XML and Web Services in VisualAge Smalltalk

In this lecture, we will learn some of the basics of XML and look at the capabilities of the VisualAge for Smalltalk XML feature.

We will then discuss the exciting new XML-based technology called Web Services which holds much promise for the future of computing.

What is XML?

XML (eXtensible Markup Language) is a markup language for describing data. It is simpler than predecessor SGML (Standard Generalized Markup Language) and more versatile than HTML (Hypertext Markup Language).

XML is an open standard for defining and sharing data across diverse network topologies. XML documents are self-describing and XML documents can be used to describe other XML documents.

XML data representation is human-readable, application-neutral, and language-neutral enabling universal interchange of data.

XML terminology

World Wide Web Consortium (W3C): A standards body that develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential as a forum for information, commerce, communication, and collective understanding.

Document Object Model (DOM): A W3C standard which describes mechanisms for software developers and Web script authors to access and manipulate parsed XML (and HTML) content. The DOM is both platform-neutral and language-neutral.

Document Type Definition (DTD): A specification of the elements and attributes that are permitted in an XML document.

Outline for Lecture 21

I. What is XML?
II. Why use XML?
III. VisualAge XML support in V5.5
   - Parser
     - DOM level-2
     - SAX-2
     - Mapping parser (new)
   - Serialization
IV. Web Services
   - What are they?
   - Who will use them?
   - What is SOAP?
   - What is WSDL?
   - What is UDDI?
Well-formed XML document: An XML document that conforms to basic XML rules

Valid XML document: A well-formed XML document that conforms to the rules specified in a DTD

Simple API for XML (SAX): A standard interface for event-based XML parsing


Mapping specifications: Instances of the VisualAge class AbtXmlMappingSpec which contain rules for mapping XML elements and attributes into Smalltalk objects

Interface specifications: Instances of the VisualAge class AbtInterfaceSpec which describe the attributes, actions, and events supported by an object

DTD
<![-- Describes the structure of the objects required for the VA Stock trading demo -->]
<!ELEMENT Portfolios (Portfolio*)>
<!ELEMENT Portfolio ( portfoliolid, availableCash, Holdings)>
<!ELEMENT Holdings ( Holding)*>
<!ELEMENT Holding ( symbol, TransactionRecord* )>
<!ELEMENT TransactionRecords ( TransactionRecord* )>
<!ELEMENT TransactionRecord ( numShares, price, timestamp, id )>

<!ELEMENT symbol (#PCDATA)>
<!ELEMENT numShares (#PCDATA)>
<!ELEMENT purchaseCost (#PCDATA)>
<!ELEMENT purchaseTimestamp (#PCDATA)>
<?xml version="1.0"?>
<!DOCTYPE Portfolios SYSTEM "vaportfolio.dtd">
<Portfolios>
  <Portfolio>
    <portfolioId>000001</portfolioId>
    <availableCash>10000</availableCash>
    <Holdings>
      <Holding>
        <symbol>YHOO</symbol>
        <TransactionRecord>
          <numShares>100</numShares>
          <price>100.01</price>
          <timestamp>2000-07-13-17.24.25.000000</timestamp>
        </TransactionRecord>
      </Holding>
      <Holding>
        <symbol>IBM</symbol>
        <TransactionRecord>
          <numShares>1000</numShares>
          <price>120.01</price>
          <timestamp>2000-07-13-17.24.25.000000</timestamp>
        </TransactionRecord>
      </Holding>
    </Holdings>
  </Portfolio>
</Portfolios>

XML schema: A specification of the elements and attributes that are permitted in an XML document along with the datatypes for these artifacts

<xsd:schema
 xmlns:tns="http://www.vast.com/VAStock"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.vast.com/VAStock">
 <xsd:complexType name="Portfolio">
   <xsd:sequence>
     <xsd:element minOccurs="1" maxOccurs="1" name="portfolioId" type="xsd:string"/>
     <xsd:element minOccurs="1" maxOccurs="1" name="availableCash" type="xsd:decimal"/>
     <xsd:element minOccurs="0" maxOccurs="unbounded" name="Holding" type="tns:Holding"/>
   </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
Parsing XML in VisualAge Smalltalk

"Validating parser used to read well-formed XML and verify that the contents conform to the DTD referenced in the XML."
| domDocument domElement |
| domDocument := AbtXmlDOMParser newValidatingParser parseURI: '..\ss2001\common\wedding1.xml'.
| domElement := domDocument getElementById: 'Planner_1'.
"Non-validating parser used to read well-formed XML data."
| domDocument domElements |
| domDocument := AbtXmlDOMParser newNonValidatingParser parseURI: '..\ss2001\common\wedding1.xml'.
| domElements := domDocument getElementsByTagName: 'Address'.

