Tect

Web Services and Application Frameworks (.NET and J2EE)

Gunjan Samtani
Dimple Sadhwani

Published by Tect,
29 South LaSalle St.
Suite 520
Chicago
Illinois
60603
USA

Contents

Purpose 3

What are Application Frameworks? 3

Flavors of Application Frameworks 4

- Microsoft .NET Framework 4
- Java 2 Platform, Enterprise Edition (J2EE) Framework 5

Web Services: All About Interoperability 6

- Classification of Web Services 6

Application Frameworks and Web Services 6

- Microsoft .NET 7
- Java 2 Platform, Enterprise Edition (J2EE) 8
- Differences Between J2EE and .NET Frameworks for Web Services Support 9

How to Choose an Application Framework for Web Services 10

- The Ten Most Important Deciding Factors 10
- Application Servers and Packaged Application Providers 12
- A Word of Caution 12

An Example of Application Servers and Web Services 12

Conclusion 13

Purpose

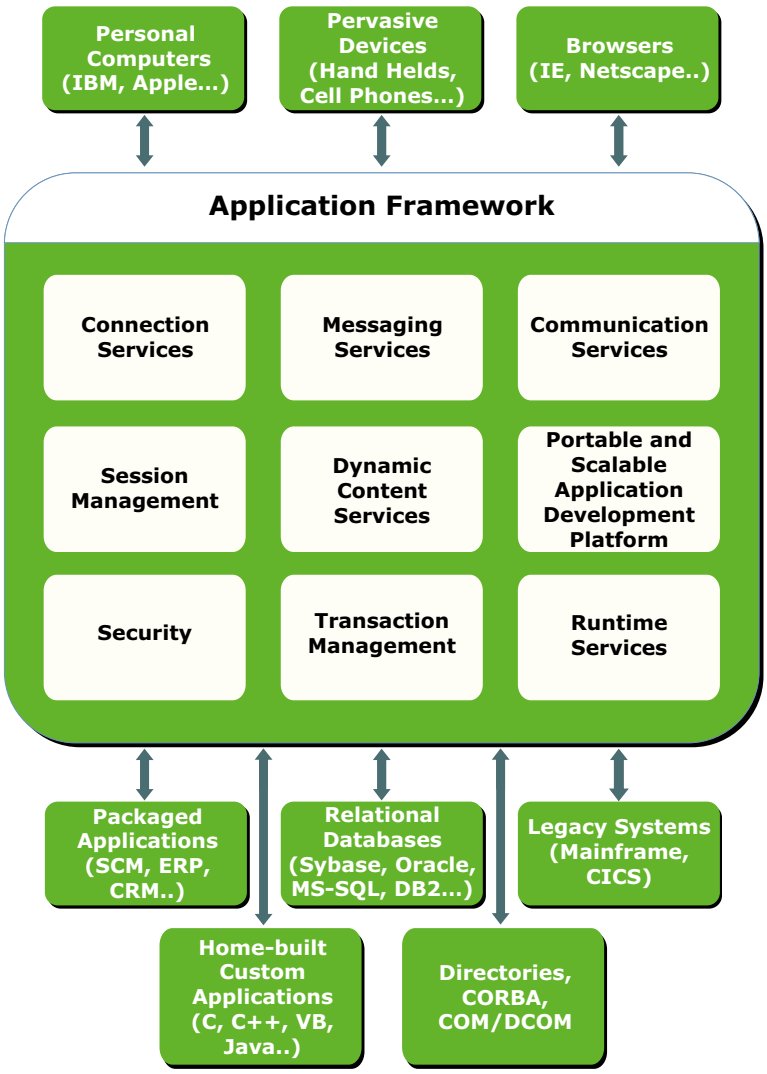
This paper discusses the fundamentals of application frameworks, the different flavors of application frameworks, the fundamentals of Web Services, and the relationship between application frameworks and Web Services. We include discussion of the current state of Web Services standards support by the .NET and J2EE frameworks, the key differences between the J2EE and .NET frameworks as far as their support for Web Services is concerned, and detailed discussion on how companies should choose between the Microsoft .NET and J2EE frameworks. We round off with a look at Web Services support by application server vendors and third party packaged applications, and an elaborate example of Web Services and application servers.

What are Application Frameworks?

Application frameworks are a holistic set of guidelines and specifications that provide platforms, tools, and programming environments for addressing the design, integration, performance, security, and reliability of distributed and multi-tiered applications. An application framework includes the presentation services, server-side processing, session management, business logic framework, application data caching, application logic, persistence, transactions, security, and logging services for applications.

