but recently it has been made available on UNIX-based machines. Although DCOM did not strictly fit our definition of an enterprise solution, it approaches it more so today.

The Common Request Broker Architecture (CORBA)

The CORBA movement was born shortly after the advent of COM and DCOM from Microsoft. Another way of looking at this is that CORBA is the rest of the world's answer to DCOM. The CORBA movement involved several hundred organizations and it resulted in a CORBA specification. Various implementations of this specification are available today. The key concepts behind it were:

- Protect software that we write today and allow a choice of API.
- Make software interoperate; we cannot rewrite most of it.
- Concentrate on control and invocation interfaces.
- Expect technology to evolve.

It is fair to say that CORBA has gained popularity because it permits a heterogeneous arrangement of hardware and software and is more aligned with the enterprise solution than perhaps DCOM.

Remote Method Invocation (RMI)

The development of Java from Sun Microsystems brought yet another distributed-object technology in the form of RMI, which is also based on RPC.

The main difference between Java/RMI and the other two distributed-object technologies is that RMI requires that the server and the client be written in Java. More accurately, they must present Java calling semantics to each other. This means that although the server and client can be written in a "native" language, they must be "wrapped" in Java.

We can argue that Java also fits into our definition of an enterprise solution because the Java Virtual Machine (JVM) has been ported to a wide range of platforms.

Java Beans

The Java answer to components was the "Java Bean." The Java Bean is a Java program that conforms to well-defined rules regarding how it should be written. Another very simplistic way of looking at it is that it conforms to a "programming standard" of sorts. I will present a more detailed discussion on Java Beans in a future article.

An important aspect of Java Beans is that it permits software developers to write small components much the same as in COM and DCOM and then build larger pieces of software using the smaller building blocks. Eventually a complete application can be constructed using this method. Typically, graphical user interfaces lend themselves well to this strategy.

Enterprise Java Beans (EJB)

Enterprise Java Beans form part of a standard component system for application servers. Some features of EJB technology are:

- It is an interface standard for which vendors provide implementations.
- It has no relation with client Java Beans.
- It has gained wide acceptance in industry (35 vendors provide EJB-compliant interfaces).

The EJB package is part of the Java 2 Enterprise Edition (J2EE), which was announced in April 1997. EJB 1.0 was released in March 1998. The specification was defined by Sun in partnership with BEA and Oracle.

Some objectives were to:

- standardize Java application servers
- enable reuse of business components without access to source code
- make component and application development easy.

The EJB specification specifies the interface between:

- a client and an EJB
- an EJB and the EJB container.

An EJB application server includes:

- the server
- development and management tools
- standard infrastructure services such as RMI, security services, naming services, transaction services, database connectivity, and messaging services.

Summary

I have presented a brief overview of the common distributed-object technologies. The development of DCOM, CORBA, and RMI were all driven (in part) by the desire to support the notion of "enterprise" solutions. Enterprise Java Beans, arguably, bring new meaning to this expression. It remains to be seen whether EJB technology will live up to this reputation.