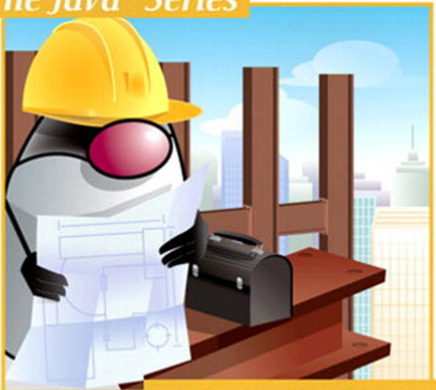
B. V. Kumar · Prakash Narayan · Tony Ng

Implementing SOA Using Java™ EE

Forewords by Robert Brewin and Raj Bala

The Java™ Series



...from the Source

Library of Congress Cataloging-in-Publication Data:

Kumar, B. V. (Balepur Venkatanna), 1959-
Implementing SOA using Java EE / B.V. Kumar, Prakash Narayan, Tony Ng.
p. cm.

ISBN 978-0-321-49215-9 (pbk. : alk. paper) 1. Service-oriented architecture (Computer science)
2. Java (Computer program language) I. Narayan, Prakash, 1960- II. Ng, Tony. III. Title.

TK5105.5828K95 2010
004.6'54--dc22

2009041877

ISBN-13: 978-0-321-49215-9
ISBN-10: 0-321-49215-3

Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.

First printing December 2009

Editor-in-Chief
Mark Taub

Acquisitions Editor
Greg Doench

Development Editor
Songlin Qiu

Managing Editor
Kristy Hart

Project Editor
Anne Goebel

Copy Editor
Apostrophe Editing Services

Indexer
Lisa Stumpf

Proofreader
Karen A. Gill

Publishing Coordinator
Michelle Housley

Cover Designer
Alan Clements

Senior Compositor
Gloria Schurick

Forewords

Robert Brewin

Recently, seasoned analysts like Anne Thomas Manes have said that SOA is dead and that it has failed to deliver its promised benefits. There have been opposing viewpoints to this. ZDNet blogger Joe McKendrick hosted a panel discussion on "Avoiding SOA Disillusionment," and the panelists concluded that any perceived disillusionment stemmed from lack of planning and measurement on the part of the Enterprises and not from a failure of SOA. In fact, Enterprises that have been working with SOA practices and methodologies remain bullish on the approach and recognize that SOA continues to hold promise as a model for integration and helping to tactically reduce costs in tough times. The promise of SOA is that it offers an architectural approach to support the proliferation and adoption of reusable services. This is an approach that companies should adopt to streamline their development processes and improve the quality and maintainability of their code.

At Sun, we developed the Java Platform, Enterprise Edition (Java EE) as an industry standard, and it forms the ideal foundation upon which developers can implement Enterprise-class SOA and next generation web applications. I am pleased to see this book by Kumar, Narayan, and Ng, which takes a practical approach to implementing SOA with Java EE. The focus is on real implementation techniques, leveraging the GlassFish Application Server and NetBeans IDE. By taking this approach, the authors have demystified SOA from an alphabet soup of Web Services standards and shown how readers can implement SOA in their Enterprise readily and easily. In addition to explaining the concepts of SOA and the concepts of Java EE, the authors dive deep into implementing SOA with

Java EE and show how services can be delivered within different tiers of an Enterprise architecture.

Architects, developers, managers, other IT professionals, educators, and students will benefit from different aspects of this book from concepts to architecting to implementation, configuration, and tuning. I trust that you will find this book beneficial and enlightening.

Robert Brewin
Chief Technology Officer, Software
Sun Microsystems

Raj Bala

Now more than ever, concepts like availability, leveragability, scalability, expandability, extendibility, and security permeate every discussion on technology architecture. As companies become more aware of harvesting maximum sustainable value from technology investments, the architecture fraternity has always cried loud for how the fundamentals matter. Architectural integrity is measured by all the "itys" that I mentioned in my first sentence, and it is heartening to see how the answers have been around and, in fact, getting better.