Thus, an application framework provides the following:

- Transaction Management
- Scalability
- Security
- State Management
- Application Integration Services
- Administration Services
- Run-time Services
- Connection Services
- Messaging Services
- Application Development, Deployment, and Execution Platform
- Web Services
- Business Process Management Services
- Support for various graphical user interfaces including Web-browsers and Wireless devices



Application development tools and application servers are built on top application frameworks. The application frameworks aim to provide a single and unified software infrastructure (a fully integrated application environment, reaching from the need to include and integrate a wide variety of legacy applications through to the need to create and deploy Web Services) that reduces the number of enterprise software products to support, maintain and integrate.

Flavors of Application Frameworks

Application frameworks for client-server and Web-based applications can broadly be classified into two distinct fundamental physical architectures and technologies - Microsoft .NET and Java 2 Enterprise Edition (J2EE).

Let's have a brief look at these two frameworks:

Microsoft .NET Framework

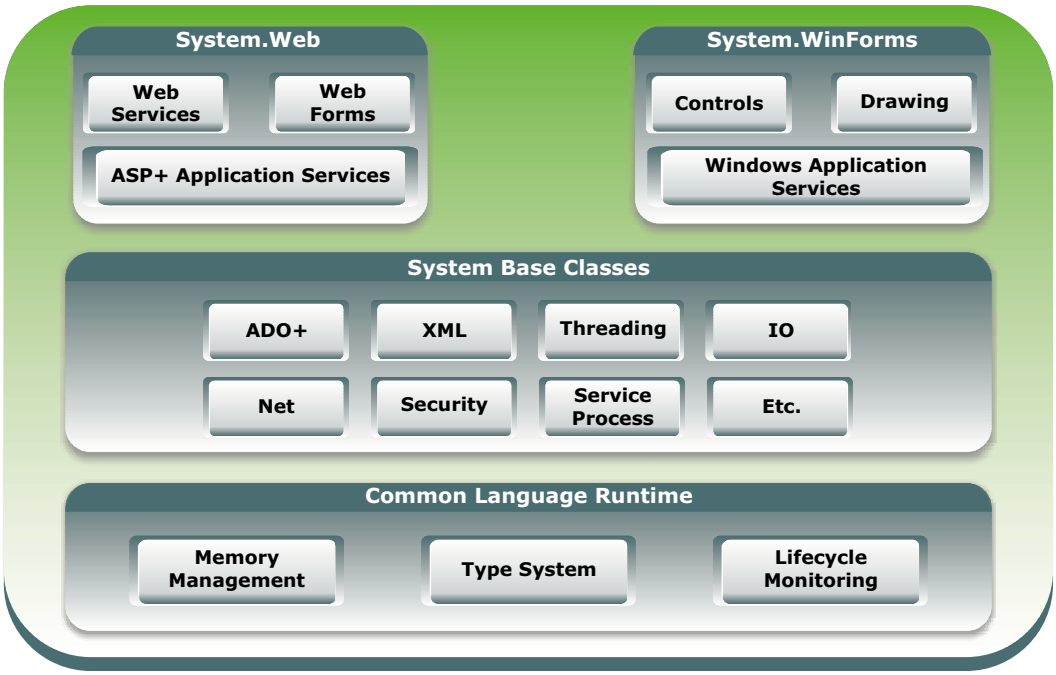
Microsoft .NET is a platform that comprises of servers, clients and services. The .NET framework includes everything from basic runtime libraries to user-interface libraries - the common language runtime (CLR), the C#, C++, VB.NET, JScript.NET languages and the .NET Framework APIs (Application Programming Interfaces). It comprises the following:

1. *.NET Platform:* This includes the tools and infrastructure to build .NET services and.NET device software.

2. *.NET Product and Services:* This includes Microsoft .NET-based enterprise servers, which provide support for the .NET framework, such as BizTalk Server 2002 and SQL Server 2000, Windows.NET, Visual Studio .NET, and Office.NET.

3. *Third-party (Vendor) .NET Services:* Third-party (vendor) services built on .NET platform.

