# Java 2 Enterprise Edition (J2EE)

# versus

# The .NET Platform

by

Felicia Cheng

Jared Zheng

Jonathan Card

Peng Li

Xiao He

Nov 26, 2004

# Introduction

Web Services are software components that can be housed in an application on a local network or the Internet, and are accessible by applications that are connected to the above. One of the big promises of web services is universal interoperability because of the use of platform and language independent protocols. Prior to the advent of web services, enterprise application integration was very difficult due to differences in programming languages and middleware used within organizations. With web services, any application can be integrated so long as it is Internet-enabled.

In general, Web services use the following technologies:

- Service Description - the provider defines the web services in WSDL (the Web Services Description Language).
- Service Publishing - the provider registers its services in UDDI registries.
- Service Discovery - a user finds the service by queuing  a UDDI registry
- Service Binding - the user's application invokes the service using SOAP

And the foundation of web services is XML messaging over standard web protocols such as HTTP:

- Web server receives a request formatted in XML from an application
- Web server performs a task
- Web server returns a XML-formatted response

By following these standards of web services, you can integrate two businesses, departments, or applications quickly and cost-effectively.

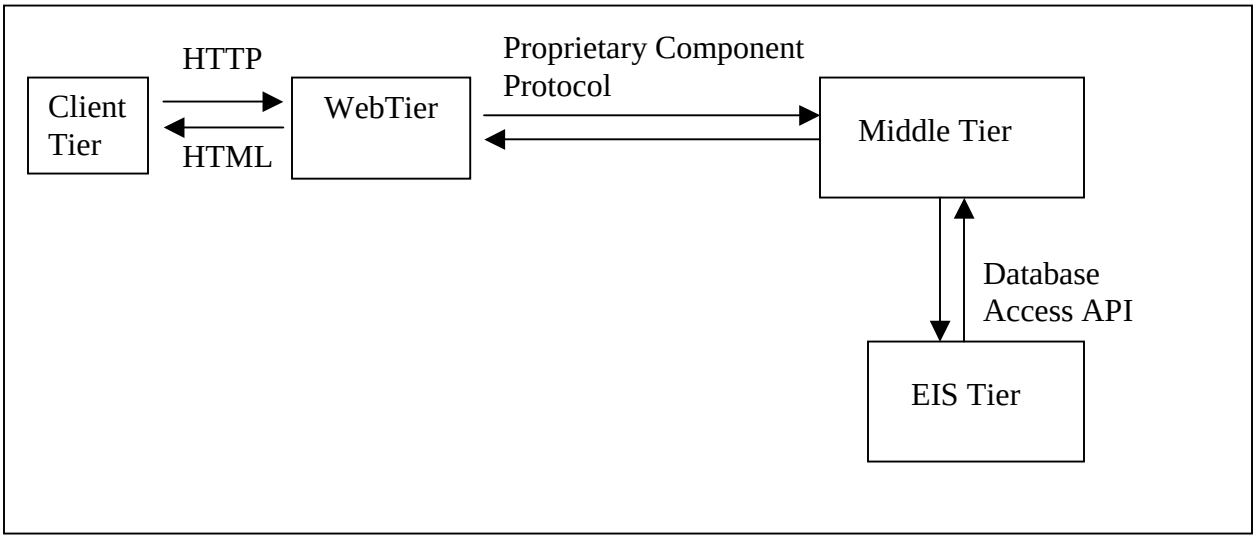
Today, there are two technical visions for building Web services. One of these is the Microsoft vision and goes by the overall name of the .NET platform. The other is the Sun vision and goes by the overall name of Java 2 Enterprise Edition (J2EE).

In this report we will focus on compare the overall J2EE vision as defined by Sun (the owner of the specification) to the overall .NET platform vision as defined by Microsoft.

## A Typical Enterprise Application Architecture

Today's web-based enterprise application architectures are typically based on four tiers architecture, as shown in Figure 1.

### Figure 1. Multi-Tier Application Architecture




The *client tier* is responsible for working with clients.

The *web tier* accepts an HTTP requests from a web browser and returns an HTML page that the browser can then display.

The *middle tier* is where much of the business logic is implemented, so it is often called the *business tier*. In J2EE, this middle tier infrastructure in called Enterprise JavaBeans(EJB). In the .NET framework, it is called .NET management component.