Using the SAX parser

To create a custom SAX handler in VisualAge Smalltalk, the following steps must be followed:

Create subclass of AbtXmlSaxDefaultHandler
Override SAX interfaces to customize behavior

- ContentHandler
- DTDHandler
- EntityResolver
- ErrorHandler

Set overridden interfaces in SAX parser instance prior to parsing
Creating objects from XML. Below is a sample VisualAge XML mapping specification file. The mapping specification contains rules that enable conversion of XML into user-defined objects.

```xml
<?xml version="1.0"?>
<!-- This file contains information required to transform XML representations of objects into actual Smalltalk objects. -->

<!DOCTYPE XmlMappingSpec SYSTEM "..\abtxml.dtd" >
<XmlMappingSpec Name="VaStockMappings">
  <ClassElementMapping ElementTagName="StockQuote" ClassName="VaStockQuote">
    <AttributeMapping ClassAttribute="symbol">
      <Attribute>symb</Attribute>
    </AttributeMapping>
    <AttributeMapping ClassAttribute="currentPrice" StringConversionMethod="asScaledDecimal">
      <Attribute>currentPrice</Attribute>
    </AttributeMapping>
  </ClassElementMapping>
  <ClassElementMapping ElementTagName="StockQuotes" ClassName="VaStockMappings">
    <AttributeMapping ClassAttribute="items">
      <SubElement>StockQuote</SubElement>
    </AttributeMapping>
  </ClassElementMapping>
  <ClassElementMapping ElementTagName="Portfolio" ClassName="VaPortfolio">
    <AttributeMapping ClassAttribute="portfolioId">
      <Attribute>portfolioId</Attribute>
    </AttributeMapping>
    <AttributeMapping ClassAttribute="holdings">
      <SubElement>Holding</SubElement>
    </AttributeMapping>
    <AttributeMapping ClassAttribute="availableCash" StringConversionMethod="asScaledDecimal">
      <Attribute>availableCash</Attribute>
    </AttributeMapping>
  </ClassElementMapping>
</XmlMappingSpec>
```
Creating objects from a DOM

```smalltalk
| mappingSpec dom |
mappingSpec := AbtXmlMappingSpec fromFile:
'..\ss2001\common\wedding.map'.
" Parse the wedding.xml file and map it to a Smalltalk object "
dom := AbtXmlDOMParser newValidatingParser parseURI:
'..\ss2001\common\wedding1.xml'.
dom mapUsing: mappingSpec
```

Web services

e-business drivers...

- 1-to-1 $\Rightarrow$ Many-to-Many Collaborations
- Packaged Software Applications $\Rightarrow$
  - Self-contained, Interoperable, modular components
- Rigid, point-to-point App Integration $\Rightarrow$
  - JIT Sourcing and “Software Assemblies”
- Software as a product $\Rightarrow$
  - Software as service (utility)
- ASP + e-Marketplace $\Rightarrow$ P2P Networks

What is a Web service?

Web services are self-contained, modular application that can be:

- Described
- Published
- Found
- Bound
- Invoked
- Composed

What is a Web service for?

- A web service is about integration
  - Application integration
  - Independent of platform, programming languages, etc…
• Allows “just in time” integration of
  – Business process
  – Dynamic e-business

Who will use Web services?
• Businesses will start with integrating internal applications
  E.g. inter-divisional systems efficiency
• Over time (1-3 years), evolution towards supply-chain integration
  As more standards are established
• Expose to the transactional web key business processes that your partners/suppliers/customers will want to interact with.

Service-oriented architecture
• Web Service Definition Language: An XML based interface definition language for network based services
• Universal Description Discovery & Integration: A standards based architecture specification for service description and discovery. (www.uddi.org)
• Simple Object Access Protocol: A lightweight XML based protocol for the exchange of information in a decentralized, distributed environment.