Note: Readers are advised to get the details about Microsoft .NET Framework at Microsoft's web site: http://www.microsoft.com/net

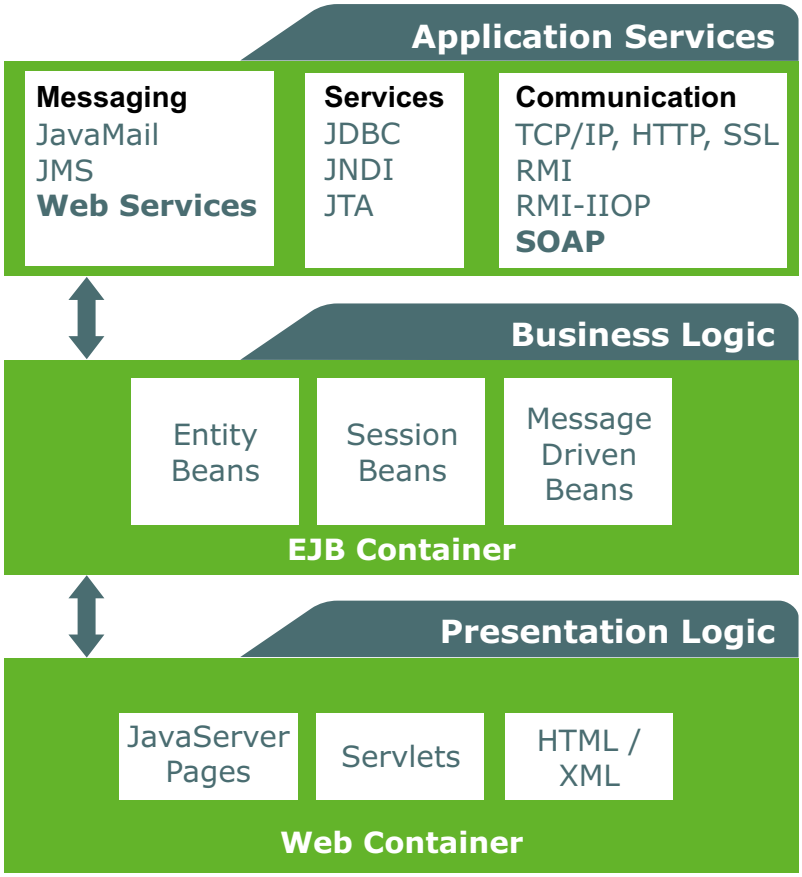


Java 2 Platform, Enterprise Edition (J2EE) Framework

J2EE is a set of specifications, which define the standard for developing multi-tier enterprise applications with Java. The J2EE platform provides a complete framework for design, development, assembly, and deployment of Java applications built on multi-tiered distributed application model. The J2EE specification defines numerous API services and multiple application programming models for developing applications and integrating them with the enterprise systems.

The latest J2EE 1.3 APIs include the following:

- Enterprise JavaBeans (EJB) 2.0
- J2EE Connector Architecture (JCA) 1.0
- JDBC 2.0 (for database connectivity)
- JavaServer Pages (JSP) 1.2
- Servlet 2.3
- Java Transaction API (JTA) 1.0.1
- Java Messaging Service (JMS) 1.0.2
- Java Name and Directory Interface (JNDI) 1.2
- Java RMI 1.0
- RMI/IIOP 1.0
- Java Authentication and Authorization Service (JAAS) 1.0
- JavaMail 1.1
- Java API for XML Parsing (JAXP) 1.1



Note: Readers are advised to get the details about J2EE Framework at Sun's web site: http:/java.sun.com/j2ee.

Web Services: All About Interoperability

Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network. They are an emerging technology that are based on service-oriented architecture (SOA) and enable new and existing applications to be integrated using of XML as data format over standard network protocols such as HTTP for transportation.

Web Services are based on "open" environment and standards, which ensures that a Web Service can be located and used, no matter where it is, what platform it runs on, or who developed it.

Classification of Web Services

Web Services can be classified as follows:

User-centric Web Services: User-centric Web Services are used to provide user personalization, interface customization, and support for multiple languages, thereby greatly enhancing user experience. They logically separate the layout (presentation) in formats such as HyperText Markup Language (HTML) from the actual data in Extensible Markup Language (XML).

*Application-centric Web Services***:** Application-centric Web Services are used to integrate enterprise and business-to-business applications. Application-centric Web Services enable companies to integrate applications and business processes without the constraints of a proprietary infrastructure, platforms and operating systems.

Both user- and application-centric Web Services, make full use of open standards, including HyperText Transfer Protocol (HTTP), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Discovery, Description, and Integration (UDDI).

Application Frameworks and Web Services

As we discussed in the above section, XML-based Web Services architecture allows programs written in different languages on different platforms to communicate with each other in a standards-based way. It is the application frameworks that pro-

vide guidelines and infrastructure for the deployment, management, and execution of Web Services. The application frameworks have to provide support for all the Web Services standards.

