EXCLUSIVE: MICROSOFT .NET FOCUS ISSUE

February 2002 Volume 2 Issue 2

WebServices JOURNAL

.NET J2EE XML

WSJ2.COM

web services world tour EDGE 2002

WASHINGTON, DC	FEBRUARY 26
NEW YORK, NY	MARCH 19
SAN FRANCISCO, CA	APRIL 22

Page 52 Register for Web Services Edge 2002 East Gold Passport, Attend the World Tour FREE!

GETTING GREATER BUSINESS VALUE from a WEB SERVICE

PAGE 6

RETAILERS PLEASE DISPLAY UNTIL MARCH 31, 2002

$6.99US $7.99CAN

SYS-CON MEDIA

Sonic Software

www.sonicsoftware.com

Sonic
Software
www.sonicsoftware.com

SpiritSoft

www.spiritsoft.net/downloads

WebServices JOURNAL
.NET J2EE XML

SUBSCRIPTIONS
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS, PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT.
SUBSCRIPTION HOTLINE: SUBSCRIBE@SYS-CON.COM
COVER PRICE: $6.99/ISSUE
DOMESTIC: $69.99/YR (12 ISSUES)
CANADA/MEXICO: $99.99/YR OVERSEAS: $129.99/YR
(U.S. BANKS OR MONEY ORDERS)
BACK ISSUES: $10 EA., INTERNATIONAL $15 EA
SUBSCRIBE@SYS-CON.COM

EDITORIAL OFFICES
SYS-CON PUBLICATIONS, INC.
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637
WEB SERVICES JOURNAL (ISSN# 1535-6906)
IS PUBLISHED MONTHLY (12 TIMES A YEAR) BY SYS-CON PUBLICATIONS, INC., 135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
PERIODICALS POSTAGE PENDING
POSTMASTER: SEND ADDRESS CHANGES TO:
WEB SERVICES JOURNAL, SYS-CON PUBLICATIONS, INC.
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645

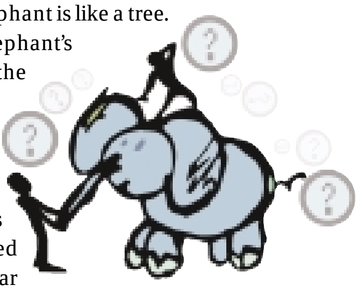
Microsoft .NET–Past the Hype to Reality

written by
Sean Rhody

Author Bio:
Sean Rhody is the editor-in-chief of **Web Services Journal**. *He is a respected industry expert and a consultant with a leading Internet service company.*

SEAN@SYS-CON.COM

There's an old story about what happens when several blind men encounter an elephant. One, feeling the leg, says that an elephant is like a tree. Others, touching various other parts of the elephant's anatomy, describe it as other things. The point of the story, besides not hiring blind men to do anatomical surveys, is that depending upon your perspective, a particular object can look different.



Nothing in my career in IT has had more similarity to this story than .NET. Perhaps that's because, unlike the blind men, the parties involved have a vested interest in having their particular version of the truth prevail. Because, you see, .NET is a myth. No, .NET is a reality, and J2EE is legacy code. .NET is full of security holes. No, .NET is robust and the platform itself will protect the developer and the user. .NET is a risk because it collects personal information.... All of these statements have been bandied about in the press.

This issue of ***WSJ*** focuses on the .NET platform. This technology, based on SOAP, UDDI, and the entire Windows platform, is the critical software suite on which Microsoft is staking its hopes of revolutionizing the industry.

Microsoft faces a number of challenges in driving toward that goal. The Windows server platform, particularly IIS, has been assailed by hackers who took advantage of its ease-of-use features and turned upon the very platform itself. These incidents have made it difficult for people to accept that the .NET platform can be secure, and because a Web services platform is intended to expose the key business functionality of an enterprise, this difficulty translates into hesitancy in adoption of .NET.

Nevertheless, Microsoft is making great efforts toward improving security and removing the issue as a stumbling block toward adoption. But a second concern, privacy, also looms large. That's because MS Passport, a part of .NET, collects a large amount of personal information, and, when used in conjunction with commerce sites, may evolve into one of the most comprehensive demographic databases available.

But this data collection isn't necessarily bad. If we can get past the superficial cross selling to a point where actual intelligence kicks in, it might be worth it. For example, I buy a lot of music online but my tastes are eclectic. One week I may buy a country album, the next some adult alternative music, the following week some light jazz, and the next something mainstream. A site that digests my tastes and can recommend new artists intelligently would get my vote immediately. And because I like new music, I'd be willing to allow the site to collect that kind of information. If Passport can fulfill this vision, I'm for it.

It really comes down to a vision. Microsoft has one vision of Web services based on the .NET model. While remaining Microsoft-centric and leveraging traditional MS tools, it achieves a lot of platform interoperability, which is a big plus in the corporate development world. And, while retaining the benefits and uniqueness of the Windows platform, .NET allows other platforms to integrate with Windows in ways that were not possible before.

Of course, this worries the others involved in Web services. In particular, the J2EE vision of Web services finds the platform irrelevant, and focuses more on the application server as the platform. Certainly, this leverages multiple-hardware architectures effectively. But it represents a divergence from the .NET strategy. And yet, because it encompasses XML, UDDI and sometimes SOAP, it provides a great deal of interoperability with the .NET platform, although the focus is on assimilation rather than true cooperation.

For these vendors, .NET is a worry. It's the strongly leverageable Windows desktop moving into the enterprise server environment where UNIX and the mainframe live, and where Java has most successfully penetrated. Certainly .NET threatens this position.

So it's not surprising that the press, in covering the various vendor positions on .NET, makes it look like the situation with the blind men. Each has some grain of truth, some viewpoint of .NET as it relates to Web services, and still it seems like the big picture somehow escapes us. Hopefully, this issue will help you make up your mind and determine which part of the elephant .NET is for you.

.NET FEATURE

written by Paul Hernacki

Thinking of bigger and wider opportunities

Okay, so you've developed what you believe is a useful piece of code and exposed it to the world as a Web service. All comers may now rejoice in the warmth and glow of your artful coding. But who are the people and what are the business systems benefiting from your Web service? Are you reaching all the possible consumers of the service? What more would you have to do to make it as useful as possible to an even greater audience? And, to ask a simple but important question, do you even know exactly who's on the other end using the service?

The truth is, there's only so far you can go in order to make certain that the people you want to have access to your service are the ones actually using it. Thus begins the careful balancing act between concepts like functionality and performance...and the increasingly important issue of security. Wondering where this is going? Keep reading.

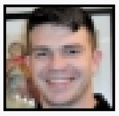
Moving Beyond Tic-Tac-Toe

Microsoft's .NET vision of XML Web services, together with the powerful set of tools MS provided for Web services deployment and use, helped create a new paradigm with tremendous impact. The ability to provide services *between systems* in the same way we use Web pages to deliver services creates value, as it gives us the opportunity to realize the promise of distributed computing among disparate platforms.

The current use of Web services focuses more on solving real business problems and less on performing the trivial functions initially used to illustrate the concept. (Although personally I still think playing tic-tac-toe via SOAP requests is a great way to experiment with new technology!) Businesses adopt Web services technology to generate revenue and lower costs. In doing so, they deal with the challenge of exposing their data as a service to a particular audience. When fulfilling that service, they decide what degree of security is needed so only that audience (and hopefully it's a paying one) has access to it.

Multiple Audiences, Multiple Security Levels

If you're reading this article you've probably already written a Web service or two, and you've probably also read plenty of white papers and articles on authentication and authorization options in Web services. The examples such articles use frequently imply that a Web service either has a particular level of security or it does not. This in turn suggests that a Web service has only a single audience, for whom that level of security is sufficient.



Paul Hernacki is a technical evangelist for Extreme Logic and is the bridge between the technical and business audiences. He articulates Extreme Logic's solutions and explains how the Microsoft .NET platform delivers business value.
PHERNACKI@EXTREMELOGIC.COM

As developers, we want our code to be as useful as possible to the greatest possible number of users. Why? Well, job security may be one reason. More often it's pride and hubris (first noted as some of the virtues of a truly great programmer by Larry Wall and others in *Programming Perl*, a book now standard in the personal library of most developers). We love the idea that something we created is out there helping to make the world a better place.

But for us to keep playing with a new technology like XML Web services, somebody has to be footing the bill. You got it… business users! And what do they want? They want the greatest possible revenue for the least possible investment. Unfortunately, business users don't often believe that first-person shooter games played over the company network embody the greatest revenue opportunities.

So you need an approach to developing Web services that gives your code the greatest possible use and produces the greatest possible revenue. This means moving away from the idea of services that only meet the needs of a singular group of stakeholders. And this in turn means getting away from the idea that a service is either secured or it isn't. We need to think bigger and more broadly about what a Web service can do for the business.

Different Strokes for Different Folks

I was up late last night trying to think of some good examples to explain this approach. Okay…that's a lie. I was supposed to be up late working on this article but instead I was online checking the stats of my fantasy hockey team. It's annoying, because it takes forever to go through all the box scores and then calculate in my head how many points I've earned using our league's scoring system. Of course I could have just waited until the next morning when the fantasy site grabs the box scores from a major online news provider and updates the official league scoring. But that requires waiting and I hate waiting (impatience being another of the virtues of a programmer).

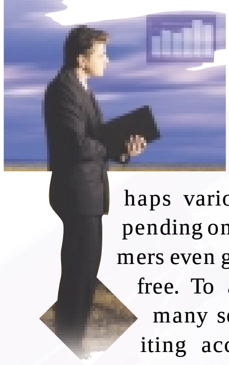
It then occurred to me there's an organization out there that controls the origination of that statistical data and publishes that data to subscribing sources. Who's interested in that data? Sports sites, portal/content providers, newspapers, and of course, fantasy hockey sites – to name just a few. Each one of these data consumers varies in its needs as to the timeliness of the data. Accordingly, each attaches a particular value to the data. The sports Web sites may want the data immediately with extensive detail, while the weekly newspaper in a small town may prefer consolidated information weekly.

The important point to consider here is that there's an initial owner of that data who then publishes it out to multiple consumers (and don't forget to consider both the potential internal and external consumers of the service). Honestly, I've no idea how this happens today, but the idea illustrates my point well. Data already exists that can be delivered to multiple stakeholders either for profit or, in other instances, as a cost-cutting measure.

The goal then is to create Web services that

Interval or Timeliness of Data Updates	Potential Consumers	Value Attached to Service	Security Required
Minute	Sports Web Sites	`High	Medium
Hour	General News and Info Portals, TV Networks	Medium	Low
Day	Daily Newspapers, Fantasy Sports Sites	Low	N/A
Week	Magazines, Weekly Newspapers	Low	N/A

FIGURE 1 **Major League Sports Game data and statistics**



provide different levels of service or data to those specific consumers. Perhaps various rates are charged depending on the service. Perhaps consumers even get the lowest level of service free. To accomplish this, there are many security issues, such as limiting access according to service bought, while providing sufficient functionality and performance.

Figure 1 illustrates the initial model for this approach. If the hockey stats example doesn't seem relevant enough to you and your company, then let's consider other possibilities. How about a manufacturer that updates its production schedule every few hours based on actual orders received through trading partner Web sites and other sales order entry systems (see Figure 2)? While supply chain partners would benefit in receiving hourly updates on materials needed for production, other stakeholders – such as company managers and executives – may need only weekly roll-ups of the data, and this information could be supplied to them through personal portals on the intranet. Still other employees may be interested only in monthly roll-ups, just to give them an idea of how the company is doing. Additionally, investors or analysts may be allowed access only to quarterly roll-ups on a delayed basis. Get the idea? It's all about thinking bigger, in terms of all the different people benefiting from the service.

Interval or Timeliness of Data Updates	Potential Consumers	Value Attached to Service	Security Required
Hour	Supply Chain Partners	High	High
Day	Procurement Managers, Scheduling & Planning Systems	High	High
Week	Company Managers and Executives	Medium	Medium
Month	Emplyees	Low	Low
Quarter	Analysts, Research Groups, Press	Lowest	N/A

FIGURE 2 Manufacturing Production Schedule Data

So how do you develop and implement a service to answer the broad range of needs of all stakeholders? The answer is not much different from the way you're *already* developing Web services. This article is aimed more at discussing the overall approach to planning and using Web services, and how business should think of them, than it is about a change in the way we actually develop.

A Working Example

For discussion purposes, let's walk through an example of how we might do this. Let's go back to the hockey game because…well…I like hockey.

We completed the first steps. We identified the data under our control that is useful and valuable to a number of consumers. Then, we identified the possible consumers and determined levels of value they might attach to the data and the probable details desired. And then we considered the degree of security required.

For our example, let's assume that, as games are played, the data is entered using an application at the stadium location and then submitted to a central data store that we control. Using VisualStudio.NET, we create a Visual Basic COM object that queries the data store and formats the data into an XML document every 60 seconds (including trimmed down versions at the various delayed levels) as a Web service. We'll name it Statistics Service, which is made available through an ISAPI Listener. The Microsoft development environment makes it easy for us to create test clients to be certain the service works as planned.

TABLE 1: Authentication Mechanisms for HTTP

Basic	Use for nonsecure or semisecure identification of clients, as username and password are sent as base64-encoded text, which is easily decoded. IIS will authorize access to the Web service if the credentials match a valid user account.
Basic over SSL	Same as Basic authentication except that the communication channel is encrypted, thus protecting the username and password. A good option for Internet scenarios, however using SSL has a significant impact on performance.
Digest	Uses hashing to transmit client credentials in a secure way. However, may not be widely supported by developer tools for building Web service clients. IIS will authorize access to the Web service if the credentials match a valid user account.
Integrated Windows authentication	Useful primarily in Intranet scenarios. Uses NTLM or Kerberos. Client must belong to the same domain as the server, or belong to a trusted domain of the server's domain. IIS will authorize access to the Web service if the credentials match a valid user account.
Client certificates over SSL	Requires each client to obtain a certificate. Certificates are mapped to user accounts, which are used by IIS for authorizing access to the Web service. A viable option for Internet scenarios, although use of digital certificates isn't widespread at this time. May not be widely supported by developer tools for building Web service clients. Only available over SSL connections, thus performance may be a concern.

Authorization and Authentication

On receiving a request for the document, we identify the principal making the request, then determine the authorization level. Finally, we answer the request with the XML document the subscriber is authorized to receive. If the subscriber is one of the sports Web sites requesting up-to-the-minute information, we want to corroborate the subscriber, since a premium is paid for this data. But if the requestor is permitted access only to the most recent weekly roll-up, then corroboration isn't critical. In fact, if a request comes in and we can't identify the principal or for some reason authentication or authorization fails, we simply send the weekly roll-up – offering the lowest level of service as a default.

Let's start with the first piece – identifying the principal. The Web service should be designed to handle several security levels accordingly. Since this service will be available over the Internet we won't know the specific IP addresses of all of the consumers, so we want to leverage the authentication features of the protocol used to exchange messages, i.e. HTTP.

Microsoft Internet Information Server 5.0 (IIS) supports all of the various authentication mechanisms for HTTP shown in Table 1 (which is borrowed from MSDN).

Regardless of the security mechanism chosen at each level, we'll create Windows 2000 accounts for each consumer allowed to access the Web service and configure ACLs appropriately on the resources that make up the service. For default we also create an anonymous guest account that will have access to the lowest level of service.

For a nonpaying user there's no need for authentication – we'll allow a standard or guest

"There's nothing groundbreaking about the coding methods here. What's different is the business approach to delivering the service of a single coding effort to the greatest possible number of consumers, despite varying needs surrounded by varying security levels."

login. Paying users will have their own unique usernames and passwords. For daily and up-to-the minute users there should be a more secure method that discourages hackers. This data is usually possible to obtain by alternative methods (e.g., one could view it free on a subscribing sports site or watch all the games on home satellite and write really, really fast). However, since some subscribers are paying a relative premium to get this data fed directly into their systems on request, it should be just difficult enough to make hacking it, or hijacking the session, not worth the trouble.

A reasonable mechanism in this case is to use hashing to transmit client credentials. Usernames and passwords should not be transported as plain text, so SSL would be used. But using SSL for document transmission would be detrimental to performance. One option to consider is splitting Statistics Service into two services, Stats Service (occurring over HTTP) and Logon Service (which precedes Stats Service for paying customers and will require SSL). Additional options for security mechanisms based on your specific business requirements can be found in the Global XML Web Services Specifications – which offer a security language for Web services. The specifications describe enhancements to SOAP messaging providing three capabilities: credential exchange, message integrity, and message confidentiality. These three mechanisms can be used independently or in combination to accommodate a wide variety of security models and encryption technologies.

Logon Service

In our example the client application begins by sending a logon request over SSL to the Logon Service that contains two parameters (username, password). The request is dispatched to a Logon COM object that accepts the username and password as input. They are authenticated against the known user accounts. The COM object then generates a hashed logon key (hashedkey) which is passed back to the client application and will expire in minutes or hours. The key is generated using Windows Cryptography APIs. The hashedkey becomes the first parameter of each request to Stats Service. The hashedkey will be used as an identifier to map to each user's authorization level according to their corresponding ACL so each one can then be served the corresponding XML document. If the username/password combination sent to Logon Service was the guest ID, then a hashedkey is still returned (always a standard and static hashedkey for guest). When the subsequent request comes to Stats Service, it's mapped to the guest access level. The Stats COM object will only grant access to the weekly roll-up of the statistics and dispatch the stripped-down XML document back to the client.

Statistics Service also needs to maintain some level of service in the event that authentication and authorization fail. This is done at two levels. The first potential failure is that Logon Service or Logon Object fails to find the username and password supplied. In Logon COM object, if it can't locate a non-guest username in the list of known users, it will by default return the guest hashedkey back to the client. This needs to be known by the client application if it's a paying customer and receiving limited "guest" information is not acceptable or even useful. Two parameters go back to the client application – hashedkey and accesslevel. The client application now makes the determination whether or not to place the subsequent request to Stats Service based on needs. The next consideration is if the Logon Service or Object failed altogether. In this instance, the client service receives no hashedkey. The client service submits that generic, static hashedkey that permanently maps to guest level authentication. (Someone wanting guest access would never even need to use Logon Service, but we'll anticipate the fact it may happen nonetheless.)

The description of these in and out parameters, as well as the static guest key, would be published through UDDI in the WSDL for the services. For a public service you may choose to publish it at http://uddi.org, but for internal or more private services the Microsoft platform makes this part even simpler since Windows.NET server (when released) will have UDDI built in.

So for Logon Service there are the following parameters:

```
   Logon([in]username, [in]password,
[out]hashedkey, [out]accesslevel)
```

And for Stats Service:

```
Stats([in]hashedkey, [out]statsdoc)
```

There's nothing groundbreaking about the coding methods here. What's different is the business approach to delivering the service of a single coding effort to the greatest possible number of consumers, despite varying needs surrounded by varying security levels.

There are further security measures surrounding the COM objects that you can take advantage of through using Common Language Runtime (CLR). However, this article isn't about security, you can find out more about that by studying it separately elsewhere. Even in a more common scenario like the manufacturing example, the goal is still to allow code written once to meet the needs of various stakeholders; and, for the default, to always offer the option for the lowest denominator of service. For highly secure data, a variation of this allows you to include a check for a known IP Address and/or Certificate from a given partner before you publish to them the requested XML document. Without that, it still offers the rolled-up delayed data.

It's a Web service for the masses…or at least for a whole lot more people than I might have tried to target previously. Now if only there was a Web service that could have accurately predicted that my star goalie would be out several weeks with a twisted ankle.

SUN

Microsystems

www.sun.com/sunoneinfo

SUN
Microsystems
www.sun.com/sunoneinfo

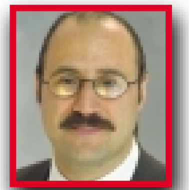
Written by Paul Lipton

MS Gets It At Last!

A Java architect plays with the Beta 2 of .Net... and is pleasantly surprised

My first thought, when I heard about .NET, was "Here we go again!" It sounded like yet another attempt on the part of Microsoft to revitalize what appeared to be a fading technology and vision. Increasingly, as a programmer and architect, I was getting the impression that the exciting stuff was happening in the Java universe. For example, there were innovative ideas coming out of the open source community, such as the Apache Struts and Cocoon projects. Also, many of the most interesting books and papers being written were using Java code to illustrate important concepts. All in all, it seemed that many informed minds had already made their language choice. (Besides, I must confess that I've always hated Basic.)

Like many programmers who are comfortable with C++ and Java, I secretly believed that the best programmers preferred object-oriented languages that at least understood what inheritance was, could handle programming exceptions in a mature fashion, and had a syntax that generally didn't remind us of how we used to code back in the pre-PC days (around the time of stone knives and cave dwelling). Also, considering how many times MS had flip-flopped on the issue of Java, there was a general impression that, despite Microsoft's money and its dominance on the desktop, the corporation was lost. They just didn't "get it."



Author Bio:
Paul Lipton is the director of Object Technology at Computer Associates International, Inc. (CA). He has been an architect and developer of enterprise systems for more than 20 years. Paul participates in various standards organizations such as the ODMG (Object Data Management Group) where he participates on the C++ and Java committees. He has participated in the Java Community Process, having served on the expert committee for JDO.
PAUL.LIPTON@CA.COM

Sea Change for Microsoft?

Well, I've been playing recently with the Beta 2 of .NET, and the biggest shock isn't the dependence on XML or the emphasis on Web services: it's simply that Microsoft finally "gets it." Don't get me wrong; there's not much in .NET that hasn't been done with other technologies. However, .NET does have a consistent technical vision that's at least comparable to what I've seen in competing technologies. Furthermore, that vision is pretty far along.

I won't go down the full list, but see if this sounds familiar to you. As in Java, there is a virtual machine with garbage collection and other enhanced runtime capabilities. On top of this are sets of libraries that demonstrate good object-oriented design principles like encapsulation. Not exactly new ideas, but it's nice to see Microsoft admit that such an approach makes sense. And Microsoft has played to its traditional strengths, as well. The development environment is a thing of beauty. It's comprehensive and carefully thought out.

> "Everything a developer would want is there to facilitate construction"

Essentially, everything a developer would want is there to facilitate construction of basic Web services.

.NET and J2EE

What does this mean to the many companies that have made sizeable investments in technologies such as J2EE? Expect there to be much marketing hype and misinformation from all sides. For example, Microsoft points out that the .NET virtual machine could run on diverse platforms beyond Windows, which is true enough – but don't expect the major hardware/OS vendors like Sun, HP, and IBM to start lining up for the privilege of porting .NET to their own environments – Microsoft's record on porting to other platforms isn't good. Sure, there are lots of smaller companies that might be able to port .NET, but will major corporations care to depend on these ports?

The same situation exists with languages. Regardless of the legal and marketing issues, Microsoft's recent addition of J# is still a letdown for serious server-side Java developers, as it doesn't support the standard J2EE libraries. However, it does provide a migration path for the small number of loyal J++ users still out there, so all credit to Microsoft for taking care of its own.

More importantly, MS has made a big point of showing how .NET supports multiple programming languages. Picture something like Oberon, Eiffel, and VB programmers all working together. Each, supposedly, would be using the perfect language for the task at hand. But, rightly or wrongly – no flames, please! – most new development today is in VB, C++, and Java (with C# potentially a strong contender just around the corner). The ability to have VB, C#, and C++ work together will be very useful for existing Microsoft customers, but allowing a wider variety of languages may be less useful to many corporations than being able to use a wide variety of vendors. History, of course, will be the judge.

On the one hand, the J2EE vendors are all busy trumpeting the relative maturity of the underlying J2EE platform. On the other hand, Microsoft appears to be one of the clear leaders in Web services, and .NET appears to be built solidly upon a fairly well thought out foundation. So how long is that "maturity lead" that the J2EE community claims going to last?

Timely Focus Issue

That's why this special .NET issue of ***Web Services Journal*** is so important. Web services is a much more comprehensive, enterprise-level issue than most vendors want to admit: developing, integrating, securing, protecting and managing Web services are all part of what's needed to be successful. No one vendor, not even Microsoft, will have all the answers. Especially for large enterprises, there are still big questions about how well .NET fits into existing, heterogeneous environments. Some of the most important questions to ask are how do I manage .NET as part of the diverse platforms, systems, and applications already running my business? What are the best techniques to integrate .NET applications with legacy applications, and what are the best techniques and solutions?

In the meantime, the industry will witness Microsoft banging the drum loudly when discussing certain industry standards, and remaining conspicuously silent on others. The world of Web services is more than just SOAP, UDDI, and WSDL. There are many other important issues and standards that informed consumers may need to consider. The devil, as they say, is in the details.

