Java™ Technology + .NET: Bridging the Gap

Ted Neward

Neward & Associates
http://www.tedneward.com

TS-8029

Credentials

- Who is this guy?
  - Independent consultant, architect, mentor
  - Instructor, PluralSight (www.pluralsight.com)
  - BEA Technical Director, Microsoft MVP Architect
  - Java Specification Request (JSR) 175, 250, 277 EG member
  - Founding Editor-in-Chief, TheServerSide.NET
  - Author
    - *Effective Enterprise Java* (Addison-Wesley, 2004)
    - *Effective .NET* (Forthcoming)
    - *Pragmatic XML Services* (Forthcoming)
    - *Pragmatic .NET Project Automation* (Forthcoming)
    - *Server-Based Java Programming* (Manning, 2000)
    - *C# in a Nutshell, 2nd Ed* (OReilly, 2003)
    - *SSCLI Essentials* (w/Stutz, Shilling; OReilly, 2003)
  - Papers at http://www.tedneward.com
  - Weblog at http://blogs.tedneward.com

Objectives

- Java platform, .NET: each with different philosophies and approaches to solving problems
  - Neither is right, neither is wrong, they're just different
  - Instead of fighting over which is the best one-size-fits-all solution…
  - … how do we leverage the best of both worlds?
- Note: NO POLITICS!
  - I don't work for Microsoft, I'm not selling anything
  - I just look for the best* solutions for my customers

    "best" is a relative term, a value judgment…

    …and it's my clients who make that judgment

The Problem of Platforms

- Bridging two platforms comes in 3 forms
  - Migration: rewrite the code from one to the other
    - Expensive in terms of time and money
    - Multiple source bases lead to complication of maintenance
    - Lack of centralization complicates atomic processing
  - Portabililty: taking the code as-is and recompiling
    - No clear common platform between .NET and Java platform
    - Web services represent a new platform of its own
  - Interoperability: using the compiled system as-is
    - Quickest, in many respects (which is why it's done so frequently)

Why Interoperability?

- Analysts predict by 2005 80% of all enterprises will be joint Java 2 Platform, Enterprise Edition (J2EE™ platform)/.NET environments
  - Market share split between the two
    - J2EE platform 35–40%
    - .NET 35–40%
  - Neither is “going away” any time soon

This is the “You-Have-to-Do-It” reason

Why Interoperability?

- Both Java platform and .NET have something to offer
  - WPF offers a huge wealth of new UI offerings
  - Workflow provides a new way of integrating “knowledge workers” into the software development landscape
  - Java platform dominates the server landscape with a proven track record of scalability and maintenance
  - Or, perhaps, “nobody gets fired for choosing Java platform”
- The Evil Twin Brothers have each followed their own path—make use of that!

This is the “You-Want-to-Do-It” reason

The Players

- NetFX 3.0
  - Windows Presentation Foundation (“Avalon”)
    - Graphical layer leveraging modern graphic card capabilities to enhance user-interface experience
  - Windows Communication Foundation (“Indigo”)
    - Unified communication toolkit and runtime focused on interoperability and service-orientation
  - Windows Workflow Foundation
    - Programming layer designed for long-running processing, spanning multiple users and programs

Office

- Microsoft Office
  - External automation: any out-of-proc use of the object models exposed by Office applications
  - In-proc automation: in-process use of the object models exposed by Office apps (COM, .NET)
  - Add-ins: components hosted inside Office apps to extend application functionality
  - Smart tags: elements of text in an Office document that are recognized as having custom actions
  - Smart documents: DLLs that implement custom code associated with an XML schema attached to a doc
  - Excel real-time data: in-proc/out-of-proc components bringing data into Excel in real-time
  - Web services and research services
  - XML documents
  - Custom task panes, custom ribbons, custom form regions, custom digital signatures, custom blog extensions…

Java Platform

- Java Platform, Enterprise Edition (Java EE)
  - Swing: UI toolkit aimed at flexible look & feel on multiple platforms
  - Servlets, JavaServer Pages™ (JSP™ page) technology: HTTP-based access
  - JDBC™ API: Call-level access to relational data
  - Java Message Service (JMS): Messaging
  - EJB™ architecture: Transactional application server
  - Java RMI: Point-to-point object RPC
  - Java Architecture for XML Binding (JAXB) software: Java API for XML Binding
  - JAX-WS: Java API for XML Web Services
- … and about four billion different OSS projects
  - Spring, WebWork, SWT, Tapestry…

Scenarios

- .NET rich client, Java platform back end
  - Office-to-Java platform
    - Office is the pre-eminent rich client platform for the Windows platform—back it up against Java platform services
  - WPF-to-WCF-to-Java platform
    - Windows Presentation is the next-generation presentation layer, designed to take advantage of full graphics card functionality—let it display Java platform data in rich and new ways over WCF/JAX-WS
  - WPF-to-Java platform-generated XAML
    - Windows Presentation can also “browse” XAML directly, generated from within servlet/JSP file pages