The tools and servers built on top of the application frameworks provide a programming model with a development and runtime environment for building both user-centric and application-centric Web Services. Further, they have to implement all the Web Services standards as supported by the underlying framework. The following diagram shows the relationship between application frameworks, Web Services servers, and Web Services standards.

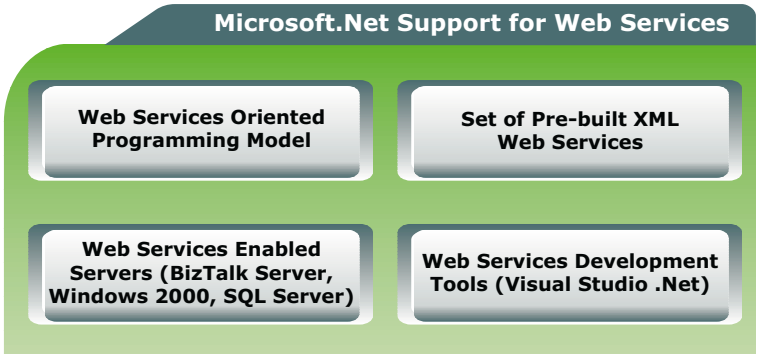
Let's discuss how the .NET and J2EE frameworks are providing support for Web Services:

Microsoft .NET

Microsoft .NET is the Microsoft XML Web Services platform. It provides built-in support for building and consuming standards-based Web Services. It enables the creation and use of XML-based applications, processes, and Web sites as Web Services. Through just a single line of code and/or setting a value of an attribute, it is possible to turn an application into a Web Service in the .NET environment. Furthermore, by default all inter-process communication is done using the SOAP standard. According to the president and CEO of Microsoft Mr. Steve Balmer, "to the .NET framework, all components can be Web Services, and Web Services are just a kind of component".

As shown in the following diagram, Microsoft .NET provides:

- A programming model to build XML Web Services and applications.
- Web Services development tools such as Visual Studio .NET to define and develop application functionality and architecture for XML Web Services and applications.
- Web Services enabled servers such as BizTalk Server 2002 and SQL Server 2000. BizTalk Server Toolkit for Microsoft .NET enables Web Services orchestration through its integration with Visual Studio .NET. Further, the SQL Server 2000 Web Services Toolkit enables the usage of Visual Studio .NET to extend the capabilities of applications built on SQL Server 2000.
- A set of pre-built user-centric XML Web Services such as Microsoft .NET My Services.



Java 2 Platform, Enterprise Edition (J2EE)

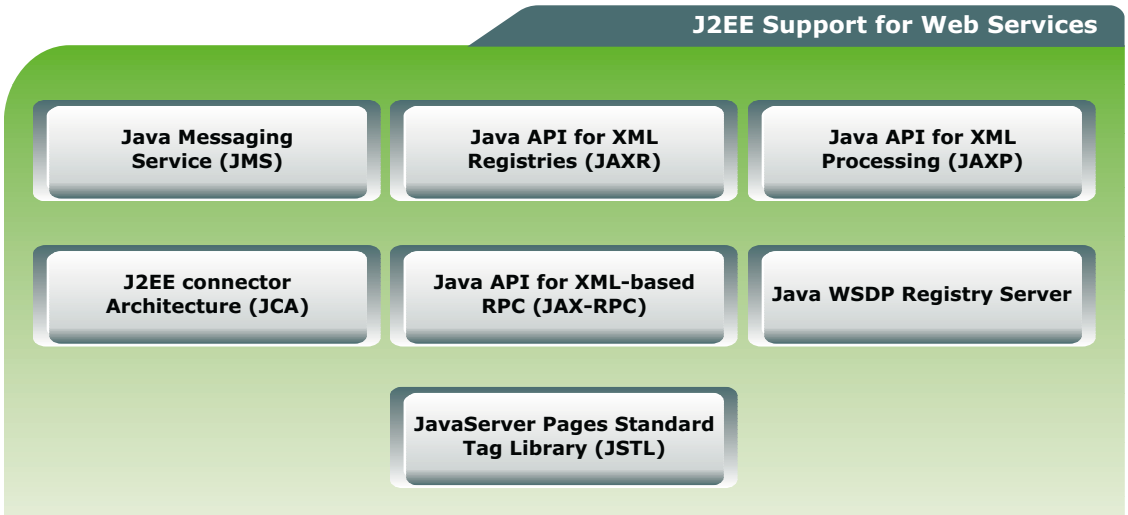
The new APIs released by Sun, as part of J2EE 1.3, provide a top-to-bottom, end-to-end solution for a Web Services-based architecture. J2EE 1.3 simplifies integration with new technologies for Web Services, such as Java Messaging Service (JMS), J2EE Connector Architecture (JCA), Java API for XML Messaging (JAXM), Java API for XML Processing (JAXP), Java API for XML Registries (JAXR), and Java API for XML-based RPC (JAX-RPC).