Written by Scott Dietzen

J2EE vs .NET: Friends or Foes?

What will determine whether one prevails...or ensure that they can co-exist?

Java 2 Enterprise Edition (J2EE) has not yet achieved critical mass as a Web application platform. Today, for example, over 10,000 customers are using BEA WebLogic, the J2EE application server market leader. However, it's a safe bet that J2EE will eventually reach critical mass.

Microsoft's .NET is also a safe bet to get to critical mass. No other Web application infrastructure software platform has the traction of these two leaders, and while some developers may be drawing battle lines between J2EE and .NET, it seems to me that their peaceful co-existence will be the norm.

Most sophisticated IT organizations will deploy both platforms. J2EE already has a strong position in enterprise applications, and enterprise ISVs (independent software vendors) need an application code base that can be deployed on whichever platform their customers demand. On the other hand, much of Microsoft's existing ISVs and small enterprise marketplace is sure to jump on the .NET bandwagon.

Author Bio:

Scott Dietzen, is chief technology officer for BEA Systems, Inc.

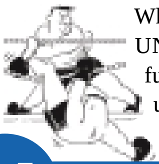
SDIETZEN@BEA.COM

Interoperability Via Web Services

The good news for all of us is the promise of easy interoperability between the two platforms, via the shared Web services stack (SOAP, WSDL, UDDI, URI). The leading J2EE platforms already provide Web services that were delivered before the advent of .NET. The real fisticuffs will continue to be Java vs C# and J2EE vs .NET.

Both J2EE and .NET are server-side platforms. J2EE, of course, came out of the recognition that the sweet spot for Java was more than just client devices. Rather, Java was and remains an outstanding platform for server-side applications. J2EE extends Java with the standard application programming interfaces (APIs) required for server-centric Web applications. J2EE does for Web applications what SQL did for relational applications – protects investment in programs and programmers.

While there's some truth to the "write once, test everywhere" claim of Sun's Java, interface specifications and rigorous compliance testing have ensured that the Java community has done a far better job of protecting programming investment than prior standards like SQL and POSIX. What's more, the portability offered by a specific J2EE platform like BEA WebLogic is comparable to that offered by .NET, but applies across virtually all of the hardware and operating systems on the Web. This multiplatform support remains one of the most fundamental value-adds for J2EE. While C# may indeed be ported to UNIX, it seems unlikely that the full .NET environment will show up there (that is, Basic – "the language" – may be multiplatform, but Visual Basic/ VB.NET – "the platform" – is Windows-centric).



"The real fisticuffs will continue to be Java vs C# and J2EE vs .NET."

Java/J2EE Is a Community Effort

While competition between J2EE and .NET helps the industry by driving innovation in both market camps, how this innovation comes about highlights another of the fundamental differences between the two platforms: .NET APIs are the product of a single company, while Java/J2EE represents the collective innovation of the hundreds of companies that make up the Java community. Universities, too, have bought into the Java paradigm, making it the de facto standard in computer science curriculums.

For those that are looking to handicap the J2EE/.NET race, I would summarize the technology advantages for J2EE as follows:

- ***Maturity:*** J2EE has four years of scaled production under its belt. How many years did it take to fully harden Microsoft SQL Server (which is of comparable complexity to .NET)? And SQL Server was derived from a mature product.
- ***Richness:*** Today, the J2EE component model supports sessions (with session protection), asynchrony (messaging), caching, replication, and automated persistence. Moreover, J2EE is language-specific, which offers a more natural model for intra-application invocations (Web services are the better fit for inter-application calls).
- ***Pluggability:*** J2EE offers service-provider interfaces (SPIs) for messaging and events (JMS), resource adapters (JDBC, J2EE connectors), XML processing (JAX), and so on. These SPIs allow a level of plug and play not afforded by .NET.
- ***Learning Curve:*** The existing Visual Basic programmer will find that .NET demands a steep learning curve. This disconnect is going to have a multi-year impact on the reworking of Microsoft applications and the retraining of MS developers. The Java/J2EE community is simply further down this path, and J2EE applications require zero rework for Web services.

Microsoft and Java

Microsoft's strategy with respect to Java is actually good news for companies like BEA. Had Microsoft elected to stay in the Java business, I suspect it is unlikely that BEA would be having so much success enabling J2EE for Windows. Today, WebLogic is one of the most Windows-friendly of the J2EE application servers, including support for native bindings for MS SQL Server, IIS, COM+ (bi-directionally), and, last but not least, "out of the box" interoperability with Microsoft clients and servers via Web services.

Future Factors

What will ultimately determine whether J2EE or .NET prevails are the innovations still to come. Here are just a few examples of future factors:

1. How reliable and secure each is for asynchronous Web services
2. How dramatically each reduces the complexity of application integration
3. The extent to which each simplifies aggregation and personalization of user interfaces
4. The extent to which each lowers the total cost of ownership

The ultimate winner(s) will have integrated platforms that make even the most difficult things easier. Stay tuned!

Quintessence

www.in2j.com

Written by Norbert Mikula

Standards–What a Mess

Only at first glance...WSJ makes some sense of it all

There are plenty of jokes regarding the world of standards development, from "Standards are like sausages – you're better off not knowing how they were created" to the old-time paradox: "The good thing about standards is that there are so many to choose from."

As a person who spends a considerable amount of time trying to keep track of the e-business technology landscape, I can appreciate the pain and frustration of anyone trying to make sense of the dozens of "standards" that are being developed by different organizations.

Useful Reference Model

While I can't promise to completely untangle the mess out there, over the last three months I've started to use a reference model that I've found to be a useful tool to navigate the maze of e-business standards. Published by the Business Internet Consortium (BIC: www.businessinternetconsortium.org), the model was developed under the leadership of Dr. Jackson. He of Intel. While not – yet – perfect and only in first revision, it is, in my view, a very promising approach, and I've already used it successfully in my own presentations. The very positive feedback I've received prompts me to use this forum to introduce it to a broader group of people.

Author Bio:
Norbert Mikula has more than 10 years of experience in building and delivering Internet and e-business technologies. He serves as vice-chairman of the board of directors of OASIS and is industry editor of Web Services Journal. Norbert is recognized internationally as an expert in Internet and e-business technologies and speaks regularly at industry events.
NORBERT@SYS-CON.COM

Since I'm not representing BIC (so if you don't like my explanations, blame me, not them), I'll also weave in my own thoughts and will try to associate some of the current e-business–related OASIS activities with the respective layers (this isn't an official OASIS view, by the way; it's just my own ramblings). I choose OASIS, in particular, because as a long-time fan and board member, I'm probably as familiar as anyone with its current initiatives.

Underlying Philosophy

The basic principles underlying the design of the BIC Reference Model – that's the name I use for it – are *open standards* and a *layered approach*.

I use open standards simply because the model includes only standards where a group of experts worked together to collaboratively come up with a solution for a problem, and where a potentially unlimited number of companies, countries, and/or individuals could exercise influence and vote on matters and the final release of the intellectual newborn.

"Standards are like sausages – you're better off not knowing how they were created"

This approach hopefully avoids proprietary vendor-specific specifications. We'll leave aside the fact that in reality standards development is a fantastic mix of full-contact combat sports and UN Security Council–style political games and charades.

I have a layered approach because ultimately we want to end up with a model where we have clearly defined layers that are functionally well specified, with a lower layer forming a solid foundation for an upper layer. The ISO/OSI networking model would, for example, be an excellent example of such an approach.

While I can see why this approach is necessary, one shouldn't forget that it can only work in practice if all the participants agree on what the layers and their defined interfaces look like. Having said that, it's still my hope that over time we may see some convergence fostered by discussions around this model. Certainly one thing the model achieves is that one is able to put into broad conceptual "buckets" many of the initiatives currently under way. Given that the model is a conceptual one, this is just fine.

On a high level there are three layers

(and the BIC makes reference to Gartner here): the lowest layer provides the base-level communications and integration infrastructure; the middle layer provides the syntaxes and schemata to define business objects and business processes; and finally the top layer houses the actual business messages and processes.

Base-Level Layers

Backend Integration

Most (if not all) e-business architectures will require us to provide connectivity to the back-end systems that host our mission- and business-critical data. I personally would see the J2EE Java Connector Architecture (JCA) as an important emerging standard. Also good old JDBC and ODBC might fit into this layer.

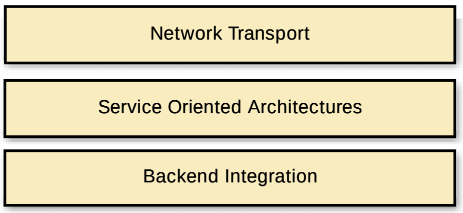


FIGURE 1 Technical Conceptual Layer - Base level layers

Service-Oriented Architecture

Here we have all the tools and systems that allow us to build e-business systems, including both Java 2 Enterprise Edition (J2EE) and Microsoft's .NET architecture. In this layer we find all the components that help us write the code that allows us to connect to our back-end systems but also the business logic of the e-business software itself.

Object Management Group

www.omg.org/mda

Network Transport

Network transport is, not surprisingly, one of the best-understood layers. In this section we can find the likes of HTTP, FTP and SMTP. This layer really provides the foundation upon which we move around our business data and objects.

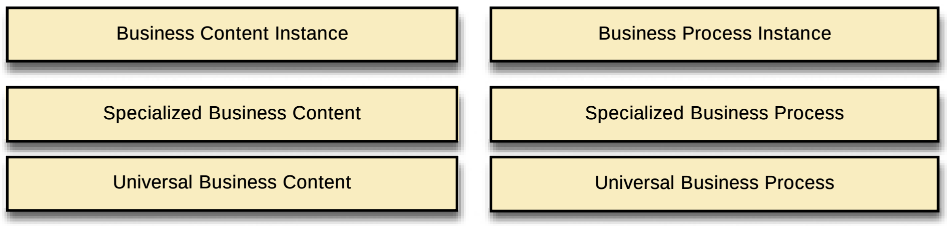

Business Content Instance
Business Process Instance
Specialized Business Content
Specialized Business Process
Universal Business Content
Universal Business Process

FIGURE 3 | Technical Conceptual Model - Business Content and Processes

Syntax and Schemata

Core XML Standards

The core XML standards are what the W3C has been focusing on for the last few years. Here we can include XML, XSL, and many other of the X<insert-name-here/> specifications. From an OASIS perspective, this layer includes the interesting RELAX NG initiative that just finalized its approach to developing lightweight XML schemata. Other relevant efforts at OASIS include an XML version for the entity resolution mechanism developed originally for SGML, as well as the topic map-related initiative, "Topic Maps – Published Subjects."

Messaging

In the messaging layer are the standards that help us to define the structures of both the envelopes, as well as the actual contents of our business objects. One example in the spotlight right now is the Simple Object Access Protocol (SOAP). We can also mention here the OASIS TRP (Transport Routing Protocol) specification, which uses SOAP at the core but adds more "business strength" messaging qualities on top of it. SOAP and OASIS TRP are a great example of how within one and the same layer we might find initiatives that actually address a similar problem-space but are very different in their approach.

Service Description Languages

A lot of attention is currently being given to this layer (it's even one of the core subjects of an entire magazine – ***Web Services Journal***!). Here BIC puts initiatives such as WSDL (Web Services Description Language), but a more recent initiative worth mentioning, one that was started after the model was published, is the Web Services Component Model (WSCM), a new OASIS Technical Committee.

Repository

Repositories can provide a very diverse set of functionalities, such as the storage of business object definitions and business processes, as well as data-dictionaries. Many repository initiatives have emerged over the last years. Examples are the RosettaNet business and technical dictionaries as well as the OASIS Registry and Repository specifications.

Directory/Registry Services

Registries and repositories are often combined, as in the case of the OASIS Registry and Repository work. While the repository stores actual objects, the registry provides the management interfaces and protocols to register and discover the entries of a repository. Another well-known example is UDDI (Universal Description Discovery and Integration). Also underway is the OASIS DSML (Directory Services Markup Language), which provides an XML-based interface to LDAP.

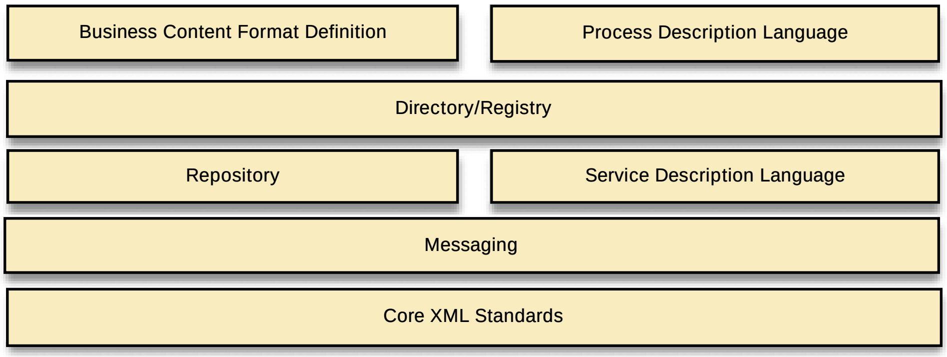

Business Content Format Definition
Process Description Language
Directory/Registry
Repository
Service Description Language
Messaging
Core XML Standards

FIGURE 2 | Technical Conceptual Layer - Mid-level layers

Business Content Format Definition

The specifications in this layer enable the actual definition of all aspects of our business objects. This layer provides the basic building block for business messages. Examples listed by the BIC include RosettaNet Technical Dictionary Structure, RosettaNet Business Dictionary Structure, RosettaNet PIP Service Content, OAGI Business Object Document, as well as ebXML Core Components.

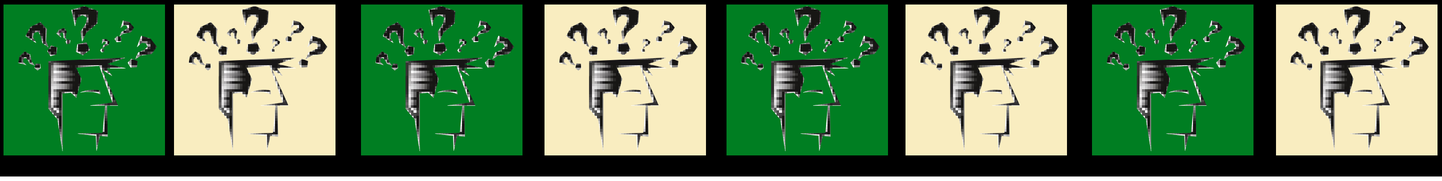
Business Conceptual Model

Universal Business Content

This specifies business terminology and accepted values that may be universally used in business messages that support a broad range of industries, business models and locales; it's the vocabulary used to construct the business content of a message. Examples are RosettaNet Business Dictionary, OAGIS, ebXML Core Components, and many others.

Jon Bosak, former chair of the W3C XML Working Group, is spearheading UBL (Universal Business Language) with the objective of developing an XML business library under the OASIS umbrella. There are two "religions" on how one should define business objects and business processes. One is bottom-up – i.e., construct the individual components based on their specific requirements – and the other is top-down, following a modeling approach such as the Unified Modeling Language. Ultimately there are ways to combine them, but if you want to see a good catfight between experts, then just bring up this subject. (But don't quote me…)

OASIS examples here include OASIS CIQ (Customer Information Quality) working on a global scheme for specifying address information and the OASIS Provisioning Services Technical Commit-tee. Finally, there's OASIS DocBook, a schema for documenting software and hardware (and incidentally the first OASIS specification ever to be the subject of a full-length book – published by O'Reilly).



Specialized Business Content

In many ways you can't separate this layer clearly from the "universal" business content layer. Certainly, the model is correct in assuming that there will be vocabularies used only in a specific industry or context, but specific approaches often contain more universal components that are embedded into and strongly tied to these specialized components, and thus can't be separated out.

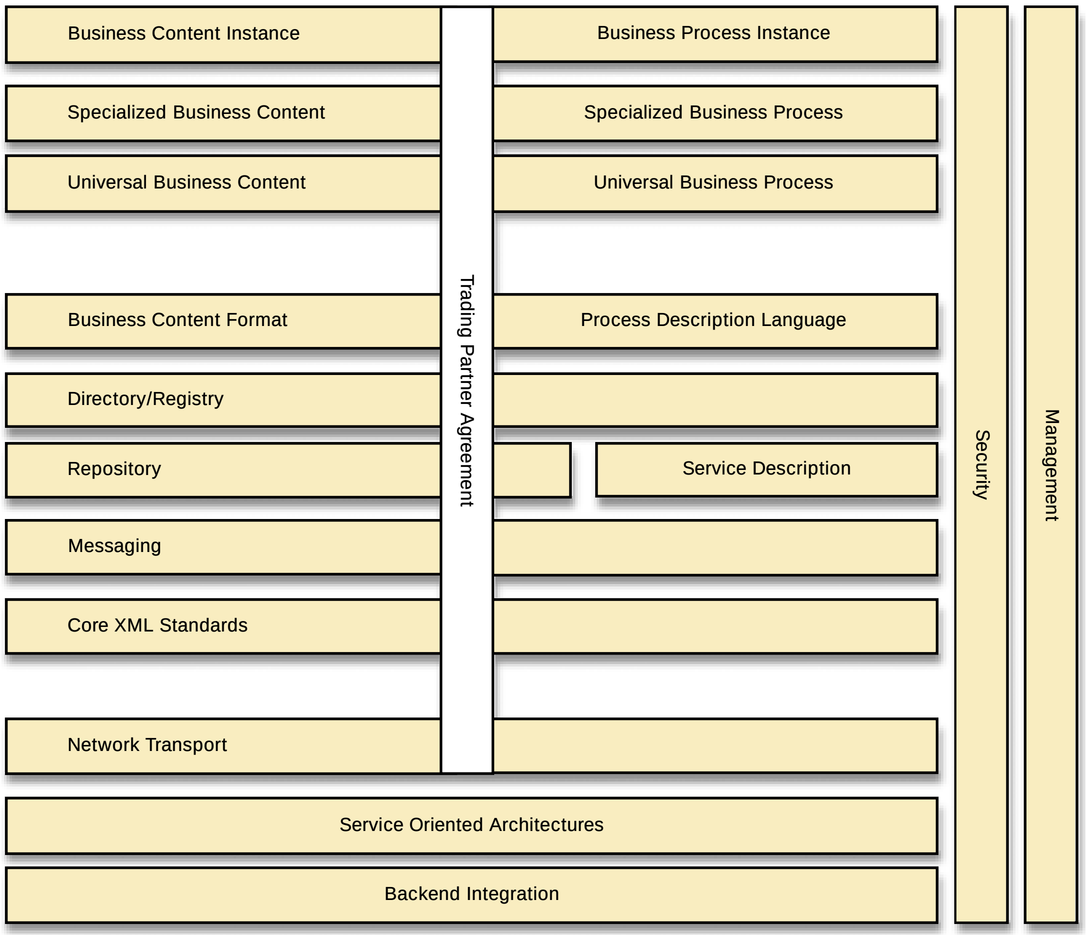
Business Content Instance

Finally, we have the actual business objects and messages themselves. Many still hope that someday we'll be able to build our business content instances from the building blocks found in universal and specialized vocabularies.

Process Descriptions

As with business objects, we have a set of three layers containing specifications regarding the descriptions of business processes. Specifications for Process Description Languages are plentiful and include approaches such as UML, XLANG, ebXML BP, BPML, and the Web Services Flow Language (WSFL). The Universal Business Process layer and the specialized Business Process layer contain – just like their counterparts on the business object side – the more generic and the more specialized business process building blocks respectively. In the actual Business Process Instance layer we can finally find the final business process descriptions that define the sequence of interactions between one or more business partners.

An example of an OASIS effort in the process description layer is the OASIS BTP (Business Transactions Protocol) – which develops specifications to describe distributed and long-lasting business transactions.



Business Content Instance
Business Process Instance
Specialized Business Content
Specialized Business Process
Universal Business Content
Universal Business Process
Business Content Format
Process Description Language
Directory/Registry
Repository
Service Description
Messaging
Core XML Standards
Network Transport
Trading Partner Agreement
Service Oriented Architectures
Backend Integration
Security
Management

FIGURE 4 | Full BIC Reference Model

Vertical Layers

TPAs, or Trading Partner Agreements, are created when two or more parties agree on all aspects of their e-business communications – including protocols, message schemas, and business processes. In the future, a business will publish its trading partner profile by making explicit its preferences and capabilities, and autonomous agents will then be able to use negotiation mechanisms to create trading partner agreements between these parties. To give but two examples, we've already seen such ideas in the shape of UDDI and its t-Models as well as, in a more complete and complex form, in ebXML.

Security

Security is a layer that crosses all the vertical buckets discussed so far. Digital certificates and XML Signature are mentioned by BIC, but I would include here the OASIS XAML (Access Control Markup Language), which is concerned with expressing policies for information access over the Internet. I'd also include the very important OASIS SAML (Security Assertions Markup Language) – addressing, among other things, the Internet-wide Single-Sign-On problem.

Management

This layer is largely undiscovered. From a B2B and Web services perspective, while we do have tools based on standards such as SNMP (Simple Network Management Protocol), a lot of research has to be done here and many opportunities exist in this space for new startup companies.

Try the Model Out

While the model I present here definitely won't cure the common cold, you may want to try it out by trying, when you next hear about a new specification, to find the "bucket" you think it belongs to. A more distant hope on my part is that this model may serve as a guide to help all standards organizations to make a better job of determining how and where they and their respective initiatives fit in – and to figure out how we can achieve what we're all looking for...convergence.

Written by Eric Lynn

Implementing and Obtaining ROI From Web Services in the Here and Now

Introducing a new programming paradigm to a mission-critical application

As I speak both formally and informally to developers about Web services, the same questions always come up: Is anyone we know doing it? Don't we have to wait for .NET or the next version of J2EE? Is it secure enough to be trusted?

The best way to answer these questions is to look at leading-edge companies that have already successfully implemented Web services. Barry-Wehmiller is just such a company. Located in St. Louis, it is an e-commerce marketplace that brings together buyers and sellers, as well as seekers and suppliers of information, to provide value-added alternatives to the packaging industry. The company recently completed a Web services application based on current Microsoft technologies that provides insight into the questions.

Author Bio:
Eric Lynn, a software architect with 11 years of experience in both Microsoft and Java technologies, has authored several courses on Microsoft Distributed Technologies and has taught publicly as well as privately (for Fortune 500 companies, including Intel Corp). He's CEO of DeltaPoint Solutions, a company focused on helping development teams make a successful transition to new technologies, and is a frequent speaker on technology trends.
ELYNN@DELTAPOINTSOLUTIONS.COM

Barry-Wehmiller Thinks Forward ...and Adopts Web Services

In Q1 of 2000, Barry-Wehmiller, like many forward-thinking organizations, began evaluating their strategic development and reviewing technologies that could provide a competitive edge. The company had successfully developed a marketplace that provided a meeting place for manufacturers, brokers, consultants, and material suppliers with a variety of end users, such as plant engineers, purchasing agents, and maintenance managers. Suppliers could post changes to their catalogs and receive orders in near real time – but with a catch. Proprietary systems and a significant trust relationship were required for the successful integration of the supply chain.

The technology solutions combined the use of asynchronous messaging through Microsoft Message Queue (MSMQ) with a VPN (see Figure 1). This solution, though effective, incurred significant maintenance costs, set up serious barriers to entry, and raised security concerns. Internal and acquired suppliers could be trusted as part of the VPN, but what of outside suppliers? Would the cost of implementing MSMQ and a VPN, in addition to the integration of data formats, raise the bar too high? Would suppliers running non-Microsoft platforms pass on the business opportunity, rather than be forced to maintain a Microsoft server (including maintaining MSMQ)?

Web services offered them a way to lower the barrier to participation in the supply chain while increasing the company's overall security – with the added benefit of eliminating the VPN, along with its associated cost and maintenance.

“Don't we have to wait for .NET or the next version of J2EE?”

Clear Business Value Proposition

The business value proposition in this particular case is an instance of one of the overall key value propositions of Web services in general. Barry-Wehmiller wanted to develop a complex and secure technological bridge to an abstract class of business partners (in this case a supplier) through standards-based service mechanisms. Before Web services, the complexity of the technology interface would need to be evaluated with each new supplier and (unless there was a lucky alignment of the technology stars) the construction effort would be repeated. The elimination of this repetition of effort and the associated cost in time, effort, and expertise leads to new business opportunities in which relatively new business relationships may engage in complex transactions between disparate operating and security environments.

