XML and Web Services in VisualAge Smalltalk

Bryan Hogan

What is XML?
- eXtensible Markup Language
  - Markup language for describing data
  - Simpler than predecessor SGML (Standard Generalized Markup Language)
  - More versatile than HTML (HyperText Markup Language)
- Self-describing: XML documents can describe other XML documents (i.e. XML schema)

An open standard for defining and sharing data across diverse network topologies

Mark Weitzel (VisualAge Smalltalk)

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

XML protocols will soon replace languages and APIs in significance. (Microsoft keynote at XML DevCon 2000)
### Terminology

- **World Wide Web Consortium (W3C)**
  - develop 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. ([www.w3.org](http://www.w3.org))

- **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.

- **XML Schema**
  - a specification of the elements and attributes that are permitted in an XML document along with the data types for these artifacts.

### Terminology (continued)

- **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.

- **Simple Object Access Protocol (SOAP)**
  - A lightweight XML based protocol for exchange of information in a decentralized, distributed environment.
Terminology (continued)

- Extensible Stylesheet Language Transformations (XSLT)
  - A language for transforming XML documents via the application of stylesheets

- Mapping specifications
  - Instances of the VisualAge class AbtXmMappingSpec 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

Example document type definition

```xml
<!-- Describes the structure of the objects required for the VA Stock trading demo -->
<!ELEMENT Portfolios (Portfolio*)>
<!ELEMENT Portfolio (portfolioId , 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 schema example

```xml
<xs:schema
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    targetNamespace="http://www.vast.com/VAStock">
  <xs:complexType name="Portfolio">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="portfolioId" type="xs:string"/>
      <xs:element minOccurs="1" maxOccurs="1" name="availableCash" type="xs:decimal"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="Holding" type="tns:Holding"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TransactionRecord">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="numShares" type="xs:int"/>
      <xs:element minOccurs="1" maxOccurs="1" name="price" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```
Example XML document

```
<?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>
    </Holdings>
  </Portfolio>
</Portfolios>
```

VisualAge XML support: Parser

- XML 1.0 specification
  - http://www.w3.org/TR/1998/REC-xml-19980210
- DOM level-2 core interfaces
  - http://www.w3.org/TR/DOM-Level-2-Core/
- SAX 2.0

Example: Using the DOM parser

```
* Validating parser used to read well-formed XML and verify that the
  contents conform to the DTD referenced in the XML. *
  | domDocument domElement |
  domDocument := AbXxxDOMParser newValidatingParser parseURI:
  ../../../ss2001/common/wedding1.xml.
  domElement := domDocument getElementById: 'Planner_1'.

* Non-validating parser used to read well-formed XML data. *
  | domDocument domElements |
  domDocument := AbXxxDOMParser newNonValidatingParser
  parseURI: ../../../ss2001/common/wedding1.xml.
  domElements := domDocument getElementsByTagName: 'Address'.
```
Converting XML to objects

- Create SAX handler to automatically create desired object from parsed XML
- Use VisualAge XML mapping support

VisualAge XML support: Serialization

- Create instances of Smalltalk objects from parsed XML documents
- Create XML documents from Smalltalk objects
- Mapping layer to reconcile XML named elements to Smalltalk classes
  - Leverages the defined public interface for existing parts
  - Mapping layer and interface specification can be defined in XML

Creating custom SAX handlers

- Create subclass of AbtXmlSaxDefaultHandler
- Override SAX interfaces to customize behavior
  - ContentHandler
  - DTDHandler
  - EntityResolver
  - ErrorHandler
  - VAST (contains protocol required by the parser but not present in SAX 2 specification)
- Set overridden interfaces in SAX parser instance prior to parsing

**NOTE:** SAX interfaces are identifiable in the Smalltalk image using method categories. Each supported SAX interface is categorized with the naming convention: `AbtSax<interface name>`.
Converting XML to objects

- Write code to extract contents from DOM and create objects (see AbtClassElementMapping
  >fromDOMElement:<int>)
- Create DOM “wrapper” classes (see AbtDOMElementWrapper)
- Create SAX handler to automatically create desired object from parsed XML
- Use VisualAge XML mapping support