J2EE server products are already providing basic Web Services support such as accessing J2EE components using the SOAP 1.1 protocol. Furthermore, J2EE-based application servers, such as iPlanet, WebSphere, and WebLogic are also supporting the automatic generation of Web Services interfaces, including the WSDL file that describes the service, and the facilities for marshalling and un-marshalling the SOAP request to back-end EJB components.

Recently released, Sun Microsystems' Java Web Services Developer Pack (WSDP) and Java XML Pack contains:

- JAXP (Java API for XML Processing) - to support the creation, receipt and manipulation of XML data, including XML Schema
- JAX-RPC (Java API for XML-based Remote Procedure Calls) - to enable the creation of Web Services using SOAP and WSDL
- JAXM (Java API for XML Messaging) to support XML messaging via SOAP
- JAXR (Java API for XML Registries) to support access to UDDI registries

Lastly, as far as Web Services security is concerned, apart from the security package that is already a part of the Java Developers Kit 1.3, Sun has released the JSSE (Java Secure Socket Extension) API as part of the Java 2 SDK, Standard edition 1.4. This API supports data encryption, authentication on the server side (and optionally on the client side), message integrity, SSL (Secure Sockets Layer), and transport layer security across any TCP/IP connection.



Differences Between J2EE and .NET Frameworks for Web Services Support

J2EE and the Microsoft .NET frameworks both hold the promise of being the predominant Web Services framework. In this section, we compare and contrast between the two frameworks, ignoring all the hype associated with their marketing campaigns.

CRITERIA	J2EE FRAMEWORK	.NET FRAMEWORK
Fundamental Design and Support for Web Services	J2EE is supporting Web Services through a pack of APIs such as Java API for XML Messaging (JAXM), Java API for XML Processing (JAXP), Java API for XML Registries (JAXR), and Java API for XML-based RPC (JAX-RPC).	Web Services are built right into the platform and Microsoft .NET framework provides ready support for Web Services standards such as SOAP, WSDL, and UDDI.
Implementation	The implementation of Web Services in J2EE will typically be done through Enterprise JavaBeans (EJBs). You can, however, also have standalone Java applications providing Web Services implementation. It all depends on how the business processing and data logic layer of an application is designed and built.	The implementation of Web Services in the .NET framework will typically be done in NET managed components, including managed classes and COM/COM+ components.
Pricing	Expensive as compared to MS.NET, however, if a company already has J2EE-based application server platform, it makes much more sense to leverage the existing infrastructure and assets. On an average (as a ball park estimate), if J2EE-based application servers are run on UNIX platforms, it would cost a company five times more to have Web Services implemented on J2EE platform vs. .NET platform. This factor of five includes the hardware and the software cost. It is worth stressing that the actual price would also include the cost of development and maintenance.	Much cheaper as compared to J2EE-based application servers. J2EE, however, is still a better choice for industry-strength server side applications.
Portability	Java code can be ported across multiple platforms including Windows, UNIX, OS390, and AS400. Thus, Java driven Web Services may be developed on one platform, but deployed and executed on another.	.NET ties primarily to Microsoft's operating systems. The .NET framework, however, includes the **Common Language Runtime (CLR)**, which is analogous to the Java Runtime Environment (JRE). The CLR acts as an intermediary between .NET source code and the underlying hardware. The .NET code runs within the CLR. Once the CLR is ported to another platform, .NET software should run there as happily as the platform it was written on.