The Technology Picture

So, the business case was clear. But what about the choice of technology? In this case, Barry-Wehmiller chose to implement SOAP across https using the Web technologies already in place – i.e., Microsoft. Its current Web offerings were developed on top of Windows NT 4.0 and Internet Information Services (IIS) 4.0 using Active Server Pages (ASP), VBScript, and JavaScript. Microsoft SQL Server and MSMQ provided data services on the Web farm and asynchronous messaging. Business objects were developed with Visual Basic 6.0 and for the most part ran in the context of Microsoft Transaction Server (MTS).

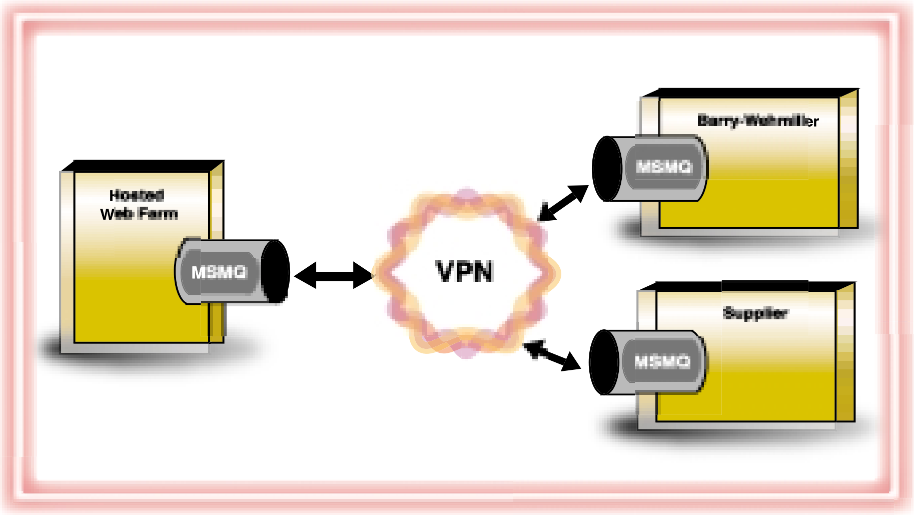


FIGURE 1 | Proprietary supply chain network

BEA
eWORLD

www.bea.com/events/eworld/2002

Barry-Wehmiller had the choice of either being an early adopter of Microsoft .NET or working with an older version of IIS/ASP to implement XML-based RPC. Fortunately, Microsoft had already provided the MS SOAP Toolkit 2.0 and some key samples. Some quick proof-of-concept attempts showed that the available functionality was sufficient for its needs.

I should mention here how surprised I am that so many developers fail to realize that Web services are already available with the current toolsets in both the Microsoft and Java world. The conversations I have with them usually go something like this:

Me: "Hey, have you had a chance to mess around with Web services?"

Developer: "Well, no, I haven't installed .NET on my machine yet."

The Tools Are Already Here

For Microsoft developers, the SOAP Toolkit 2.0 (you know, the one that really works) has been out since early Q1 of 2000. Get it! Play with it! (Sorry about that, I know *you* know that: but would you please tell your friends? OK, I got that off my chest.)

I understand the disconnect: Microsoft touts .NET as the premier Web services platform and developers make the assumption that the current technologies are insufficient. Visual Studio.NET certainly increases development productivity by being intimately aware of SOAP and XML and providing some interesting and powerful language extensions through its attribute-based programming features, but the current toolset is sufficient and quite productive.

How productive? With two developers and one half-time project manager, Barry-Wehmiller completed the conversion to Web services in less than two months! Operations were up and running on budget and on time – a significant feat when you consider that the company was introducing a new programming paradigm to a mission-critical application.

Secrets of a Successful Transition

As a consultant who helps development teams transition to new technologies, I was interested in understanding how the Barry-Wehmiller team was able to make this successful transition. "Training was absolutely critical," Chris Brauch, manager of site development, told me. As I talked to the developers and support staff, I asked whether the transition to Web services was as difficult as the transition from client/server to Web development. The answer was a resounding "Yes!" They'd attended several seminars and read articles and books, but a formal class was what proved critical in enabling managers, developers, and support staff to come up to speed. Consultants were important as well: Barry-Wehmiller engaged G.A. Sullivan to provide critical insight and assistance with the SOAP implementation. The mentoring of a knowledgeable consultant in the context of this real-world project gave the company the ability to apply skills gained in the classroom to its problem-space.

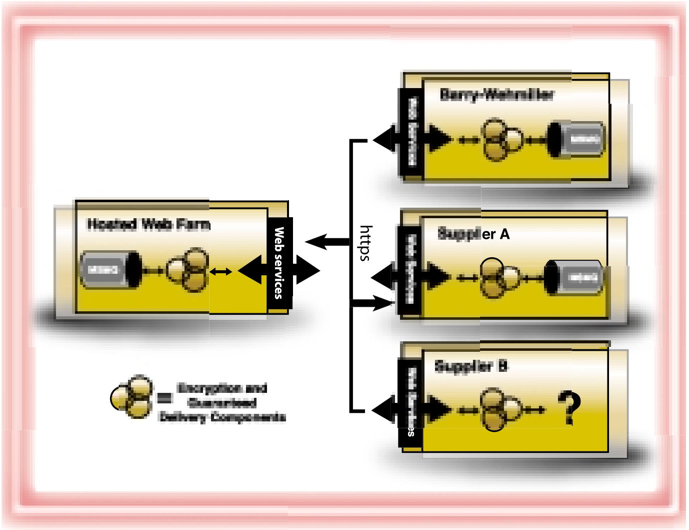


FIGURE 2 Web services supply-chain network

Missing Jigsaw Pieces

The final technical solution is indicative of the present state of Web services. Though SOAP proved sufficient for the transmission protocol, work external to the Web services framework was required for two of the key elements of the solutions: encryption and guaranteed delivery (see Figure 2). The requirement for external components to handle encryption and guaranteed delivery detracted from the overall goal of loosely coupled services, and required that multiple implementations be constructed depending on the supplier operating environment. That was exactly what the company was trying to avoid!

Recent submissions to the W3C by IBM and Microsoft (www.w3.org/2001/03/WSWS-popa/paper51) highlight the still-missing pieces in the Web services jigsaw puzzle, such as standards for binary attachments (eliminating the performance cost of converting to and from a textual representation), message routing, transactions, digital signature support, guaranteed message exchange, and encryption. Barry-Wehmiller is looking at these developments to enhance its Web services offerings and further reduce the barrier to entry into its supply chain.

Despite these missing pieces, the investment made has produced the return expected. New suppliers are able to come on board with minimal investment. The VPN has been eliminated and suppliers don't have to participate in any trust-relationship with the corporate network. Performance has been increased by 150% over the MSMQ/VPN solution. Maintenance costs have been significantly reduced.

Further, with the expertise gained and a solid success under its belt, the development team at Barry-Wehmiller is poised to embrace new Web services technologies that will enhance and expand the service offerings and thereby maintain their leadership in the market.

Conclusions

So, back to those three questions I keep getting asked...

Is anyone we know doing it? Yes, and they've been successful with a surprisingly low investment. Their ROI has been solid and they're poised to capitalize further on their initial success.

Don't we have to wait for .NET or the next version of J2EE? Absolutely not. While these new tools provide some impressive support for SOAP and XML, toolkits are available as effective bolt-ons to current technology.

Is it secure enough to be trusted? Probably not by itself, not yet. So consider third-party tools or home-grown mechanisms in order to add features such as encryption, digital signature support, and guaranteed delivery – and watch the W3C for fast action on these critical missing puzzle pieces.

JavaOne

http://java.sun.com/javaone

Written by Derek Ferguson

Mobile .NET Web Services Part II

A step-by-step guide to invoking Web services from virtually any mobile device

At the end of my last article (***WSJ***, Vol. 1, issue 4), I promised that in this article I'd show you an even better way to invoke .NET Web services from Pocket PCs. In addition, before this month's article is finished, you'll learn how to invoke .NET Web services from Java-powered devices.

The .NET Compact Framework

In the first half of this article, we'll focus on invoking .NET Web services from Windows CE-powered devices (including Pocket PCs) via Microsoft's newest .NET technology: the Compact Framework.

What Is It?

The .NET Compact Framework is the portion of Microsoft's new .NET technology initiative specifically targeted for execution on mobile devices. The best way to get a clear picture of this in your head is to contrast it to the rest of .NET.

The rest of .NET allows you to create:

- ***Windows applications:*** Full-featured GUI applications intended for use on desktop computers running the Windows OS.
- ***Web applications:*** Apps that serve up various kinds of markup (HTML, etc.)
- ***Web services:*** Apps that, you probably know, substitute XML for the various flavors of presentation markup used by Web applications.

The .NET Compact Framework adds the whole concept of Mobile Device applications to the mix. A Mobile Device application is like a Windows application, in that it is a full-featured GUI application. Unlike Windows applications, however, Mobile Device applications are intended to be used on devices instead of desktop computers. Unlike Web applications and Web services, these applications don't require the presence of an Internet connection in order to be used.

Author Bio:

Derek Ferguson is chief technology evangelist for Expand Beyond Corp. He is a world-renowned author, speaker, and developer who has been recognized throughout the IT industry for his work

DEREK@XB.COM

Develop for Devices As You Develop for Windows

The entire goal behind creating a subset of .NET that is intended for execution on devices rather than desktops is to allow the .NET developers to create applications for devices just as they would for desktops. This ability to create device-based applications directly from Visual Studio .NET gives the developer:

- The eventual ability to use all of the .NET languages (VB, C#, etc.)
- Access to a large subset of the .NET Framework classes
- A drag-and-drop approach to creating application GUI's
- Step-by-step debugging, just like on the desktop!

"The .NET Compact Framework is the portion of Microsoft's new .NET technology initiative specifically targeted for execution on mobile devices"

How Do You Get It?

To create a successful solution using the .NET Compact Framework, you need two things. The first of these is the software needed to turn Visual Studio .NET into a development environment suitable for creating device-based applications. The other is the set of runtime classes for installation onto your device(s) in order to allow them to "speak .NET" (as it were).

At the time of writing, both of these things are available from Microsoft as a product called the Smart Device Extensions for Visual Studio .NET. This product is currently bundled with the Release Candidate version of Visual Studio .NET itself.

There are two ways to get the Release Candidate:

1. Be a Microsoft Developers Network subscriber
2. Order it on DVD from http://msdn.microsoft.com/vstudio/nextgen/getrc.asp

Once you get it, simply run the Windows installer. My best experiences with this have been on completely clean machines. So, if you encounter any errors during setup, you might want to consider trying it on a completely "fresh" computer (if you have access to one).

What Do You Do With It?

In this section you'll find sample code for retrieving a stock quote using the .NET Compact Framework. Better yet, of course, you'll see how to use it to create a Visual Studio .NET project that you can compile and deploy to your Pocket PC or emulator.

Coding Your Project

In order to code your project, you must first create the graphical user interface (GUI), then populate it with the appropriate event-handling code.

Drawing the GUI

The Forms Designer portion of Visual Studio .NET isn't yet usable with the Smart Device Extensions, as of the Technology Preview release. In order to create a GUI for your application, then, we're going to have to exercise a bit of a workaround:

1. Launch Microsoft Visual Studio .NET
2. Create a new Windows application project named "GUI Builder"
3. Size Form1 to be 240 x 320 (the size of a Pocket PC screen)
4. Drag a textbox, label, and button onto Form1
5. Press F7 to enter Code View
6. Press Ctrl-A to select all of the code, then Ctrl-C to copy it to the clipboard
7. Close this project
8. Create a new Pocket PC Visual Basic Project called "NETcfQuote"
9. Verify that the project opens in Code View
10. Press Ctrl-A to select all of the code, then Ctrl-V to paste the code from step #6 above
11. Comment out the line of code that begins "Me.AutoScaleBaseSize"

At this point, you've transferred the GUI created in the GUI Builder project into your NETcfQuote project. With any luck, this workaround will not be needed in later versions of the Smart Device Extensions.

Mongoose Technology

www.portalstudio.com

Walking Through the Code

We're attempting to write a mobile client for a .NET Web service. It makes sense, then, that one of the first things we should do is add a reference to our .NET Web service to our project. One of the most wonderful parts of the Smart Device Extensions for Visual Studio .NET is how easy it makes to reference Web services for device-based code:

1. Right-click "References" in the Solution Explorer
2. Enter the address of the .NET Web service we created in last month's article (http://localhost/Chapter10Server/Service1.asmx)
3. Once the page has loaded, click "Add Reference"

You're now ready to access the get Quote method on this Web service from a remote device just as easily as you would normally reference methods on a local component! Could it get any easier? Chalk one up to Microsoft's side in the battle for simplicity.

At this point, you're ready to add the code from Listing 1 to your application. Simply copy it, exactly as it appears, immediately below the line that reads:

```
#End Region
```

As you might have guessed, this code is intended to run whenever the single button on the application's GUI is clicked. First, it declares a reference to the remote Web service. Then it calls the getQuote method on this reference, passing in the contents of the textbox as its first parameter, and an empty variable – price – as its second. If the method returns successfully, the label is populated with the contents of price, otherwise an error message is placed in it.

Compiling and Deploying

In order to compile and test your application, first select "Pocket PC emulator" from the dropdown that says "<select device>." Then, press F5. If there are errors, Visual Studio .NET will tell you that there were problems with the build and prompt you to continue or abort. If compilation succeeds without a hitch, the new Pocket PC emulator will launch and your application will appear.

At this point, you should enter the symbol of a NASDAQ stock in which you are interested into the textbox on the application's GUI. Then, click the button. Shortly, the most recent price of that stock should appear in the label on the display.

In order to try your application on a real Pocket PC, you need only:

- Place your Pocket PC in its cradle (attached to your computer via a cable)
- Choose "Pocket PC Device" from the dropdown instead of "Pocket PC Emulator"

Java 2, Micro Edition (J2ME)

We'll conclude our look at mobile .NET Web services by demonstrating how they may be invoked in a truly cross-platform manner by using the Java platform.

What Is It?

Java is Sun Microsystems' platform for creating software capable of running on a wide variety of hardware and operating system combinations. Software written completely in Java can often run on systems as different as UNIX and Windows without even requiring recompilation.

With the release of Java 2, Sun divided their platform into three distinct flavors:

- Enterprise Edition (J2EE)
- Standard Edition (J2SE)
- Micro Edition (J2ME)

J2SE might be referred to as "regular Java" – it's intended for use with standard, desktop computer applications. J2EE, on the other hand, is intended for the creation of server-based software, such as Web applications and distributed, component-based applications.

J2ME differs from both of the above by focusing on Java code intended for execution directly on mobile devices. In this sense, it's Java's direct analog for Microsoft's .NET Compact Frameworks.

How Do You Get It?

The great thing about J2ME is that almost all of the software needed to create applications for it is freely available for download off the Internet. On the other hand, the software needed in order to run these applications on the devices must typically be hard-wired into devices by the manufacturer (as opposed to being installed after-the-fact by end-users).

Acquiring the Bits

The first thing that you must download is the Java 2 Software Development Kit (SDK). This is the bit that takes standard text files full of Java sourcecode and converts them into binary files full of Java bytecode. You can get it from http://java.sun.com/j2se/1.3/Text.

The second thing that you should try to lay your hands on is the J2ME Wireless Toolkit. Although not absolutely essential for creating J2ME applications, it contains all of the extra libraries that you would need, plus a great assortment of emulators and development tools. Since it's also free, why not download it? Go to http://java.sun.com/products/j2mewtoolkit/Text.

Installation

Both of the above products use standard Windows setup engines to install and configure themselves on your machine. Just make sure that you install the JDK first, and then the Wireless Toolkit. (This is so that the Wireless Toolkit can configure itself to know exactly where your JDK is – thus making automated compilation and testing that much easier!)

What Do You Do With It?

Well, this is the point that we've been building to ever since the start of last month's article: actually invoking a .NET Web service from a mobile device that has nothing whatsoever to do with Microsoft.

The key to our approach, as you'll see, is a little bit of open-source code known as kSOAP. In this section, you'll see how to use kSOAP to build and deploy a J2ME application that can invoke .NET Web services from virtually any J2ME-powered device.

Discussing the Code

Let's begin by looking at Listing 2. It begins, like most Java code, with a series of import statements. The first statement imports the standard Java IO implementation for MIDP. The next few lines import various packages specific to MIDP Java. The final four import statements include support for kSOAP into our MIDlet.

kSOAP can be downloaded free from http://ksoap.enhydra.org. Also included in the package is some basic Javadoc documentation and sample code. In my book, *Mobile .NET*, I go into greater detail than present space allows about the workings of this fine technology. However, when you've finished entering and running this code listing, you should have a fairly strong basic grasp of how the technology is used.

Inside the FergieStock class, we first find several member variable declarations. The first of these sets the namespace, which must match the namespace we chose for our .NET Web service. After this, a number of GUI elements are declared, and added to the screen by the default constructor, as you can see.

The pauseApp() and destroyApp() methods are present only because the MIDlet specification requires them. The startApp() method is also required and, in this case, serves the purpose of displaying our initial form on the screen.

All of the real magic in our MIDlet happens when users press the single command button on the GUI to transmit their stock symbol up to our Web service. kSOAP's SoapObject component is used to serve as a proxy for the remote .NET Web service. Its ClassMap object is used to encapsulate the parameter that will be passed across the wire to the remote service. Both of these objects are passed into kSOAP's SoapEnv elope component, which will take care of serializing all of the kSOAP objects into appropriate SOAP XML for transmission to our .NET Web service.

MIDP's built-in HTTP connectivity is used to facilitate communications with the remote server. First, the XML for our SOAP request is transmitted, then the response is received back via the same HttpConnection method.

kSOAP's SoapEnvelope object provides the technologies needed in order to interpret the XML that's been returned to our client by the server. Once the price of the stock has been parsed out, it's placed into the label on our display. On the other hand, if an error has been raised, the word "Error" is placed inside the label.

Compiling and Testing

In order to try out the code shown above, follow these steps:

1. Save the code from Listing 2 as Fergie Stock.java
2. Save the code from Listing 3 as Fergie Stock.jad
3. Start the kToolBar application from the Windows Start menu (in the J2ME Wireless Toolkit program group)
4. Select "Open Project," then navigate to "FergieStock.jad" and choose "Open Project" again
5. Click Build and verify that there are no errors
6. Click Run

At this point, one of the J2ME Wireless Toolkit's emulators should open, allowing you to test your new J2ME stock quote client. Once you've entered a symbol and gotten back its price, congratulate your-self: you've just talked successfully to Microsoft .NET from Java!

Final Words

There you have it – how to invoke .NET Web services from virtually any kind of mobile device. If you're still hungry for more information about .NET technologies targeted at mobile computing, be sure to check out my MSN community at www. MobileDotNet.com.

Listing 1

```
Public Class Form1
    Inherits System.Windows.Forms.Form

    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As _
System.EventArgs) Handles Button1.Click

        Dim price As String
        Dim ws As localhost.Service1 = New
localhost.Service1()

        Label1.Text = ""

        If ws.getQuote(TextBox1.Text, price) Then
            Label1.Text = price
        Else
            Label1.Text = "No such stock!"
        End If

    End Sub
End Class
```

Listing 2

```
import java.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import org.ksoap.*;
import org.kxml.*;
import org.kxml.io.*;
import org.kxml.parser.*;

public class FergieStock extends MIDlet implements
CommandListener {

static final String serviceNamespace =
"http://pocketdba.com/webservices/";
Form mainForm = new Form ("Fergie Stock");
TextField fromField = new TextField ("Symbol :", "ALGX", 4,
TextField.ANY);
StringItem resultItem = new StringItem ("", "");
Command getCommand = new Command ("Get", Command.SCREEN,
1);

public FergieStock () {
mainForm.append (fromField);
mainForm.append (resultItem);
mainForm.addCommand (getCommand);
mainForm.setCommandListener (this);
}
public void startApp () {
Display.getDisplay (this).setCurrent (mainForm);
}

public void pauseApp () {
}

public void destroyApp (boolean unconditional) {
}

public void commandAction (Command c, Displayable d) {
try {
String from = fromField.getString ();
ByteArrayOutputStream bos = new ByteArrayOutputStream ();
SoapObject request = new SoapObject
(serviceNamespace, "getQuote");
request.addProperty ("symbol", fromField.getString ());
ClassMap classMap = new ClassMap();
classMap.prefixMap = new PrefixMap
(classMap.prefixMap, "tns",
"http://pocketdba.com/webservices/");
SoapEnvelope envelope = new SoapEnvelope (classMap);
envelope.setBody (request);
XmlWriter xw = new XmlWriter (new OutputStreamWriter
(bos));
envelope.write (xw);
xw.flush ();
byte [] data = bos.toByteArray();
System.out.println (new String (data));
HttpConnection connection = (HttpConnection) Connector.open
("http://localhost/Chapter10Server/Service1.asmx",Connector
.READ_WRITE);
connection.setRequestMethod (HttpConnection.POST);
connection.setRequestProperty ("Content-Type", "text/xml");
connection.setRequestProperty ("Connection", "close");
connection.setRequestProperty
("Content-Length", ""+data.length);
connection.setRequestProperty("SOAPAction",
"\"http://pocketdba.com/webservices/getQuote\"");
OutputStream os = connection.openOutputStream ();
os.write (data);
os.flush ();
InputStream is=((StreamConnection)
connection).openInputStream ();
int buffSize=200;
byte[] buff = new byte[buffSize];
ByteArrayOutputStream bytebuff=new
ByteArrayOutputStream(buffSize);
int eof=0;
while(eof!=-1) {
eof=is.read(buff);
if (eof!=-1) {
bytebuff.write(buff,0,eof);
}
}
is.close();
String response = new String(bytebuff.toByteArray());
System.out.println (response);
Reader reader = new InputStreamReader(new
ByteArrayInputStream(bytebuff.toByteArray()));
XmlParser parser = new XmlParser (reader);
SoapEnvelope resultEnvelope = new SoapEnvelope ();
resultEnvelope.parse (parser);
resultItem.setLabel ("amount :");
String result = (String)resultEnvelope.getResult();
result = result.substring(0,result.indexOf(".")+3);
resultItem.setText (" "+result);
os.close ();
reader.close ();
connection.close ();
}
catch (Exception e) {
resultItem.setLabel ("Error:");
resultItem.setText (e.toString ());
}
}

private static byte[] extractXml(byte[] extractFrom){
byte[] result;
int off=0;
boolean nonStop=true;
for(int i=0;nonStop;i++){
if (extra ctFrom[i]==60) {
off=i;
nonStop=false;
}
}

result=new byte[extractFrom.length -off];
for(int i=off;i<extractFrom.length;i++){
result[i-off]=extractFrom[i];
}
return result;
}
}
```

Listing 3

```
MIDlet-1: FergieStock, , FergieStock
MIDlet-Jar-Size: 40598
MIDlet-Jar-URL: FergieStock.jar
MIDlet-Name: FergieStock
MIDlet-Vendor: Sun Microsystems
MIDlet-Version: 1.0
```

Download the code at
sys-con.com/webservices

ParaSoft

www.parasoft.com/wsj2

Written by Michael Sick

Peopleclick and HR.NET

How can Web services and .NET work for developers today?

Though often spoken of as a technology of tomorrow, it's important to understand that Web services are already proving to be a key component of some of the products and projects of today.

This month's article examines how Web services and .NET are becoming critical components of Peopleclick's enterprise workforce management product line. We'll also look into the evolutionary forces that have led Peopleclick to select Microsoft's .NET platform. Finally, we'll examine current Web service technologies and those being considered for future use.

Before we start, it's worth noting that this article is not meant to be a detailed technical study of a Web service architecture. There are no code samples and concepts are discussed solely at the overview level. More technical questions can be forwarded directly to the author.

Author Bio:
Michael Sick is an independent Java architect helping clients solve complex product definition and design problems. He has more than eight years of experience in the construction of distributed information systems and Internet technology, holding positions including architect and VP of Development. He holds undergraduate degrees in both geology and political science from Guilford College.
MIKE_A_SICK@YAHOO.COM

A Little About Peopleclick

Peopleclick is a provider of enterprise workforce management services designed to help their customers recruit new employees, manage their existing workforce, and ensure compliance with the various affirmative action regulatory laws. All these services are Web-based and offered in an Application Service Provider (ASP) model. With over 300 employees of its own – and having recently achieved profitability – Peopleclick is getting noticed within its industry and in the ASP community in general.

