

# WEB SERVICES

Real Estate Concepts Workshop

# Objectives

2

- See why web services are being used to build large scale distributed applications
- Discuss the components of web services and their purposes
- Map these components to RETS 2

# Intended Audience

3

- Software application developers
- Web developers
- Project managers
- Team leads

# Outline

4

- Web services today
- Utilizing XML tools
- Building extensible schemas
- Analyzing WSDL
- Quickly creating a web service consumer

# Web Services

What are they?

Why are they important?

What are the component parts?

Who is using them?

How are they being used?

# Web Services: What Are They?

6

- Provider/Requestor model
- Utilizes core W3C standards
  - XML
  - XML Schema
  - SOAP
  - WSDL
- Essentially: XML over HTTP, packaged in an envelope, with explicit type, messaging and structure requirements rigorously defined

# Web Services: Important?

7

- Deliver in key areas
  - Interoperability
  - Ease of development
  - Vendor tool support
  - Application of standards

# Web Services: Components

8

- XML
  - ▣ Tag based
  - ▣ Self-describing
  - ▣ Freely available parsers
  - ▣ Platform independent
  - ▣ Vendor neutral
  - ▣ Loads of tools (also freely available)

# Web Services: Components

9

- XML Schema
  - A W3C specification
  - Support on all platforms
  - Is an XML document
  - Provides structure, name and type constraints
  - Can bind to classes (tools such as JAXB, castor)
  - Tool of choice in web services

# Web Services: Components

10

## □ XML Schema example

```
<xs:complexType name="CounterOffer">
  <xs:sequence>
    <xs:element name="OfferPrice" type="offer:OfferPrice"/>
    <xs:element name="DateReceived" type="commons:DateReceived"/>
    <xs:element name="ExpirationDate" type="commons:ExpirationDate"/>
    <xs:element name="DateResponded"
type="commons:DateResponded"/>
    <xs:element name="OfferDetails" type="offer:OfferDetails"/>
    <xs:element name="CounterOffer" type="offer:CounterCounterOffer"
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="offer:offerStatus" use="required"/>
</xs:complexType>
```

# Web Services: Components

11

- SOAP
  - ▣ Messaging framework
  - ▣ Documents are XML
  - ▣ Outlines enveloping scheme and fault communication
  - ▣ Designed to cover many scenarios
    - Request/Response
    - Intermediaries
    - One-way

# Web Services: Components

12

## □ SOAP Example

- ▣ Utilizing version 1.2
- ▣ Message packaging scheme

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
      <n:priority>1</n:priority>
      <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

# Web Services: Components

13

- WSDL: Web Service Description Language
  - A W3C specification
  - Is an XML document
  - Describes the interface (similar to IDL in Corba or DCOM)
  - Defines types (via XSD), messages, ports, operations, bindings, and services
  - Using version 1.1

# Web Services: Components

14

- Additional components
  - ▣ WS-I: Web Services Interoperability
    - Provides guidelines for achieving interoperability
    - Offer implementation guidance and best practices
    - Provides tools and sample applications
  - ▣ WSS: Web Services Security
    - Security at the message level
    - Username token, nonce, created passed within SOAP header
    - May use SAML or X.509

# Web Services: Components

15

- Additional components
  - ▣ MTOM: Message Transmission Optimization Method
    - Improves performance, selectively encoding message parts
    - Particularly well suited for sending files
    - Response may be parsed as received, rather than waiting for the entire message to arrive
  - ▣ WSE: Web Service Enhancements
    - Provides implementations for WS-Security, WS-Addressing, WS-Policy

# Web Services: Public APIs

16

- Some companies providing public web service APIs
  - Google Search
  - Amazon
  - Yahoo! Search, Yahoo! Maps
  - eBay
  - Geocoder
  - FedEx
  - Etc.