Service oriented architecture (SOA) as a fundamental fix to future problems has evolved to newer and more advanced frontiers. Saddling on ever-perfected technologies such as Java EE, SOA is becoming more appealing and compelling than ever before.

At Cognizant, we have been developing and delivering Enterprise solutions using SOA. And it is my privilege to write a Foreword for a book for one of our own—Kumar is a coauthor along with Prakash and Tony. The book carefully unravels the vast topic of service oriented architecture through a definitive and illustrative approach. It segments web services across First Generation Web Services for services composition, Second Generation Web Services for wiring these services into the process/workflow of the enterprise, and WS-* for addressing the nonfunctional needs of the Enterprise application. This book will also double-up as an effective implementation guide on the advanced features of the new Java Platform, Enterprise Edition and indicate how different APIs, such as JAX-WS and JAXB, of the new platform help in different aspects of service orientation for the Enterprise application.

This book should be extremely relevant to a variety of stake holders including architects, senior enterprise developers, and application integrators. This book is also a great reference material for students of computer science, software, and systems architecture.

From academics to architects, practitioners to pedants, students to specialists, coders to CXOs, this book could be a vital source of SOA inspiration—of how to build great architecture without compromising on the "itys."

Raj Bala
VP and Chief Technology Officer
Cognizant Technology Solutions

3

Evolution of Service Oriented Architecture

The requirement of service orientation for the enterprises first emerged with the advent of the Internet and World Wide Web. The IT world has since witnessed numerous paradigm shifts, as newer technologies such as XML and Java impacted enterprise solution requirements. The business of "service delivery" started gaining momentum among the enterprises and their collaborators. But the IT definition of the term "service" was not aligned with that of the business definition, and this cohesion was crucial for the enterprises to remain competitive in the dynamically changing market conditions. Evolution of business components such as Enterprise JavaBeans, as a part of J2EE technologies, on the one hand, and the emergence of core constituents of web services such as SOAP, WSDL, and UDDI, on the other, provided the opportunity to draft service definitions in alignment with the business requirements. Furthermore, the eventuality of loosely composing these services and binding them with the business process of the enterprise resulted in the arrival of Service Oriented Architecture *(*SOA*).*

The idea of SOA is not completely new. Different forms of service orientations were previously attempted and implemented as enterprise solutions by many vendors on different businesses and enterprises during the era of client/server. These architectures were implemented as enterprise solutions with different degrees of success, but they were never known or termed as SOAs during those eras. Regardless, none of these attempts could be considered successful implementation of SOAs. Before the arrival of XML and other web services, SOAs, (though not referred to as such), were implemented as a solution, without snazzy name and fanfare. In this chapter, we first explore the concept of service orientation and then analyze how the emergence of different architectures' combined paradigm shifts in enterprise technologies led to the evolution of web services and SOA.

Services Oriented Architecture—The Description

SOA can be described as a unique style of architecting and designing the enterprise solution using business services throughout the life cycle—from concept to retirement. SOA also enables for provisioning the IT infrastructure of the enterprise so that disparate applications[1] can exchange data as a part of the business process.

Business services can be defined as a set of actions or tasks an organization provides to different service stakeholders. Some of the service stakeholders are customers, collaborators, clients, employees, and so on. Consider that whereas an SOA can be defined as an approach to building IT systems, the business services are considered the key organizing principle for aligning IT systems for business needs.

The key point here is *business services* and *alignment of IT infrastructure* as per business services and business process requirement. Service orientation, therefore, enables the architects to focus on the description of the business problem rather than any development or execution environment of the enterprise solution. Because these two are delinked, a business solution that is architected as per SOA would be loosely coupled, flexible in nature, and allow implementation of dynamic needs of the enterprise business requirements.