The Workforce Management Suite

Peopleclick provides a suite of services designed to help companies manage all aspects of their workforce. The services focus on:

- ***Recruiting:*** Peopleclick offers services that allow human resource (HR) managers to define and execute a custom recruiting process. Services include defining open positions, posting the positions on the corporate Web site and automating background and credit checks.
- ***Workforce Planning:*** Once employees have been hired, Peopleclick provides services for the management of "human capital." The software tracks skills, roles, and personalities, facilitates performance reviews, and helps managers determine training needs.
- ***Affirmative Action:*** Many companies emphasize diversity and compliance with affirmative action regulations. Peopleclick's affirmative action software allows companies to determine their level of diversity and manage it effectively.

"The Peopleclick development environment can be summed up in one word: Microsoft."

The Development Environment

The Peopleclick development environment can be summed up in one word: Microsoft. On the client side, Internet Explorer 5.0 and above is supported for power users of the system. Technologies such as Javascript, XML, XSL, and the Microsoft XML-HTTP control enable the application to have a robust desktop-like user interface.

The middle tier is composed of several "Web farms" running Internet Information Server (IIS) and a thin layer of Active Server Pages that invoke COM+ components performing core business logic. The COM+ components are primarily built in Visual Basic but also have been written in C++ and C#. These components are responsible for all data retrieval and processing using ActiveX Data Objects (ADO), the Microsoft XML Parser, and general file I/O. The entire middle tier runs on Windows 2000 on various Intel based computers.

The third tier is a mix of SQL7 and SQL2000 running in a clustered mode on 4-way Intel boxes. Third-tier servers also provide services such as message queuing using MSMQ and content searching using the Microsoft Indexing Service.

A Web Services Evolution

As we've seen, Peopleclick has incorporated many facets of XML and .NET technology into their solution. This isn't the result of a desire to be "buzzword compliant" by keeping up with the latest technology fads. Instead, each of these technologies has been adopted to solve challenging real-world issues.

Starting With XML

Peopleclick's initial move towards Web services was their rearchitecting of the suite around XML technologies. As the first version of the Peopleclick suite was developed, it became clear that both flexibility and configurability were essential to the product's success. The system was redesigned using XML to represent core business objects and for customer configuration. David Anderson, Director of Technology, says "As a Web services provider, offering a high degree of configurability without changing your core code on a per-customer basis is essential in scaling the business. Using technologies like XML, XSL, and XML RPC requests has given us a powerful and flexible user interface and has enabled us to offer the configurability our enterprise customers demand."

One of the key features was the use of the XML-HTTP control by the Javascript in the browser layer. This allowed the UI to request data that was returned as XML and then parsed and/or styled, which enabled Peopleclick to offer a robust UI, and was in many ways a precursor to the use of SOAP in the product suite.

Standardizing on HR-XML

As Peopleclick began to formalize XML representations of their business objects, they realized that others in their industry were doing the same and so chose to extend their use of XML by helping to drive standards in the HR-XML consortium (www.hr-xml.org). The consortium's self-described mission is "the development and promotion of standardized XML vocabularies for human resources." These standards have facilitated a greater degree of

New Atlanta Communications

www.newatlanta.com

integration with third-party systems like Internet job boards and human resource management systems.

SOAP for B2B Communication

As the product suite matured, Peopleclick was driven to find the best way to exchange business objects and data to enable B2B communications with several partner companies. They turned to the then emerging SOAP Toolkit from Microsoft and began to integrate SOAP into their applications. A portion of their suite utilizes the SOAP Toolkit to enable B2B communication with several of their business partners that offer value-added services. In one case, Peopleclick offers its customers the choice of sending their postings to third-party job sites like Monster.com and Dice.com. Rather than build this functionality in house, Peopleclick integrated its suite with E-worker Technologies' (www.eworkertech.com) posting engine using SOAP enabled transactions (see Figure 1). Rick Purcell, CEO of E-Worker Technologies, states: "The use of SOAP as a data exchange model allowed E-worker and Peopleclick to trade value and services quickly and with a minimum of technical hassle. SOAP drastically reduces the overhead needed to exchange data and services." This model of rapid integration with third-party services is allowing Peopleclick to expand its product offering with lower engineering investment.

Moving Towards C#

Peopleclick is currently developing products using C# and the .NET development suite. The affirmative-action portion of the product suite is being rearchitected using C# as the core development language. Visual Basic has served Peopleclick well in this regard, but has some fundamental limits in threading and networking that make C# very attractive. The C# portion of the suite is expected to be in production by the summer of 2002. Peopleclick expects that many future projects will be developed using C# and VB.NET.

The Evolutionary Environment

As described above, Peopleclick's use of Web services technology was an evolutionary rather than a revolutionary process. It seems worthwhile to review some of the factors that Peopleclick described as key to success in this area.

Peopleclick credits in-house expertise and individual initiative with much of their success. Some developers became initial contributors to the HR-XML consortium while others have become industry recognized leaders in C# development. These efforts have made it possible for Peopleclick to examine emerging technologies early and to develop the expertise to train and mentor additional staff.

Another key factor is Peopleclick's strong relationship with Microsoft on many levels. Peopleclick's developers are part of the Microsoft Developer Network (MSDN) and participate in a number of beta programs for SQL Server, Windows .NET Server, and Visual Studio.NET. And as a member of Microsoft's Premier Support program, Peopleclick has the access it needs to Microsoft resources. Peopleclick CTO Jim Grundner states, "Our relationship with Microsoft is mutually beneficial. They provide us with excellent advice and consulting, while we provide them with a software platform that challenges and advances their .NET vision." Additionally, Peopleclick works closely with Microsoft's consulting arm to understand, prototype, and implement emerging technologies.

Whatever the method, getting started early and having access to expert advice is critical to incorporating new technologies into an existing product in an orderly manner. Remember that Peopleclick, as an Application Service Provider, must be able to upgrade its suite for all of its customers at the same time with little or no downtime. Peopleclick's evolutionary environment is founded on a history of success that has allowed experiments and individual initiative to shape their product for the better.

What's .NEXT ?

As the Peopleclick suite evolves, it seems that its ties to .NET and Web services will only deepen. One direction being considered is making use of Microsoft's Passport for user management and profiling. The benefit of outsourcing this service and offering single sign-on to Passport members is compelling. Another technology under consideration is the evolving BizTalk server for B2B messaging and transactions. Additionally, Microsoft's service management suite Application Center is also being considered as a way to ensure that existing and future Web services are well managed and monitored. Whatever products are eventually selected as part of the Peopleclick suite, we can be sure that Web services and .NET will play a critical role.

Summary

Peopleclick is a successful provider of enterprise workforce management products and they're betting that Web services and .NET will be a part of their continued success. They've been using XML and related technologies for over four years. Now they're starting to Web service–enable their products and are already reaping the rewards of lower development costs and faster time to market. The process of incorporating Web services into its products has been facilitated by strong vision, individual initiative, and excellent relationships with their technology providers. Peopleclick views its use of .NET and Web services as a strategic advantage and plans to continue to invest time and energy in this direction.

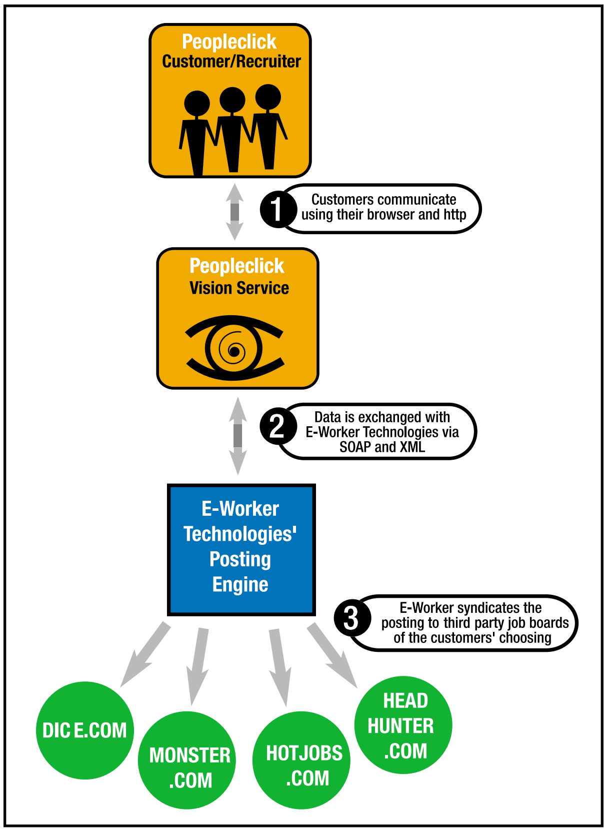

Peopleclick Customer/Recruiter
1 Customers communicate using their browser and http
Peopleclick Vision Service
2 Data is exchanged with E-Worker Technologies via SOAP and XML
E-Worker Technologies' Posting Engine
3 E-Worker syndicates the posting to third party job boards of the customers' choosing
DICE.COM
MONSTER.COM
HOTJOBS.COM
HEAD HUNTER.COM

FIGURE 1 SOAP for B2B Communication

LOOX

Software

www.loox.com

Written by Gunjan Samtani & Dimple Sadhwani

Security and the .NET Framework

Grave danger awaits if your security isn't robust

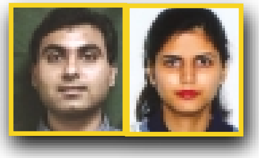
Labeled as the coming nirvana for enterprise application integration and business-to-business (B2B) integration, Web services technology is nonetheless vulnerable to a wide array of security threats such as denial of service and spoofing.

In this article, we'll review Web services security requirements, the factors that determine them, and how Microsoft's .NET Framework supports them.

Web Services as Interfaces to Business Methods

The common belief is that the security requirements of Web services are the same as those of a Web site. But this assumption completely underestimates the security requirements of Web services, which are, in fact, much higher than for a simple information-oriented Web site.

Here's why: a Web service is an interface that describes a collection of business operations that are network-accessible through standardized XML messaging based on Simple Object Access Protocol (SOAP), which is a standard messaging layer used to exchange XML documents in Web services. In other words, Web services expose applications supporting business operations, encapsulating business logic, and accessing business data over the network using invokable interfaces. Unless the Web services platform provides robust security features, it can pose grave dangers for the corporation as a whole.



Author Bios:

Gunjan Samtani, divisional vice president of information technology at UBS PaineWebber, has several years of experience in the management, design, architecture, and implementation of large-scale EAI and B2B integration projects. He's the primary author of *B2B Integration: A Practical Guide to Collaborative E-Commerce* (forthcoming 2002) and has presented research papers at several national and international conferences.
GUNJAN_SAMTANI@YAHOO.COM

Dimple Sadhwani, an independent consultant in the field of e-commerce technologies, 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. As well as coauthoring *B2B Integration: A Practical Guide to Collaborative E-Commerce*, she has also authored several articles in the field of Web services.
DIMPLE_SADHWANI@YAHOO.COM.

EAI and B2Bi

To understand the security issues involved with Web services, it's important to discuss the different domains for which Web services can be used. Web services can potentially be used for two distinct domains – enterprise application integration (EAI) and business-to-business integration (B2Bi). EAI is the process of creating an integrated infrastructure for linking disparate systems, applications and data sources within the corporate enterprise. B2Bi is the process of secured coordination of information among businesses and their information systems, enabling cross-enterprise business applications such as collaborative e-commerce, collaborative networks, supply-chain management (SCM), and customer relationship management (CRM) across multiple channels of delivery, including wireless devices and the Internet.

“The key security requirements for the usage of Web services are authentication, authorization, data protection, and nonrepudiation”

The security requirements for the EAI domain are a subset of those of B2Bi. It's much easier to control, manage, find, execute, and maintain Web services within an intranet than to use them over the Internet across the corporate firewall.

Some of the essential factors that determine the security requirements for the use of Web services are:

1. What's the purpose of the Web service?
2. Who are the subscribers? Are they trusted trading partners or can any company invoke the Web service over the Internet?
3. What's the level of access that the Web service provides to the underlying application? Should the access be based on authorization and entitlements?
4. Is the Web service transaction-oriented?
5. What's the network protocol through which the authentication and data transmission occur between the service requestor and provider?
6. Is there a need to guarantee that the sender of the Web service request and response message is the same as the creator of the message?
7. How many distinct entities are involved in the usage of the Web service, i.e., does it have an entity-chaining feature?
8. Is there an application- and component-chaining feature in the implementation code of the Web service?

Security Requirements for Web Services

The key security requirements for the usage of Web services are authentication, authorization, data protection, and nonrepudiation.

Authentication

Authentication is a security requirement that ensures each entity involved in the usage of a Web service – the requestor, the provider, and the broker (if there is one) – is what it actually claims to be. Authentication involves accepting credentials from the entity and validating them against an authority.

While Web services for EAI should have one level of authentication and rarely make use of encryption, Web services for B2Bi may involve multiple levels of authentication and should always use encryption. Of course, there will always be a tradeoff between robust security and performance.

Authorization

Authorization is a security requirement that determines whether the requestor has been granted access to the Web service by the service provider. Basically, authorization confirms the service requestor's credentials – checks that the service requestor is entitled to perform the operation, which may range from invoking the Web service to executing a certain part of its functionality.

Data Protection

Data protection is a security requirement that ensures that the Web service

request and response haven't been tampered with en route. Data protection requires securing both data integrity and privacy. It's worth mentioning that data protection doesn't guarantee the identity of the message sender.

In the case of B2Bi projects, zthe messages corresponding to Web service request and response should always be encrypted, using one or more of the following: cryptography, digital signatures, secured socket layer (SSL), and so on. The use of SSL should be avoided, as far as possible, for Web services used within the corporate network for EAI projects.

Nonrepudiation

Nonrepudiation guarantees that the sender of a message is the same as the creator of the message. This is very useful for Web

services used in the B2Bi domain, as otherwise a malicious sender can later disavow having created and sent a specific message.

How the .NET Framework Supports Security Requirements

The .NET Framework's security solution is built on the concept of managed code, in which the security rules are enforced by the common language runtime. CLR enables the execution of both managed code (typically verified for type safety and disciplined behavior of other properties) and unmanaged code (requires special privileges and permissions to run). Let's review how the .NET Framework supports the security requirements for Web services.

Authentication

For authentication, the .NET Framework supports the Microsoft Windows operating system identity authentication, HTTP, message authentication codes (MACs), Digest, Kerberos, and Microsoft Passport.

- ***Basic HTTP:*** In basic HTTP authentication, the authentication information passed over the network isn't encrypted; this mechanism should only be used for Web services for EAI projects.
- ***Message Authentication Codes:*** A MAC is an authentication tag (also called a checksum) derived by applying an authentication scheme, together with a secret key, to a message. However, it isn't foolproof; hackers can also generate these codes and misuse them.
- ***Kerberos:*** Kerberos, a network authentication protocol, is the default authentication security protocol of the .NET Framework. The Kerberos protocol uses strong cryptography so that a Web service requestor can prove its identity to a Web service provider and vice versa across an insecure network connection such as the Internet.
- ***Microsoft Passport:*** In the future, the .NET Framework will be supporting Passport and cookie-based authentication for Web services. Passport's identity service stores sensitive consumer information on Microsoft's servers, which will restrict its corporate usage, as companies will be unwilling to cede control of sensitive authentication information.

Authorization

The business logic of XML Web services in the .NET Framework is often implemented using COM components or managed classes. The .NET Framework allows defining authorization security on these components to be done administratively and/or programmatically.

- ***Role-based Security:*** Developers, system administrators, and component creators can define role-based security for components in the .NET Framework. A role corresponds to a specific set of access rights and privileges. Role-based security is defined using administrative tools and configuration settings.
- ***Programmatic Security:*** The security of a component (which may contain all the implementation business logic for the Web service), including the client's permissions and privileges, can also be defined programmatically in the .NET Framework. Based on specific permissions and privileges, such as authority to read and write files or read and write environment variables, a certain part of the component's code is executed.

Data Protection

For data protection, the .NET Framework supports cryptography, encryption, digital signatures, SSL, Kerberos, hashing, and random-number generation.

- ***Cryptography:*** This protects the privacy of sensitive information being transmitted over the Internet. Through the use of sophisticated mathematical formulas and computer algorithms, it ensures that the information is unintelligible if it falls into the wrong hands. The .NET Framework provides CryptoAPI, which includes features such as data encryption, support for PKCS #10, PKCS #7, X.509, and adding and retrieving certificates from certificate stores.
- ***Kerberos:*** After a Web service requestor and provider have used Kerberos to prove their identities, they can also encrypt all of their communications to assure privacy and data integrity as they exchange XML-based messages.
- ***Digital Signatures:*** .NET Framework supports the use of digital signatures as a security solution. A digital signature is the electronic equivalent of a pen-and-paper signature and is used for authenticating the sender of the message. It

provides a means by which information cannot be repudiated by binding a communication to the entity that signed it. It also guarantees that the message wasn't modified after it left the sender. Digital signatures rely on public key systems.

In the future, the .NET Framework will also support the XML Signature specification, an XML-based language for representing digital signatures.

• ***Secured Socket Layer:*** The .NET Framework supports SSL to encrypt authentication and data for HTTP and NNTP transmissions. SSL is the cornerstone security technology for the use of single-level Web services in the B2Bi domain. By "single-level," we mean that there's only one Web service provider and one Web service requestor. SSL can secure the entire channel between these two entities end-to-end. This ensures that none of the devices involved, such as computers and routers, can interpret or maliciously corrupt the information flowing through them in the form of encrypted XML documents.

However, the use of SSL for multilevel Web services in the B2Bi domain is very limited. "Multilevel" implies the participation of multiple entities in the use of a Web service such that each entity can interpret and modify the data. The encryption and re-encryption of data at each entity isn't a viable solution, as it severely slows down the performance of the entire system. XML Signature will be useful in such scenarios.

Nonrepudiation

Nonrepudiation, a term typically used in secured transactions, ensures that the different entities involvd in the usage of a Web service cannot escape their obligation. This validation provides greater confidence in the data confidentiality process, an important factor in the wide adoption for Web services in the future. For guaranteed nonrepudiation, apart from Web services requests and response message authentications, the sender and receiver authentications must also be performed.

The .Net framework provides support for all Web service ransactions to be digitally signed and stored in a audit trail that is tamper-proof. Further, all transactions can also be fully traced and profiled.

Security Is the Key

Secured interoperability for Web services is the key to its becoming a successful technology in the EAI and B2Bi domains. Companies should perform a full security audit and risk analysis of the chosen Web services platform before moving any business processes to Web services.

The .NET Framework does provide support for the key security requirements of Web services: authentication, authorization, data protection, and nonrepudiation. But some of the security solutions are still evolving, so remember to stay tuned.

EnginData Presents

2002 Developer Market Survey Reports

$2,995
Java Developer Market Survey 2002
$2,495
Web Services/XML Developer Market Survey 2002
$1,995
Wireless Developer Market Survey 2002

Our comprehensive reports offer insight and strategy to guide your most critical business decisions in today's fastest growing technologies...

✓ **Establish your product and marketing strategy**

✓ **Understand your customer's needs**

✓ **Evaluate Technology & Trends**

engindata.com

engindata✓ RESEARCH

Written by Dave Rader

The Largest Web Application in the World

Is this what .NET My Services will one day become?

"One key area in which Microsoft must prove itself is in the security of the data stored by the .NET Services. Because Microsoft is a favorite target of hackers, and the personal, transactional, and payment data will be so valuable, this data store is sure to be plagued by people trying to break the security system. Passport has been shown to contain possible security vulnerabilities with cross-site scripting and automatic logins. In order to convince users and partners of the safety and privacy of their data, Microsoft will need to be vigilant against any possible threats."

As part of its .NET initiative, Microsoft has developed a set of Web services – originally codenamed "Hailstorm" – called .NET My Services.

The hope is that these services will have the potential to spark the creation of integrated applications that enable users to share data and collaborate more easily. Certainly .NET My Services will be a major example of the new, open Web services design currently sweeping the industry.

Author Bio:
David Rader is a partner with Fusion Technologies, a consulting company specializing in application integration and Web services development. Most recently, he's been developing XML-based applications in the small business portal space.
DAVIDR@FUSIONTECH.COM

What Are They?

The first set of services to be offered includes basic information storage, such as profile data (name, address), contacts and addresses, payment data, application settings, and document storage, along with more functional services, like calendaring, inboxes, and messaging. Many of these are next-generation implementations of existing, limited-use Web sites, but with easy application integration options.

Open Architecture

The .NET Services are designed to be cross platform and are built using open Web services standards. Accessible using SOAP over HTTP, the services return standard XML data packets with well-defined schemas. Microsoft has demonstrated .NET Services being accessed by Windows, UNIX, and Macintosh computers, as well as Palm OS and Pocket PC handheld devices.

All user data is stored in a central repository. This means that users can access documents, settings, and personal information from any location or device with an Internet connection. Over time, as applications are built to leverage this universal access to information, users can expect better integration between Web sites, desktop applications, and businesses.

"In addition to making it easier to reach customers with important information, the information can reach the customer in the right place at the right time"

Interlocked Services

.NET My Services are applications per se – they'll have pieces of user interfaces, for instance to register and manage .NET Services subscriptions. They'll also share user interface elements with partner applications, such as credit card entry screens, and are designed to be building blocks that can be used by Microsoft and any other business to quickly and easily assemble personalized online applications.

The various .NET My Services work together to provide access to data or user touchpoints for .NET My Services–enabled applications.

Next-Generation Passport

The .NET Passport authentication service forms the foundation for single sign-on applications. This service is the next-generation incarnation of Microsoft's Passport Web site, which now has over 150 million registered users, and is already integrated into many online applications. The .NET Passport system will enable affiliated Web sites or applications to authenticate a user, establish unique user credentials, and (with the user's permission) retrieve profile information and preferences.

Profile, Contacts, Calendar, Inbox

Other .NET My Services include an upgraded version of the Microsoft Wallet, an online address book, calendar, and, of course, e-mail. Each of these has taken a step forward in terms of interoperability, and in many cases they have been revamped from a specific application to working in a more general-purpose setting. For instance, the .NET Presence service can be thought of as a super version of the "friends" features available in online gaming and instant messaging applications. In addition to the common ability to set your current online/o ffline/ busy status, you can also choose how you want your .NET Alerts to be delivered to you.

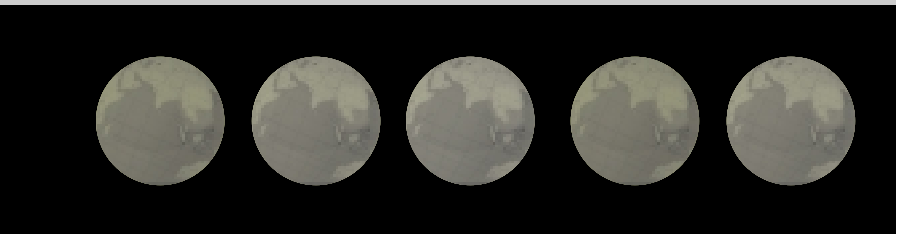
Extensibility

Although officially .NET My Services isn't called a framework or platform, Microsoft is encouraging third-party developers and partners to extend the services already announced with compatible Web services.

Envision, for example, is an investment portfolio service. By leveraging the user's preferences and notification choices, such a portfolio-tracking service could send detailed news stories or price change alerts to the specific device a user has configured. Or imagine a wish list service that could automatically send your list to selected members of your .NET Contacts around the time of your birthday.

No Duplicate Data Entry

Perhaps the most useful benefit of the .NET



My Services system would be the elimination of the need for duplicate data entry. In some of the most successful consumer software products (for instance, QuickBooks), this is a number one tenet, and certainly most Internet users today would benefit from having to enter their address and billing information in only one location.

Of course, having all your personal data in one central location immediately raises concerns about who'll have access to that data. The privacy and security of the information that users enter will thus be key factors in the success of the .NET My Services in the marketplace. Microsoft is addressing this challenge directly and in a unique way, by providing the user with control over who's allowed access to the data. Because the .NET My Services data store is centrally managed, all requests for access to the information can be authenticated. In order for a partner Web site or application to query the .NET My Services data store, the user must explicitly grant access to the application. This access can be revoked, enabling users to cancel a subscription or turn off a service if they choose.

Putting the User in Control

This user control is a new idea in the online realm. Web sites today have a privacy statement, or other "terms of use" for the data that users enter. Few of these policies prohibit selling the information, or sharing it with partner sites. In setting the terms of use for the .NET My Services, Microsoft has guaranteed that users will remain in control of their personal information.

What Does the Future Hold?