Scenarios

- Java platform front-end, .NET back-end
  - Eclipse RCP hosting Office
    - Utilize Eclipse's ability to host COM components to host Word, Excel automation objects
  - SWT fronting for WPF
    - Use SWT but take advantage of WPF/Aero features
  - Struts driven by Workflow
    - Use Windows Workflow as a page-level driver to make it easier to go between Struts pages (in essence, replacing the action mappings XML file)
  - Spring/SpringMVC hosting Workflow
    - Spring receives incoming calls, feeds them to Workflows, which use activities to "call out" as necessary to other Spring services

How Interoperability?

- Tiers vs. layers
  - Tier: physical node in the network topology
  - Layer: software abstraction intended to ease development and maintenance of code
- 3 Layers
  - Layers: presentation, business, resource
- 3 Tiers
  - Tiers: client, middle, server
- Crossing tiers isn't the problem
- Interop within a single layer is the problem

How Interoperability?

- 3 Modes
  - Intra-process
  - Inter-process
  - Resource-oriented
- Combinations of modes and layers/tiers
  - Presentation interoperability: sharing session state
  - Presentation/business interop
  - Business interop: EJB architecture calling COM+, or vice versa
    - As part of transaction or independently
  - Resource interop: message brokers, database, etc.

Basic Principles of Interoperability

- Key problems of any interop technology
  - Agreement on data types (endian-ness, size, etc.)
  - Agreement on invocation semantics (pass-by-ref, pass-by-val)
  - Lifecycle and identity management
  - Security protocols
  - Lookup model
  - State management model (persistence)
  - Processing model (propagating transactions)
  - Threading model
  - Synchronization model
- The more tightly coupled the principals, the more difficulties involved
  - Key: keep things loosely-coupled as much as possible!

How Interoperability?

- The Interoperability Continuum of Complexity
  - Top-down (simple to complex)
  - Start from the top, work your way down
  - With power comes complexity; with complexity, power

3 Approaches to Interop

- Resource-based interop
  - Database: “Everybody knows SQL”
  - Filesystem: XML is your friend here
  - Filesystem: Java platform/J# Serialization also works
  - “Brokers”: BizTalk, MQSeries, established software in place
- Out-of-process interop
  - Simple protocols: raw HTTP, SMTP/POP3, sockets
  - REST: leveraging the infrastructure of the Web
  - Binary messaging: vendor toolkits, messaging style
  - Binary RPC: vendor toolkits, CORBA for RPC semantics
  - Web services: the new platform (both RPC and messaging)
- In-process interop
  - IKVM: translating Java bytecode on the fly to CIL
  - JuggerNET: in-proc generated code proxies
  - JNI™ API/Managed C++: hosting Java platform and .NET together

3 Approaches to Interop

- Resource: database access
  - Relational database is everybody's friend
    - Well-known, well-understood paradigm
    - Schema defines strong constraints around data
  - Java platform:
    - JDBC API, SQL/J, RowSets for direct relational access
    - Java Data Objects, EJB architecture Entity beans, Hibernate for object access
    - Stored procs for procedural-based access
  - .NET:
    - ADO.NET, DataSets for direct relational access
    - ObjectSpaces, others for object-based access
    - Stored procs for procedural-based access
  - Works for other platforms, too

3 Approaches to Interop

- Resources: filesystem/formats
  - Office 11, Office 12 have well-known formats for storing documents (CSV, WordML, OpenXML)
    - These documents can be generated from lots of different sources, including Java platform web apps
  - Office also has a prior binary format that requires **much** greater work to use
    - See Apache POI

3 Approaches to Interop

- Resource: filesystem/XML
  - XML is the **lingua franca** of the enterprise
    - XSD defines constraints for data
  - filesystem is well-known, well-understood, always available
    - No surprises here
    - Systems have been using it for decades
  - Java platform: Java Architecture for XML Binding (JAXB)
    - Other approaches include Apache XMLBeans
  - .NET: XSD.exe, XmlSerializer
  - Works for other platforms, too
  - Key: “Start from the middle”; in this case, XSD
    - Or RelaxNG, or…
    - XSD just happens to be better supported

3 Approaches to Interop

- Resource: filesystem/serialization
  - Java Object Serialization can also serve as a convenient middle ground between Java technology and .NET
    - Java 2 Platform, Standard Edition (J2SE™ platform) is backwards-compatible to JDK 1.1 software…
    - …and J# supports JDK 1.1…
    - …which means Serialization works both ways
  - Key: Start from the middle (object model, in this case)

