

---

# **Java + .NET: Why, exactly?**

Ted Neward  
Neward & Associates  
<http://www.tedneward.com>

# Credentials

---

- Who is this guy?
  - Independent consultant, architect, mentor
  - Instructor, PluralSight ([www.pluralsight.com](http://www.pluralsight.com))
  - BEA Technical Director, Microsoft MVP Architect
  - 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, 2<sup>nd</sup> Ed* (OReilly, 2003)
    - *SSCLI Essentials* (w/Stutz, Shilling; OReilly, 2003)
  - Papers at <http://www.tedneward.com>
  - Weblog at <http://blogs.tedneward.com>

# Objectives

---

- Java, .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 its 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 & money**
    - **Multiple source bases lead to complication of maintenance**
    - **Lack of centralization complicates atomic processing**
  - Portability: taking the code as-is and recompiling
    - **No clear common platform between .NET and Java**
    - **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 J2EE/.NET environments
  - Market share split between the two
    - J2EE 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 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 dominates the server landscape with a proven track record of scalability & maintenance

**Or, perhaps, “nobody gets fired for choosing Java”**
- 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**

- Microsoft Office

- External automation: any out-of-proc use of the object models exposed by Office apps
- 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 Enterprise Edition
  - Swing
    - **UI toolkit aimed at flexible look & feel on multiple platforms**
  - servlets, JSP
    - **HTTP-based access**
  - JDBC
    - **Call-level access to relational data**
  - JMS
    - **Messaging**
  - EJB
    - **Transactional application server**
  - RMI
    - **Point-to-point object RPC**
  - JAXB
    - **Java API for XML Binding**
  - JAXWS
    - **Java API for XML Web Services**
- ... and about four *billion* different OSS projects
  - Spring, WebWork, SWT, ...

# Scenarios

---

- .NET rich client, Java back end

- Office-to-Java

- Office is the pre-eminent rich client platform for the Windows platform—back it up against Java services**

- WPF-to-WCF-to-Java

- Windows Presentation is the next-generation presentation layer, designed to take advantage of full graphics card functionality—let it display Java data in rich and new ways over WCF/JAX-WS**

- WPF-to-Java-generated XAML

- Windows Presentation can also "browse" XAML directly, generated from within servlet/JSP pages**

# Scenarios

---

- Java front-end, .NET back-end
  - Eclipse RCP hosting Office
    - utilize Eclipse's ability to host COM components to host Word, Excel automation objects**
  - 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 & layers/tiers
  - Presentation interoperability: sharing session state
  - Presentation/business interop
  - Business interop: EJB 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/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/Managed C++: hosting Java 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:
    - JDBC, SQL/J, RowSets for direct relational access**
    - JDO, EJB 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 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: JAXB (Java API for XML Binding)
    - 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 & .NET
    - J2SE is backwards-compatible to JDK 1.1...**
    - ... 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: java.net.\***
    - .NET: System.Net**
  - HTTP: basic exchange of information (MIME)
    - Java: java.net.\* (URLConnection)**
    - .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: JMS
    - **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**
  - 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**
    - Java: J2SE, 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: JAXWS, fragmented vendor support
    - JAXWS 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/Managed C++
  - both the JVM and CLR are fundamentally "just" DLLs
    - Java: JNI talks natively (C/C++) to the OS**
    - CLR: supports C++/CLI as a bridge**
    - use MC++ to write JNI DLLs/JNI Invocation code**
  - Warning: lots of tricky issues to be aware of
    - data transcription from one type system to another**
    - awkwardness of working with JNI model (JACE!)**
    - thread affinity, synchronization scoped to JVM/CLR**

# 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, but in the CLR**
  - Principally useful when the code is important, not data
- In-process: JaCIL
  - Open-source project that seeks to do as IKVM 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

# Questions

---