While it will take several (or many) years for the full effects of .NET My Services to be seen, the vision of comprehensive digital access to information from anywhere is very exciting.

When scheduling a meeting or trip, it should be possible to simply review both parties' calendars and find a convenient time, even if you're standing in the middle of a convention floor. With a single change to your .NET Alerts preference, you can have important messages routed to your cell phone rather than your e-mail inbox, without any change to the configuration of the sending applications.

Benefits To Partners

Companies who embrace .NET My Services will find a streamlined registration process for users, reduced application development time and costs, and the opportunity for more rewarding customer interaction. By leveraging the information already entered into the .NET My Services data store, a Web site may be able to pre-fill most of the registration information, and skip to providing personalized service or information to the user. By providing these streamlined interactions, an application can more quickly "win" the user and demonstrate its value potential, thereby driving higher customer satisfaction and increased revenue.

Benefits to Developers

.NET My Services will benefit application developers in the same way that the component "revolution" improved desktop development. It will be possible to plug in user authentication and profile management modules and notification services, and easily store settings for use on other devices. These services will enable developers to deliver increased functionality in their applications in less time.

As an example, consider the process of communicating changes in information to your customers. Most Web applications have built an e-mail system to send notices when something important occurs. The full development cost was spent to get a text message sent via SMTP to the customer's mailbox.

If a new application leverages the .NET Alerts service, sending a text (XML) message to the .NET My Service via HTTP will achieve significantly

greater functionality for about the same development cost. Rather than being limited to one delivery mechanism (e-mail), your message can be delivered via e-mail to a pager, cell phone or instant messenger. In addition, customers have the ability to configure where and how they would like to receive alerts via .NET Presence, functionality that you didn't have to rebuild for this particular application. (see Figure 1).

Benefits to Customers

So, in addition to making it easier to reach customers with important information, the information can reach the customer in the right place at the right time, making it more actionable. Customers can choose to receive traffic alerts on their cell phone so that while they're driving they can react to the traffic jam, rather than have the notice merely sit in their inbox on their desktop. By making your information more available and more actionable, your service becomes more useful to your customer.

Benefits to E-Business

A final, less tangible, benefit to businesses is the repackaging of online services in a new business model and a new development paradigm. By breaking down some of the resistance to fee-for-service systems and showing that consumers are willing to pay for a useful service, Microsoft is opening new doors for online services. In addition, .NET My Services provides a model for other businesses on how to provide open, secure, accessible services to a consumer population.

Challenges

Microsoft offers a compelling vision of simplified use, deep integration, and pervasive information sharing; however, there are significant challenges to making this successful. If Microsoft proves able to achieve even a fraction of its goal for the use of .NET My Services, they'll quickly become the largest Web application in the world. With hundreds of millions of users, and billions of transactions moving through the system, the engineering requirements are enormous.

Microsoft has experience with very large-scale Web sites already (microsoft. com, msn.com, msnbc.com, and hotmail. msn. com are all among the top 10 most-trafficked sites). The company has publicly shared some of the massive redundancy and distribution of the .NET My Services platform to alleviate concerns over reliability and scalability. With vast arrays of load-balanced servers and multiple physically dispersed data centers, everything is built on the lessons learned throughout the industry on ways to handle the expected high traffic volumes.

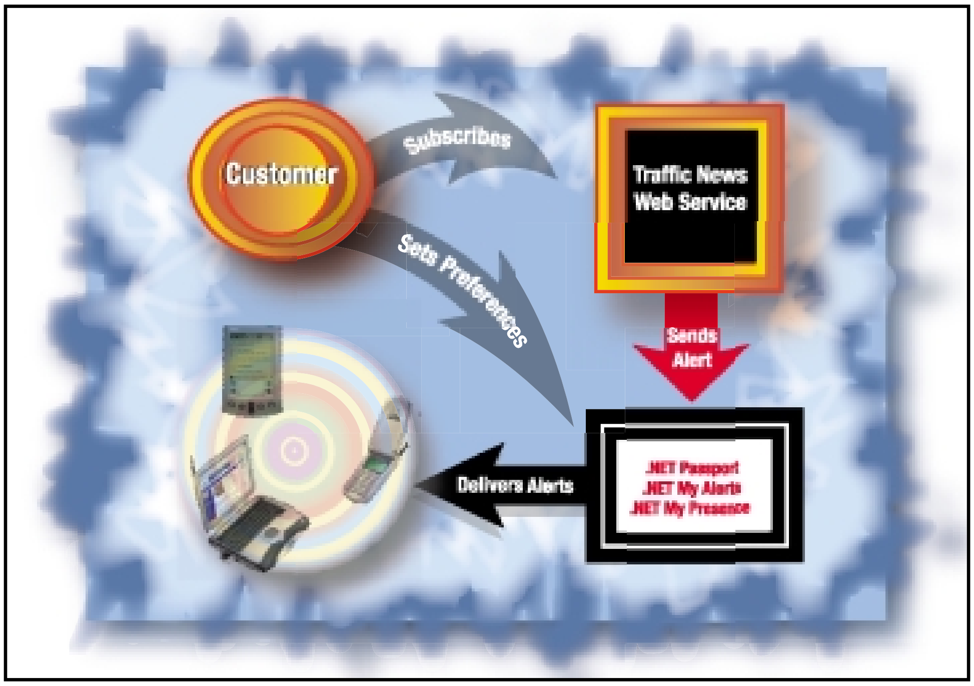


FIGURE 1 | .NET My Services

The .NET My Services interaction model helps to limit some of the fears around distributed denial-of-service attacks. By requiring each partner application to sign up with Microsoft, and receive a unique Site ID, the "frontline" servers (or hardware packet filtering devices) can easily screen out malicious requests from nonaffiliated sources without taxing the application servers. In addition, there are built-in bandwidth and transaction throttle controls that can be used to slow or completely stop requests from what appear to be affiliated sources that exhibit suspicious behavior patterns.

Even with the rigorous engineering and thorough design to ensure the .NET My Services platform is able to meet the scalability and reliability challenges, fears linger about possible failures. Whenever businesses begin relying on an outside party to authenticate users, carry vital message to customers, and store payment information about customers, the reliability of that third-party becomes a critical factor in success or failure of the business. If an outage of the .NET My Wallet service were to prevent participating sites from taking orders from customers during a busy period, those sites would stand to lose millions of dollars in revenue, and customers would turn to alternative suppliers. In such a situation, the afflicted sites would certainly move to quickly eliminate their dependency on the .NET My Services platform.

One key area in which Microsoft must prove itself is in the security of the data stored by the .NET Services. Because Microsoft is a favorite target of hackers, and the personal, transactional, and payment data will be so valuable, this data store is sure to be plagued by people trying to break the security system. Passport has been shown to contain possible security vulnerabilities with cross-site scripting and automatic logins. In order to convince users and partners of the safety and privacy of their data, Microsoft will need to be vigilant against any possible threats.

Summary

Potentially, .NET My Services could serve to usher in a new wave of connected applications and services, in which users will have a wholly new level of control over the information they give to service providers, and gain a new ability to easily share that data between desktop applications, Web sites, and online services – regardless of where they are or what device they use.

JAVA: INDUSTRY NEWSLETTER

JAVA DEVELOPERS JOURNAL Industry >Newsletter

Monday, December 10, 2001

BROUGHT TO YOU BY...

aligo Wireless Development Strategies - A Free White Paper. How can you use J2EE/J2ME to successfully deploy Wireless Applications? Aligo, the leader in Java based wireless application servers, offers a free technical paper on Java and Wireless Development. Click Here! for more information.

JAVA DUDES

JDJ NEWS DESK EXCLUSIVE

Oracle Targets Top Spot With New Oracle9iAS and JDeveloper

(December 10, 2001) – I returned just yesterday from a fantastic trip to San Francisco for Oracle's OpenWorld conference. It wasn't clear what to expect when I accepted Oracle's invitation to give a presentation, but I was delighted to find the show had a high-energy crowd that was very tuned in to Java, especially J2EE and Web services. The whole show was great, but the most interesting and important items came from Oracle itself - Oracle9i Application Server Release 2 and Oracle9i JDeveloper.

WEB SERVICES: INDUSTRY NEWSLETTER

WebServices JOURNAL Industry >Newsletter

Monday, December 3, 2001

BROUGHT TO YOU BY...

ORACLE Catch the Oracle9i JDeveloper wave and see how built-in profilers and Code Coach makes your Java code tighter and faster than ever before. Based on the latest J2EE and XML standards, JDeveloper combines Java coding, UML modeling, and RAD on NT/Linux/Unix. Click Here! for more information.

ONLINE EXCLUSIVE: FULL "KEYNOTE PANEL" TRANSCRIPT

XML: INDUSTRY NEWSLETTER

XML JOURNAL Industry >Newsletter

Tuesday, December 18, 2001

BROUGHT TO YOU BY...

flamenco Secure Web services. Reliable Web services. Manageable Web services. If you need Web services then you need a Web services network. Security, guaranteed delivery, and management services with no code changes. Plug into the Flamenco Network and get going! For a limited time, bring your Web services and test-drive the Flamenco Network for free.

XML NEWS

Xerces C++ XML Parser Version 1.6.0 Provides Full Support for W3C XML Schema Recommendation

(Toronto) - The XML Parsers Development division of the IBM Toronto Laboratory has announced the much-anticipated release of the Apache Xerces-C++ XML parser version 1.6.0, including full support for the W3C XML Schema Recommendation version 1.0. Xerces-C++ is a validating XML parser written in a portable subset of C++; Xerces-C++ makes it easy to give your application the ability to read and write XML data.

THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

FREE E-Newsletters

SIGN UP TODAY!

Go to www.SYS-CON.com

The most innovative products, new releases, interviews, industry developments, and plenty of solid *i*-technology news can be found in SYS-CON Media's Industry Newsletters. Targeted to meet your professional needs, each e-mail is informative, insightful, and to the point. They're free, and your subscription is just a mouse-click away at www.sys-con.com.

SELECT THE INDUSTRY NEWSLETTERS THAT MATCH YOUR NEEDS!
CHOOSE ONE – OR TRY THEM ALL!

JAVA Industry >Newsletter	WebLogic Industry >Newsletter
WebServices Industry >Newsletter	WebSphere Industry >Newsletter
XML JOURNAL Industry >Newsletter	COLDFUSION Industry >Newsletter
wireless Industry >Newsletter	LINUX WEEK Industry >Newsletter

Don't Delay!
Subscribe for FREE!
at www.sys-con.com

Exclusively from the World's Leading *i*-Technology Publisher SYS-CON MEDIA

From Subroutines to Web Services – An Evolutionary View

Beyond Client/Server Computing: The Next Generation of Life on the Web Is Here

In many respects, Web services are nothing new. They are just a natural evolution of an approach to building systems that dates back 40 years. On the other hand, they hold the promise of truly transforming computing in the same way that client/server computing did over the past 15 years. This article tries to address the question, "What's all the excitement about?"

What is a Service?

Let's begin by defining the term "service." A service is a reusable piece of self-contained logic that knows how to do a task, but does not know why it is being called upon. A water fountain is a good example of a service: it is quite skilled at providing drinking water, but it has no idea that its client is thirsty.

Software developers long ago recognized the need for services. Without reusable and self-contained programming logic, they had to write the same code multiple times, and then maintain each of these multiple instances. In the early days of computing, when systems were limited to a single host platform, reusable services were called subroutines or external program calls. Regardless of the name, though, these were early versions of services.

As computing spread across multiple computers, the need arose for reusable services that could operate across machine boundaries. An entire industry evolved around this need, which became known as client/server (i.e.,client/service) computing. Fortunes were made by companies that built tools to help programmers build systems that interoperated across machine boundaries. Note that the concept of a service had not changed at all. Only the underlying environment had changed – transitioning from single to multiple machines.

Web Services

The evolving maturity of the Web takes the notion of a service to yet another level. In the world of client/server computing, service invocations are typically made within a single, private network, or at least within a pre-established network of partners. In addition, client/server components are generally "tightly coupled," which means that the client needs to have registered to use the service in advance. (This is analogous to a water fountain working for you only if you have registered in advance for its use.)

Web services, on the other hand, are typically invoked across the Internet or an intranet, and are always loosely coupled. In other words, there is no technical requirement for a Web service to know in advance which clients will access it. Indeed, new clients can "add themselves" simply by using the correct calling parameters and demonstrating proper authorization.

Since Web services can be made publicly available so easily, they promise to bring new possibilities to the way organizations work with each other, transforming B2B commerce over the next few years. This is the most common view of Web services, and is what most of the excitement is about.

Interestingly, though, Web services are emerging even more quickly as an integration mechanism within individual organizations for Application-to-Application communications (A2A). The loosely coupled architecture of Web services delivers a flexible, extensible architecture for companies trying to integrate disparate systems. Important companies in this space include webMethods, Tibco Software, SeeBeyond Technology, and Vitria Technology.

For externally-accessed Web services, the idea is that a service can be placed onto the Web, where any organization's applications can find it and use it easily – much like a drinking fountain. Once again, there has been no change to the concept of a service. Rather, the underlying environment in which services operate has changed fundamentally. And these changes in turn fundamentally transform the way we do business.

Some Examples

Let's take a look at how Web services are already beginning to change the face of computing to get a taste of this new technology's power:

An early example of Web services is Dun & Bradstreet's Global Access Toolkit (reviewed in ***WSJ,*** Vol. 1, issue 1). D&B provides creditworthiness and other information about businesses worldwide. Last year they implemented a Web services system (Global Access) that allows authorized customers to access D&B data across the Internet as if that data was local to their own system. For example, a commercial bank's loan application system might call out to Global Access with a client company's key identification information (name, address, etc.) while the rest of the loan application is still

Author Bio
Andy Astor has held technology leadership positions in such diverse industries as financial services, software development tools, and integration consulting. Most recently, he managed worldwide customer access systems for Dun & Bradstreet, where his work included the development and launch of Global Access, one of the earliest commercial Web services.
AASTOR@HOME.COM

being entered by a user. Within seconds, the loan system receives D&B's credit rating, enabling a credit decision to be rendered even before the application is completed.

A much more broad-reaching Web services example is Microsoft's .NET My Services formerly code-named Hailstorm. Based on Microsoft's .NET platform, My Services promises to put many individual services into Web service form, including management systems for contact lists, calendars, inboxes (both e-mail and voice mail), money, and far more. Web services are so fundamental to Microsoft's strategy that they openly speak about redefining the company over the next five years as a provider of Internet software-based services, rather than a software producer.

In truth, nearly every developer-centric software company has jumped onto the Web services bandwagon. Major platforms include Microsoft's .NET, Sun's Open Net Environment (Sun ONE), IBM's WebSphere, BEA's WebLogic, Cape Clear, Iona, SilverStream, OASIS, Apache, Borland, and many others.

Buzzwords – Four Key Technologies

Now that we understand the conceptual definition of a Web service, I'll mention the key underlying technology standards upon which Web services are based. The technologies listed below reflect current thinking, and are widely supported. Bear in mind, however, that other variants exist; this is a fast-changing world. That said, following are the key technologies behind Web services today:

- ***XML (eXtensible Markup Language):*** An extension to the HTML language that allows data definitions to be transmitted along with the data itself. Prior to XML's development, it was difficult to write software that would interpret the contents of a Web page because the pages simply contained text and graphics. XML allows the individual fields in a page to be described, so that a piece of software can understand them. XML is by far the most universally accepted standard for Web service enablement.
- ***SOAP (Simple Object Access Protocol):*** A standardized way of using the XML language to call a Web service. Just as the English language needs grammatical rules to be understood, SOAP is a "grammar" that uses XML to call a Web service.
- ***UDDI (Universal Description, Discovery, and Integration):*** A central directory of Web services where services can be registered, much like the Yellow Pages. This permits potential customers (i.e., automated programs) to browse available services, and identify the location of these services and what information they require. For example, a system looking to purchase 100 tons of steel can browse a UDDI directory, find the services that vend steel, and submit price queries instantly and automatically.
- ***WSDL (Web Services Description Language):*** A standardized way in which the interfaces (i.e., inputs and outputs) of Web services are described. A WSDL file is generally part of a UDDI directory entry, and thus enables the client software to determine how to use the published service.

Key Issues

The power of Web services is no longer in question. Over the next few years, they will fundamentally change the way most companies build and use software. Of course, as with any important new technology, Web services also raise new challenges. I will touch on just a couple in the following paragraphs. Each of these items could easily fill a complete introductory article in its own right, but it is worthwhile to at least mention them here.

- ***Security:*** When we open our services to the world, all sorts of security issues arise. Some obvious examples include authentication, authorization, and bill/payment processing. Every situation is different, but all require a security analysis.
- ***Discovery:*** A Yellow Pages for Web services is very powerful, but also complex. There is a large difference between publishing phone numbers categorized by business and publishing the very mechanisms for doing business with companies. Standards are not yet firm in this area, and they have a long way to go before reaching universal acceptance. In addition, there is a real question as to how popular UDDI or other methods will be beyond a certain class of universal services.

Conclusions

My hope is that this article has communicated two key messages that might at first seem contradictory. First, Web services are nothing new; they are simply the next step in service-oriented computing, which has been evolving for more than 30 years. There is no great mystery here, and they are not the solution to world hunger.

The second key point is that Web services change everything. They provide the mechanisms to automate the Web, and to bring enormous efficiencies to the marketplace. All technology leaders should begin now to look for the B2B initiatives within their organizations that can benefit from this key technology. Web services may become the most significant new technology category for many companies during the next several years.

Many of us are excited about Web services. But be cautious: they do not solve all of your problems. Do not let your over-enthusiastic developers make them the answer to all things IT. If it doesn't make sense to you, don't do it. Also, remember that Web services are in their infancy and it would be unwise to over-commit. Put a toe in the water. Get some experience. Experiment. Don't lock into a single vendor until you understand the domain.

"**Web services are so fundamental** to Microsoft's strategy that they openly speak about **redefining the company over the next** five years as a provider of Internet software-based services, **rather than a software producer**"

Author Bios

Kyle Gabhart is the director of the Java Division of Objective Solutions (www.objectsoln.com), a high-end engineering services company based in Richardson, TX. Kyle is a prolific writer, with more than a dozen technical articles and books to his name and he is the original contributor of the kXML-RPC source code, which is now part of the EhydraME platform.
JAVA@OBJECTSOLN.COM.

Jason Gordon is an Associate Architect for Verizon's Technology Integration and eInfrastructure group (www.verizon.com). He is a core member of the KXML-RPC design team for EnhydraME and also serves as Region 5 Telecommunications Chair for the National Society of Black Engineers (www.nsbe.org). J
JASONTGORDON@HOTMAIL.COM

written by Kyle Gabhart & Jason Gordon

Wireless Web Services with J2ME Part II

SOAP or XML-RPC? The answer depends on your needs

Last month in Part I (*WSJ* Vol. 2 Issue 1) we discussed J2ME and accessing Web services from wireless devices using the XML-RPC protocol. In this article, we will consider SOAP as a vehicle for accessing Web services from wireless devices, comparing and contrast-ing it with XML-RPC. Our sample application will again be a J2ME midlet, however, we will use EnhydraME's kSOAP rather than kXML-RPC to provide the protocol's implementation.

Overview of SOAP

The Simple Object Access Protocol is, according to the 1.1 specification, "a lightweight protocol for exchange of information in a decentralized, distributed environment." The protocol is entirely based on XML, vendor-neutral, and one of the cornerstone technologies in the Web services revolution. It is quite similar to XML-RPC, but we will examine that more closely in the next section.

SOAP was originally conceived and developed at Microsoft between 1998 and 1999. SOAP did not, however, gain widespread attention until DevelopMentor, IBM, Lotus, and Microsoft submitted the SOAP 1.1 version to the W3C on April 26, 2000. With both IBM and Microsoft behind it, the industry began to give SOAP some serious consideration. As of this writing, the SOAP 1.2 specification is being drafted by the W3C's XML Protocol Working Group. The latest draft of the 1.2 spec can be found at www.w3.org/TR/soap12/.

When examining SOAP, it is important to identify the three main components of any SOAP message: the SOAP envelope, rules for encoding data, and a request/response in-

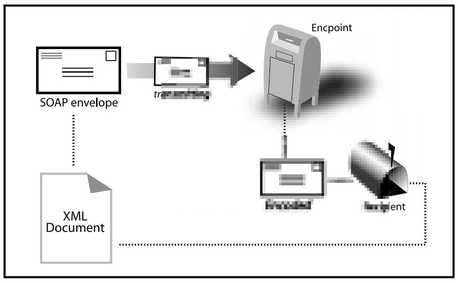


FIGURE 1 SOAP messaging architecture

teraction mechanism. The SOAP messaging architecture can be compared to a postal system (see Figure 1). A document is enclosed in an envelope and that envelope is transmitted via the transport mechanism, the mail system in our analogy, or HTTP across a network in the case of SOAP. With the mail, a

Special Introductory Offer SAVE $13.89*

WebServices JOURNAL

The world's leading independent Web Services information resource

Get Up to Speed with the Fourth Wave in Software Development

- Real-World Web Services: Is It Really XML's Killer App?
- Demystifying ebXML: A Plain-English Introduction
- Authentication, Authorization and Auditing: Securing Web Services
- Wireless: Enable Your WAP Projects for Web Services
- The Web Services Marketplace: An Overview of Tools, Engines and Servers
- Building Wireless Applications with Web Services
- How to Develop and Market Your Web Services
- Integrating XML in a Web Services Environment
- Real-World UDDI
- WSDL: Definitions and Network Endpoints
- Implementing SOAP-Compliant Apps
- Deploying EJBs in a Web Services Environment
- Swing-Compliant Web Services
- and much, much more!

Only $69.99 for 1 year (12 issues)
Newsstand price $83.88 for 1 year

THE NEW PROGRAMMING PARADIGM???

November 2001 Volume 1 Issue 2

WebServices JOURNAL

WSJ2.COM

FULL CONFERENCE COVERAGE

web services EDGE conference & expo

pg. 44

SEPT. 23-26 IN NYC

A Web Services Success Story

DUPONT PROTECTIVE COATINGS

pg. 10 by Catherine Marchand

From the Editor
The Information Paradox
by Sean Rhody pg. 5

Product Review
Cape Connect Two for J2EE
by Cape Clear
Reviewed by Brian Barbash pg. 16

Industry Insight
Mission Critical Web Services
by Norbert Mikula pg. 30

From the Industry
What's Happening in Web Services
by Steve Benfield pg. 60

News
pg. 64

Content Management & Web Services

eContent Services–the services behind Web Services

Page 6

Business Basics: Web Services Fundamentals: From Whence It Came *A look at future possibilities* Phil Karecki 18

WSJ Feature: Web Services: Enabling Technology or New Programming Paradigm? *The new integration model* Rebecca Dias 22

Business Collaboration: Understanding the Depth and Breadth of the Business Opportunity *Revenue growth* David Russell 26

Technology: Information Syndication Using Web Services *A dynamic way to share data and boost profitability* Darren Govoni 34

WSJ Feature: A Web Services Container *A better environment for real-world deployment* Ben Bernhard 38

Java: JAXM: Interoperable SOAP Communications for the Java Platform *Extensibility expedites e-commerce* Dave Chappell 48

Another View: The Real Niche for Web Services Part 2 *Tying up the loose pieces of evolving systems* Kurt Cagle 52

Open Source: Atoms vs Bits and the Digital Middle Mile *Agreement on standards expedites sharing among partners* Noel Clarke

UDDI: Understanding the UDDI tModels and Taxonomies *The key integration point*

SYS-CON MEDIA

www.wsj2.com

SYS-CON MEDIA

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of Web Services.

**Offer subject to change without notice* zip code identifies the locale an envelope should be delivered to; with SOAP, HTTP header data provides such routing information. At this point, the analogy breaks down. Whereas the "last mile service" provided by a postal carrier is directed by the street address on the envelope, it is information encoded in the body of the XML document itself that ultimately directs the SOAP request to the specific service on the server.

XML-RPC vs SOAP

Although SOAP and XML-RPC have similar roots (XML-RPC is based on a subset of the original SOAP spec that was developed at Microsoft in the late 1990s), they are very different animals. Both XML-RPC and SOAP are XML-based protocols for communication and data exchange, so when should one be chosen over the other? We'll take a look at their strengths and weaknesses, and then evaluate these two protocols from a business perspective.