Tools and Servers	These companies have already started supporting Web Services creation, deployment and execution within their products The level of sophistication for it and support for Web Services standards differs from product to product.	Microsoft's cornerstone development tool (IDE - Integrated Development Environment) for Web Services is Visual Studio .NET. As of this writing, there is no doubt that Microsoft's tool Visual Studio .NET is ahead of its competition in terms of its support for Web Services. Web Services enabled servers from Microsoft include BizTalk 2002 and SQL Server 2000.
Promoting Companies	Multiple (independent) companies including IBM, BEA Systems, Oracle, HP, and Sun Microsystems. All these companies will be providing support for Web Services in their J2EE-based development tools and application servers. This is a comforting factor, as there are competing products in this technology, which also means that there is no monopoly of a single company.	There are multiple companies that have built IDEs and application servers based on J2EE framework. A majority of the tools, servers and technology are controlled by a single company - Microsoft. Although there is no question about Microsoft's stability and commitment towards Web Services technology, yet, without competition the technology promoted and offered may not be best one.
Maturity of Platform	J2EE has proven to be a robust, scalable and a mature platform over the last four years. Addition of support for Web Services is just another feature for this platform.	Although .NET inherits a lot of features from Windows DNA architecture, it is still relatively new and has to prove itself to be able to offer an enterprise-wide framework.

On a final comparison note on popularity, according to a poll (conducted in December 2001) of enterprise IT professionals run by ZDNet UK's Tech Update channel, majority implementations of Web Services will be based on Java (79%) rather than Microsoft's .NET (21%) alternative. It is, however, worth mentioning that this poll was taken before the release of the key Microsoft product that provides Web Services development tools and servers, which were launched in February 2002. It will be interesting to note what the actual position will be in December 2002, once J2EE and .NET-based products have been out in the market place for a while.

How to Choose an Application Framework for Web Services

If your company is debating, how to choose between such architecturally different frameworks for Web Services implementation, you are the not the only ones. This debate is going on at every level within IT groups in several companies - right from developers to mid-level managers to senior executives. And it is not an easy choice - if a company is not totally committed to either one of the frameworks.

The Ten Most Important Deciding Factors

Here are ten of the most important factors that should be carefully considered before making a conclusive decision on J2EE and/or Microsoft .NET as a framework for Web Services within your company:

1. What is the existing framework within your company? Is it J2EE or Microsoft/.NET or a mix of both?

2. Implementation of which framework will yield a higher return on investment (ROI)? Products built on which frame work fit within your budget?

3. Which framework fits both in the short-term and long-term IT and business strategy of your company?

4. Which framework can be easily supported within your IT infrastructure, eventually leading to lower cost of ownership?

5. What technologies are your developers' experts in?

6. Products built on which framework are evolving more rapidly and closely to the Web Services standards, which are still being defined?

7. Products built on which framework are more robust, scalable and most importantly meets your integration needs with less complexity?

8. Products built on which framework offer greater security so that you feel comfortable in using Web Services for business-to-business integration?

9. Products built on which framework offer greater flexibility for integrating third-party (vendor) services?

10. Which framework offers greater support for aggregation and personalization of user preferences?

Now that we have discussed the questions, here are their answers, in the same order. It is very important to mention at this juncture that these answers are not the ONLY correct answers and that there may be several answers to these questions; it is quite possible that the answer to these questions may not **have** a single answer. In such cases, companies have to keep their "long-term" goals in mind and make decisions. The decision may very well be to implement both frameworks within the company, and that may be perfectly right for a specific organization. After all, Web Services address connectivity needs by enabling open interaction between systems implemented on disparate platforms such as Microsoft's .NET and Java 2 Enterprise Edition (J2EE).

1. Evaluate the existing infrastructure (applications, development tools, and application servers) within your company. If the answer is J2EE, then use J2EE to implement Web Services within the firm. If the answer is Microsoft-based technology, then use .NET. Try to make the best use of your existing infrastructure as much as possible.

   If there is a mix of both the platforms within your company, then decide on a project-by-project basis. In some cases J2EE would make more sense and in others .NET.

2. Try to apply the following formula when deciding which platform to use for Web Services implementation:

   Increased Revenue + Decreased Cost + Improved Efficiency = Higher Profitability

   Obviously, you have to consider your budget as well when deciding between the two frameworks. As mentioned in the comparison table between J2EE and .NET frameworks, J2EE is much more expensive as a new proposition when compared to .NET.

3. There is no definite answer to this question. You have to do the homework, yourself, in evaluating the short-term and long-term goals of your company. Choose a platform that fits them. See also answer seven for more details.

4. The cost of ownership is defined as follows:

   Cost of Ownership = Cost to implement + Cost to maintain

   The framework that has a lower cost of ownership, however, may not be suitable for your company - based on the existing infrastructure, goals, etc.

5. This is one of the most important of these decision factors. If the developers in-house are experts in Microsoft-based technology, introducing and implementing J2EE is a very tough call. No matter how good the technology is it is its correct usage that makes it useful for a company.