The web tier communicates with the middle tier through a method transport protocol. For the .NET platform, this protocol is usually either DCOM or SOAP. For J2EE, it is RMI/IIOP.

The fourth tier is the EIS *tier*, where actual data is stored. Communication between the middle tier and the EIS tier use a specific API, ADO.NET for the .NET framework and JDBC for J2EE.

# The .NET Platform Architecture

NET is a set of software technologies for connecting information, people, systems, and devices. This new generation of technology is based on Web services.

The overall .NET platform architecture can be divided into four main areas:

- .NET Infrastructure and Tools: The infrastructure and tools to build and operate application systems, including .NET Framework, Visual Studio.NET, and .NET Enterprise Servers,
- .NET Foundation Services: The .NET services are a set of information sharing services for the Internet, such as Passport.NET (for user authentication) and services for file storage, user preference management, calendar management. These services will be offered by both Microsoft and Microsoft partners.
- .NET User Experience: This will be a broader, more adaptive user experience, where information is delivered in a variety of ways on a variety of different devices.
- .NET Devices: This device software enables a new breed of smart Internet devices that can leverage Web services.

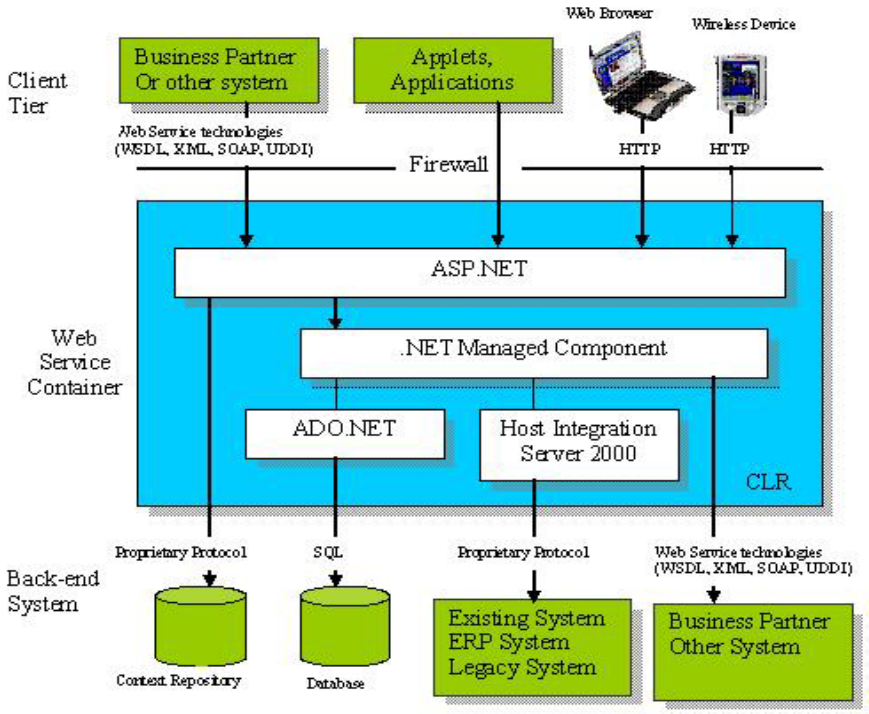
## .NET and Web Services

Microsoft .NET is designed for Web services. It is largely a rewrite of Windows DNA, which was Microsoft's previous platform for developing enterprise applications. Web Service is implemented inside .NET framework. It can be implemented by .NET managed component, such as Managed Class, COM/COM++ (Reference Figure-1, .NET Web Service model). Also, .NET uses UDDI registry service as its internal mechanism of discovering and publishing Web services

.NET has the following characters:

- Independent of  developing language
- Provide enterprise level extendibility and reliability
- Integrated security
- Easy to implement
- Distributed operation
- Support standard (SOAP, UDDI, WSDL)
- Reliable operation and management
- Provide strong developing and debugging tools

The Web services model of .NET is showed in Figure-1:




The layers are the visible faces of our application.

**Clients**, **Web browser and wireless devices**: connect to ASP.NET, which render user interfaces in HTML, XHTML, or WML.

**.NET application:** is hosted in a container, which provides qualities of service for enterprise applications, such as transactions, security, and message services.