XML-RPC is an extremely lightweight mechanism for invoking XML-based services. It is a clean, simple protocol that provides the minimum overhead necessary to invoke remote services and exchange data in a platform-, language-, and vendor-neutral manner. It defines six simple data types and two complex types. The result is that XML-RPC is a highly-efficient lightweight protocol. Messages are simple to construct, simple to parse, simple to debug, and are easily human-readable. XML-RPC requires a minimal amount of active memory for processing, building, and parsing messages, and produces a thin message body to be exchanged between client and server.

In contrast, SOAP is a fatter protocol (although generally considered lightweight). In exchange for some additional overhead, SOAP provides namespace awareness, a sophisticated data-typing mechanism, and a flexible messaging paradigm. The W3C's XML Namespaces specification is leveraged to provide namespace awareness, and the XML Schema specification provides the data-typing mechanism. As such, SOAP supports over 40 standard data types and provides the capabilities to define custom simple and complex data types. This provides a tremendous degree of flexibility in terms of describing robust data structures with intricate relationships with other data contained in the message body.

The SOAP architecture also introduces a flexible messaging architecture, supporting a variety of messaging paradigms, including unidirectional, bidirectional, multicast (publish-subscribe), and sequentialmessaging (multiple parties chained together in a particular order). One aspect that facilitates these paradigms is the ability of a SOAP message to include a header section. This allows security, transaction, and routing information to be exchanged between multiple parties. For example, a client can send a SOAP request with an intended destination, but also declare in the SOAP envelope's headers that a particular party should receive the message, process the pertinent header information, and then pass the message on to the next party in the chain (or the intended recipient if no other parties have been declared). One final distinction is that SOAP supports asynchronous messaging, while XML-RPC requires a synchronous communication between the two parties in an exchange.

With all of these differences between the two protocols, executives, project managers, and wireless architects alike are wondering – "How do I make a choice between SOAP and XML-RPC for a particular wireless project or system?" Well, even if you weren't wondering that, we're going to tell you anyway.

In a nutshell, XML-RPC provides a fast, compact protocol for exchanging data and invoking services in a neutral, standardized way. If application size, memory, and bandwidth are your top priorities, then XML-RPC is the way to go. On the other hand, if your application requires a robust data-typing mechanism, extensive security or transaction support, or a flexible messaging architecture, then SOAP is your answer. Also, SOAP is more mainstream than XML-RPC, so if your application needs to access a variety of services that you have no control over, SOAP may be a better match from an interoperability perspective. To break it down into more detail, Table 1 that outlines the business justifications for using each protocol.

Wireless SOAP Example

In our previous article we developed a sample application using kXML-RPC (http://kxmlrpc.enhydra.org), an open-source implementation of the XML-RPC protocol for micro devices maintained by Enhydra. kSOAP (http://ksoap.enhydra.org) is another Enhydra Micro Edition project, providing SOAP support for mobile devices. We will create a MIDP interface with the kSOAP libraries underneath to activate a SOAP-based Web service

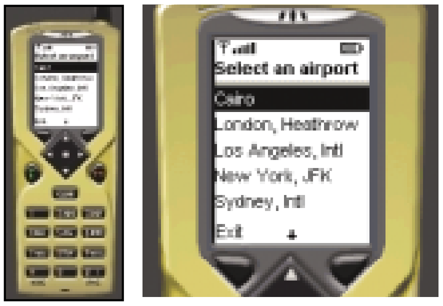


FIGURE 2 Airports for which information can be retrieved

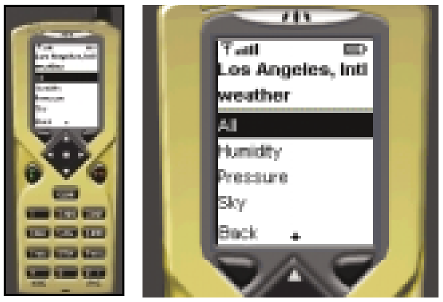


FIGURE 3 Weather information services for selected airports

TABLE 1: Business Justification for using each protocol

Business Priority	Appropriate Protocol
Tiny memory footprint	XML-RPC
Speed and efficiency	XML-RPC
Interoperability with other parties beyond influence or control	SOAP
Reduced bandwidth	XML-RPC
Easy maintenance and debugging of system	XML-RPC
Exchange of robust data structures with intricate relationships	SOAP
Flexible messaging architecture	SOAP
Creation of named custom data structures	SOAP
Creation of simple, non-named custom data structures	XML-RPC
Asynchronous communication	SOAP
Additional security beyond HTTPS and SSL3	SOAP
Transaction support	SOAP

located on a remote HTTP server.

For our application, we will be using three classes from the kSOAP library to handle the marshalling and unmarshalling of data via SOAP:

- ***HttpTransport:*** A convenient API that enables SOAP calls via HTTP using the J2ME Generic Connection Framework
- ***ClassMap:*** Provides namespace support and a two-way mapping between namespace-qualified XML names and Java classes
- ***SoapObject:*** A generic object used to represent any SOAP object within the body of a SOAP request or response. SOAP objects can also be nested

Rather than creating both the client and the service we are accessing, we will simply access an existing service. Xmet hods.com provides a Web service registry for publicly available Web services. We will be invoking a service provided by Cape Clear that gathers and disseminates airport weather information on behalf of a client. The details regarding that service can be located at: www.xmethods.com/detail.html?id=129, and the source code for this midlet (Airport Weather.java) can be downloaded from the kSOAP Web site at http://ksoap. enhydra.org /soft ware/down loads/index. html.

In creating our sample SOAP midlet (AirportWeather.java), the process is identical to the one we used in Part 1. This time, the packages and names have been changed to protect the innocent. We will import, extend, and use classes from the kSOAP library rather than the kXML-RPC library.

The first step, obviously, is to import the necessary packages and declare the MIDP components that will be used in the application. After this, we define the midlet's constructor, initializing all the UI components, and add them to the display as necessary. With that complete, we need to fill in the three other lifecycle methods. In the startApp() method, we simply bring the MIDP display into action. Since we don't use any shared resources, the pauseApp() method is blank. Finally, the destroyApp() method releases the local resources that we have allocated for our midlet.

As with Part 1, all the action takes place in the commandAction() method. This method is called anytime the user performs a command event (pressing a key, selecting an item from a list, etc.). The Command and Displayable objects are then queried to determine which component has actually been activated/ deactivated, and the appropriate actions are performed. When our midlet is launched, it displays a list of popular international airports for which weather information can be retrieved (see Figure 2). It also displays an exit button to allow the user to exit the midlet.

Upon selecting an airport, a list of weather information services for that airport is displayed (see Figure 3). A back button is also displayed to allow the user to return to the airport menu. The user then selects the weather information service he or she is interested in and the midlet sends a SOAP request to the server to collect the desired information. A SOAP response is returned, parsed by the midlet, and the result is displayed on the screen. So how does all of that work? See Listing 1 for the command Action() method.

The outer level of the commandAct ion() method checks to see what component has been activated. We'll look at each of these four events individually, beginning with the airport menu:

```
  if ( dis == airportMenu && com ==
List.SELECT_COMMAND ) {
     currentAirport =
airportMenu.getSelectedIndex();
     servicesMenu.setTitle( airports[
currentAirport ] + " weather" );
     display.setCurrent( servicesMenu
); //display the list of services
```

When an airport is selected, we set the currentAirport variable, set the title for the services menu to reflect the selected airport, and display the services menu.

Next we'll look at the services menu. There are seven possible weather services that can be retrieved: wind conditions, temperature, pressure, humidity, sky conditions, visibility, and a complete summary containing the six individual pieces of information:

```
  else if ( dis == servicesMenu &&
com == List.SELECT_COMMAND ) {
String result = null;
int choice =
servicesMenu.getSelectedIndex();
```

The first thing to do is to declare a String variable to store the result of the service call that will be made. Next, determine which service has been selected and store the index value in an integer variable and perform a switch on that variable. The switch statement can be seen in Listing 2. The first case retrieves a summary and the other cases retrieve individual weather items. After the switch statement, the results are displayed on the screen via an Alert object.

Within each case statement in the switch block, the callSer vice() method is used to handle the exchange of SOAP messages. See Listing 3).

Four essential things take place in this method. An HttpTran sport object is created, a PrefixMap namespace is created for the SOAP envelope, a SoapObject is created to represent the request, and the request object is sent via the HttpTransport class's call() method.

The next else/if block checks to see if the back command has been selected, in which case the list of airports is displayed:

```
 else if ( com == backCommand ) {
   display.setCurrent( airportMenu );
```

Finally, the exit command is checked to allow the application to be exited:

```
else if ( com == exitCommand ) {
       destroyApp( true );
         notifyDestroyed();
```

That's all there is to it. Download the source code to get the details for the interface and then you are ready to build the midlet interface, compile the code and access this weather service from a J2ME phone using SOAP!

Conclusion

Web services are sweeping the industry and changing the face of business. Mobile computing and wireless access to information at any time, from anywhere, is an increasingly popular idea. The combination of the two is explosive! In these two articles, we've explored XML-RPC and SOAP as protocols for accessing Web services from wireless devices. XML-RPC provides a highly-efficient, extremely lightweight mechanism for invoking Web services that is ideal for mobile and embedded devices. SOAP provides a slightly heavier protocol with increased functionality and flexibility. Between these two, an appropriate protocol will likely be found for any wireless project.

Listing 1

```
public void commandAction( Command com,
                           Displayable dis ) {    *
  if ( dis == airportMenu &&
       com == List.SELECT_COMMAND ) {
 // An airport has been selected
  }
  else if ( dis == servicesMenu &&
            com == List.SELECT_COMMAND ) {
 // A weather service has been selected
  }
  else if ( com == backCommand ) {
// The back button returns to airport menu
  }
  else if ( com == exitCommand ) {
 // The exit button exits the midlet
  }//end if/else
}//end commandAction(Command, Displayable)
```

Listing 2

```
switch( choice ) {
 case 0:
  SoapObject obj = (SoapObject)
     callService( currentAirport, "getSummary" );
  if (obj != null ) {
   result = "The weather at " +
    obj.getProperty(0) + " is " +
    obj.getProperty(3) + " with a " +
    obj.getProperty(2) + " sky, and wind " +
    obj.getProperty(1) + ". The humidity is " +
    obj.getProperty(4) + ", the pressure is " +
    obj.getProperty(5) + ", and visibility is " +
    obj.getProperty(6) + ".";
  }//end if (obj != null)
  break;
 case 1:
  result = (String) callService( currentAirport,
                               "getHumidity" );
  break;
 case 2:
  result = (String) callService( currentAirport,
                               "getPressure" );
  break;
 case 3:
  result = (String) callService( currentAirport,
                          "getSkyConditions" );
  break;
 case 4:
  result = (String) callService( currentAirport,
                            "getTemperature" );
  break;
 case 5:
  result = (String) callService( currentAirport,
                                      "getOb" );
  break;
 case 6:
  result = (String) callService( currentAirport,
                                    "getWind" );
  break;
}//end switch();

//display the result
response.setString( result );
display.setCurrent( response );
```

Listing 3

```
private Object callService( int choice,
                            String methodName ) {
 Object result = null;

 try {
  transport = new HttpTransport( serviceUrl,
               soapAction + "#" + methodName );
  transport.debug = true;

  classMap = new ClassMap();
  classMap.prefixMap = new PrefixMap(
   classMap.prefixMap, "air", serviceNamespace );

  transport.setClassMap( classMap );

  request = new SoapObject( serviceNamespace,
                            methodName );
  request.addProperty( "arg0",
                      airportCodes[ choice ] );
  result = transport.call( request );
 } catch( Exception e ) {
    e.printStackTrace();
    System.out.println( "Request: \n" +
                      transport.requestDump );
    System.out.println( "Response: \n" +
                      transport.responseDump );
    result = null;
 }//end try/catch

 return result;
}//end callService()
```

Download the code at
sys-con.com/webservices

dot.com
- buyer's guide
- web services forums
- mailing list
- web services jobs
- web services store

magazine
- advertise
- authors
- customer service
- editorial board
- subscribe

content
- archives
- digital edition
- editorial
- features
- interviews
- product reviews
- source code

conferences
- web services EDGE conference & expo — june 24–27 2002
- web services EDGE conference & expo — oct 1–3 2002

search

Search

specials

Collect All Four!

JDJStore.com
$229.99

JDJ, XML, CFDJ, CFAdvisor ALL Four!!!-The Complete Works

best sellers

1. Plantronics M-175 Headset for Cellular & Cordless Phones
$39.95

2. Garmin: StreetPilot III GPS
$629.99

3. Cape Clear: Cape Studio (Single Developer's License)
$445.00

4. WebGain Visual Cafe Standard Edition
$97.99

5. RoboHELP Office 9.1 from eHelper V8.0
$899.95

What's Online

www.sys-con.com/webservices

Join the fourth wave of technology and become part of the newest paradigm in software development!

WSJ2.com

Can't get to the newsstand in time? Let www.wsj2.com be your source for industry events and happenings. Check in daily for up-to-the-minute news, events, and developments, and be the first to know what's going on in the latest paradigm of technology.

Readers' Choice Awards

Vote now at www.sys-con.com/webservices for your favorite Web services products. The awards, known as the Oscars of the software industry, will be presented at Web Services Edge 2002, June 24-27, 2002.

WSJ READERS' CHOICE AWARD

Welcome to WSJList!

The online mailing list community that keeps you at the center of discussions, technical questions, and more... Join the WSJList now to monitor the pulse of the Web services industry!

Digital Edition

Don't have your print edition on hand? Can't wait for the next issue to arrive in the mail? Our digital edition is just what you need. As long as you have your computer with you, you can read ***Web Services Journal*** anytime, anywhere.

Web Services Edge 2002 International Conference & Expo

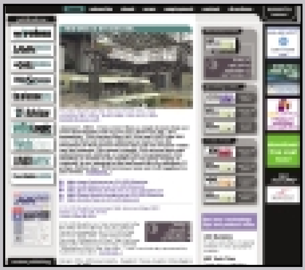
Plan now to be at the Jacob Javits Convention Center in New York City, June 24-27, 2002, for the largest Web services conference and expo of the year.

Conference tracks will offer invaluable information on Web services, Java, XML, IT strategy and much more. Go to www.sys-con.com for updates, news, and registration information.

WebServices JOURNAL Industry >Newsletter

Subscribe now for up-to-the-minute news of what's happening in Web services. Exclusive stories and the inside news on Web services, the newest paradigm in software development.

Welcome to WSJList!



QUICK POLL

Will Web Services be "The Next Big Thing?"
A- Yes
B- No

Vote

Results

Free XML Starter Kit
software AG

The Truth About Web Services
IONA Webcast Series
IONA

Click for a FREE 30-day trial
AltoWeb

CLICK HERE NOW
SilverStream eXtend
Web Services

Get SonicMQ®
sonic SOFTWARE

See it at the BEA eWorld developer conference.

Delphi 6 and Kylix 2 Web Services

Easy-to-build Web services and SOAP Applications

When Borland shipped Delphi 6 in May of last year, one of its new features was support for SOAP – most notably in the form of Web services. Borland Kylix 2 (for Linux) is now also available with the same capabilities, and as I write this article, Borland has just announced the Borland Web Services Kit for Java, which will enable Borland JBuilder to create and consume Web services using WSDL and SOAP. Finally, by the time you read this article I expect Borland C++ Builder 6 to be announced (or available) with the same SOAP and Web services capabilities that Delphi 6 and Kylix 2 currently have. Perhaps even a bit more, since SOAP is ever evolving.

In this article, I want to demonstrate the ease of use of Delphi 6 and Kylix 2 (two RAD tools) by developing a Web service in a Delphi 6 server (running on Win32) and consuming it in a Kylix 2 client running on Linux.

What is a Service?

For the Web Service "engine," I'm using Delphi 6 Enterprise. You can download a free 60-day trial edition of Delphi 6 Enterprise from their Web site at www.borland.com/down loads (although you may not deploy applications written with this trial edition, you can follow the instructions and steps in this article to see how easy it is to write your own Web services).

To create a Web service, start Delphi 6 Enterprise and from the main menu do File | New and then select Other to get the so-called Object Repository dialog. Go to the Web Services tab, which contains three icons: one for a Soap Server Application, one for a Soap Server Data Module, and one for the Web Services Importer. If you've registered your copy of Delphi 6 Enterprise (the real version),

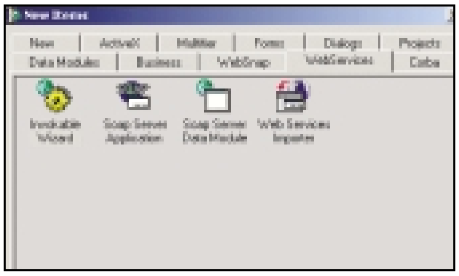
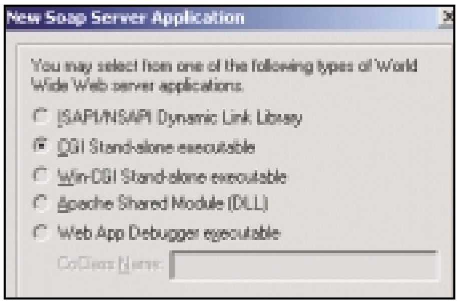


FIGURE 1 **Delphi 6 Web Services Wizard**

you can download a few additional tools and wizards, which among others result in a fourth icon here, namely the Invocable Wizard (see Figure 1).

To start a new Web service, double-click on the Soap Server Application icon, which gives a dialog (see Figure 2) in which we will specify the Web server application that will contain our Web service. Note that although the latest SOAP protocol specifies more than just the HTTP protocol for communication (including FTP and SMTP), Delphi mainly supports HTTP using the wizards from Figure 1, and hence uses a simple Web server application as wrapper.

Figure 2 shows that we have a number of choices here. For a real-world Web service that will be used (consumed) a lot, you should look at ISAPI/NSAPI or Apache DLL possibilities.


New Soap Server Application

You may select from one of the following types of World Wide Web server applications.

- ISAPI/NSAPI Dynamic Link Library
- [✓ CGI Stand-alone executable]
- Win-CGI Stand-alone executable
- Apache Shared Module (DLL)
- Web App Debugger executable

CoClass Name:

FIGURE 2 **New SOAP Server Application Wiard**

For debugging purposes, you can start with a Web App Debugger executable (you can always transform your Web server project to another target). For our example, I want to select a simple CGI standalone executable because I want to deploy the resulting Web service on the Internet, and since it's only an example with this article, I don't reckon it will be consumed that much (so I don't need to turn it into a DLL for improved efficiency).

Web Module

Once we've made our choice, we can click on the OK button and a new Soap Server application will be generated, including a Web module (the core of our Web server application). Save the Web module in file SWebMod.pas and the project in D6WebService.dpr before you continue. As you may have noted by now, the Web module already contains three configured components, which form the basis of our Web service.

From left to right, the HTTPSoapDispatcher is used to receive incoming SOAP requests and dispatch them to another component (defined by the Dispatcher property) that can interpret the request and execute it. The latter will be done by the HTTP SoapPascalInvoker component, which receives the incoming SOAP request, executes

Author Bio

Bob Swart (aka Dr.Bob; www.drbob42.com) is an independent author, trainer and consultant based in The Netherlands who has spoken at Delphi and Borland Developer Conferences since 1993. Bob is co-author of the Revolutionary Guide to Delphi 2, Delphi 4 Unleashed, C++Builder 4 Unleashed, C++Builder 5 Developer's Guide, Kylix Developer's Guide, and Delphi 6 Developer's Guide, as well as a contributing author for numerous computer magazines

DRBOB@CHELLO.NL

(invokes) a Pascal method, and produces the response back to the HTTPSoapDispatcher. But before the HTTPSoapPascalInvoker can actually invoke the requested Pascal method, it checks to see if the method's interface and implementation class have been registered (in the invocation registry; we'll get back to this in a moment). While the first two components are used when the Web service is actually consumed, the third component – WSDLHTML Publish – is used to produce the WSDL (Web Service Description Language) that defines the Web service itself.

Invokable Wizard

The three components on the Web module have already been configured, so you don't have to work on them to make your Web service function. In fact, right after you've created the Web module and saved your project files, you should continue with the next step, which involves starting the Invokable Wizard from the Object Repository. Again, do File | New | Other, go to the Web Services tab and double-click on the Invokable Wizard. This will result in the Invokable Interface & Class Wizard where we can define the (file)name of our Web service engine (see Figure 3).

Because we're building an example Web service, I've designed a simple engine

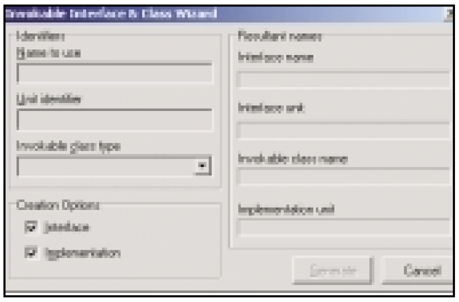


FIGURE 3 | Invokable Interface & Class Wizard

which does nothing more than convert some input value to an output value. For this example, I've implemented a little Roman number conversion routine – from decimal to Roman and back. As a result, I would like to use the name "Roman" (the editbox in the upper-left corner). Typing Roman here seems to result in a sensible value for everything except the Invokable class type combobox, which offers us two choices: TInterfacedObject or TInvok ableClass. The latter is easier to manage, so select the TInvokableClass here.

Finally, click on Generate to create two new units: RomanIntf.pas with the IRoman interface definition and Roman Impl.pas with the TRoman class that implements the IRoman interface.

Without the Invokable Wizard to generate the two units, you can write the units Roman Intf.pas and RomanImpl.pas yourself (see the two listings that follow), so with the trial edition you need to enter more code.

IRoman interface

Both the definition of IRoman and implementation of TRoman are still empty, of course, so now is the time to actually write some code. For our Roman number conversion, I want two methods: IntToRoman and RomanToInt. The first would get an Integer as argument and produce a String as result; the second would get a String as input argument and would produce an Integer as result. In Object Pascal, the IRoman definition should look like Listing 1 (unit RomanIntf.pas).

The code inside the initialization section will register the IRoman interface in the Invokable Registry (where the HTTPSoapPascalInvoker will look for it later). Note that you needed to write only two lines (the functions IntToRoman and RomanToInt), and that the generated comments already suggest you should use the stdcall calling convention.

TRoman Implementation

Once we've saved the RomanIntf.pas file again, we can move to the RomanImpl.pas unit to finish the TRoman class. Since TRoman is implementing the two methods from IRoman, we must copy these declarations inside the class definition of TRoman. With only two methods, that's easy, but with a few dozen it can be a bit messy. Fortunately, Delphi offers us some support with Code Insight. Just move to the class definition of TRoman in the code editor and click Ctrl+Space to show the available methods of TRoman and IRoman. The Code Insight window will start with a number of methods in Red – these are the abstract methods from IRoman that need to be implemented. Just select all the Red methods and press enter to insert them all inside the class definition of TRoman.

Once the class definition is complete, you can generate empty placeholders for the actual implementation by pressing Ctrl+C. This will produce the skeletons for the IntToRoman and RomanToInt functions (see Listing 2).

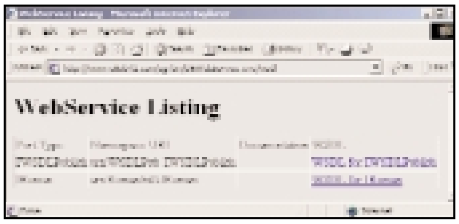
Again, the code inside the initialization section will register the TRoman class in the Invokable Registry (where the HTTPSoapPas calInvoker will look for it).

The actual implementation of IntToRoman and RomanToInt is rather lengthy, and not important for the Web service topic itself, so I haven't included it here in the listing of unit RomanImpl.pas.

Deployment

This is it. After you've saved your project files, you can compile the Web service application. In order to use it, you must now deploy it as a Web server application. For your convenience, I've done that already, and the D6Web Service.exe is available to use as an exercise on my Web site at www.eBob42.com/cgi-bin/D6WebService.exe

Note that this direct URL won't do anything. You actually have to pass the additional PathInfo/wsdl to get the WSDL (automatically produced by the WSDLHTMLPublish component; see Figure 4).


WebService Listing

Port Type	Namespace URI	Documentation	WSDL
IWSDLPublish	urn:WSDLPub-IWSDLPublish		WSDL for IWSDLPublish
IRoman	urn:RomanIntf-IRoman		WSDL for IRoman

FIGURE 4 | Web Service Listing of D6WweService.exe/wsdl

If you click on the link for WSDL for IRoman (or www.eBob42.com/cgi-bin/D6WebService.exe/wsdl/IRoman) you get the full WSDL for the IRoman interface.Now we'll move over to Linux and start Kylix 2 Enterprise to write a Web Service consumer for this application.