XML mapping specifications

- AbtXmlMappingSpec - a collection of class/element mappings that define rules for converting to/from XML
- AbtClassElementMapping - rules that describe how an XML element and its child elements and attributes can be mapped to/from a Smalltalk object instance
- AbtAttributeMapping - rules to determine how a simple XML element/attribute is mapped to/from a Smalltalk attribute

Class/element Mapping

- A rule for mapping an XML element to a Smalltalk class

```xml
<ClassElementMapping ElementTagName="Address"
  ClassName="STSolutionsAddress" >
  <!-- … Attribute mappings go here … -->
</ClassElementMapping>

<Address>
  <Street>29 Oak St.</Street>
  <City>Raleigh</City>
  <State>NC</State>
  <Zip>99999</Zip>
</Address>
```
Attribute Mapping

- A rule for mapping an XML attribute or text-only subelement into a Smalltalk attribute.

```xml
<AttributeMapping classAttribute="street">
  <Attribute>Street</Attribute>
</AttributeMapping>
```

```xml
<Address>
  <Street>29 Oak St.</Street>
  <City>Raleigh</City>
  <State>NC</State>
  <Zip>99999</Zip>
</Address>
```

Subelement Mapping

- A rule for mapping a complex XML subelement into an attribute of a Smalltalk object.

```xml
<ClassElementMapping ElementTagName="CeremonyLocation"
  ClassName="STSolutionsEventLocation">
  <AttributeMapping ClassAttribute="address">
    <SubElement>Address</SubElement>
  </AttributeMapping>
</ClassElementMapping>
```

```xml
<CeremonyLocation>
  <FacilityName>The Vatican</FacilityName>
  <Address>
    <!-- ... Subelements for Address go here ... -->
  </Address>
</CeremonyLocation>
```

Subelement Attribute Mapping

- A rule for mapping an attribute from an XML subelement into an attribute of a Smalltalk object.

```xml
<ClassElementMapping ElementTagName="CeremonyLocation"
  ClassName="STSolutionsSimpleLocation">
  <AttributeMapping ClassAttribute="street">
    <SubElement>Address</SubElement>
    <Attribute>Street</Attribute>
  </AttributeMapping>
</ClassElementMapping>
```
XML mapping specification example

```xml
<?xml version="1.0"?>
<!DOCTYPE XmlMappingSpec SYSTEM "..\..\xml\abtxmap.dtd">
<XmlMappingSpec Name="AddressMapping">
    <ClassElementMapping ElementTagName="Address" ClassName="STSolutionsAddress">
        <AttributeMapping ClassAttribute="street">
            <Attribute>Street</Attribute>
        </AttributeMapping>
    </ClassElementMapping>
    <ClassElementMapping ElementTagName="CeremonyLocation" ClassName="STSolutionsEventLocation">
        <AttributeMapping ClassAttribute="name">
            <Attribute>FacilityName</Attribute>
        </AttributeMapping>
        <AttributeMapping ClassAttribute="address">
            <SubElement>Address</SubElement>
        </AttributeMapping>
    </ClassElementMapping>
</XmlMappingSpec>
```

Example: Mapping DOM objects to business objects

```ruby
<table>
<thead>
<tr>
<th>mappingSpec</th>
<th>dom</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappingSpec := AbtXmlMappingSpec fromFile: '..\ss2001\common\wedding.map'.</td>
<td></td>
</tr>
<tr>
<td>&quot;Parse the wedding.xml file and map it to a Smalltalk object.&quot;</td>
<td></td>
</tr>
<tr>
<td>dom := AbtXmlDOMParser newValidatingParser parseURI: '..\ss2001\common\wedding.xml'.</td>
<td></td>
</tr>
<tr>
<td>dom mapUsing: mappingSpec</td>
<td></td>
</tr>
</tbody>
</table>

NOTE: Special packaging instructions are required for classes and methods that are mapped from XML but contain no direct references in the execution path for the application.

AbtXmlMappingParser (in VAST 6.0)

- Uses mapping specification during parsing to map directly into objects
- Uses schemas to validate XML structure and convert simple types
- Uses pluggable configurations to enable custom SAX handlers for tags that require special processing
AbtXmlMappingParser (Example)