**The business (middle) layer:** This layer of .NET applications is built using .NET managed components. It connects database using ADO.NET, or existing systems using Microsoft Host Integration Server. It can also connect to business partners using web services technologies (SOAP, UDDI, WSDL,) through the JAX APIs

## .NET Framework

The most important part of the .NET platform is the .NET Framework; it provides basic functions for developing and running .NET applications, including programming developing library, compiler, components configuration, running management and more. It includes the component-oriented middle-tier infrastructure (COM+), the Common

Language Runtime (CLR) environment, a just-in-time compiler, and a set of operating system libraries packaged using the .NET component model.

## Common Language Runtime

CLR is the core of .NET framework. The runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. CLR makes it easy to design components and applications whose objects interact across languages. Objects written in different languages can communicate with each other, and their behaviors can be tightly integrated. Common Language Runtime includes the following main functions:

- Just-in-time compiler
- Language independent
- Garbage collection (Memory management)
- Thread management
- Error handler

CLR is very similar to Java JVM. The CLR provides the following benefits for application developers:

- Vastly simplified development
- Seamless integration of code written in various languages
- Evidence-based security with code identity
- Assembly-based deployment that eliminates DLL Hell
- Side-by-side versioning of reusable components
- Code reuse through implementation inheritance
- Automatic object lifetime management
- Self describing objects

## Class Libraries

.NET framework provides a common, consistent development interface across all languages supported by it. It provides almost all common classes required for application developing. Through .NET framework class libraries, programmer can focus on developing business logic instead of details of programming. The general class libraries including:

- Base classes: provide standard functionality such as input/output, string manipulation, security management, network communications, thread management, text management, and user interface design features.
- ADO.NET classes: enable developers to interact with data accessed in the form of XML through the OLE DB, ODBC, Oracle, and SQL Server interfaces.
- XML classes: enable XML manipulation, searching, and translations.
- ASP.NET classes: support the development of Web-based applications and Web services.

- Windows Forms classes: support the development of desktop-based smart client applications.

## COM+

COM+ is the name of the COM-based services and technologies first released in Windows2000. COM+ brought together the technology of COM components and the application host of Microsoft Transaction Server (MTS). COM+ automatically handles difficult programming tasks such as resource pooling, disconnected applications, event publication and subscription and distributed transactions.

# The J2EE Architecture

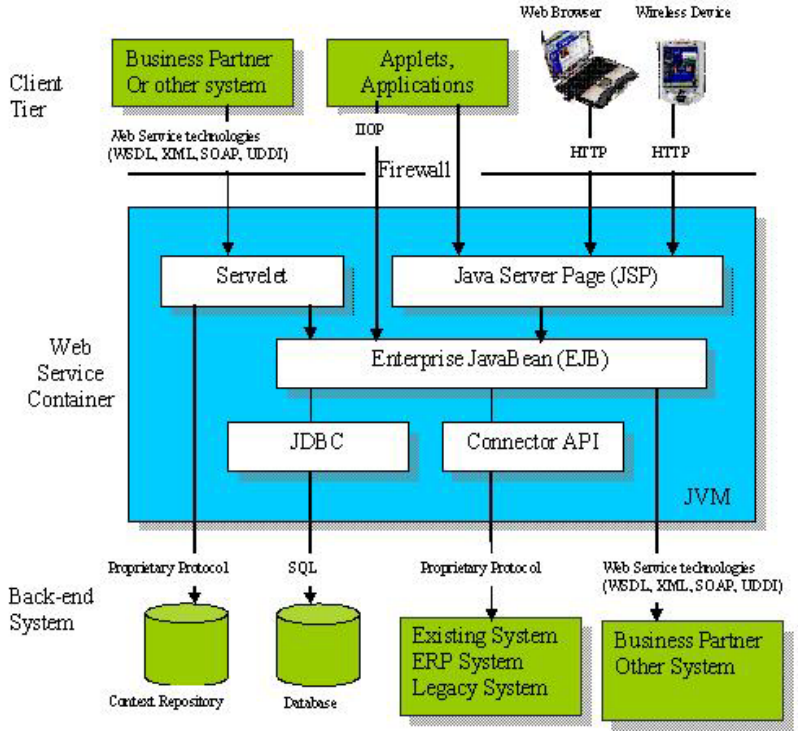
The Java 2 Platform, Enterprise Edition (J2EE) is an open and standard based platform for developing, deploying and managing multi-tier, web-enabled, and component-based enterprise applications.