6. After the Visual Studio .NET and .NET-based enterprise server release in February '2002, Microsoft has indeed taken a lead over any of the J2EE framework based products, such as BEA's WebLogic and IBM's WebSphere, as far as support for Web Services in concerned. But this may very well be a short-term lead as several J2EE product vendors (BEA, IBM) have already or are about to release their version of products that would provide complete support for Web Services and their standards.

7. As far as scalability and robustness are concerned, there is no question that J2EE beats Microsoft technology hands down. But if you a small-to-medium size organization where these are secondary issues, then Microsoft's .NET may fit your organizational needs. The complexity of development, deployment, and maintenance is much less in Microsoft technology based products. Visual Studio .NET is a marvelous example of how Microsoft eases the development work required for Web Services.

8. As of now, both the frameworks appear to be at par as far as their security features are concerned. The role-based security model of .NET may have an edge over J2EE, as it will be widely used in Web Services.

9. As far as open architecture goes, J2EE is far superior to .NET. But if this is a deciding factor for your company or not is something that can only be determined by the need for using third-party vendor services within your company. One point worth mentioning here is that all the major packaged application providers have announced supporting both J2EE and .NET. So, check with your packaged application vendor companies as well.

10. Microsoft .NET offers a much wider support for personalization of user preferences through .NET My Services.

An important piece of advice: The answers to these questions should be sought both from top-to-bottom (senior management to developers) and bottom-to-top (developers to senior management) within an IT organization.

Application Servers and Packaged Application Providers

Most of the Application server vendors have already started providing at least partial support for Web Services and their standards. Based on the industry trend and press releases by these vendors, it appears that full and complete support for Web Services by all the major application server vendors is now a matter of dotting the i's and crossing the t's. Thus, the probability that you will be able to utilize your current EAI and B2B integration infrastructure in deploying and using Web Services is very high.

Furthermore, all the major packaged application software vendors are racing to embrace Web Services. These include the enterprise resource planning (ERP), the supply chain management (SCM) and the customer relationship management (CRM) software providers. The trend among these packaged application providers is to stay on the sidelines of this emerging battle between J2EE and .NET - which by the way is the correct approach too. They are doing so by supporting both these frameworks. For example, for SAP - the biggest ERP vendor - has recently announced that their implementation of Web Services would include support for both J2EE and .NET. SAP's Web Application Server (which is being touted as the next generation Web Application Development Platform) would allow application components to be provided as Web Services.

A Word of Caution

Everything mentioned above is good, but companies are looking for answers to where do the support for Web Services by different application server vendors stand today? The fact of the matter is that Web Services standards don't yet define security, operational management, workflow, business rules, transactional integrity and other elements necessary for an enterprise-ready computing platform.

An Example of Application Servers and Web Services

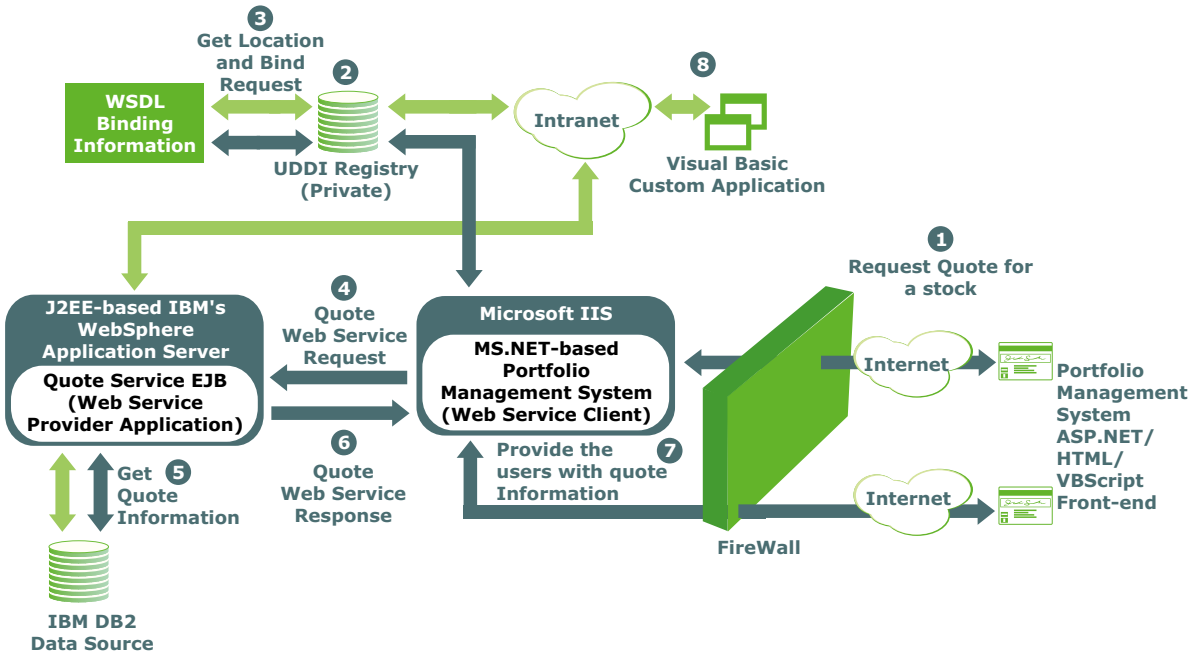
In this example, the retail clients and in-house clients of a financial company use the portfolio management portal to monitor their investments. The front-end of the portal is built using Microsoft technology (Active Server Pages .NET (ASP.NET), Internet Information Services (IIS) Web Server, VB Script, etc.). One of the features provided within the portal application is quote information. Using this feature, the clients can retrieve real-time quotation for any stock. When a client requests a quote for any stock, the request is sent from the browser to the Web Server.