It is important to notice here that the description of SOA does not mention the requirement of web services technologies as a prerequisite. Technologies such as CORBA or J2EE can still be efficiently and effectively used to implement the enterprise solution so that enterprise architecture is service oriented. However, what is crucial in the context of service orientation is the possibility of separating the *service interface* from the *execution environment*. An SOA that is appropriately implemented provides a scope in which it is possible to mix and match the execution environment.

Early Architectures

Earlier approaches to building enterprise solutions essentially focused on functional aspects of the enterprise problem. These approaches tended to directly use the specific implementation environments, such as object orientation, procedure orientation, data or information orientation, message orientation, and so on to solve business problems. This resulted in enterprise solutions that were often tied to features and functions of a particular environment technology. Some of the popular technologies that evolved were *Information Management Systems (IMS)*, *Customer Information Control Systems (CICS)*,[2] *Common Object Request Brokered Architecture (CORBA)*, *Component Object Model/Distributed Component Object Model (COM/DCOM)*, and *Message Oriented Middleware (MOM)*.

Enterprise architectures have evolved tremendously since the Mainframe era or the Centralized Model of mainframe architecture. The progression in architectures such as client/server architectures, distributed architecture, or web architectures discussed in Chapter 1, "Introduction," are generic in nature. Specific architectures on mainframe systems, such as IMS, CICS, CORBA, and DCOM, have evolved as environment-specific distributed architectures. You need to analyze some of these technologies and their contribution to the evolution of enterprise architectures.

IMS

IMS is one of the earliest technologies to lay the foundation for more advanced data accessing technologies such as DB2 and Universal Database. IMS was developed by IBM in the late 1960s to manage data for NASA's Apollo Moon Landing project. This technology was later released as the world's first commercially sold Database Management System. IMS technology's data management was based on the earliest data model called the Hierarchical Data model. This premier database and transactional management system was implemented to handle many commercially critical, online operational and on-demand business applications and data that enabled information integration, information management, and scalability.

The IMS technology essentially is composed of two subsystems: a Database Manager called IMS DB and a Transactional Manager called the IMS TM. We explore briefly these two subsystems in the next section.

IMS as Database Manager

The IMS DB is basically a large system Hierarchical Database Management System. When introduced, IMS DB was an enormous success, and many large organizations employed IMS DB for managing the enterprise information. Subsequent research and development efforts by IBM resulted in the revolutionary way of handling the data. The *Relational Database Management System* (*RDBMS*) by E. F. Codd in 1971 prompted IBM to introduce a radical product called the DB2. Following the introduction of DB2, IBM intended to replace the Hierarchical Data Management System with relational databases and replace IMS DB with DB2. However, IBM was not entirely successful in replacing IMS because a number of major IMS-based organizations were not interested in replacing the otherwise stable and satisfactorily running IMS-based applications. As a result, IBM continues to develop newer products and packages around the IMS technologies that help those organizations that continue to maintain IMS-based legacy products on their mainframe systems.

IMS as Transactional Manager

The IMS TM is a robust transactional management system that primarily functions on the IBM mainframe systems. This Transaction Manager was initially designed as an interactive system that interacts with an end user, through a combination of 3270 screens and VTAM communication mode to process business transactions. In coordination with IMS DB, IMS TM technology uses a messaging and queuing methodology to implement the transactions in the business processes.

When the user initiates a transaction through a 3270 screen, the IMS Control Program receives a transaction identification number and stores it on a message queue. The Transaction Manager, thereafter, invokes a scheduler on the queued transaction to initiate the business process. The message processing region of the IMS TM then retrieves the transaction from the IMS message queue and processes the same. The processing could involve reading/writing/updating the information on the IMS DB.[3] Based on the system design and the architecture of the enterprise application, the IMS TM could respond and return an output message to the user who initiated the transaction on the 3270 terminal.

CICS

CICS from IBM is a transaction server that runs primarily on IBM mainframe systems under operating environments such as z/OS. CICS is now available for other operating environments such as OS/2, AIX, Microsoft Windows, and Linux. The z/OS implementation of CICS is, by far, the most popular and significant implementation of the CICS technologies.