# Web Services: Public APIs

17

- Typical features
  - ▣ Registration required for use
  - ▣ Multiple ways to use them
    - SOAP
    - REST (Representational State Transfer, traditional HTTP)
    - POX (Plain Old XML)
  - ▣ Provide relatively simple, discrete services
  - ▣ Online support and documentation

# Web Services: Public APIs

18

- Forms of documentation
  - Online API (similar to Javadoc)
  - Blogs
  - Wiki
  - Newsgroups

# Web Services: Usage

19

- Utilize public services
  - ▣ Search
  - ▣ Mapping
  - ▣ Package tracking
- Provide building blocks for creating applications
  - ▣ Tight integration with online auction
  - ▣ Facilitate sales by providing access to search inventory

# Web Services: Future

20

- Hot, hot hot
  - ▣ Activity around combining simple services
  - ▣ Rapid, agile development
  - ▣ Highly interactive sites
  - ▣ B2C
- Longevity
  - ▣ Robust functionality
  - ▣ B2B quality transactions

# Component Technologies

How can XML help build robust, distributed services?

# Component Technologies

22

- Recall the web service components
  - XML
  - XML Schema
  - SOAP
  - WSDL
  - WS-I
  - WSE
  - MTOM

# Component: XML

23

- Implementers are free to use tools, toolkits, frameworks, etc. to facilitate data handling

# Component: XML

24

- Some ways to retrieve data from XML
  - ▣ Freely available parsers – Java, .NET, Perl, C++, etc.
  - ▣ XSLT, Extensible Style Sheet Language for Transformations also freely available
  - ▣ Language environments also provide specialty tools
    - Language object/XML bindings
    - Optimized reader/writer classes

# Component: XML

25

- In Java:
  - Sun provides a JAX
    - JAXP: Java API for XML Processing (included in J2EE)
    - JAXB: Java Architecture for XML Binding (serialization)
    - JAX-WS: Java API for XML based Web Services
    - JAX-RPC: Java API for XML based RPC (SOAP 1.1)
    - JAXM: Java API for XML Messaging (document-oriented)
    - JAXR: Java API for XML Registries (discovery)

# Component: XML

26

- In Java: (cont.)
  - ▣ JWSDP: Java Web Services Developer Pack replaced with GlassFish (J2EE Web Services Implementation)

# Component: XML

27

## □ JAXP

- Core for working with XML
- Included in J2EE, J2SE (5.0)
- Features
  - Validation against XSD
  - XPath for node location and searching
  - DOM
  - SAX
  - XSLT

# Component: XML

28

- Thanks to JAXP, parsers are pluggable
  - ▣ Factory API used to create instances of
    - SAX Parser
    - DOM Document Builder
    - XSLT Transformer
  - ▣ Various parser implementations
    - Crimson (Sun)
    - XML4J (IBM)
    - Xerces (Open Source, off XML4J)

# Component: XML

29

- JAXB
  - ▣ Binding between XSD and Java class
  - ▣ Allows developer to work with objects, rather than XML directly
- Plenty of other tools to do the same thing

# Component: XML

30

- XML in .NET: System.Xml
  - Streaming classes
    - XmlReader
    - XmlWriter
  - Document class
    - XmlDocument W3C DOM
  - Optimized for transformations
    - XPathDocument

# Component: XML

31

- **Serialization in .NET**
  - ▣ Can be completely hands off
  - ▣ Can specify attribute, element mappings
  - ▣ Has interfaces to facilitate SOAP encoding/decoding
- **XSLT in .NET 2.0**
  - ▣ Performance improved significantly over .NET 1.1

# Component: XML

32

- Microsoft MSXML component
  - ▣ Active X
  - ▣ Distributed with Internet Explorer, Windows, and various other Microsoft products
  - ▣ First browser implementation of XML
  - ▣ Includes support for standards
    - DOM
    - SAX
    - XSLT
    - Validation

# Component: XML

33

- XML is a string
  - ▣ Therefore any programming language that can handle a string, can handle it
  - ▣ Many other languages/environments support XML natively or through libraries
    - PHP
    - Perl
    - Ruby
    - Fortran
    - C++

# XML In RETS 2

34

- XML in RETS2
  - ▣ Implementers are free to use tools, toolkits, frameworks, etc. to facilitate data handling
  - ▣ Some ways to retrieve data from XML
    - Freely available parsers – Java, .NET, Perl, C++, etc.
    - XSLT, Extensible Style Sheet Language for Transformations also freely available
    - Language environments also provide specialty tools
      - Language object/XML bindings
      - Optimized reader/writer classes

# Why XML?

35

- Clearly easily and consistently handled
  - ▣ Tools available for a variety of programming languages, platforms
  - ▣ Any operating system
- Simple rules for “good” XML
  - ▣ Well-formed
    - Small set of rules
    - Non-negotiable
  - ▣ Validity

