AC 2010-1701: LEVERAGING THE POWER OF JAVA IN THE ENTERPRISE

Javad Shakib, DeVry University

Mohammad R Muqri, DeVry University

Leveraging the Power of Java in the Enterprise

The ability to acquire, manage and utilize information has never been more instrumental. Without doubt, technology has been the most influential force behind the growth in economy. The Business is placing greater emphasis on information technology. Traces of information technology can be found from sales, to marketing, to inventory, to R&D. An integral component of technology is that it is highly dynamic. Technology changes at a rapid pace and the rate of change increases as well.

Enterprise applications require much more sophistication, because they have become more complex and larger than before. Even isolated systems are no longer absolutely isolated.

Java has emerged as the premier language for development of network-ready applications. The Java language and its extensions, provide a complete and robust environment for creating mission-critical and enterprise-wide applications. At the premise of network-centric computing, the web server becomes the focal point for enterprise applications. Java has a number of server-side capabilities which makes it an ideal fit for this environment.

Introduction

The ability to acquire, manage and utilize information has never been more instrumental. Without doubt, technology has been the most influential force behind the growth in economy. The Business is placing greater emphasis in information technology. Traces of information technology can be found from sales to marketing, to inventory, and to R&D. An integral component of technology is that it is highly dynamic. Technology changes at a rapid rate and the rate of change is increasing.

The Java programming language is one such instrumental change that has taken the industry by storm. Its introduction was followed by a huge growth in the computer industry. It is ironic, that such a success would be caused by a failure. Java was a descendent of a failed project at Sun Microsystems in its attempt to get into the interactive TV business. From one perspective, Java was at the right place and at the right time. The World Wide Web was just emerging as a technology that may finally allow every computer to communicate with others. The problem was lack of interactivity. Java did that in the form of shiny and multi-media rich applets.

The Enterprise wasn't very impressed, however. Enterprise applications require much more sophistication. Partly due to the World Wide Web phenomenon, the network became an integral part of the enterprise. Corporate LANs turned into WANs. Internet, Intranets, and Extranets became commonplace. The growth of the networks shifted the strategy for much of the enterprise. For one thing, the enterprise became more complex and larger than before. Isolated systems were no longer isolated. This paper begins with programming language comparison and delves into network centric computing, issues in enterprise development, and leveraging the power of java in enterprise.

Programming Language Comparison

There is a plethora of programming languages and new ones are being created on a constant basis for a number of applications. In order to provide some general guidelines for someone who wishes to decide which popular object oriented language(s) to learn and make a judicious selection for a certain application the following table is presented to help make evaluation and comparison depending upon the application.

	Visual Basic	C++	Java	C#	Perl
Object-Orientation	Ample Support	Yes	Yes	Yes	Add-On / Hybrid
Static / Dynamic Typing	Static	Static	Static	Static	Dynamic
Inheritance	None	Yes Multiple	Yes Single class, Multiple	Yes Single class, Multiple	Multiple
Method Overloading	No	Yes	Yes	Yes	No
Operator Overloading	No	Yes	No	Yes	Yes
Garbage Collection	Reference Counting	No	JVM Managed	Yes	Reference Counting
Class Variables / Methods	No	Yes	Yes	Yes	No
Access Control	public, private	public, protected, private	public, protected, private	public, protected, private, internal	None
Multithreading	No	Libraries	Yes	Yes	No
Regular Expressions	No	No	Standard Library	Standard Library	Built-in
Language Integration	C	C, Assembler	C, Assembly, some C++	All .NET Languages	C, C++
Built-In Security	No	No	Yes	Yes	Yes

Network-Centric Computing

A new paradigm had emerged which considered the network as an integral portion of any computing environment and with that great emphasis was placed on the server. The traditional client/server architecture was modified. The network-centric paradigm allows the client to be very thin. The client could be a cellular phone. That is because, the server not only holds the data, but it also hosts the applications. The client is merely a display device or gadget with some processing power.

Java is a natural fit for this paradigm. The Java virtual machine could be embedded in small devices which could then interact with application servers. Such servers can be deployed in a traditional client/server environment or the network-centric paradigm discussed above.