CICS is a transaction processing system designed for both batch and online business transactions. On large IBM mainframe systems, CICS technology supports a large number of transactions in a given time. The CICS technology has enabled IBM to retain a dominant position in the mainframe-oriented enterprise computing. Initially CICS applications were written in COBOL. Presently, CICS applications can be created using a variety of modern programming languages, such as PL/I, C, C++, REXX, and Java. CICS is one of the world's most durable software products on the IBM mainframe system. Supported by a variety of applications and tools, CICS is known for its reliability, security, and performance, particularly on IBM mainframe systems. Thanks to the aggressive marketing by IBM and rich research and development efforts in the United States and the UK, many of the Fortune 500 giants that invested into these systems during the Mainframe era continue to rely on core parts of enterprise applications based on CICS technologies.

The CICS applications programs are basically screens, popularly known as 3270 screens.[4] The initiation of a CICS program signals the initiation of a transaction, and the system initiates a transaction identification number. The CICS screens are sent as "maps" or "pages" using a programming language such as COBOL. The end user, on the other end of the system, inputs data that is made available to the CICS program by receiving a map. CICS screens essentially contain textual information. The textual information is presented to the end user in different formats. This includes highlighted text, colored fonts, or even blinking text.

CORBA

CORBA is not that different from the RPC technologies introduced in Chapter 2, "Evolution of IT Architectures." Developed and supported by *Object Management Group* (*OMG*), CORBA technology can be considered a generalization of RPC technology and includes several improvements on the data objects and on the data primitives. The purpose of this technology and architecture was to enable the development of distributed applications and services that can interoperably communicate with other disparate applications over the network. The CORBA architecture was essentially developed to bring about a discipline to implement portability and interoperability of applications across different hardware platforms, operating environments, and disparate hardware implementations. CORBA technology uses a binary protocol called *Internet Inter-ORB Protocol* (*IIOP*) for communicating with the remote objects.

DCOM

A bit of background is required here. In the mid-1990s Microsoft Corporation introduced a technology popular as the COM.[5] This technology enabled the development of software modules called *components* for integrating applications over the client/server architecture. To build these components, developers must adhere to the COM specification so that the components can operate interoperably within the network. The DCOM technology, introduced sometime in late

1990s, enabled interaction among network-based components to bring in the *Distributed Communication Environment* (*DCE*). DCOM technology is essentially built on an object RPC layer, which in turn is on top of DEC RPC to enable the communication among the remote objects. DCOM technology uses a binary protocol, termed *Object Remote Procedure Call* (*ORPC*), for distributed communication among remote objects. Technologies such as *Object Linking and Embedding* (*OLE*), ActiveX, and *Microsoft Transaction Server* (*MTS*) are some of Microsoft's technological advancements built on COM and DCOM technologies.

Paradigm Shifts

We previously indicated that the field of information technology has witnessed many paradigm shifts.[6] These paradigm shifts are affecting the enterprise businesses in many ways—specifically in how they conduct business and communicate. These paradigm shifts can be primarily attributed to technological innovations in the field of hardware, software, and operating and networking environments. Some of the paradigm shifts[7] that are of importance to the enterprise businesses are

- Internet and World Wide Web
- Java and Java 2 Enterprise Edition
- Extensible Markup Language
- Web Services—XML-RPC and SOAP
- Influence of the Internet and the World Wide Web

The arrival of both the Internet and the World Wide Web ushered in a paradigm shift to the enterprises, specifically in the way business transaction takes place. You might be aware that extensive research and development work sponsored by the Department of Defense[8] resulted in the foundation of what is now the Internet. The evolution of the web, in fact, ensured fundamental changes in the way B2C and B2B partners interact. More revolution than evolution, the Internet and World Wide Web has enormously grown, thanks partly to the contribution from several companies, organizations, academic and research institutions, and even the individual professionals all over the world. On the technology front, the web has not only rendered TCP/IP as the default business protocol, it also has brought forth a new type of client called the *browser client*.