# Well-Formed XML

36

- Very few rules
  - ▣ One root element
  - ▣ Tags must be properly nested, case sensitive
  - ▣ Elements must have start and end tag
  - ▣ Attributes appear in start tag, value must be quoted
  - ▣ Attribute names must be unique per element
  - ▣ Element and attribute names must start with a letter or underscore

# Valid XML

37

- Validity is an extra level of constraint
  - ▣ Formerly specified using DTD (Document Type Definition)
    - Written in BNF
    - No data types
    - No inheritance
  - ▣ Replaced with XML Schema

# Component Technologies

How can XML Schema be used to provide a common core, while allowing customizations?

# Component: XML Schema

39

- XML Schema
  - ▣ Supported by most XML parsers
  - ▣ Constrains structure, type and name
  - ▣ Conforming documents are considered valid
  - ▣ Inheritance
    - Restriction
    - Extension
  - ▣ Reusable groups
    - Model groups
    - Attribute groups

# Component: XML Schema

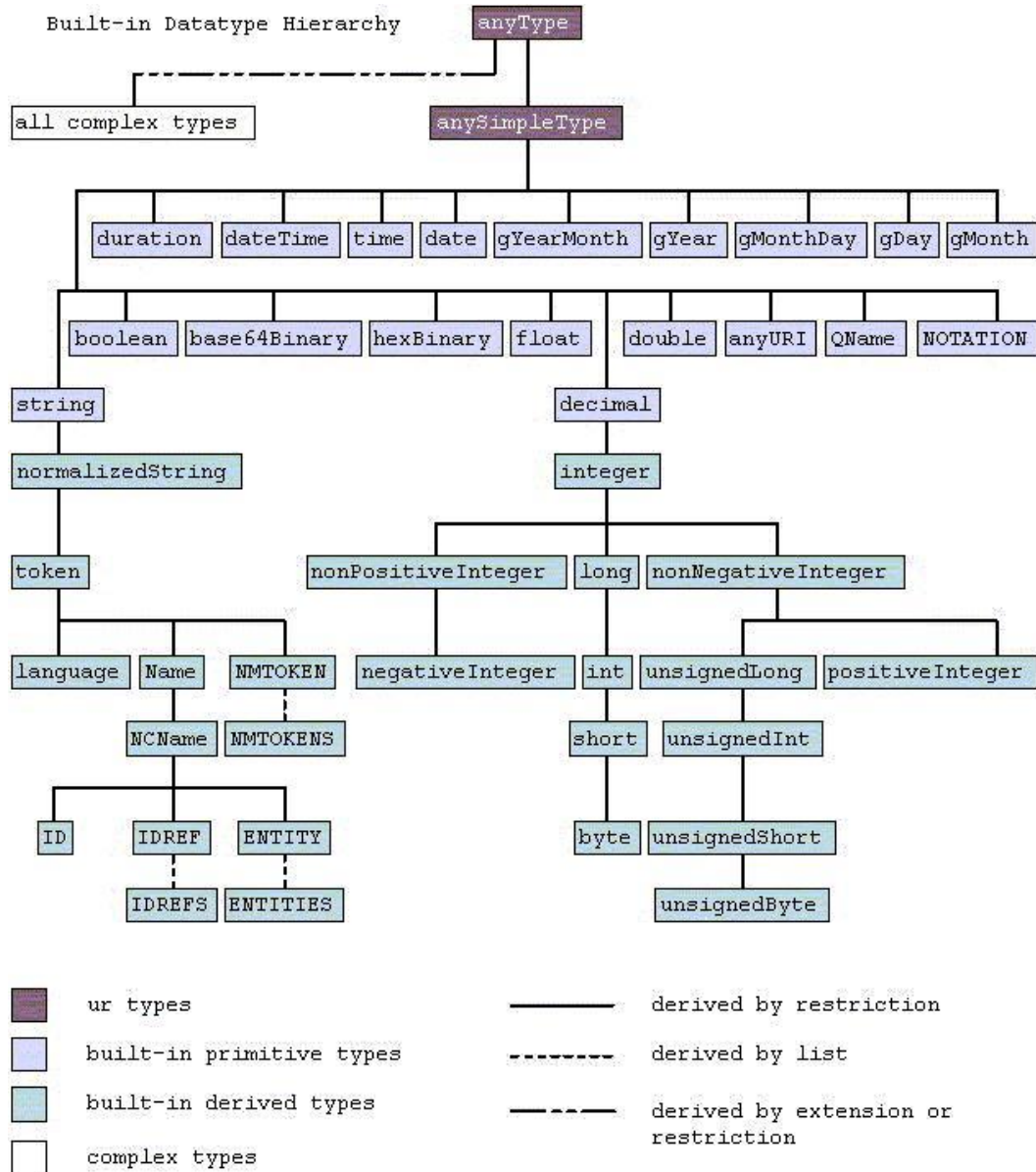
40

- Built ins
  - Primitives
  - Derived
- Type definitions
  - Simple
  - Complex
- Attributes
- Elements
- Groups

# Component: XML Schema

41

- Special concerns
  - ▣ Design patterns
    - Russian doll
    - Venetian blind
    - Salami slice
  - ▣ Extensibility
  - ▣ Versioning



# XML Schema Visibility

43

- Elements, attributes, and types may be defined
  - ▣ Top-level (global)
    - Must have name
    - Reusable via ref (element, attribute) or type (simple, complex)
  - ▣ Local
    - Appear inside a given element, attribute
    - No name, anonymous
    - Not reusable

# XML Schema: Simple Types

44

```
<xs:simpleType name="securityClass">  
  <xs:restriction base="xs:string">  
    <xs:enumeration value="Public" />  
    <xs:enumeration value="Confidential" />  
    <xs:enumeration value="Secret" />  
  </xs:restriction>  
</xs:simpleType>  
  