3 Approaches to Interop

- Resource: “brokers”
  - Products like BizTalk, MQSeries, and others have already solved a certain set of interoperability issues… if you buy in!
  - Many of them also address higher-order issues as part of the overall package, like workflow/orchestration
  - Fuzzy area—can easily be pegged elsewhere in the list, depending on how you use them (messaging, etc.)
  - “Legacy systems” fall into this category a lot

3 Approaches to Interop

- Out-of-process: simple protocols
  - TCP/IP: basic data exchange
    - Java platform: java.net.*
    - .NET: System.Net
  - HTTP: basic exchange of information (MIME)
    - Java platform: java.net.* (HttpUrlConnection)
    - .NET: System.Web
  - Still have to agree on data exchange format
    - Arguably just an extension of filesystem interop
    - XML works well here (see above)

3 Approaches to Interop

- Out-of-process: REST
  - REpresentational State Transfer: Leverage the infrastructure of the Web the way it was intended to be used
  - Using HTTP verbs (GET, PUT, DELETE, HEAD, TRACE, OPTIONS, POST) to indicate the action desired
  - Exchange data as XML documents in body of HTTP request (or in some other mutually-acceptable form)
  - Takes full advantage of Web infrastructure (caching, proxy servers, etc.)
  - Simple to develop and maintain
  - Doesn't handle security, transactions, routing…
    - Left to you to deal with

3 Approaches to Interop

- Out-of-proc: messaging
  - Communication style that focuses on independent, context-complete packages of information
    - Messaging exchange patterns provide flexibility
    - See Enterprise Integration Patterns
  - AQMP: Cross-vendor wire messaging standard
  - Java platform: JMS technology
    - Sonic MQ, Fiorano, SpiritSoft, Oracle AQ, and more
    - Some have .NET/COM bindings
  - .NET: MSMQ, WCF, SQL Server Service Broker
    - Service Broker is particularly interesting, since its access is through JDBC API
  - Email is the Internet's original messaging system
    - Portable, scalable, well-understood solutions
    - What else do you want from a messaging system?
  - RDBMS, filesystems also make good messaging layers
  - SOAP 1.2 works (very) well here for message payload as transport-agnostic messaging framing and extensibility rules

3 Approaches to Interop

- Out-of-proc: binary RPC
  - CORBA's been here since '94
    - Well defined in terms of J2EE platform
    - Java platform: J2SE platform, other vendors
    - .NET: Borland Janeva, IIOP.NET, C++ vendors (using MC++)
    - Offers security, transaction propagation, and so on
    - Lacks "sexiness" of Web services, lots of emotional baggage
  - Others (JaNET, JNBridge) offer similar capabilities
    - Usually built around similar ideas (naming service…)
    - Not as widespread; proprietary vendor products
  - Key: Start from the middle, work your way out
    - CORBA: IDL
    - Others: usually a language interface
    - Be careful to stick with consumable types on both ends!

3 Approaches to Interop

- Out-of-proc: Web services
  - Both RPC-style and messaging
    - WSDL currently fronts widely for RPC
    - Messaging not well-supported (yet)
  - Large number of specs (> 25) to handle “heavy lifting”
    - Security, transaction management, activity/orchestration
    - Automated policy exchange
    - Automated code generation of language-based proxies
  - Java platform: JAX-WS API, fragmented vendor support
    - JAX-WS API Providers and Handlers are quite possibly your godsend here
  - .NET: ASP.NET/ASMX (legacy), WCF
    - SOAP Extensions are quite possibly your godsend here
  - Key: Start from the middle, work out
    - This means writing WSDL first! (sort of)

3 Approaches to Interop

- In-process: JNI™ API/Managed C++
  - Both the Java Virtual Machine (JVM™) and CLR are fundamentally "just" DLLs
    - Java platform: JNI API talks natively (C/C++) to the OS
    - CLR: supports C++/CLI as a bridge
    - Use MC++ to write JNI API DLLs/JNI API Invocation code
  - Warning: Lots of tricky issues to be aware of
    - Data transcription from one type system to another
    - Awkwardness of working with JNI API model (JACE!)
    - Thread affinity, synchronization scoped to JVM software/CLR

The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java™ platform.

3 Approaches to Interop

- In-process: IKVM
  - Open-source project that converts bytecode to CIL
    - Can be done either on-the-fly or ahead-of-time
  - This is actually not interoperability, so much as a flavor of migration (in a way)
    - Java code isn't executing in the JVM software, but in the CLR
  - Principally useful when the code is important, not data
- In-process: JaCIL
  - Open-source project that seeks to do as IKMV does, but going the other way
    - Still very new, very immature

Summary

- Each platform has its strengths
  - Use them!
- Goal here is not to force people to “switch”
  - But instead to leverage each technology’s advantages as they appear and as they’re necessary

Q&A

Java Technology + .NET: Bridging the Gap

Ted Neward

Neward & Associates
http://www.tedneward.com

TS-8029