Java and Java 2 Enterprise Edition

Prior to the arrival of Java, the software development for any enterprise application needed to be developed on many programming environments, on different hardware and operating environment. Frequently a software application would need to be developed and delivered on multiple hardware platform and operating environments so that functionally they delivered repeatable results. Developed by Dr. James Gosling of Sun Microsystems, Java technology was introduced in 1995. The arrival of Java as a programming language ushered in yet another paradigm shift in the world of software development. A Java Virtual Machine would behave the same way on any platform, and therefore, applications developed using Java programming language would behave reliably and consistently on any platform. Java programming has brought about acronyms such as *WORA* (*Write Once Run Anywhere*), *WORE* (*Write Once Run Everywhere*), and *WORD* (*Write Once and Run on any Device*).

Java and J2EE technologies have witnessed tremendous growth over the past decade and Java, in particular, has been the most widely employed programming environment in the world today. Java is easily considered the most successful programming language. Some of the features and attributes that popularized the Java platform are object oriented, platform independent, portable, secure, robust, multithreaded, and more.

One of the prime reasons for the widespread industry adoption of this environment could be because the environment has been the product of the industry movement toward the requirement of portable and interoperable applications that can work over the web. Other contributing factors include reliable web component technologies, such as Servlet and *JavaServer Pages* (*JSP*), and distributed components such as *Enterprise JavaBeans* (*EJB*) that can enable the developers to deploy these components in a variety of container/component environments. These components essentially use a binary protocol called Java Remote Method Protocol (RMI over IIOP) for communicating with remote objects.

Since its introduction over a decade ago, Java has grown from the status of a mere programming language to a full-fledged platform on a variety of systems and environments,[9] including devices such as PDAs, mobile phones, set-top boxes, rings, cards, chips, and so on. A community called the *Java Community Process* (*JCP*) now governs the development of this language. Most of the industry leaders and key players in the IT field participate in shaping the development of this remarkable technology.

Extensible Markup Language

John Bosak of Sun Microsystems is credited with the revolutionary work on *Extensible Markup Language (XML)*. The idea of XML essentially emerged from the other nonexpendable markup languages such as *Generalized Markup Language (GML)* from IBM, *Standardized Generalized Markup Language (SGML)* from ISO, and *Hypertext Markup Language (HTML)* from ECRN. XML's popularity essentially stems out of its extensible capability. One of the biggest contributions of XML is its capability of interoperability.

The development of XML resulted in its adoption by a variety of industries—both vertical and horizontal. This has resulted in the creation of a large number of XML vocabularies that cater to the interoperability needs of different industries. The biggest contributions of XML for enterprise solution needs are the SOAP, WSDL, and UDDI technologies. Part II, "Service Oriented Architecture Essentials," discusses this in detail.

Web Services—XML-RPC and SOAP

Introduced by Dave Winer, XML-RPC is an RPC protocol that is text based. As the name indicates, the XML-RPC protocol enables the exchange of XML data between remote objects. The idea of transporting XML as a payload over transport protocols such as HTTP has resulted in laying the foundation of web services such as SOAP and WSDL. Initial work on XML-RPC resulted in a simple and portable way of making text-based RPC in a distributed environment. This pioneering work resulted in the opening of a new perspective in the history of middleware technologies. Further work in this direction resulted in a new message-oriented protocol called SOAP and brought the interoperability one step closer to business automation.

Arrival of Web Services and SOA

Earlier in this chapter we highlighted the Remote Procedure Call and its influence in the distributed communication technologies such as CORBA, DCOM, and J2EE. The protocols used in these technologies, IIOP, ORPC, and RMI/IIOP, respectively, are the binary protocols used for communication between remote objects over the corporate networks. This laid the foundations for a radically new protocol and resulted in the development of extensible vocabularies such as SOAP, WSDL, and UDDI. These extensible languages are referred to as *First Generation Web Services*. These languages provide fundamental level support for enterprise applications and enable them to be web service-oriented at the functional level. However, for enterprises, nonfunctional requirements take priority over functional requirements. The web services extensions that attempt to meet the nonfunctional aspects of enterprise requirements are referred to as the

Second Generation Web Services extensions, and we explore them briefly in the following sections.