<xs:attribute name="isgSecurityClass" type="securityClass">
```

Top-level, named, reusable type

Enumeration of string values

Attribute defined using securityClass

Example instance of the isgSecurityClass attribute with valid value

```
<elmt isgSecurityClass="Public"/>
```

# XML Schema: Simple Types

45

Top-level  
attribute,  
reusable, named

```
<xs:attribute name="documentType">  
  <xs:simpleType>  
    <xs:restriction base="xs:token">  
      <xs:enumeration value="PropertyDescription" />  
      <xs:enumeration value="ClosingStatement" />  
      <xs:enumeration value="MortgageLoanApplication" />  
      <xs:enumeration value="InsuranceApplication" />  
      <xs:enumeration value="TitleInsuranceRequest" />  
      <xs:enumeration value="PromissoryNote" />  
    </xs:restriction>  
  </xs:simpleType>  
</xs:attribute>
```

Referencing  
defined, named  
attribute

```
<xs:attribute ref="doc:documentType" use="optional" />
```

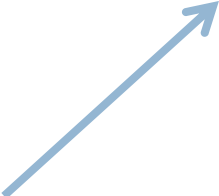
It is optional

# XML Schema: Elements

46

```
<xsd:element name="ResourceName" type="xsd:NMTOKEN" />
```

Element, type of built in NMTOKEN



Element instance, value is  
NMTOKEN type

```
<ResourceName>Image</ResourceName>
```



# XML Schema: Complex Types

47

- May include many things
  - ▣ Sequence, choice, all
  - ▣ Attributes
  - ▣ Complex content
  - ▣ Simple content

# XML Schema: Complex Types

48

```
<xs:complexType name="Category">  
  <xs:simpleContent>  
    <xs:extension base="xs:string">  
      <xs:attribute ref="commons:isgSecurityClass" use="required" />  
    </xs:extension>  
  </xs:simpleContent>  
</xs:complexType>
```

Simple content may extend a primitive, built in, or simple type

Simple content may include attributes

```
<xs:element name="Category" type="act:Category" minOccurs="0" />
```

Elements are generally complex types.  
Attributes cannot be complex types.