Issues in Enterprise Development

So far we have used the term enterprise computing in a very generic manner. Characteristics of enterprise computing can be classified into four distinct categories which are discussed below.

Applications

Applications form the soul of the enterprise. Users interact with the enterprise through the applications. Applications include legacy, more conventional client/server, and desktop. Unfortunately all classes of applications are subject to their own problems. For example, it is no secret that legacy applications are extremely hard to maintain and their integration into the enterprise is somewhat of a challenge. Many client/server applications suffer from incompatibility problems. This incompatibility is evident both through the languages and operating systems on which these applications function. The incompatibility problem is magnified because of the close interactions among applications made possible by advances in computer networking. Consider a typical department that has chosen a word processing package as its standard incompatibility problems would be minimal for documents shared among the employees in the same department. However, due to the growth in the network that same department must interact with the rest of the enterprise for everyday business functions. Since the rest of the enterprise may not be using the same word processing package, incompatibility does become a problem.

Infrastructure

Infrastructure forms another important aspect of enterprise computing. The complexities of an enterprise infrastructure are not something everyday users appreciate, but as soon as a glitch is developed those very same users are handicapped in getting their work done. The same way that desktop applications may prove to be incompatible, server applications and server configurations can vary throughout an enterprise. Desktop clients are part of the infrastructure and as the debate on total cost of ownership has shown, desktop maintenance throughout the enterprise is a costly endeavor. This is partly due to the complexity all of the applications themselves which in turn causes the desktops to become more complex.

Data

By data we refer to the large collection of raw data stored in many databases throughout the enterprise. There is incompatibility in the format and nature of this data. This in turn causes incompatibility among the many applications that access such data. While it is comprehensible that the raw data is stored in different formats, a layer must exist to provide a uniform "business" view of the data. In other words, the business meaning of the data must be used same throughout the enterprise. One problem that most enterprise applications face is data access both at the "raw" level and at the higher "business" levels.

Support

Finally, any enterprise must be supported in order for it to stay operational. As the complexity of the applications increase, the support issues become more complicated. Ask any CIO about their latest "upgrade" effort and they will surely come up with a few good stories. With fat clients scattered throughout the enterprise, support resources must also be dispersed which effectively adds to the cost. The complexity of support issues is really a direct function of the complexity of the enterprise itself. If we can "simplify" the enterprise, then supporting it ought to simplify as well.

Java in the Enterprise

Let's continue to discuss some more ways Java can aid the enterprise at a higher level. Java shifts much of the computing burden to servers. It encourages deployment of a variety of servers for specific tasks. Data continues to be stored in database servers. Applications are mostly stored in application servers. Such servers can either be used to download all or part of the application to the client or to run the application themselves. Web servers continue to be a source of interactive communication. Boot servers are responsible for making sure thin clients have an operations system ready when they are turned on. File, Mail, Print, and Directory servers continue their traditional roles in the enterprise. Java computing encourages a cohesive strategy for bringing all of the above under one umbrella otherwise known as the corporate network. Management and maintenance of much of enterprise operations are simplified by this massive shift to servers. Many organizations that have developed and deployed intranets are already experiencing the rewards for this strategy.

On the client side, Java computing relies on thin clients to achieve its overall objectives. While fat clients are necessary for certain operations, many times, such clients are used to perform a single operation. Under these conditions, the cost of a full-blown client is not justifiable. Java computing encourages design of server-centric applications that are component-based for easy development and deployment. It also encourages usage of a network-centric paradigm for applications.

Java provides a very complete and unique server-based approach to computing. From Enterprise JavaBeans to Java Database Connectivity and Remote Method Invocation, Java has what it take to be a premier enterprise computing environment. From an architectural point of view, the server side of Java is perhaps its greatest contribution to enterprise computing. The following is

an overview of some specific Java technologies that are useful for development of server-based application.

Java Database Connectivity

Java Database Connectivity (JDBC) is a set of API and a specification for connecting to database systems through Java. A uniform view of the data and method for accessing the data is essential in an enterprise environment JDBC delivers on this point. The specification requires a JDBC driver for each data source (Oracle, Sybase, Informix, etc.) The JDBC API is then built on top of that driver.

