COMPUTERWORLD

by Byron Taylor Estes and Oriel Maxime, Blackwell Consulting Services

J2EE vs .Net: The choice depends on your needs

Opinion

Aug 19, 2003 • 7 mins

Enterprise Applications Enterprise Architecture

You'd have to be living in a vacuum not to be aware of the debate that rages between proponents of the two dominant development platform contenders: Java 2 Enterprise Edition and .Net. Many articles have been written declaring one or the other the winner based upon performance, scalability, features, portability, vendor independence, ease of use, lines of code, robust security, productivity tools and enterprise connectivity, just to name a few of the hot issues.

The tale of the tape is long and confusing, but does it really get at the heart of the issue?

You Say Potato, I Say …

Much of the debate has sought to differentiate the two in order to convince customers that one is the clear and obvious choice. The most obvious conclusion seems to be that the architects are most skilled at conceptually copying and enhancing the best of one another's platforms. As a result, J2EE and the .Net Framework are far more similar than many people realize and will probably become even more so in the future.

Sun Microsystems started things off when it made the Java language syntactically similar to C/C++, but improved it by making it object-oriented from the ground up, by adding its own features and by employing common design patterns in the Java Development Kit (JDK).

Not to be outdone, Microsoft made its own improvements with .Net and the Common Language Runtime to enable code that's written in many languages but deploys as if written in one. Microsoft embraced object-oriented programming instead of using a pseudo-object-oriented notation. Its most important language, C#, is amazingly similar to Java, and the .Net Framework employs many of the same design patterns embraced in the JDK.

The Stacks and Just the Stacks, Ma'am

We can look at each as a "stack" of common services that each platform provides to applications that are built with or run on the platform. There are plenty of articles out there comparing and contrasting the stacks of .Net and J2EE. Here's a list—not necessarily an exhaustive one—that begins to make the point that these technologies are growing together conceptually and support a similar set of services, albeit implemented differently.

Stack Function	.NET	J2EE
Relational Database Access	ADO.NET	JDBC
Web Client	ASP.NET	Java Server Pages (JSP) and Servlets
Standalone Client	Windows Forms	AWT/Swing
Distributed Components	.NET Remoting	RMI/IDL
XML	System.Xml and .NET in general is built around XML.	JAX Pack (JAXM, JAXR, JAXB, JAXP)
Messaging	Microsoft Message Queuing (MSMQ)	Java Messaging Service (JMS)
Web Services Support	Built directly into .NET and Visual Studio	Java Web Services Developer Pack (JWSDP) as well as vendor specific tools.
Enterprise Components/ Transactions	COM+	Enterprise Java Beans (EJB)
Integration	Host Integration Server	J2EE Connector

Integration	Host Integration Server, BizTalk Server	J2EE Connector Architecture
Component Registration	Active Directory	Java Naming and Directory Interface JNDI

It's getting harder, not easier, to pick a clear winner, because J2EE and .Net are so similar. Consider the cola wars—Coke or Pepsi. As fundamental products, they're not that different. Both are colas, with carbonation, caffeine and celebrity endorsements.

With J2EE and .Net, selection may be based less upon intrinsic merits of the platforms and more on your existing environment (e.g. resources, investments) and personal preference or style.

Don't blame Sun or Microsoft—they are companies in a free-enterprise system. They want market share and need to provide their shareholders with a return on their investment, but there's no need to get caught up in the hype or allow these superficial debates to cloud your judgment on the more important issues that we all face.

At this point, many of you are probably asking yourself, "If my selection of J2EE or .Net isn't as critical as I thought, what is?"

From the Victrola to the MP3

Replacing your favorite music every time the industry changes formats is one thing, but can your company really afford to rewrite every application each time technology changes? J2EE and .Net are hot right now, but they will eventually be replaced by the platform du jour. It's not "if," it's when. The only real question is whether it will have already emerged by the time this article is published.

Byron Taylor Estes is a senior consultant at **Blackwell Consulting Services [https://w**

ww.bcsinc.com]. Blackwell, where he is responsible for the design, development and implementation of solutions to meet customers' business and integration needs. He holds a bachelor of science degree in business from Indiana University.

Oriel Maxime is an architect and technical manager at Blackwell. He has more than seven years of experience in translating complex business and technical problems into effective software solutions and is MCSD and MCSD.Net certified. He has a bachelor of science degree in applied mathematics from the Illinois Institute of Technology.

The reality is that most companies, especially midsize and larger, already have heterogeneous environments resulting from previous technology advances. You already have technologies that may include Cobol, CICS, IMS, DB2, Java, C/C++, Perl, PowerBuilder, Java, various incarnations of Microsoft technology and many more, all of which have become or will become legacy applications using legacy technologies.

Technology and business are changing at ever-increasing rates. There is no reason to believe that this trend will subside; instead, it's more likely to continue and sharpen. Given that we can't escape these realities, what can a company do to avoid technical churn or at least accept it and make it part of the plan?

'Green' Architecture and Recyclable Applications

The thing that's more important than the platform you choose today is how you choose to implement the platform and design the applications you run.

Software object reuse has never really lived up to the hype. There are a lot of reasons, but one of the biggest is that objects live on platforms that change too quickly. When the platform changes, the objects written on that platform have to change too. Companies spend more time upgrading their toolbox of objects than reusing them.

Enter Web services — the concept is to build an application, not as a monolithic system, but as an aggregation of smaller systems that form a community, not unlike the human body with organs or cells that specialize but work together toward a common purpose. In this model, you reuse entire systems, not just the objects that compose them. Because systems have a longer lifetime than any particular object model, you get more return on your investment.

Web services can provide one mechanism to make this approach more viable. The key here is that you can always reuse a Web service application, no matter what platform it was written on. This protects you from the J2EE/.Net choice today. And tomorrow, it will protect you from whatever the choice turns out to be. Whether you decide to use J2EE or .Net, you should build your systems like you expect them to talk to a completely different platform. That way, when you actually need to cross platforms, there won't be any huge surprises.

Web services standards, and the potential they unleash in the form of "service-oriented architectures," are the real secrets behind getting value from your J2EE or .Net investments.

Conclusion

Instead of choosing your platform based on marketing hype or technical bias, you can look at more bottom-line factors:

- What assets does your company already possess (software, hardware, middleware)?
- What level of experience do you have in people who know both your business and your implemented technology?
- Will upgrading that system asset result in a positive ROI for the business?

Adopting this philosophy of architecture will provide the greatest stability and ROI over time, not whether you choose J2EE or .Net. By adopting a service-oriented architecture, it won't matter if you pick the wrong horse in this year's platform derby or the next because you can incorporate them into the mix.

Naturally, you will still have to support what you've created, but you aren't forced to rewrite it to take advantage of the next trend.

Careful consideration must still be given to which technologies are chosen to support and deploy, but this philosophy provides options and time to decide which application assets can benefit from advances and which ones are just fine the way they are.

Sponsored Links

Empower your cybersecurity team with expert insights from Palo Alto Networks.

Secure AI by Design: Unleash the power of AI and keep applications, usage and data secure.

Accelerate impactful results with Elastic on Microsoft Azure. Seamlessly access Elastic Search, Observability, and Security within the Azure portal to quickly derive and act on data insights.

Optimum Business | Ensure continuity with Business Connection Backup.