# XML Schema: Complex Types

49

```
<xs:complexType name="Agents" mixed="true">  
  <xs:sequence>  
    <xs:element name="ListingAgent" type="agency:Agent" minOccurs="0" />  
    <xs:element name="CoListingAgent" type="agency:Agent" minOccurs="0" />  
    <xs:element name="SellingAgent" type="agency:Agent" minOccurs="0" />  
    <xs:element name="CoSellingAgent" type="agency:Agent" minOccurs="0" />  
    <xs:element name="EntryAgent" type="agency:Agent" minOccurs="0" />  
    <xs:element name="OtherAgent" type="agency:Agent" minOccurs="0" />  
  </xs:sequence>  
</xs:complexType>
```

Sequence means child elements must appear in order

minOccurs indicates the minimum number of times the element may appear  
0 would mean the element is optional

# XML Schema: Complex Types

50

```
<xs:element name="Prospects">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="Prospect" type="prospect:Prospect" maxOccurs="unbounded" />  
    </xs:sequence>  
    <xs:attribute name="versionTimestamp" type="xs:dateTime" use="required"  
      fixed="2006-04-07T00:00:00Z" />  
  </xs:complexType>  
</xs:element>
```

maxOccurs unbounded means there is no upper bound to how many times it may appear

Locally defined complex types are not reusable

Fixed value must always be what is provided  
If absent, parser supplies given value

Required means it must appear on every element instance in the instance document

# XML Schemas: Special Concerns

51

- Design patterns
  - ▣ Russian doll
    - Single root element definition
    - All types anonymous and local
  - ▣ Salami slice
    - Many top level elements defined
    - All types anonymous and local
  - ▣ Venetian blind
    - Few top level elements defined
    - All types top level and reusable

# XML Schema in RETS 2

52

- Design pattern for RETS 2 is Venetian Blind

```

<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:message="http://rets.org/ns/MessageOfDay/200604"
  targetNamespace="http://rets.org/ns/MessageOfDay/200604"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:simpleType name="Details">
    <xs:restriction base="xs:string" />
  </xs:simpleType>
  <xs:simpleType name="Timestamp">
    <xs:restriction base="xs:dateTime" />
  </xs:simpleType>
  <xs:element name="Messages">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Message" type="message:MessageType" minOccurs="0"
maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="MessageType">
    <xs:sequence>
      <xs:element name="Timestamp" type="message:Timestamp" />
      <xs:element name="Details" type="message:Details" />
      <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="versionTimestamp" type="xs:dateTime" use="required" fixed="2006-04-
07T00:00:00Z" />
  </xs:complexType>
</xs:schema>

```

# XML Schemas: Special Concerns

54

- Extensibility
  - ▣ Type derivation and substitution
    - Elements added at the end
    - Core schema requires no change or special indicator
  - ▣ Use of the <any> element
    - May appear anywhere within the content model
    - Can also specify how many additional elements appear
    - Indicates namespace for additions

# XML Schema in RETS 2

55

- Many schemas specified for RETS 2
  - [http://www.ftc2.com/rets/ws\\_files/index.html](http://www.ftc2.com/rets/ws_files/index.html)
  - Extensibility points using <any>

```
<xs:complexType name="MessageType">  
  <xs:sequence>  
    <xs:element name="Timestamp" type="message:Timestamp" />  
    <xs:element name="Details" type="message:Details" />  
    <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded" />  
  </xs:sequence>  
  <xs:attribute name="versionTimestamp" type="xs:dateTime" use="required"  
    fixed="2006-04-07T00:00:00Z" />  
</xs:complexType>
```

# XML Schema: Special Concerns

56

- Versioning
  - Facilitate schema evolution
  - Provide attributes for identification
  - Versioning in namespaces

# XML Schema in RETS 2

57

- Versioning taken into consideration
  - ▣ Namespace includes versioning information
  - ▣ Version timestamp

```
<xsd:schema  
  targetNamespace="http://www.rets.org/ns/Listing/200604"  
  xmlns="http://www.rets.org/ns/Listing/200604"  
  xmlns:xs="http://www.w3.org/2001/XMLSchema"  
  versionTimestamp="2006-137 04-07T00:00:00Z">
```

# XML Schema in RETS 2

58

- RETS Payloads documents provides more information for schema authors
  - ▣ Guidelines for element and type naming
  - ▣ Guidelines for schema authors
    - Derivation by extension recommended over restriction
    - Any element must use the ##other namespace stipulation

# Demonstration

59

- Namespaces in 20 seconds or less
- How to validate instance document using a schema
- How to extend a RETS 2 schema
- Avoiding non-determinism in extensions
- Validating an instance document against an extended schema

# Component Technologies

What does SOAP look like?

What does WSDL do for a web service?