One major benefit of JDBC is that applications written based on the JDBC API are considered database independent. ODBC brought the same degree of flexibility to the Windows operating system. The major drawback of JDBC is that it is a least common denominator. Some database specific operations and syntax is not available through JDBC. There is also a performance penalty, but optimized drivers and faster processing speeds (especially on the server) seem to reduce the performance impact.

Java Servlets

Servlets are the equivalent of applets except they reside on the server. While CGI is still the most popular mean of creating Web applications, its deficiencies are well known. The partial solution to the CGI dilemma is server-specific APIs such as NSAPI and ISAPI; however, they too have a deficiency. In the enterprise, we strive to create systems that are compatible. Server-specific APIs lack in the compatibility and portability category. Java servlets[6] are a uniform standard API for creating web-based applications. The technology is extensible, however, and could be used for development of any kind of server.

Servlets run in the same process space as the Web server. Servlets have access to the name-value pairs passed from HTML forms. After initialization, the servlet can interact with other systems and generate appropriate output. From an efficiency point of view, servlets are much more efficient than CGI. Additionally, servlets can utilize all the aspects of the Java environment many of which are unavailable to other languages such as Perl.

RMI and Distributed Objects

Data access and Web servers are integral parts of enterprise computing. Right up with them are distributed objects and all the programming benefits they bring. Distributed objects are built upon the idea of component-based software. The idea is rather then developing one "large" and complex application, one should focus on the individual components. A distributed object environment such as Common Object Request Broker Architecture (CORBA) or Remote Method Invocation (RMI) can then be used to link these components together.

Java is object-oriented in nature and that's a major plus compared to procedural methods of distribution such as RPC. Also, Java has a native object-to-object communication mechanism called Remote Method Invocation. RMI allows Java objects to invoke methods on other objects residing on other machines. Such integration heavily depends on the network and is a major aspect of network-centric computing. Your application object can simply make a call to an object residing on the database server as if it were a local object. RMI makes this level of interaction seamless.

Enterprise JavaBeans

JavaBeans is a component model for developing Java clients. While not mandatory, most Beans are developed and used in visual contexts. For example, buttons, dialog boxes, and menus can be Beans. Java on the server took a major leap forward with the introduction of Enterprise JavaBeans (EJB). EJB does for the server what JavaBeans does for the client. EJB[3] components are strictly server-based and thus must support server operations such as transactions, persistence, distributed services, multi-tier operations, and security.

The idea behind EJB is to build a server component that can be used for other applications. EJB also puts an emphasis on distributed objects since distributed objects are a source of component-based software.

Enterprise Web Servers

A Web server is important part of a multi-tier application. Internet Information Server[1] (IIS) is a group of internet servers (including a Web or Hypertext Transfer Protocol server and a File Transfer Protocol (FTP) server which comes with additional capabilities for Microsoft's Windows 2000, XP, Vista and NT Server operating systems. Microsoft Personal Web Server (PWS) is a scaled-down version of IIS for a personal computer (PC). The Apache Web Server is an open source product that runs on Unix, Linux and Windows platforms.

Sun Microsystems recently released GlassFish[4] Enterprise Server v3 which includes many contributions from Java and open source community members. It provides customers with an open source based server solution which endeavors on reducing deployment complexity and will eventually enable organizations to create and deploy modern Web applications and leverage the power of the full Java EE6 platform for enterprise applications. The added bonus would be simplified programming model and enhanced productivity which will trigger the rapid development and significantly decrease the time-to market.

Java Naming and Directory Interface

The Java Naming and Directory Interface (JNDI) is part of the Java platform, providing applications based on Java technology with a unified interface. This interface provides a generic mechanism for Java applications to access multiple naming and directory services in the enterprise. Several standards are supported including Lightweight Directory Access Protocol

(LDAP). LDAP can be defined as an internet protocol that email and other programs use to look up information from a server.

Directory services are important in any enterprise environment. A central directory allows enterprise managers to effectively address issues such as authentication, bill-outs, and control application access. Statistical logs can also be used to assess how much different applications are being used which would help in planning for future growth.