What is SOAP?
• Simple Object Access Protocol
  – Supported by Microsoft, IBM, and others
  – New acronym? Services-Oriented Architecture Protocol
• XML based messaging protocol
  Specification submitted to W3C
• Completely vendor-neutral
• Works with any programming language, object model, operating system, or platform
• Goal is to provide a standard object invocation protocol
What is SOAP?

- Simple enveloping mechanism independent of transport layer
- Payload = body + headers
- Body = \{RPC | Document\}*
- Headers = other things (security, authorization, payment, etc.)

SOAP message structure

- Request and Response messages
  - Request invokes a method on a remote object
  - Response returns result of running the method
- SOAP defines an "envelope"
  - "envelope" wraps the message itself
  - message is a different vocabulary
  - namespace prefix is used to distinguish the two parts

Alternatives to SOAP?

- CORBA, RMI, DCOM, etc…
  - Makes too many assumptions about requestor and provider
- XML RPC (SOAP predecessor)
- W3C XP (SOAP successor)

Why is SOAP so great?

- Exploit Internet protocols and standards
  - XML
  - HTTP
- Flexible layering
  - Transport bindings (Http, MQ, SMTP)
- Language bindings (Smalltalk!)
- Data encodings
- Simple (for the most part)
- Services on the web
  - Model
  - Wire format
- Lots and lots of activity from lots and lots of vendors

**SOAP and standardization**
- Submitted to W3C for consideration as a standard
  - W3C has formed a working group called "XML Protocol".
  - The group is considering what SOAP offers as input to the process
  - The XML Protocol specification may or may not be SOAP at its heart, yet the problems SOAP solves are the important requirements of XML Protocol.
- Industry-specific SOAP messages will start a new round of vocabulary standards work

OK, so I can send a SOAP message, but how do I know what message to send?

**Answer:** The service description
Web Services Definition Language (WSDL)

- WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.
- Operations and messages described abstractly, then bound to a concrete network protocol and message format to define an endpoint.
- Related concrete endpoints are combined into abstract endpoints (services).

**What does WSDL do?**

- Serves as an XML Vocabulary, similar in purpose to IDL.
- Provides operational information about the service:
  - *service interface*
  - *implementation details*
  - *access protocol*
  - *contact endpoints*
What does WSDL describe?

- Types
- Messages
- Port types (groups of operations)
- Bindings (port type associated with protocol)
- Ports (associations of bindings with network address)
- Services (groups of ports)

Let’s look at some WSDL now …

Gourmet 2 GO

Smalltalk and WSDL

- WSDL addresses the same issue that we face with CORBA and RMI: Strong Typing
- Tells us the shape of the incoming and outgoing messages
- Separates the interface from implementation (a principal very familiar to Smalltalkers!)
- Multiple implementations of the same (agreed upon) interface

Why is WSDL so great?

- Emerging standard
- Nice fit with SOAP
  - but not tied to SOAP
- Very flexible, well layered
- Eventually… industry standard WSDL definitions for key web service types

OK, so I can send SOAP messages, and I know how to format them because of the WSDL… But How do I know what kind of service someone provides?
Answer: Look up their description in UDDI!

What is UDDI?

- Standards-based specifications for service description and discovery
- A set of internet-based implementations
  - Interoperating to share registrations
- Partnership among industry leaders
  - Initiated by Ariba, IBM, and Microsoft
  - Now 14 working group members
  - Nearly 200 advisors

What is UDDI?

- A business registry for services
- ‘Napster’ for Web Services
- Defines the API to publish and discover services
- Uses standard business taxonomies
  - Industry: NAICS (Industry codes - US Govt.)
  - Product/Services: UN/SPSC (ECMA)

UDDI Working Group members

- Andersen Consulting
- Ariba
- CommerceOne
- Compaq
- Equifax
- Fujitsu
- Hewlett-Packard

UDDI Overview

Registry Operation

IBM’s UDDI Operator Node
Replicates daily with Ariba & Microsoft sites

IBM Public production site:
http://www.ibm.com/services/uddi

IBM Public customer test site:
http://www.ibm.com/services/uddi/testregistry

Both sites support SOAP & Web Page access

The Specification

Programmers’ API defining

Publication of information for discovery

“I use WSDL, here’s the definition for my service type”

“I’m company X, I have these services…”

General Inquiry

Get me the service type definitions that are published

“Show me the services that company X offers”

Web Service call specifics provide

Call request/response structure semantics

SOAP details

XML details

Error handling

UDDI API

Inquiry API

Find things

find_business

find_service

find_binding

find_tModel

Get details about things

get_businessDetail

get_serviceDetail

get_bindingDetail

get_tModelDetail
 Publish API (requires SSL)

 Save things
  - save_business
  - save_service
  - save_binding
  - save_tModel

 Delete things
  - delete_business
  - delete_service
  - delete_binding
  - delete_tModel

 Security …
  - get_authToken
  - discard_authToken

What’s in the registry?