Kylix 2 Enterprise

If you don't have Kylix 2 Enterprise, you can download a 60-day trial version from the Borland Web site. But you can also write a Delphi 6 client for the Web service using the same steps needed for the Kylix 2 client, so it shouldn't be hard to play along either way.

A Web Service consumer can be any kind of application. But to keep things simple, let's just start a new default visual application in Kylix. Save the form in file MainForm.pas and the project in RomanC lient.dpr (or any other filename you wish to use now).

web services EDGE conference & expo™

INTERNATIONAL WEB SERVICES CONFERENCE & EXPO

JDJ EDGE conference & expo™

INTERNATIONAL JAVA DEVELOPER CONFERENCE & EXPO

XML EDGE conference & expo™

INTERNATIONAL XML CONFERENCE & EXPO

web services EDGE conference & expo
JUNE 2002
Jacob K. Javits Convention Center of New York

FUNDAMENTALLY IMPROVING THE SPEED, COST AND FLEXIBILITY OF BUSINESS INTEGRATION

Sponsored by:

IONA END 2 ANYWHERE · TogetherSoft · ADOS · bea · SilverStream · MERANT · Federal Computer Week · SD Times · Charles F. Goldfarb's XML TIMES.com · jCert · WROX · Webservices.org · Java Skyline · WebServices Journal · JAVA Developer's Journal · XML Journal · wireless Business & Technology · BEA WebLogic Developer's Journal · ColdFusion Developer's Journal · WebSphere Developer's Journal · CF Advisor

Sponsored by:
SYS-CON MEDIA

Produced by:
SYS-CON EVENTS

Web Services – Skills, Strategy, and Vision

For the developer. the latest tools and techniques...

For the architect, the latest designs....

For the VP/CIO, technology management issues

- Invaluable information in the form of discussions, presentations, tutorials, and case studies
- Unmatched Keynotes and Faculty - gurus in the Java, XML, .NET, and Web Services world
- The largest independent Web Services, Java, and XML Expos
- An unparalleled opportunity to network with over 5,000 *i*-technology professionals

New in 2002!

Each track will feature "Hot Breaking" Sessions to keep attendees up-to-the-minute on Emerging Technologies and Strategies!

Featuring 2 Keynote Panels and Industry Perspectives from the Visionaries shaping Next Generation Technologies and Business Strategies

2002

ONLINE
EARLY BIRD REGISTRATION
NOW OPEN!
SAVE $400

JUNE 24–27
JACOB JAVITS CONVENTION CENTER
NEW YORK, NY

OCTOBER 1–3
SAN JOSE CONVENTION CENTER
SAN JOSE, CA

FOR MORE INFORMATION
SYS-CON EVENTS, INC
135 CHESTNUT RIDGE RD.
MONTVALE, NJ 07645

201-802-3069
WWW.SYS-CON.COM

Who Should Attend…
- Developers, Programmers, Engineers
- *i*-Technology Professionals
- Senior Business Management
- Senior IT/IS Management
- Analysts, Consultants

Who Should Exhibit…
Java, XML, Web Services, and .NET Technology vendors, staking their claim to this fast-evolving marketplace

Web Services Edge Conference Committee

Alan Williamson
Java Track Chair
Editor-in-Chief
Java Developer's Journal

Java Track
- Java 1.4: What's New?
- Building Truly Portable J2EE Applications
- J2ME: MIDlets for the masses
- WebScripting Technologies: JSP/CFML/Velocity
- Where is Swing in this New Web Services World?

Ajit Sagar
XML Track Chair
Editor-in-Chief
XML-Journal

XML Track
- XML Web Services: Standards Update
- Using XML for Rapid Application Development and Deployment with Web Services
- Unlocking the Promise of XML: From Hype to How-To
- The Use of XML Technologies to Enhance Security
- Content Management with XML

Sean Rhody
Conference Tech Chair
Web Services Track Chair
Editor-in-Chief
Web Services Journal

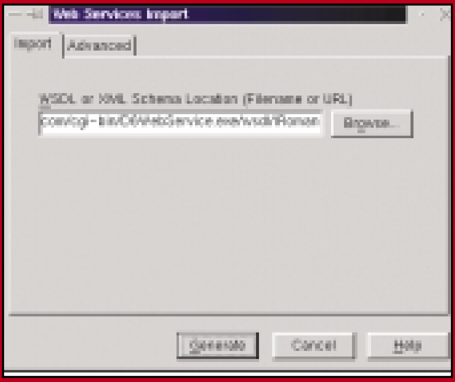
Web Services Track
- Starting Out in Web Services: Fundamentals of Web Services (WSDL, UDDI, SOAP)
- Exploring .NET myServices Initiative
- Standards Watch: Reviews and Discussions of the Interactions of All the Relevant Standards
- Guarding the Castle: Security and Web Services
- The Microsoft Way: .NET, Passport, and other MS Technologies for Web Services
- Laying Down the Rules: Web Services and Business Process Engines
- Practical Experiences with Web Services and J2EE
- The Odd Couple: Making .NET and J2EE Work Together

Andy Astor
IT Strategy Track Chair
International Advisory Board
Web Services Journal

IT Strategy Track
- Key Architectural Issues in a World of Web Services
- .NET vs J2EE – From Religious Wars to Fact-Based Decision Making
- Getting Started with Web Services
- Selecting a Framework: Toolkit, Platform, or Roll Your Own?
- Infrastructure Vendors–A Map of the World
- Extreme Programming and Web Services Projects: Defining ROI
- Standards You Need to Know About
- Best Practices You Need to Insist On
- Introduction to Business Rules and Rules Engines
- Panel Discussion- Web Service Business/ Economic Models

Jim Milbery
Vendor Track Chair
Product Review Editor
Java Developer's Journal

Vendor Track
- .NET, J2EE, JMS and other Messaging
- Case and Development Tools
- The Use of Testing Tools
- How-to Demonstrations



WSDL or XML Schema Location (Filename or URL)
com/cgi-bin/D6WebService.exe/wsdl/IRoman

FIGURE 5 | Kylix 2 Web Services Import

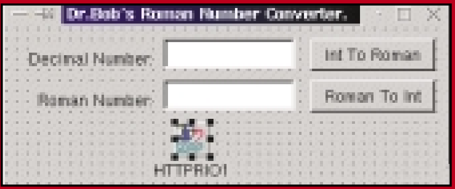
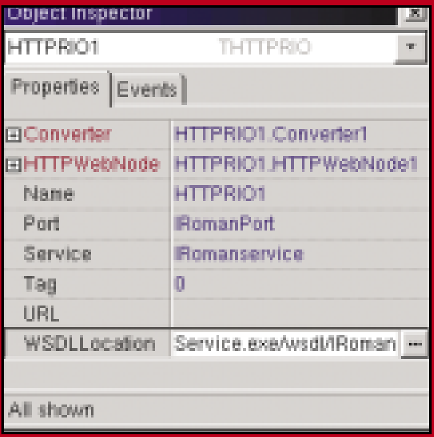


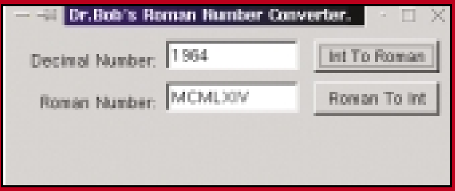
FIGURE 6 | Kylix 2 RomanClient with HTTPRIO component



HTTPRIO1 THTTPRIO

Converter	HTTPRIO1.Converter1
HTTPWebNode	HTTPRIO1.HTTPWebNode1
Name	HTTPRIO1
Port	IRomanPort
Service	IRomanservice
Tag	0
URL	
WSDLLocation	Service.exe/wsdl/IRoman

FIGURE 7 | HTTPRIO Property values



Decimal Number: 1964
Roman Number: MCMLXIV

FIGURE 8 | Roman Web Service consumer in action

Web Service Importer

Remember the Web Services tab of the Object Repository in Delphi 6? Kylix 2 has a similar one (although I haven't seen the Invokable Wizard for Kylix yet). This time, we need to use the Web Service Importer, since we must import the WSDL that defines the Web service and generate an Object Pascal import unit that defines the interface of this particular Web service. So, do File | New and go to the Webservices tab of the Object Repository and double-click on the Web Service Importer icon, which will result in the Web Services Import wizard (see Figure 5).

We now need to specify the URL to retrieve the WSDL information for our Web Service, (www.eBob42.com/cgi-bin/D6WebService.exe/wsdl/IRoman). Look at the options in the Advanced tab or directly click on the Generate button to create the Web Service import unit. This will also work with Web services that are written in another development environment, although some interoperability issues between SOAP implementations remain to be worked on.

Save the generated import unit in file Roman.pas and add it to the uses clause of the MainForm unit. The generated unit Roman.pas can be seen in Listing 3:

Note that the arguments to the two methods, IntToRoman and RomanToInt, have been declared as "const" now. Other than that, the importation of the IRoman interface looks exactly as the original IRoman interface that we defined using Delphi 6.

Now, drop two TLabel, two TEdit, and two TButton components on the main form. One label/edit pair is for the normal decimal number, and one label/edit pair is for the Roman numeral. Finally, one button will convert the contents of EditInt from Int to Roman and another button will convert the contents of EditRoman back from Roman to Int again. We also need to drop a HTTPRIO component from the Web Services tab of the Component Palette (see Figure 6 for the final layout of the main form).

HTTPRIO

The HTTPRIO component is our gateway (using HTTP) to the Remote Invokable Object. Although we have the Roman.pas import unit that defines the capabilities of our Web Service, we must tell the HTTPRIO component some information as well (such as the Service and Port). For this, we need to specify the WSDL URL again in the WSDLLocation property of the HTTPRIO component. Once you've set this property value, you can open up the combobox for the Service property, to give it the value IRomanservice; and then for the Port property, to give it the value IRomanPort (see Figure 7).

Once the HTTPRIO component is configured, we can write two event handlers and a little code to actually use the available methods. To use the HTTPRIO component, we can simply cast it to the IRoman interface (actually, the correct phrase would be to extract the IRoman interface from the HTTPRIO component, but the effect is the same). The implementation of the two-button OnClick event handlers inside the main form is shown in Listing 4.

The Web service consumer client running on Linux can be seen in Figure 8.

A final nice thing about Delphi 6 and Kylix 2 is that although the former runs on Win32 and the latter runs on Linux, the source code of their CLX projects is actually cross-platform (usually with minor changes to make it work). In this case, it doesn't involve a single change in the source code, so you can take the RomanCli ent.dpr project from Kylix 2 and compile it with Delphi 6 to produce a working client on Windows.

Conclusion

I hope I've shown you that it's easy to build Web services and SOAP applications with Delphi 6 and Kylix 2 using available components and wizards for both the server side and the client (consumer) side. And with the addition of the Borland Web Services Kit for Java (for JBuilder) and the forthcoming release of C++Builder 6, we'll have a full range of Borland RAD tools and languages on platforms supporting SOAP and Web services.

For interoperability with other vendors and their SOAP implementations, Borland participates in the SOAP Interoperability studies, which can be found on the Borland Website at http:// soap-server.borland.com/WebServices

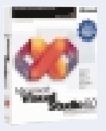
SHOP ONLINE AT **JDJSTORE.COM** FOR BEST PRICES OR CALL YOUR ORDER IN AT **1-888-303-JAVA**

BUY THOUSANDS OF PRODUCTS AT GUARANTEED LOWEST PRICES!

GUARANTEED BEST PRICES
FOR ALL YOUR
WEB SERVICES
SOFTWARE NEEDS

JDJStore.com

MICROSOFT
$1446.99 Visual Studio 6.0 Enterprise

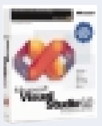


The Visual Studio 6.0 suite contains many new features to help organizations build enterprise solutions. Visual Studio 6.0 includes new versions of the award-winning Microsoft Visual Tools, enabling developers to build component-based solutions. These tools provide comprehensive support for building all aspects of a multitier solution based on the Microsoft Windows® Distributed interNet Applications (Windows DNA) architecture.

ALTOWEB
$3,540.00 Application Platform Release 2.5

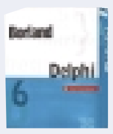
The AltoWeb Application Platform lets you build, deploy, and manage J2EE applications and Web services up to 10x faster without requiring extensive J2EE or Web services expertise. How? By replacing lengthy, custom and complex J2EE, XML and Web services coding with rapid component assembly and reuse.

MICROSOFT
$964.99 Visual Studio Professional



The Visual Studio 6.0 suite contains many new features to help organizations build enterprise solutions. Visual Studio 6.0 includes new versions of the award-winning Microsoft Visual Tools, enabling developers to build component-based solutions. These tools provide comprehensive support for building all aspects of a multitier solution based on the Microsoft Windows® Distributed interNet Applications (Windows DNA) architecture.

BORLAND
$2,999.00 Sale Price $2,849.99 Delphi



Delphi 6 makes next-generation e-business development with Web services a snap. BizSnap Web services development platform simplifies business-to-business integration by easily creating Web services. DataSnap Web services-enabled middleware data access solutions integrate with any business application. Check out our Web site for system requirements.

SITRAKA
$3,999.00 Sale Price $3,799.00 JClass ServerChart V1.1

If you've used JClass Chart in your client-side applications, you already know the value of data visualization to your end-users. With JClass ServerChart, you can bring a wide range of charts and graphs right to their Web browsers, giving them real-time access to business-critical data in an intuitive environment.

SILVERSTREAM
$495.00 Extend Application Server
Developer Edition (5 User)

SilverStream eXtend allows you to rapidly create, assemble, consume and deploy Web services. It includes an advanced and intuitive development environment. This workbench provides everything for developers and business analysts to create, assemble, consume and deploy services. And it integrates with your favorite code editors and source control systems.

W W W . J D J S T O R E . C O M OFFERS SUBJECT TO CHANGE WITHOUT NOTICE

SAVE UP TO $100 ON MULTIPLE SUBSCRIPTIONS

Special online offers

Pick 4 or 5 and Subscribe
for one special low price

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTION

Wireless Business & Technology • Java Developer's Journal
Web Services Journal • WebLogic Developer's Journal
XML-Journal • WebSphere Developer's Journal
ColdFusion Developer's Journal • PowerBuilder Developer's Journal

SYS-CON MEDIA

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

WWW.SYS-CON.COM/SUBOFFER.CFM

Listing 1

```
{ Invokable interface declaration unit for IRoman }

unit RomanIntf;
interface
uses
  Types, XSBuiltIns;

type
  IRoman = interface(IInvokable)
    ['{01304E12-D6DE-46E8-8585-2D3660B7D9CC}']
    // Declare your invokable logic here using standard Object
Pascal code
    // Remember to include a calling convention! (usually
stdcall)
    function IntToRoman(I: Integer): String; stdcall;
    function RomanToInt(R: String): Integer; stdcall;
  end;

implementation
uses
  InvokeRegistry;

initialization
  InvRegistry.RegisterInterface(TypeInfo(IRoman), '', '');
end.
unit RomanIntf.pas
```

Listing 2

```
{ Invokable implementation declaration unit for TRoman,
  which implements IRoman }

unit RomanImpl;
interface
uses
  RomanIntf, InvokeRegistry;

type
  TRoman = class(TInvokableClass, IRoman)
  public
    // Make sure you have your invokable logic implemented in
IRoman
    // first, then use CodeInsight(tm) to fill in this
implementation
    // section by pressing Ctrl+Space, marking all the
interface
    // declarations for IRoman, and pressing Enter.
    function IntToRoman(I: Integer): String; stdcall;
    function RomanToInt(R: String): Integer; stdcall;
    // Once the declarations are inserted here, use
ClassCompletion(tm)
    // to write the implementation stubs by pressing
Ctrl+Shift+C
  end;

implementation

{ TRoman }

function TRoman.IntToRoman(I: Integer): String;
begin

end;

function TRoman.RomanToInt(R: String): Integer;
begin


end;

initialization
  InvRegistry.RegisterInvokableClass(TRoman);
end.
Unit RomanImpl.pas
```

Listing 3

```
Unit Roman;
```

RECEIVE $150
DISCOUNT OFF FULL CONFERENCE
WEB SERVICES EDGE REGISTRATION
web services EDGE conference&expo
2002WORLDTOUR
Learn How to Develop
SOAP Web Services NOW!
at a One-Day Tutorial...Coming to a City Near You!

```
interface
uses
  Types, XSBuiltIns;

type
  IRoman = interface(IInvokable)
    ['{01304E12-D6DE-46E8-8585-2D3660B7D9CC}']
    function IntToRoman(const I: Integer): WideString;
stdcall;
    function RomanToInt(const R: WideString): Integer;
stdcall;
  end;

implementation
uses
  InvokeRegistry;

initialization
  InvRegistry.RegisterInterface(TypeInfo(IRoman),
'urn:RomanIntf-IRoman', '');
end.
Unit Roman.pas

Listing 4
unit MainForm;
interface
uses
  SysUtils, Types, Classes, Variants, QGraphics, QControls,
QForms, QDialogs,
  Rio, SOAPHTTPClient, Roman, QStdCtrls;

type
  TForm1 = class(TForm)
    HTTPRIO1: THTTPRIO;
    Label1: TLabel;
    Label2: TLabel;
    EditInt: TEdit;
    EditRoman: TEdit;
    btnIntToRoman: TButton;
    BtnRomanToInt: TButton;
    procedure btnIntToRomanClick(Sender: TObject);
    procedure BtnRomanToIntClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
{$R *.xfm}

procedure TForm1.btnIntToRomanClick(Sender: TObject);
var
  Int: Integer;
begin
  Int := StrToIntDef(EditInt.Text,0);
  EditRoman.Text :=
    (HTTPRIO1 as IRoman).IntToRoman(Int);
end;

procedure TForm1.BtnRomanToIntClick(Sender: TObject);
var
  Int: Integer;
begin
  Int := (HTTPRIO1 as IRoman).RomanToInt(EditRoman.Text);
  EditInt.Text := IntToStr(Int);
end;

end.
Unit MainForm.pas
```

Download the code at

sys-con.com/webservices

EDITOR'S NOTE: The author of “Street Fighting Comes to Web Services” (*WSJ*, Vol. 1, issue 3) contacted all of the companies mentioned in that article. The following Microsoft response was received after our copy deadline. It is included below in its entirety.

What is .NET?

Microsoft is creating an advanced new generation of software that melds computing and communications in a revolutionary new way, offering every developer the tools they need to transform the Web and every other aspect of the computing experience. We call this initiative Microsoft .NET, and for the first time it enables developers, businesses, and consumers to harness technology on their terms. Microsoft .NET will allow the creation of truly distributed Web services that will integrate and collaborate with a range of complementary services to serve customers in ways that today's dot-coms can only dream of. Microsoft .NET will drive the Next Generation Internet. It really will make information available any time, any place, and on any device.

The fundamental idea behind Microsoft .NET is that the focus is shifting from individual Web sites or devices connected to the Internet to constellations of computers, devices, and services that work together to deliver broader, richer solutions. People will have control over how, when, and what information is delivered to them. Computers, devices, and services will be able to collaborate with each other to provide rich services, instead of being isolated islands where the user provides the only integration. Businesses will be able to offer their products and services in a way that lets customers seamlessly embed them in their own electronic fabric. It is a vision that extends the personal empowerment first offered by the PC in the 1980s.

Microsoft .NET will help drive a transformation in the Internet that will see HTML-based presentation augmented by programmable XML-based information. XML is a widely supported industry standard defined by the World Wide Web Consortium (W3C), the same organization that created the standards for the Web browser. It was developed with extensive input from Microsoft, but is not a proprietary Microsoft technology. XML provides a means of separating actual data from the presentational view of that data. It is a key to the Next Generation Internet, offering a way to unlock information so that it can be organized, programmed, and edited; a way to distribute data in more useful ways to a variety of digital devices; and allowing Web sites to collaborate and provide a constellation of Web Services that will be able to interact with each another.

Microsoft .NET is comprised of the following:

- ***Microsoft .NET platform:*** The .NET Framework includes .NET infrastructure and tools to build and operate a new generation of services; .NET user experience to enable rich clients; .NET building block services, a new generation of highly distributed mega-services; and .NET device software to enable a new breed of smart Internet devices.
- ***Microsoft .NET products and services:*** Includes Windows.NET, the .NET Enterprise servers and a core integrated set of building block services; .NET My Services; MSN .NET; personal subscription services; Office.NET; Visual Studio .NET; and bCentral for .NET.
- ***Third-party .NET services:*** A vast range of partners and developers will have the opportunity to produce corporate and vertical services built on the .NET platform.

Microsoft .NET will take computing and communications far beyond the one-way Web to a rich, collaborative, interactive environment powered by this advanced new software. It will harness a constellation of applications, services, and devices to create a personalized digital experience – one that constantly and automatically adapts itself to your needs and those of your family, home, and business. It means a whole new generation of software that works as an integrated service to help you manage your life and work in the Internet Age.

For consumers, that means the simplicity of integrated services; unified browsing, editing, and authoring; access to all your files, work, and media online and off; a holistic experience across devices; personalization everywhere; and zero management. It means, for example, that any change to your information – whether input via your PC or handheld or smart credit card – will instantly and automatically be available everywhere that information is needed.

For knowledge workers and businesses, it means unified browsing, editing, and authoring; rich coordinated communication; a seamless mobile experience; and powerful information-management and e-commerce tools that will transparently move between internal and Internet-based services, and support a new era of dynamic trading relationships.

.NET
written by Robert McGovern
STREET FIGHTING Comes to WEB SERVICES

For independent software developers, it means the opportunity to create advanced new services for the Internet Age – services that are able to automatically access and leverage information either locally or remotely, working with any device or language, without having to rewrite code for each environment. Everything on the Internet becomes a potential building block for this new generation of services, while every application can be exposed as a service on the Internet.

The Microsoft .NET vision means empowerment for consumers, businesses, software developers and the entire industry. It means unleashing the full potential of the Internet. And it means the Web the way you want it. www.microsoft.com/business/whitepapers/net/net.asp

What's the business case for a Web services developer choosing .NET?

.NET extends the ideas of both the Internet and operating systems by making the Internet itself the basis of a new operating system via Web services. It provides the easiest and most extensive Web services platform. .NET was designed from day one as

the platform for distributed computing. Web services are built in rather than bolted on. Ultimately, this will allow developers to create programs that transcend machine, platform, and device boundaries. Businesses will thus benefit from radically increased efficiency and productivity as .NET brings employees, customers, data, and business applications into a coherent and intelligently interactive place. In short, .NET promises a world of business without boundaries.

.NET offers the richest Web services support and hence the widest level of interoperability thus providing developers and businesses with the most opportunity to leverage their investments. .NET was built as a platform for Web services and distributed computing. It has the best tools and support for Web services and is therefore the best choice for Web service developers.

Is there in fact a choice that needs to be made between J2EE and .NET?

Yes and no. The clear choice is to use .NET as the platform for application development and legacy application integration. Does that mean that legacy J2EE applications need to be rewritten? While many of our customers have chosen to do so for performance and cost reasons, this is not a requirement and developers using .NET can easily integrate and leverage these legacy systems by utilizing the Web services capabilities of .NET. It thus provides a platform for moving forward and for leveraging existing investments.

.NET has a proven track record of reducing costs, reducing time to market, increasing productivity and increasing flexibility.

Is .NET an open platform?

Yes. .NET is an open platform based on open standards. Microsoft has committed to taking the core protocols of .NET to standards bodies. Our track record in this space is stellar. We have led the charge in carrying the important standards for Web services to standards bodies – for example, SOAP, WSDL, WS-Inspection, and XML. Furthermore, we participate heavily in cross-platform interoperability testing of our implementations to these standards with community groups such as SOAPBuilders. Web services are an open standard and as the primary integration and interoperability mechanism for .NET, they make .NET the most open platform.

What's the argument against delay in making a choice?

Early adopters of .NET are seeing significant competitive and cost advantages. Companies who begin to make the move now see strong advantages over their competitors. Furthermore, the paradigms of application design, deployment, and integration are rapidly shifting to a new distributed model which .NET was specifically designed to support. The distributed model offers significant advantages and failure to capitalize on it could be catastrophic.

What security safeguards are built into .NET?

There are many aspects of security in .NET, from the capabilities built into Windows .NET and the .NET enterprise servers through the .NET Framework and on the Web services.

The .NET Framework provides the core of application and platform security. In today's software environment, applications come from many sources and perform many tasks. Trust in the application code is a key requirement because we don't want to risk damage to software or information. A security policy that grants permissions that it shouldn't may allow inappropriate access to sensitive information, or expose the local machine to malicious programs or even just plain buggy code.