It is important for you to realize that J2EE is a standard, not a product. It was first released in 1998, and is the result of a large industry initiative (including IBM, Oracle, BEA) led by Sun Microsystems.

## J2EE and Web Services

J2EE is not born for web services; it has historically been an architecture for building server-side applications in Java programming language. J2EE has recently been extended to include support for building XML-based web services as well. To support Web Service, J2EE includes a group of API, including JAXM, JAXP, JAXR, and JAX-RPC.

J2EE web services model in Figure-2




The layers are the visible faces of our application.

**Web browser and wireless devices** connect to JSP by using HTML, XHTML or WML

**Clients**: such as applets or applications, they can connect to the JSP, or directly connect to EJB layer through the IIOP (Internet Inter-ORB Protocol).

**J2EE application:** is hosted in a container, which provides qualities of service for enterprise applications, such as transactions, security, and persistence services.

**The business (middle) layer:** In large-scale J2EE applications, business logic is built using EJB components. It connects database using JDBC or SQL/J, or existing systems using the JCA. It can also connect to business partners using web services technologies (SOAP, UDDI, WSDL,) through the JAX APIs

## J2EE 1.4 Contents

- J2SE: Java 2, Standard Edition
- JAX-RPC: Java API for XML-based RPC
- JSP: JavaServer Pages Technology
- Servlet
- EJB: Enterprise JavaBeans
- JMX: Java Management Extensions.
- JMS: Java Message Service
- JavaMail
- JCA: J2EE Connector Architecture
- JTA: Java Transactions APIs
- JACC: Java Authorization Contract for Containers

## J2EE Platform Technologies

The goal of the Java 2 Platform, Enterprise Edition (J2EE platform) is to define a standard of functionality that helps meet those challenges and thus increases the competitiveness of enterprises in the information economy. To achieve this goal, J2EE platform specifies technologies to support enterprise applications. These technologies fall into three categories: component, service, and communication

### Component Technology

A *component* is an application-level software unit. There have two categories of component, one is client side component—including Applets and application clients, the other is server side component—including EJB and Web Components

#### Applets and Application Clients

*Applets* and *application clients* are client components that execute in their own Java virtual machine

#### Web Components

A *Web component* is a software entity that provides a response to a request. A Web component typically generates the user interface for a Web-based application.

- Servlets: a servlet is a program that extends the functionality of a Web server. Servlets receive a request from a client, dynamically generate the response, and then send the response containing an HTML or XML document to the client
- JavaServer Pages Technology: JSP technology provides an extensible way to generate dynamic content for a Web client. A JSP page is a text-based document that describes how to process a request to create a response.

**Enterprise JavaBeans component**

Enterprise JavaBeans (EJB) component is the core of J2EE, and is the main reason why J2EE gets widely support from the industry. One main goal of J2EE is to simplify the developing the enterprise applications, and let programmers focus on business logic development. EJB components are scalable, transactional, and secure. EJB defines three types of beans:

- Entity bean components: are the object representations of data maintained in a data store. These components manage persistent data, either managing the persistence on their own or depending on the container to manage their persistence.
- Session bean components: usually provide services to a single client, and their state cannot be recovered after a server crash.
- Message-driven beans components: enable clients to access business logic contained within enterprise bean components in an asynchronous manner.

## Services

The J2EE platform standard, to ensure that components are portable, requires a conforming platform provider to make certain services available. The platform's services are:

- Naming service: allows symbolic access to EIS resources and components within a naming environment.
- Deployment service: a deployment service allows changes to component behavior at deployment without the need to change a component's source code.
- Transaction service: frees the component developer from having to include code to handle such transactional issue as mutil-user access .
- Security service: ensure that components and resources are accessed by only those authorized for access.

## Communication

Communication technologies bring the platform's components and services together, making the J2EE platform an integrated, standard platform for developing portable, interoperable enterprise applications and Web services

- Internet protocols: enable communication between components and between components and their clients. Such protocols as TCP/IP, HTTP, SSL and so forth.
- Remote Method Invocation (RMI) protocols: J2EE platform supports the Java RMI, and uses IIOP to turn local method invocations into remote method invocations
- Messaging technologies: enable asynchronous communications
- Web service technologies: support Web service standards such as SOAP, UDDI and WSDL

# J2EE versus Microsoft .NET

## Similarities between J2EE and .NET platform

There are lots of common characters and futures in J2EE and .NET platform even thought they come from different company. A list of equivalencies between J2EE and .NET platform is given in Figure-3.