```smalltalk
mappingSpec mappedObject |
mappingSpec := AbtXmlMappingSpec fromFile: '..\ss2001\common\wedding.map'

" Parse the wedding.xml file and map it to a Smalltalk object."
mappedObject := AbtXmlMappingParser newNonValidatingParser
mappingSpec: mappingSpec
parseURI: '..\ss2001\common\wedding1.xml'
```

Mapping objects to XML

- `#abtXmlPrintString` default behavior creates an XML string based upon the class name and instance variables. Protocol in Object can be overridden to customize default behavior.
  - `#abtXmlCacheKey` - an object used to identify XML artifacts in the XML object cache. Used to resolve defaults for items listed below.
  - `abtXmlDtd` - a DTD used to determine whether Smalltalk attributes should be rendered as XML attributes or elements
  - `abtXmlMappingSpecification` - an AbtXmlMappingSpec used to map Smalltalk names into XML names
  - `abtXmlOutputSerializer` - Specify a custom serializer to convert the object to XML

Mapping objects to XML (continued)

- Serialization behavior can be overridden by implementing the instance method `#abtXmlPrintOn:`
  - `#abtXmlPrintOn:` follows the same pattern as the `#printOn:` base method

Tip: Check out the new class AbtXmlStream to simplify creation of XML tags and attributes.
What we learned...

- VisualAge Smalltalk offers various techniques for parsing XML and manipulating XML data.
  - DOM level-2
  - SAX-2
  - Mapping support

- Enhanced parser is in the works to parse directly into objects by applying mapping rules.
**Typical client/server environment**

- Customer
- Others

**My Company**

- Application
- Application
- Application

**e-business Drivers...**

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

*Networked economy driving evolution of e-business*

**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

---

**Web Services**

How’s it work?
**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](http://www.uddi.org))

---

**What is SOAP?**

Simple Object Access Protocol
- Supported by Microsoft, IBM, and others
- XML based messaging protocol
- Specification submitted to W3C
- Can be bound to multiple transports
  - HTTP
  - MQ
  - SMTP
- Goal is to provide a standard object invocation protocol

---

**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?

XML
- XML interfaces to everything

Simple

Standard
- Why reinvent?
Lots and lots of activity from lots and lots of vendors

How does a Web Service know what message to send?

Web Services Definition Language
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 is WSDL?

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

WSDL addresses the same issue that we face with CORBA and RMI: **Strong Typing**
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

How do you know what kind of service someone provides?

Answer: the service description was found in a UDDI registry

```xml
<Envelope xmlns="http://schemas/">
  <Body>
    <find_business generic="1.0" xmlns="urn:uddi-org:api">
      <name>seller.com</name>
    </find_business>
  </Body>
</Envelope>
```
What is UDDI?

- Provides 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)

What is UDDI?

UDDI Contains lots of information…

- White pages: List of businesses
- Yellow pages: Businesses organized by
- Green pages: Businesses publish their services

How UDDI v1 Works

Universal Business Registry

- Businesses populate the registry with descriptions of the services they support
- UBR assigns a programatically unique identifier to each service and business registration

Marketplaces, search engines, and business apps query the registry to discover services at other companies
The UDDI Registries

- Peer nodes (websites)
  - Companies register with any node
  - Registrations replicated on a daily basis
  - Complete set of "registered" records available at all nodes
  - Common set of SOAP APIs supported by all nodes
  - Compliance enforced by business contract

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

Application environment...

The Future
Application environment...
The Future

My Company
- Application
- Application
- Enterprise
  Business
  Registry
- Application

Customer
Others...

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

IBM’s goal is simple: Be the leader in defining the standards that will power the next generation of e-business.

Why we (the Smalltalk Team) like Web Services

- Promotes interoperability by minimizing the requirements for shared understanding
- Language implementation neutral
- Built on industry standards
- Great integration story!
- Wrap ‘legacy’ applications
- Just in time integration: Late binding comes to the web!

Learning more: IBM developerWorks

www.ibm.com/developerworks
Web Services building blocks

- XML = data neutral
- HTTP = transport protocol neutral
- SOAP = message neutral