First Generation Web Services

As you may recall from Chapter 1, the three pillars of web services are SOAP, WSDL, and UDDI. These technologies are advanced vocabularies of the XML and use other supportive XML vocabularies such as Namespace and *XML Schema Definition* (*XSD*). Each of these web services vocabularies address different aspects of enterprise information interchange in an interoperable manner.

SOAP

This new text-based messaging technology enables applications to exchange information in the form of messages. The messages can be interchanged in a synchronous or asynchronous manner. The design of SOAP message structure is such that the messages can be interchanged between applications through RPC invocation or through MOM technologies.

WSDL

WSDL enables description of the service through the use of a set of specialized XML elements. The service description includes the data types interchanged (this is programming language-independent), name of the service, parameters passed, transport protocol used, and so on. WSDL also enables several related services to aggregate into a service suite.

UDDI

UDDI is a specification and service that helps businesses provide a platform in such a way that the service requesters can discover service providers, zero in on appropriate partners, and enable an agreed-upon business automation. UDDI, like WSDL, uses advanced XML vocabularies to define the business and service information in an elaborate manner. As a service, UDDI registries enable the service requester to store all necessary information regarding business and service information that is suitably categorized as per industry standards.

The Second Generation Web Services

Enterprise solution requirements might be categorized into functional requirements and nonfunctional requirements. Nonfunctional requirements govern the architectural and design aspects of any enterprise solution. There are many nonfunctional requirements, and one enterprise's nonfunctional requirements list and priorities would be different from another. Some of the nonfunctional requirements that are common to most of the enterprises are

- Security
- Reliability
- Availability
- Quality of service
- Business process
- Choreography

Several web services extensions and frameworks have been proposed by various industry consortia, and there is more than one web service extension proposed by competing industry consortia. These extensions and frameworks address one or more nonfunctional enterprise requirements Although there is a general consensus among the industry consortia on some of the web service extensions, this is not the case for all web service extensions.

Some of the important web services extensions are

- WS-Security Specifications and Frameworks
- WS-Addressing Specification
- WS-Reliable Messaging Specifications
- WS-Business Process Execution Language
- WS-Choreography Definition Language
- WS-Metadata Exchange Specifications

SOA Using Web Services

We have already discussed how the arrival of XML and related technologies brought in a paradigm shift for enterprise solutions. The core web services technologies provided a sound foundation for the functional aspects of the services, its description, and invocation. The second generation web services extensions, on the other hand, brought the nonfunctional requirements into the web services fold. Together, web services technologies provide several key features and advantages that the earlier technological solutions could not. Interoperability, for example, enables a clear separation of the service interface from the execution environment. Therefore, SOA implemented using web services technologies is likely to provide a leading edge over any other technological implementation.

Using web services, it is easier to change service compositions of the enterprise application and implement the changes at a lower cost. These features help the enterprise project developers to quickly respond to the dynamic requirements of the enterprise business needs.

Benefits and Challenges with SOA

SOA with web services as an implementation route brings a host of advantages to the enterprises. This doesn't necessarily mean that service orientation of the enterprise architecture is void of any disadvantages. Some of the significant pros and cons associated with SOA are as follows:

Benefits

- Rapid integration of enterprise applications—departments and partners
- Efficient business automation
- Enhanced corporate agility
- Faster time to market for new products and services
- Reduced IT costs for the corporate long-term investment
- Improved operational efficiency of the business processes
- Better ROI

Challenges

- Identifying the need for SOA
- Significant investment in resources on rearchitecting the core IT assets
- Identifying the right kind of governance model for the enterprise
- Mind share for the right kind of professionals and stake holders
- Legacy system issues—some legacy applications cannot be service oriented

Notice here that the issues and challenges for SOA relate more to the cultural aspect of the problem than the technological or business aspects. Of course, issues such as integration of unsupported legacy systems to service orientation remain as bottlenecks to the implementation of SOA.