Figure-3

| Feature | J2EE | .NET |
|---|---|---|
| Interpreter | JVM | CLR |
| Web GUI | JSP | ASP.NET |
| Middle-Tier Components | EJB | .NET Managed Component |
| Database access | JDBC, SQL/J | ADO.NET |
| Message Service | JMS | Message Queue |
| Remote Invocation | RMI-IIOP | .Net Remoting |
| Transactions | JTA | COM+/DTC |
| SOAP,UDDI,WSDS | YES | YES |

Except a lot of similar features, there are also lots of similar ideas, purpose and concepts in J2EE and .NET.

- They both provide flexible Web service to enterprise application with higher reliability, availability, scalability and security.
- They both provide a run time environment, JVM in J2EE, CLR in .NET
- The both use container and component technologies in designing and implementing their solution
- They both support multi-tier design architecture

# Differences between J2EE and .NET Platform

## JVM versus CLR

Both J2EE and .NET provide an independent run time environment to application systems. It is JVM in J2EE and CLR in .NET platform. A list of common features of JVM and CLR is given in figure-4.

JVM is designed for platform independence and each OS/device has separate JVM. In theory, JVM bytecode is language-neutral, however in practice, this bytecode is only used with Java now. Other languages can be bridged into a J2EE solution through web services, CORBA, JNI (Java Native Interface) or the JCA.

CLR is designed for language independence. It supports all major languages except Java. Microsoft also introduced a new C# language which is equivalent to Java and is also available as a programming language within the Visual Studio.NET environment. All languages supported by the CLR are translated to MSIL/IL (Microsoft Intermediate Language) and managed by CLR.

Figure-4:

| | JVM | CLR |
|---|---|---|
| Managed execution environment | X | X |
| Garbage Collection | X | X |
| Metadata and Bytecode | X | X |
| Platform-abstraction class library | X | X |
| Runtime-level security | X | X |
| Multi-language support | ? | X |
| Multi-platform support | X | ? |

## Portability

The portability refers to the ability to move a code base from one operating system to another without having to change the code itself. J2EE has absolute advantage in this field because the JRE or JVM, on which J2EE is based, is available on any platform. Another reason is that J2EE is a standard, and so it supports a variety of implementation, such as IBM, BEA, and Sun. To ensure the portability of J2EE, Sun has built a J2EE compatibility test suite. Any companies J2EE solution pass the test will get a license from Sun.

In contrast, .NET only runs on Windows, its supported hardware, and the .NET environment. Today there some projects, such as “Mono”, try to build Linux and cross-platform applications based on ECMA 334/335, which is the specifications for C# and a subset of the .NET framework (CLI-common language infrastructure) submitted by

Microsoft. The purpose of these projects is trying to improve the portability of .NET. But without Microsoft's support, how many businesses will take this solution?

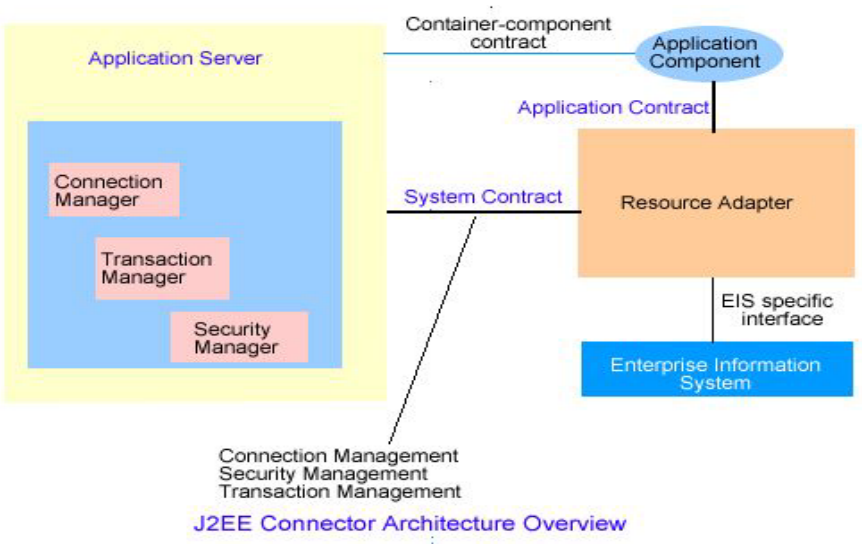
## Support Existing Systems

Many large companies have existing applications written in variety of languages, such as COBOL, C++, SAP, and Siebel. This kind of legacy integration is one of the most challenging tasks to overcome when building web service

