XML Data Binding for Java Programmers

XML One London

19th – 22nd March 2001

Presented by Andrew Fawcett

Technical Architect

CODA a Division of Science Systems Group

andrew.fawcett@coda.com

About Me! ☺

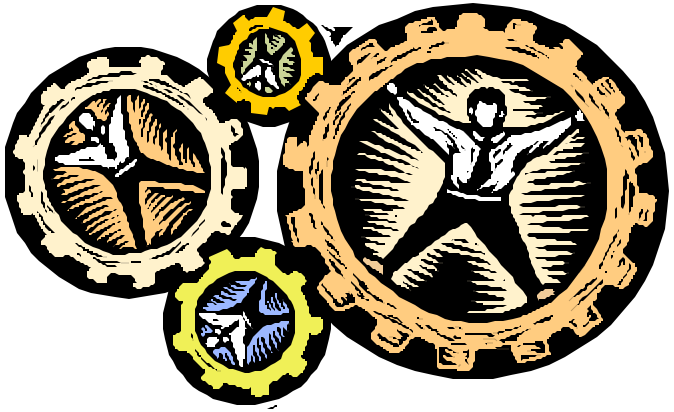
- Technical Architect for Product Development
  - Began developing on IBM AS/400 COBOL
  - Moved on to Microsoft platform
    - Visual Basic, Visual C++, Visual J++, Visual InterDev
  - Recent Projects
    - Web enabling existing applications
      - Using J2EE Java Servet API and XML based forms
    - XML enabling existing C based application server
      - Integrated with Java through XML Data Binding!
  - Currently building J2EE based application server
- Contributor to Open Source project Castor

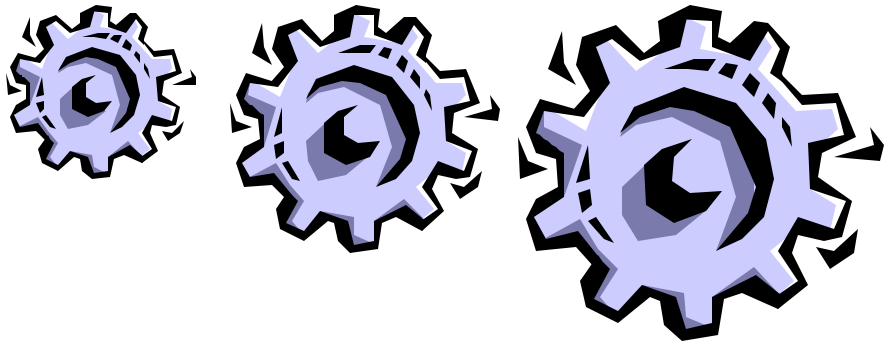
Contents

- Part 1. Introduction and Concepts
- Part 2. W3C XML Schema and Java
- Part 3. Implementations, Castor
- Part 4. Example

Sprokets.com

- Produce Widgets
  - Internet enabled
  - B2B enabled
- Processes
  - Warehouse
  - Produce Invoice
  - Send Invoice
  - Receive Invoice
  - Produce Returns
  - Process Returns

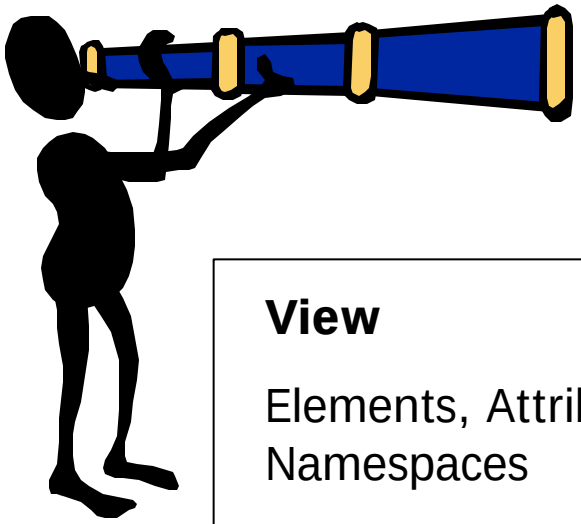




XML and Java
the story so far...

- XML Documents represent business concepts…
  - Invoice, Product, Employee…
  - Schemas can be used to define valid XML Documents
    - DTD (Document Type Definitions)
    - W3C XML Schema
- Java programmers write code to interpret and/or output the correct XML content
  - XML API's available to Java
    - SAX (Simple API for XML)
    - DOM (Document Object Model)
    - JDOM (Java Document Object Model)

XML and Java
what's your view?