SOA Implementation Technologies

Web services implementation of SOA has many crucial advantages over any other implementation strategies. Presently, there are two predominant solutions that help in web services implementation of SOA: Microsoft's .NET technologies and Sun Microsystems's Java Platform Enterprise Edition[10] technologies.

Microsoft's .NET Technologies

The .NET product suite from Microsoft enables enterprises to build enterprise-class web SOAs. The .NET product suite is largely a rewrite of Windows DNA,[11] which constitutes Microsoft's previous platform constituents for developing enterprise applications. The new .NET Framework replaces these technologies and includes the web services layer.

The .NET Environment

The .NET technologies offer language independence and language interoperability. This is an interesting aspect of the .NET technology. Accordingly, a .NET component can be written, for example, partially in different programming languages and implemented as part of the web services solution. The .NET technology converts this composite language component into an intermediary neutral language called *Microsoft Intermediate Language* (*MSIL*). This MSIL[12] code is then interpreted and compiled to a native executable file.

The .NET Framework also includes a runtime environment called the *Common Language Runtime* (*CLR*). This environment is analogous to the Sun Microsystems *Java Runtime Environment* (*JRE*).

The .NET Server Services

Microsoft has packed a number of servers as part of the .NET platform called The .NET Enterprise Servers. These servers provide vital services for hosting enterprise-class applications. Some important servers included as part of the .NET Servers are SQL Server, Exchange Server, Commerce Server, Cluster Server, Host Integration Server, and BizTalk Server.

Sun Microsystems's Java Enterprise Edition Technologies

The Java Platform, Enterprise Edition (Java EE) is a progression of the Java environment to the server side of the application software paradigm. J2EE, unlike Microsoft's .NET, could be termed a defacto industry standard and has resulted in a large industry initiative called the *Java Community Process* (*JCP*). The participants of this community include the "who's who" in the IT and related industries—IBM, Oracle, Nokia, BEA, and so on. The spirit of Java as well as the other related technologies, such as Java EE, was to free the customers from the dependency of products and tools from vendors.

Java Foundation

The launching of Java as a programming language took the industry by storm in 1995. As previously indicated, the Java programming environment provided unique features that no other programming language provided: portability, platform independence, and so on. The core feature is the *Java Runtime Environment* (*JRE*) that can be made available on any hardware or operating environment. The application is developed using the Java programming language and compiled into platform-independent *bytecodes*. This bytecode can then be deployed to run on JRE that is installed on any compatible system.

Java EE is the server-side extension of Java. The applications are not just Java objects but are also appropriate server-side components. For creating web applications, components such as Java Servlets and *JavaServer Pages* (*JSP*) are used and deployed on web servers, and these web servers run on JRE. Likewise, for creating enterprise applications, components such as *Enterprise JavaBeans* (*EJB*) are developed and deployed, optionally with web applications, in application servers. Again, these application servers also run in JRE.

Web Services Using Java Enterprise Edition

The evolution of Java EE has been steady. Java EE technologies are consistently improving with each version. These improvements are essentially driven by *Java Specification Requests* (*JSR*), and once again, this is the JCP initiative. The arrival of XML and the related advanced vocabularies has resulted in immediate adoption into the Java environment. Simply put, this is because Java, as a portable programming language, and XML, as portable information, are an excellent combination for any environment. Further, the arrival of web services, in the form of SOAP, WSDL, and UDDI, has resulted in the creation of appropriate APIs.

Java EE applications can be executed on the web and on application servers. Appropriate components are developed and assembled to create enterprise applications. The Java EE servers and containers provide all the necessary "service plumbing" support for the web and application server.