Traditionally, security architectures have provided isolation and access control based on user accounts, giving code complete access rights within those restrictions and assuming that all code run by a particular user is equally trustworthy. Unfortunately, isolating code on a per-user basis will not sufficiently protect one program from another if both programs are running on the user's behalf. Alternatively, code that is not fully trusted has often been relegated to a sandbox model of execution, where code is run in an isolated environment with no access to most services.

A successful security solution for today's applications must strike a balance between the two models of security. It must provide access to resources in order for useful work to be done, and it requires finer control of application security to ensure that code is identified, examined, and given an appropriate level of trust. The .NET Framework security model provides just such a solution. http://msdn.microsoft.com/vstudio/nextgen/technology/frame worksec.asp

.NET delivers Web services capabilities which have their own set of security requirements for building distributed applications within and across organizations, people and devices. To meet these additional needs, .NET's Web services build on existing security protocols such as SSL, XML Encryption, XML Signature, etc. to provide a cohesive open security architecture. http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000442

What compatibility does .NET have with J2EE?

Microsoft fully supports Web services across .NET as the interoperability mechanism of choice.

What real-world implementations of .NET exist?

We have many examples. You can find detailed case studies, on MSDN.

How does .NET performance and scalability stack up against competitive platforms?

See http://gotdotnet.com/team/compare

Microsoft implemented the complete Java Pet Store, Sun's primary J2EE reference application, using Microsoft .NET and C#. The .NET Pet Shop implements the same functionality as the Java Pet Store, but does so in one third the amount of code. The white paper contains direct comparisons of the .NET and J2EE implementations, including full source code to the .NET Pet Shop, a new blueprint application for the .NET platform. Microsoft also used .NET to extend the application beyond Sun's implementation with an XML Web Service and support for mobile devices.

The white paper also details how the .NET Pet Shop performs and scales versus the J2EE Java Pet Store. Microsoft took the Oracle 9*i* Application Performance Challenge and showed how the .NET Pet Shop offers 28 times faster performance than Oracle's highly optimized J2EE version of the Java Pet Store running on the same hardware configuration, based on Oracle's own published data!

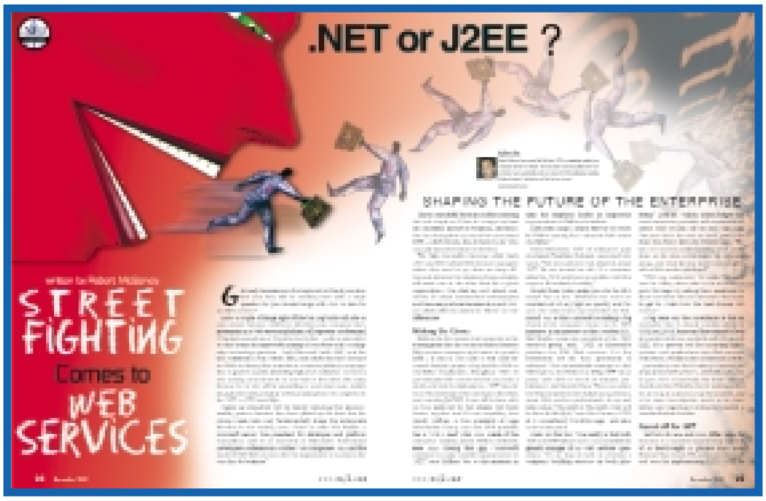
Microsoft has implemented and benchmarked .NET using the Java reference application; why aren't there TPC-C benchmarks posted against any Java app servers?

If a company chooses not to implement .NET, what will they lose?

Opportunity costs, competitive advantage, flexibility, ease of development, and deployment.

Third-party analysts are telling me .NET is "immature," "unproven." What's the argument against that?

.NET builds off nine years of services built into the Windows operating system – it's a new programming model on top of Windows.


.NET or J2EE ?
STREET FIGHTING Comes to WEB SERVICES
SHAPING THE FUTURE OF THE ENTERPRISE

WebServices JOURNAL
.NET J2EE XML

COMING IN THE March ISSUE

Secure Web Services: A Matter of Trust
by Norbert Mikula

Web Services Security
by Mark O'Neill

Scaling Web Services: JMS Meets JCache
by Steve Ross-Talbot

The Sun Rises on Web Services: An Overview of Sun's JAX Pack
by Kyle Gabhart

Getting into the Flow: WSFL
by James Snell

Web Services Are Here and Now
by Scott Durschlag

Securing Web Services
by James Ducharme

WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
BaseBeans Engineering	www.basebeans.com	866-460-2320	65
BEA eWorld	www.bea.com/events/eworld/2002	404-240-5506	21
engindata Research	www.engindata.com		37
HP Bluestone	www.hpmiddleware.com/download	856-638-6000	67
Java Developer's Journal	www.sys-con.com	800-513-7111	39
JavaOne	http://java.sun.com/javaone	888-886-8769	23
JDJ Store	www.jdjstore.com	888-303-JAVA	55
Loox Software	www.loox.com	800-684-LOOX	33
Mongoose Technology	www.portalstudio.com		25
New Atlanta Communications	www.newatlanta.com		31
Object Management Group	www.omg.org/mda	781-444-0404	17
ParaSoft	www.parasoft.com/wsj2	888-305-0041	29
Quintessence	www.in2j.com		15
Sams Publishing	www.samspublishing.com	800-571-5840	13
Sonic Software	www.sonicsoftware.com		2, 3
SpiritSoft	www.spiritsoft.net/downloads		4
Sun Microsystems	www.sun.com/sunoneinfo		10, 11
SYS-CON Events	www.sys-con.com	201-802-3069	52, 53
SYS-CON Media	www.sys-con.com	800-513-7111	41, 55
Web Services Edge Conference & Expo	www.sys-con.com	201-802-3069	56, 57
Web Services Journal	www.sys-con.com	800-513-7111	45
WebSphere Developer's Journal	www.sys-con.com	800-513-7111	65
Wireless Business & Technology	www.sys-con.com	800-513-7111	35
Wireless Edge Conference & Expo	www.sys-con.com	201-802-3004	62, 63
XML Global Technologies	www.xmlglobal.com/newangle	800-201-1848	68
XML-Journal	www.sys-con.com	800-513-7111	60

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser.

wirelessEDGE conference&expo

MAY 7–9, 2002
Santa Clara Convention Center
Santa Clara, California

STRATEGIES & SOLUTIONS FOR UNWIRING THE ENTERPRISE

International Wireless Business & Technology Conference & Expo

Wireless Edge will provide the depth and breadth of education and product resources to allow companies to shape and implement their wireless strategy. Developers, *i*-technology professionals and IT/IS management will eagerly attend.

Plan to Exhibit

Provide the Resources To Implement Wireless Strategy

The conference will motivate and educate. The expo is where attendees will want to turn ideas into reality. Be present to offer your solutions.

Who Should Attend

Mobile & Wireless Application Professionals

who are driving their enterprises' wireless initiatives:

- Program Developers
- Development Managers
- Project Managers
- Project Leaders
- Network Managers
- Senior IT and Business Executives

CONFERENCE TRACKS

TRACK 1: DEVELOPMENT
- WAP
- i-Mode
- Bluetooth / 802.11
- Short Messaging
- Interactive Gaming
- GPS / Location-Based
- Wireless Java
- XML & Wireless technologies

TRACK 2: CONNECTIVITY
- Smart Cards
- Wireless LANs incl. Bluetooth
- UMTS/3G Networks
- Satellite Broadband

TRACK 3: WIRELESS APPS
- Education
- Healthcare
- Entertainment
- Transport
- Financial Services
- Supply Chain Management

TRACK 4: HARDWARE
- Cell Phones/WorldPhones
- PDAs
- Headphones / Keyboards / Peripherals
- Transmitters / Base stations
- Tablets

TRACK 5: MANAGEMENT
- Wireless in Vertical Industries
- The WWWW
- Unwired Management
- From 3W to 4W: Issues and Trends
- "Always-On" Management
- Exploiting the Bandwidth Edge
- Unplugged Valueware
- Wireless Sales & Marketing

FOR EXHIBIT & SPONSORSHIP INFORMATION PLEASE CALL
201 802-3004

Arie Mazur
President, Chief Executive Officer, and Chairman, Slangsoft

Chris Bennett
Architect, Freedom Technologies

Daniel Elliott
Senior Vice President, Mobile Business, CompuCom

Douglas Lamont
Professor of Marketing, DePaul University

Kenneth Leung
Business Development Manager, Retail and Emerging Technologies, IBM

Keith McIntyre
Vice President and Chief Technologist, Stellcom

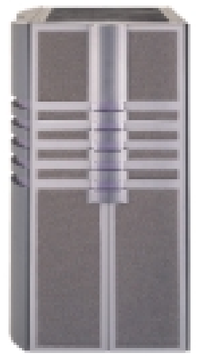
Steve Milroy
Wireless Technologist, Immedient

Tony Wasserman
Director, Mobile Middleware Lab, Hewlett-Packard Company

WWW.SYS-CON.COM/WIRELESSEDGE2002

Exclusive Sponsorship Available

Rise above the noise. Establish your company as a market leader. Deliver your message with the marketing support of SYS-CON MEDIA



ONLINE REGISTRATION NOW OPEN

WWW.SYS-CON.COM/WIRELESSEDGE2002

Or, To Register By Phone
Call: (201) 802-3069

	Development	Connectivity	Wireless Applications	Hardware	Management
MAY 8, 2002					
8:30-9:45	Embedded Java (Bill Ray, Network 23 Limited)	A Real Time Model of (3G) UMTS Access Stratum (Niloy Mukherjee, MIT Media Laboratory)	Using Mobility to Streamline Traditional Business Processes (Dan Elliott, CompuCom)	Java Software Performance On Wireless Devices: Myths and Realities (Ron Stein, Nazomi)	LMDS - The Last Mile Enabler of Class 5 Revenues with VoIP (Ed Peters, Ensemble Communications, Inc.)
10:00-11:15	KEYNOTE				
11:30 A.M.	EXPO FLOOR OPEN				
	LUNCH BREAK				
1:30-2:30	XML & Wireless Technologies (Karl Best,OASIS)	Securing Wireless Data Via Smart Card (Joseph Smith, New Dominion Software)	Collaboration for Wireless Warriors (Timothy Butler, SiteScape, Inc)	User Interactivity for Information Appliances (Arie Mazur, Slangsoft)	Leveraging Wireless In Customer Acquisition and Retention (Kenneth Leung, IBM)
2:45-3:45	KEYNOTE				
3:45-4:45	Developing Mobile Web Applications (Tony Wasserman, Hewlett-Packard Company)	In-building Wireless, the Next Frontier (Mary Jesse, RadioFrame Networks)	Turbocharge Mobile Applications with J2EE (Dr. Jeff Capone, Aligo, Inc.)	Cell Phones/ WorldPhones (Speaker TBA)	Mobilize Your Enterprise (Chris Bennett, Freedom Technologies)
5:00-6:00	VoiceXML Workshop (Bryan Michael, BeVocal)	Satellite Broadband (Speaker TBA)	Mobile Portals - The First Step Towards the Mobile Enterprise (Tony Wasserman, Hewlett-Packard Company)	PDAs (Speaker TBA)	Wireless Solution Return On Investment (ROI) In the Real World (Steve Milroy, Immedient)
MAY 9, 2002					
8:30-9:45	Brew, I-Mode, WAP, and J2ME: How the Battlefield Is Shaping Up at the Start of the Mobile War (Reza B'Far, eBuilt, Inc.)	The Future of IP Mobility (Antti Eravaara, NetSeal, Inc.)	Multimodality: Revolutionizing Our Wireless Lifestyles (Arvind Rao, OnMobile Systems)	Tablets (Speaker TBA)	Marketing Value in Automotive Telematics Through Mobile Communications (Douglas Lamont, DePaul University)
10:00-11:15	KEYNOTE				
11:30 A.M.	Guide to Developing Applications with Micro Java (Kurt Baker, Kada Systems)	EXPO FLOOR OPEN			
	LUNCH BREAK				
1:45-3:00	Building Secure Mobile Solutions (Keith McIntyre, Stellcom)	Smart Card Communications (Bill Ray, Network 23 Limited)	Developing New Applications Using VoiceXML (Jonathan Taylor, Voxeo)	PDAs (Speaker TBA)	Policies and Profiles: The Key to Mobile Data Services (Doug Somers, Bridgewater Systems)
3:15-4:30	Bluetooth™ Wireless Technology and Java™ Technology (Michael Portwood, Exuberance, LLC)	UMTS/3G Networks (Speaker TBA)	Instant Messaging and Wireless Computing Collide (JP Morgenthal, IKimbo)	Transmitters / Base Stations (Speaker TBA)	Total Business Solutions B2E (Speaker TBA)
4:45-6:00	Creating Carrier Optimized Wireless Internet Applications (David Young & Victor Brilon, Lutris)	Wireless LANs/Bluetooth (Speaker TBA)	Rapid Development of Successful Wireless Applications (Rod Montrose, AVIDWireless)	Headphones / Keyboards / Peripherals (Speaker TBA)	Wireless Sales & Marketing (Speaker TBA)

REGISTER BEFORE March 1 And Save Up To $400

Plan to Attend the 3-DAY Conference

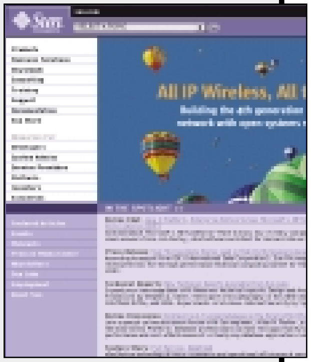
PRODUCED BY SYS-CON EVENTS

May 7 will feature full-day tutorials. Consult www.sys-con.com/wirelessedge2002 for up to-the-minute conference news.

*Program subject to change.

Sun and Bowstreet to Deliver Dynamic Web Services Assembly on Sun ONE Platform

(Portsmouth, N.H. and Santa Clara, CA) – Bowstreet and Sun Microsystems, Inc. have advanced the Sun ONE services-on-demand initiative with an agreement to port the Bowstreet Business Web Factory to the Sun ONE architecture, including the iPlanet Application Server and the iPlanet Portal Server platform. The agreement brings dynamic Web services assembly – the ability to assemble and customize the appropriate web services at run-time – to the enterprise.


All IP Wireless, All

Building the 4th generation network with open systems

As part of the agreement, Bowstreet will offer a fully integrated version of its Web services development and assembly platform, the Business Web Factory, for the iPlanet Application Server by the end of the first quarter of 2002. The Business Web Factory will also provide integrated support for the iPlanet Portal Server at the same time.

www.sun.com www.bowstreet.com

SoftWIRE for Visual Studio .NET to Simplify Computer Programming

(Middleboro, MA) – SoftWIRE Technology has announced the launch of SoftWIRE for Visual Studio .NET. SoftWIRE, a graphical programming package that integrates with the Microsoft Visual Studio .NET integrated development environment (IDE). Its intuitive nature assists users in creating powerful applications in Microsoft Visual Basic .NET and Visual C# .NET without having to learn a programming language.

SoftWIRE BUILD IT AND WIN IT CONTEST

In SoftWIRE, users can create Visual Basic .NET or C# projects without writing a single line of code. Using Microsoft .NET components, SoftWIRE provides access to advanced features such as e-mail (MAPI), database (ADO.NET), network communications (TCP/IP) and office automation.

SoftWIRE for Visual Studio .NET is available online and is scheduled to release simultaneously with Microsoft Visual Studio .NET. Prices for SoftWIRE for Visual Studio .NET start at $395.00 for upgrades from SoftWIRE 3.1 for Visual Basic 6.0. For new users SoftWIRE is $495.00, U.S. list. All run-time applications may be distributed royalty free.

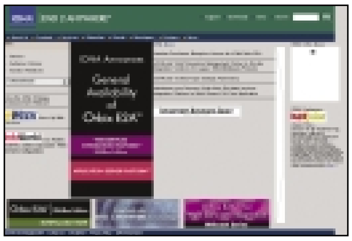
www.softwire.com

Industry's First Web Services Integration Platform Enables 'End 2 Anywhere' Integration

(Waltham, MA) – IONA, the leading e-Business Platform provider for Web services integration, has announced the general availability of its Orbix E2A Web Services integration solutions. IONA Orbix E2A offers both an integration platform and an application server platform to enable customers to easily create, connect, and manage End 2 Anywhere Web services integration solutions that liberate their information assets.

Orbix E2A offers the industry's first Web Services integration platform. Unlike solutions from other vendors that add Web services as a "feature," Web services are at the heart of Orbix E2A. This enables customers to deploy Web services integration solutions at the higher-value application and business process levels, as well as the lightweight component level supported by vendors of application servers and toolkits.

The Orbix E2A Application Server Platform is available in three editions: J2EE Technology, Standard, and Enterprise. The Orbix E2A Web Services Integration Platform XMLBus Edition is also immediately available. Pricing for both platforms starts at $495 per development license and $2,500 per CPU for a


IONA Announces General Availability of Orbix E2A

deployment license. License pricing varies by edition.

www.iona.com

IBM Boosts Web Services With New Technologies For Developers And Service Providers

(Somers, NY) – IBM has announced tools that allow developers and service providers to easily create, host, and monitor Web services. The Web Services Hosting Technology, the Web Services Gateway, and the Web Services Toolkit (WSTK) 3.0 are available for free trial download on alphaWorks, (www.alphaWorks.ibm.com), the destination for IBM emerging technology.

IBM's Web Services Hosting Technology is a set of management tools that support revenue-generating Web services. It is designed specifically for software developers and service providers, empowering them to bring Web services to a hosted environment regardless

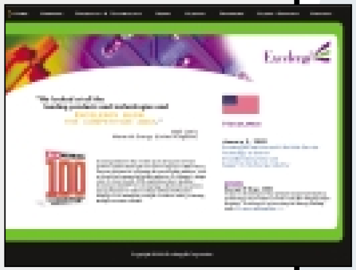
of the actual Web services implementation. It also supports the provisioning and metering of Web services without requiring code changes, decreasing time to market and development expenses for an ISV or service provider. In addition, it allows service providers to develop an integrated billing model to help them more easily track usage and collect revenue for the services.

www.ibm.com

Excelergy and Microsoft .NET Web Service Technology Deliver Value and Support to Energy Industry

(Lexington, MA) – Excelergy Corporation has announced it's working with Microsoft Corporation to combine key elements of the Microsoft .NET application development platform with Excelergy's open, component-based, native XML technology platform to deliver the most advanced Web services available in the energy and utility industry.

Excelergy eXACT 5.0 automatically accepts, translates, validates and securely transmits mission-critical business and customer information for vital processes such as forecasting, trading, scheduling, billing, and settlement in retail and wholesale energy markets. Future versions of Excelergy eXACT will incorporate Microsoft .NET technologies such as .NET Alerts, which allow users to automatically receive instant notifications and securely access corporate data from any electronic device.



www.excelergy.com

Written by Anne Thomas Manes

Identity Crisis

Passport may not fill the need for a global identity service

Do you have a .NET Passport identity? You may not realize it, but chances are reasonably high that you do. If you have a HotMail or MSN account, Microsoft assigned a Passport identity to you automatically. Microsoft claims to have more than 160 million users registered in the Passport identity service.

Pretty soon you'll need a Passport ID to have any interaction with Microsoft. In December 2001, quite a few gamesters were surprised to discover that their old accounts at the Microsoft Zone gaming site wouldn't work without a Passport ID. Microsoft also requires a Passport ID to join MSDN, to register for a Microsoft seminar, or to access Microsoft's node in the UDDI public registry. The new Windows XP Product Activation (WPA) system uses Passport by default. You can also use your Passport ID to log in to your XP system.

So just what does a Passport identity do for you? Obviously, it lets Microsoft track your activities, but that's not a particularly strong incentive for most users. Most consumers view Passport as an electronic wallet. You can associate a credit card with your Passport ID and use it to buy things at any site that supports Passport Express Purchase. This sounds pretty useful, except that Microsoft hasn't been especially successful in recruiting e-tailers to support Passport (there are less than 100 participating sites so far). Even so, Passport can fill in Web forms for you, alleviating the need to type in your name and address at every site.

But Passport has a much more useful role to play in the future, particularly in the realm of Web services. Passport provides a cross-corporate single-signon service, which is critical to allow Web services to work together.

Today most Web services work alone, but in the future we want to be able to assemble multiple Web services to create more powerful business services. First we need to provide a way to let Web services share information.

Consider how most Web services implement security today. Each business that offers a secure Web service maintains a list of authorized users, who authenticate themselves using a userid and password. When we start assembling Web services, we don't want to force the user to type in a userid and password for every Web service involved in the aggregate business service, and we don't want to force Web service providers to develop point-to-point security connections for each Web service assembly effort. Instead, we need a facility that enables single sign-on across any number of Web services operated by any number of businesses. What we need is a global identity service.

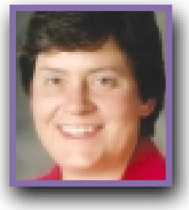
Passport could be used as a global identity service but there is one serious impediment. Passport isn't open. Microsoft intends to collect revenue from businesses that use Passport for authentication. While I will grant that Microsoft has the right to make money from its innovations, I suspect that not every business wants to pay Microsoft to manage its authentication process. And many businesses won't be inclined to let Microsoft own their customer information. Hence, I doubt that Passport will ever become the de facto global identity service.

In September Microsoft announced plans to "open up" Passport by adding support for Kerberos V5, but this feature still won't make Passport open. What it means is that Passport will be able to access your internal user management system (such as Active Directory) to retrieve user identity information, assuming, of course, that it supports Kerberos V5.0. (I probably don't need to tell you that Active Directory supports Kerberos V5.0.) Keep in mind that although you would be managing and maintaining your own list of authorized users, all identity and authentication requests still need to go through Microsoft's Passport service, allowing Microsoft to collect a toll.

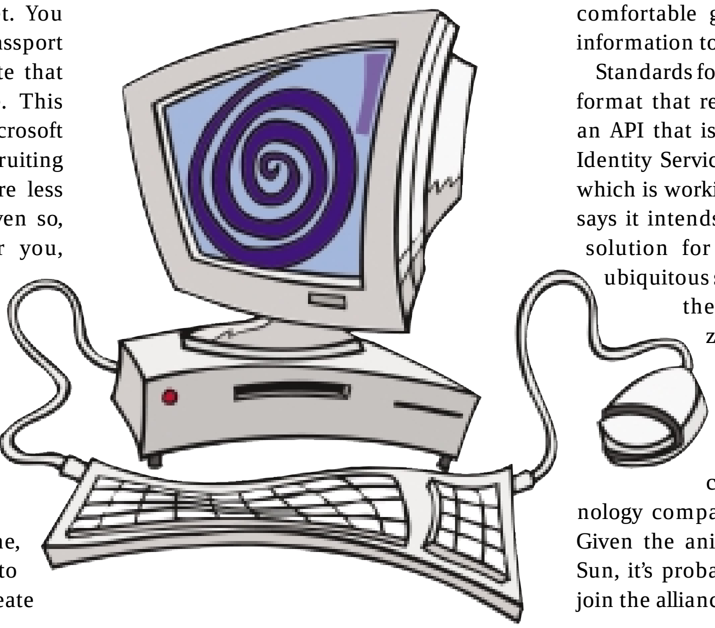
It would be a lot better if there were standards associated with Identity that would allow anyone to set up an Identity Service so that any Web service could authenticate users using any Identity Service. This approach would allow users a wide choice of Identity options. Personally, I'd feel much more comfortable giving control of my financial information to my bank than to Microsoft.

Standards for Identity would include an XML format that represents user information and an API that is used to access any compliant Identity Service. The Liberty Alliance Project, which is working to define Identity standards, says it intends "to create an open, federated solution for network identity – enabling ubiquitous single sign-on, decentralized authentication, and open authorization from any device connected to the Internet." Liberty was started by Sun Microsystems and is backed by a plethora of consumer, financial, telco, security, and technology companies. Even AOL has joined up. Given the animosity between Microsoft and Sun, it's probably unlikely that Microsoft will join the alliance. But we can only hope.

"Pretty soon you'll need a Passport ID to have any interaction with Microsoft."



Author Bio:
Anne Thomas Manes is the CTO of Systinet, a Web services infrastructure company. Anne is a recognized industry spokesperson and has published on a range of technology issues. ATM@SYSTINET.com



HP Bluestone

www.hpmiddleware.com/download