Java Message Services

One of the most prevalent message-based middleware is MQ Series from IBM. Message oriented middleware is an effective technology for connecting applications where real-time processing is not always available or not desired. Transaction-oriented systems typically fall into this category. Banking transactions, hotel and airline reservations systems typically utilize a message oriented middleware as part of their architecture. For example, the airline reservation systems must respond in real-time whether a ticket is available or not. It does not necessarily have to print the ticket or perform the billing in real-time. So as the reservations are confirmed, they are queued up for the printing and billing systems.

Message-oriented middleware is central to many enterprise applications and Java recognizes this by providing the Java Message Service (JMS). This service allows Java applications to interface with a variety of message-based middleware systems including various aspects of push/pull technologies.

Java Definition Language

In any enterprise environment, typically there are a number of systems that must interact with each other. These systems and applications operate under different environments and their hardware and software architecture are different As a result, system integration becomes an important function of any enterprise architecture. Just as English is known as the standard language around the world, a common architecture is needed to standardize the way various systems can communicate and interact with each other. One such standard is CORBA. The specification has wide industry support. When you make your applications and services CORBA-compliant, then those services can be used by any other CORBA-compliant application. New systems can be designed with CORBA in mind and existing legacy applications can be "wrapped" with CORBA.

CORBA achieves its common interface via OMG IDL (Interface Definition Language). IDL is not a programming language. It just describes interfaces between distributed components. It does not depend on any particular programming technology. As such, interfaces are defined in IDL and implemented in various languages. From IDL interface descriptions an Object Request Broker (ORB) product automatically generates code in language of your choice to effect the “glue” that connects the components and manages their intercommunication. Thus Java IDL allows you to do your implementation in Java. As a result, services written in Java can instantly

become available to the enterprise via their CORBA interface. In a heterogeneous environment, such capability is a must.

To avoid becoming too much dependent on a lone vendor, enterprises often prefer solutions that have broad industry acceptance or are open source solutions. Popular choices today include Common Gateway Interface (CGI) programs, mod Perl programs, and PHP for page authoring.

CGI programs allow the Web server to get information from other applications before responding to the browser. The mod Perl plug in for the Apache Web Server integrates Perl code with the Web server, so programmers can write extensions in Perl. Many developers use mod Perl as a replacement for the CGI interface because it addresses some of CGI's limitations. Creating CGI programs, mod Perl scripts, and writing beans or custom tag libraries requires familiarity with Java programming language. As such we can say that JSP technology essentially enables a tiered development methodology. The salient differences are summarized in the table as follows:

	JSP	ASP	Perl/CGI	Mod Perl
Scripting Language	Java	JScript, VBScript	Perl, C	Perl
Memory Leak Protection	Yes	No	Yes	No
Web Server	Multiple	Microsoft IIS or Personal Web Server	Any Web Server	Apache Web Server
Portability across servers/platforms	Yes	No	No	No
Reusability/Modular Code	Yes	No	No	No
Concurrent Access support without separate processes	Yes	Yes	No	Yes

Conclusion

The network-centric computing paradigm places great emphasis on the enterprise server. It also leads to heterogeneous environments where interoperability and communication among various systems becomes an issue. The Java language is a prime candidate for enterprise development since it is object-oriented and network-aware. The key to its power is write-once, run anywhere model. The Java Runtime Environment (JRE) translates java code into machine instructions that run on virtually all of the platforms. Extensions and enhancements to the Java language provide a variety of services which are integral to any enterprise language. These services include

transaction processing, CORBA support, middleware interfaces, and distributed processing capabilities. Enterprise servers not only provide the foundation to develop and deploy web services but further enhance value added services for management, monitoring, diagnostics, clustering, transaction management, and high availability of mission critical services.

BIBLIOGRAPHY

1. Deitel, H., et.al., Internet & World Wide Web How to program, Second Edition, Upper Saddle River, NJ: Prentice Hall, 2002.

2. Anderson, R. "The Long and Winding Road to Web-Based Apps." Network Computing 19 March 2001: 79-84.

3. Haefel, R., Enterprise JavaBeans, Fourth Edition, O'Reilly, June 2004.

4. GlassFish Community, https://glassfish.dev.java.net/ (accessed January 2010).

5. Sun Developer Network, http://java.sun.com/products/jndi/index.jsp (accessed December 2009).

6. Java Server Pages, White Paper, http://java.sun.com/products/jsp/jspguide-wp.html (accessed December 2009).