J2EE offers several ways to achieve legacy integration:

- Using JMS to integrate with existing messaging systems
- Using Web services to integrate with any system
- Using CORBA for interfacing with code written in other languages
- JNI for loading native libraries and calling locally

The most important part of the J2EE vision for integration is the JCA (J2EE Connector Architecture). Figure-5 illustrates the JCA :




JCA defines a standard architecture for connecting the J2EE platform to EIS (enterprise information system). JCA enables an EIS vendor to provide a standard resource adapter to its EIS. Resource adapter provides connectivity between the EIS, the application server, and the enterprise application.

.NET offers the Host Integration Server 2000 for bridging to data and applications on mainframe legacy systems, BizTalk Server 2002 to build XML-based business process across applications and organizations

## Tools and languages support

On .NET platform, Visual Studio.NET is the most powerful development tools for .NET developing. It provides an integrated developing environment, an easy use GUI interface and lots of developing components.

Sun does not provide strong developing tools. Forte is the only one provided by Sun with its SUN.ONE solution. Other IDEs for Java development include IBM's VisualAge for Java, and Borland's JBuilder. Also, numerous 3rd party tools and open source-code products are available.

Java is the only language support on J2EE platform. Other languages can be bridged into J2EE platform through CORBA, web services, JNI, or JCA. However, these languages can't be mixed with Java Code.

By contrast, .NET supports any language that satisfied the CLS standard (Common Language Specification) due to the new CLR. The multiple language support by CLR is clearly a feature advantage that .NET has over J2EE.

## Performance

In order to compare the performance between J2EE solution and .NET solution, we need to create a fair testing environment which needs to work for both J2EE and .NET. J2EE is a specification, which is designed to run on any platform, while .NET is a product, which design is focus on Windows environment. .NET has already been optimized in Windows environment, where J2EE does not. Since we cannot run .NET on UNIX environment, it is impossible to create a 100% fair testing environment to compare them.

To have a better understanding of the performance, we can reference the "Pet Store" war between Sun and Microsoft. The Java Pet Store is provided by Sun to demonstrate how to use J2EE to develop flexible, scalable, and cross-platform enterprise applications. When .NET was released, Microsoft also provided a Pet Store based on .NET solution, and claimed that .NET has the better performance. The following article, http://www.onjava.com/pub/a/onjava/2001/11/28/catfight.html, provides an analysis why Microsoft's solution is faster than Sun's solution, which shows Microsoft's claim is not true. The main idea is that Sun's Pet Store is designed based on MVC design pattern, and support any platform and database system, while Microsoft's solution does not use this design pattern and just support Windows platform and SQL Database. For more information about Java Pet Store, reference http://java.sun.com/developer/releases/petstore. For Microsoft's Pet Store, reference: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/bdasamppet3.asp.

Today, the "Pet Store" war between Sun and Microsoft is still continuous. The company Middleware provides the test result, http://www.gotdotnet.com/team/compare/Middleware30.pdf, which we can reference.

## Conclusions

Compared to J2EE and Microsoft .NET, there has no absolute advantage of one over another one. They have their pros and cons; have lots of common characters, and have lots of different.

Sun's J2EE vision is based on a family of specifications that can be implemented by different vendors. It is open, and any company can license and implement the technology.

One of J2EE's major disadvantages is that Java is the only choice of the programming language. And one of J2EE's major advantages is that J2EE offer operating system portability.

Microsoft's .NET platform vision is a family of products rather than specifications. The major disadvantage of this approach is that if is limited to the Windows platform. The major advantage of .NET is that it supports multiple programming languages, and for small and medium applications, .NET is cheaper than J2EE solutions.

## Reference

http://www.microsoft.com/net/

http://java.sun.com/blueprints/guidelines/designing_webservices/html/

http://www.onjava.com/pub/a/onjava/2001/11/28/catfight.html

http://java.sun.com/j2ee/connector/reference/industrysupport/index.html