As may potentially happen within any mid-to-large size company, we are assuming that the quote service is provided to multiple clients as a Web Service by a middleware application within the company, with our portfolio management portal being just one of those clients. Another client, as shown in the figure, is a VB application.

The information about Web Services offered by this middleware application (which may be published by some other group within the company) is obtained from the private internal UDDI registry and invoked over the intranet. The implementation of the business methods exposed by the Web Service is provided by EJBs contained in another application server.

This is a typical example of .NET-to-J2EE application server integration using Web Services. The binding information for frequently used Web Services, such as those for requesting quotes, can be cached by the client application, to avoid the

resource intensive and time consuming dynamic binding. In this example, Web Services loosely integrates Microsoft technology-based portfolio management application with the J2EE-based middleware application that interfaces with the Mainframe to receive the quote.



The sequence of steps is as follows:

1. The user requests quotes for a specific company on an ASP.NET/VBScript/HTML front-end that is passed over to the portfolio management portal running within Microsoft IIS. Here for the sake of simplicity it is assumed that the user has already successfully logged into the application and has a valid session established.

2. The .NET-based portal application gets information about Web Services made available by the J2EE-based middleware application by performing a look up in the private UDDI registry.

3. The location of and WSDL binding information for Web Services is sent to the portal application as a SOAP-based message.

4. The portal application invokes the Web Service published by the middleware application, passing a stock symbol as part of a SOAP-based message.

5. The actual implementation of the Web Service is provided by EJBs running within a J2EE-based application server. The EJBs use the JDBC API to get information from the data source, which in this case is IBM's DB2.

6. The EJBs send the Web Services response to the portal application as a SOAP-based message.

7. The response is formatted in HTML (using XSLT) and sent back to the browser-based client application.

8. Another VB custom application within the company intranet invokes the same Web Service, thereby being another client of the quote Web Service. The communication happens based on SOAP.

Conclusion

Application frameworks provide a platform for design, development, assembly, deployment, execution, and monitoring of applications built on multi-tiered distributed application model. Leading application frameworks (Microsoft .NET and J2EE) are providing full support for Web Services and competing to become the framework of choice for Web Services initiatives within companies. This enterprise war between J2EE and .NET is bound to go on for a few years. Ultimately, it is the Web Services technology that will be the winner, as these frameworks will push each-other to outbid their respective competitor's support for Web Services, leading to faster adoption of Web Services standards, creation of efficient tools and robust and scalable application servers and, last but definately not least, cheaper costs of development tools and servers.

About the Authors

Gunjan Samtani is Divisional Vice President, Information Technology at UBS PaineWebber, one of the world's leading financial services firms. He has several years of experience in the management, design, architecture, and implementation of large-scale EAI and B2B integration projects. He is the primary author of the upcoming book titled "B2B Integration - A practical guide to collaborative e-commerce". He has presented research papers at several national and international conferences and is the author of more than 100 articles and research publications in field of finance and technology. His email address is gsamtani@ubspw.com.

Dimple Sadhwani is Senior Software Engineer at Island ECN based in New York. She has many years of experience working for financial and telecommunication companies on large scale trading systems, CRM applications, Internet/Intranet portals, and client/server applications. She is co-author of the book "B2B Integration - A practical guide to collaborative e-commerce". She has also authored several articles in the field of Web Services. Her email address is dsadhwani@island.com.