# Component: SOAP

61

- SOAP is not complex
  - ▣ Enveloping
    - Header
    - Body
  - ▣ Body content schema defined by web service designer, not by SOAP itself
  - ▣ Package for content going across the wire

# Components: SOAP

62

## □ SOAP Envelope

- Envelope serves as container for whole message
- Carries namespace URIs
- Contains two possible child elements
  - Optional Header
    - Used for processing directives, including security parameters
    - Carries standard attributes: mustUnderstand, relay, role, encodingStyle
  - Required Body
    - Holds the content
    - Content defined in custom schema
    - In error, carries the Fault

# Component: SOAP

63

- SOAP handling generally transparent
  - ▣ Developer tools
    - Web service endpoints emit and consume it
    - Object model for creating/consuming
    - Content/object mapping for serializing/deserializing
- Styles
  - ▣ Document
  - ▣ RPC

# SOAP in RETS 2

64

- SOAP in RETS 2
  - Utilizing SOAP 1.2 (Recommendation in 2003)
  - Document style
  - MTOM

# Demonstration

65

- Building a simple web service
- Deconstructing the message

# Component: WSDL

66

- Web Services Description Language
  - Types provided by schema
    - Internal
    - External
  - Ports
  - Bindings
  - Service

# Component: WSDL

67

## □ Namespaces

```
<wsdl:definitions name="rets"  
  targetNamespace="urn:retsWsdI"  
  xmlns:rets="urn:retsWsdI"  
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"  
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"  
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
  xmlns:rl="http://rets.org/ns/ResourceList/"  
  xmlns:ll="http://rets.org/ns/LookupList/">
```

# WSDL in RETS 2

68

- Types
  - ResourceList
  - LookupList
  - SearchRequestType
  - ManifestType
  - SeverityType
  - RETSFaultList

# WSDL in RETS 2

69

- Elements
  - SearchRequest + SearchResponse
  - UpdateRequest + UpdateResponse
  - ResourceListRequest + ResourceListResponse
  - LookupListRequest + LookupListResponse
  - MetadataRequest + MetadataResponse
- Message for each element

# WSDL in RETS 2

70

- Port
  - Operation for each pair
  - Input and output for each message pair
- Binding
  - SOAP 1.2 binding for document-literal
- Service
  - Single RETS service

# Demonstration

71

- Reading WSDL
- Consuming WSDL
- Document/Literal vs. RPC discussion

# Component: WS-I

72

- Interoperability
  - ▣ Cornerstone of web services mantra
  - ▣ Hard to achieve
- WS-I Organization
  - ▣ Creates, promotes, and supports protocols for interoperable message exchange
  - ▣ Provides basic profile enumerating

# Looking Ahead

73

- Looking ahead in .NET
  - ▣ Continued commitment to web services with WCF: Windows Communication Foundation
  - ▣ Nutshell
    - Makes WS calls even more transparent to the developer
    - Stresses commitment to W3C standards, including the WS\* variety
    - Integrates WSE with .NET Framework in 3.0
    - Standard with Vista

# Looking Ahead

74

- Looking ahead in Java
  - ▣ Continued commitment to web services
  - ▣ Nutshell
    - JAX WS 2.0

# Resources

75

- List of web APIs  
<http://www.webapi.org/webapi-directory/>
- List of SOAP implementations  
[http://www.soaprpc.com/ws\\_implementations.html](http://www.soaprpc.com/ws_implementations.html)
- Simple Java to .NET MTOM example  
<http://simon.guest.members.winisp.net/webcasts/mtomblogcast.wmv>
- Perl XML <http://perl-xml.sourceforge.net>

# Resources

76

- Document literal / RPC literal discussion  
[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/rpc\\_literal.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/rpc_literal.asp)
- Document based web service article, Java JAX WS 2.0  
<http://java.sun.com/developer/technicalArticles/xml/jaxrpcpatterns/index.html>
- Extensible content models  
<http://www.xfront.com/ExtensibleContentModels.pdf>

# Resources

77

- JAXWS MTOM Blog:  
[http://weblogs.java.net/blog/vivekp/archive/2006/05/more\\_on\\_jaxws\\_a.html](http://weblogs.java.net/blog/vivekp/archive/2006/05/more_on_jaxws_a.html)
- Article on MTOM and XOP:  
[http://www.webservices.org/weblog/colin\\_adam/becoming\\_attached\\_to\\_soap](http://www.webservices.org/weblog/colin_adam/becoming_attached_to_soap)