White Pages
  - Information about a service provider
    - Business Name
    - Text Description
      - list of multi-language text strings
    - Contact info
      - names, phone numbers, fax numbers, web sites…
    - Known Identifiers
      - list of identifiers that a business may be known by - DUNS, Thomas, other

Yellow Pages
  - Business categories
    - 3 standard taxonomies in V1
      - Industry: NAICS (Industry codes - US Govt.)
      - Product/Services: UN/SPSC (ECMA)
      - Location: Geographical taxonomy
Implemented as name-value pairs to allow any valid
taxonomy identifier to be attached to the business white
page

Green Pages

- How to bind to a provider
  - Technical info about services, specified by a business: how to "do e-commerce" with them
    - references to specifications for Web services
    - support for pointers to various file and URL based discovery
      mechanisms if required

- Nested model
  - Business processes
  - Service descriptions
  - Binding information

- Programming/platform/implementation agnostic
  - Services can also be categorized

Four building blocks

- tModel (a.k.a. service description)
  A keyed reference to an API or namespace.

- Business entity
  Description of a business and their contacts

- Business service
  A named reference to a service

- Binding template
  A description of a service including its location and tModels it
  supports

tModel

- A tModel is a keyed descriptive entity that can be used for
  - Unique, identifiable specification types
    - WSDL
    - Rosetta.Net RNIF 1.1 specification
    - cXML punchout 1.1
Namespace qualification for taxonomies and identifiers
   - UNSPSC – part code specification
   - DUNS – business identification number

Business entity
   - A businessEntity is the description of a business, its contacts, and its identifiers
     - Contact Information
       - Phones, addresses, emails, etc.
     - Classifications
       - Built-in/checked: UNSPSC (product), Geo Taxonomy (location), NAICS (industry)
       - User Specified …
     - Identifiers
       - D-U-N-S®
       - Thomas Register …

Business Service
A business service is a named reference to a Web service
   - General description of the service
   - Classifications
     - UNSPSC, GeoTaxonomy, NAICS
   - Identifiers
     - D-U-N-S®
     - Thomas Register, …

Binding Template
A bindingTemplate is a description of how and where a service is invoked
   - URL for where to invoke a service
   - List of tModels associated with the service
List of access parameters that may be used by the service

Why is UDDI so great?

- Standard
- Lots of companies involved
- Fits nicely with SOAP & WSDL
- Very flexible, supports all kinds of approaches to Web services
- It looks a little complicated… ’cause it is!

UDDI Resources

- Visit http://uddi.org for:
  - Executive White Paper
  - Technical White Paper
  - UDDI Programmer’s API Specification
  - UDDI Data Structure Reference
  - UDDI XML schema
  - News Releases
  - Frequently Asked Questions
- See also: http://ibm.com/services/uddi

Let’s query UDDI now…

Putting it all together…
VisualAge Smalltalk Web Services

Gourmet 2 Go Integration
VisualAge Web Service Example

Application environment…
The Future
Open Issues
Distributed computing is still hard!
- Security
- System management
- Reliable messaging
- Quality of service
- Logging/tracing/auditing
- Performance
- Private/local UDDI registry

So, What is a Web Service?
- Web Services are self-contained, modular applications that can be:
  - Described ➔ Using WSDL
  - Published ➔ To UDDI
  - Found ➔ In UDDI
  - Invoked ➔ Using SOAP
  - Composed ➔ Orchestration

IBM’s Accomplishments
- Created UDDI with Microsoft and Ariba
- Co-author of SOAP (1.1) with Microsoft, Ariba, + 11 more
- Chair XML protocol working group on W3C
- Co-author of WSDL with Microsoft, Ariba