Java EE architecture supports the following tiers: presentation tier, business tier, and data tier (or EAI tier). Not all of them are essential, and depending on the enterprise requirement, even one of the tiers can enable the application to be identified as a Java EE application. If the presentation tier is present, Java Servlets and JSP can be designed and deployed to create the web application. The Servlets can also be configured to be the services (or clients of) web services application. If the business tier is present, EJB can be developed and deployed as part of the enterprise application. The EJBs can be Session EJBs and Entity EJBs. Although session EJBs can handle session management, Entity EJBs address persistence activity. Alternatively, session EJBs can participate in the web services interactions. Business partners can connect with the presentation tier and business tier of J2EE applications through web services technologies.

Summary

The concept of Service Oriented Architecture is not entirely new. SOA essentially promotes the separation of the service interface and the execution environment. SOA also promotes the alignment of IT infrastructure to meet the business service requirements. Although SOA can be implemented in a number of ways, utilizing web services provides several advantages, particularly because web services bring enterprise application closer to business automation. Two of the most popular technologies for implementing SOA through web services are Microsoft's .NET and Sun Microsystem's Java Platform, Enterprise Edition.

In the next part, we devote our attention toward the building blocks of SOA technology. Three elements included here are the derivatives of the extensible markup language, namely SOAP, WSDL, and UDDI, and business process-related XML vocabularies such as BPEL and CDL. Advanced elements of web services address aspects such as security, reliability, quality of services, and so on.

Endnotes

1. Different applications are exchanging the data, while participating in business processes, regardless of hardware platform, operating environment, or programming languages underlying these applications.
2. Often pronounced as "kicks."
3. The IMS DB now supports relational database management systems such as DB2 and Universal DB.
4. Pronounced "three two seven zero" screen or terminal.
5. Most of the technologies invented/introduced by Microsoft Corporation are invariably on the Windows/Intel combination. Often this combination is referred to as Wintel: Windows and Intel.
6. The term paradigm shift was first used by Thomas Kuhn in his famous book *The Structure of Scientific Revolutions*, in 1962, to describe the process and result of a change in basic assumptions within the ruling theory of science. It has since become widely applied to many other realms of human experience and the field of information technology as well. Paradigm shift can also be defined as a significant change from one fundamental view to another. Such changes are usually accompanied by discontinuity.

7. We are essentially focused on the field of enterprise solutions here. Scope of information technology is really wide, and paradigm shifts as applied to this scope, as per the interpretations of different experts, could be different. For example, as per the essays of Bioss Sari, the following three events mark the paradigm shifts in the field of information technology:
   - Invention of the microprocessor and its impact on the computer industry
   - Paul Baran's invention of the distributed network and packet switching
   - The future of computing and the end of the silicon era
8. DARPANet and ARPANet are the two revolutionary projects sponsored by the U.S. Department of Defense. DARPANet is the origin of ARPANet project. The aim of the DARPANet project was to exchange military information among analysts, scientists, and researchers located at different geographical locations of the United States. The ARPANet project was launched by DOD sometime in the late 1960s. The network infrastructure for this project was created by the U.S. Defense Advanced Research Project Agency (ARPA). The idea of ARPANet was to set up an experimental wide area network within the United States to survive the military exigencies.
9. Java technology from Sun Microsystems was initially developed as a programming environment for devices. However, when it was launched, it was launched as a "portable" programming language. However, the language grew in several directions, including the devices.
10. Sun Microsystems has rechristened the J2EE as the Java EE. This change is not just in the name. There are fundamental changes in the way web services are created as a part of web applications or enterprise applications. These aspects are discussed in detail in Chapter 9, "Java Platform, Enterprise Edition Overview."
11. Windows DNA includes many technologies that are part of Microsoft's products today. They include Microsoft Transaction Server (MTS) and COM+, Microsoft Message Queue (MSMQ), and the Microsoft SQL Server database.
12. This IL code is language-neutral and is analogous to Sun Microsystem's Java bytecode.

INDEX

I

J

K

L

P

Q

R

S

U

V

W

X–Y–Z