View	Java API
Elements, Attributes and Namespaces	startElement(String name... Document.getRootNode()
Invoice Product Employee	com.xyz.xml.Invoice.java com.xyz.xml.Product.java, com.xyz.xml.Employee.java

XML Meets Objects

```
<Invoice ordernumber="1001" rep="Fred Blogs">
  <OrderedBy>
    <CustomerNo>CUS1200JS</
    <Date>2001-02-03T16:44:
    <Address>
      <Name>Joe Smith</Name>
      <Street>123 XYZ Street</Stree
      <City>Harrogate</City>
      <State>North Yorkshire</State
      <Zip>HG1 XYZ</Zip>
      <Country>United Kindgom</Country>
    </Address>
  </OrderedBy>
  <DeliveryAddress>
    <Name>Joe Smith</Name>
    <Street>123 XYZ Street</Street>
    <City>Harrogate</City>
    <State>North Yorkshire</State>
    <Zip>HG1 XYZ</Zip>
    <Country>United Kindgom</Country>
  </DeliveryAddress>
  ...
```

Invoice
Class : Invoice

Ordered By
Class : OrderedBy

Address
Class : Address

Delivery Address
Class : Address

XML Meets Objects

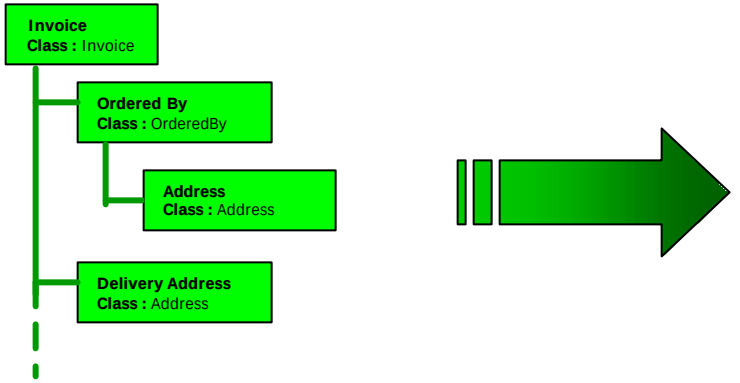
```
  ...
  <ShipDate>2001-02-03T16:44:08.715</ShipDate>
  <Items>
    <Item>
      <Code>W1000</Code>
      <Description>Small Widget</De
      <WarehouseLocation>WAB10</War
      <UnitPrice>10.50</UnitPrice>
      <Quantity>10</Quantity>
      <ExtPrice>105.00</ExtPrice>
   </Item>
   <Item>
   ...
   </Item>
  </Items>
  <NetProduct>218.10</NetProduct>
  <Postage>19.20</Postage>
  <TotalShipment>237.30</TotalShipment>
  <AmountCharged>237.30</AmountCharged>
</Invoice>
```

Items
Class : InvoiceItems
Methods:
addItem(InvoiceItem item)
removeItem(int index)
InvoiceItem getItem(int index)

Item
Class : InvoiceItem
Methods:
setCode(String code)
setUnitPrice(BigDecimal unitPrice)
setQuantity(int quantity)
String getCode()
BigDecimal getUnitPrice()
int getQuantity()

Marshalling and Unmarshalling

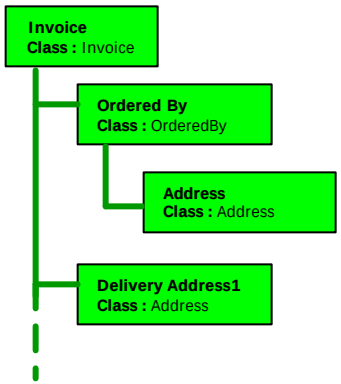
- Marshalling (Objects to XML)




```
<Invoice ordernumber="1001" rep="Fred Blogs">
  <OrderedBy>
    <CustomerNo>CUS1200JS</CustomerNo>
    <Date>2001-02-03T16:44:08.715</Date>
    <Address>
      <Name>Joe Smith</Name>
      <Street>123 XYZ Street</Street>
      <City>Harrogate</City>
      <State>North Yorkshire</State>
      <Zip>HG1 XYZ</Zip>
      <Country>United Kindgom</Country>
    </Address>
  </OrderedBy>
  <DeliveryAddress>
    <Name>Joe Smith</Name>
    <Street>123 XYZ Street</Street>
    <City>Harrogate</City>
    <State>North Yorkshire</State>
    <Zip>HG1 XYZ</Zip>
    <Country>United Kindgom</Country>
  </DeliveryAddress>
  ...
```

- Unmarshalling (XML to Objects)

```
<Invoice ordernumber="1001" rep="Fred Blogs">
  <OrderedBy>
    <CustomerNo>CUS1200JS</CustomerNo>
    <Date>2001-02-03T16:44:08.715</Date>
    <Address>
      <Name>Joe Smith</Name>
      <Street>123 XYZ Street</Street>
      <City>Harrogate</City>
      <State>North Yorkshire</State>
      <Zip>HG1 XYZ</Zip>
      <Country>United Kindgom</Country>
    </Address>
  </OrderedBy>
  <DeliveryAddress>
    <Name>Joe Smith</Name>
    <Street>123 XYZ Street</Street>
    <City>Harrogate</City>
    <State>North Yorkshire</State>
    <Zip>HG1 XYZ</Zip>
    <Country>United Kindgom</Country>
  </DeliveryAddress>
  ...
```



Sun XML Data Binding Specification

- JSR (Java Specification Request) #000031
  - http://java.sun.com/aboutJava/communityprocess/jsr/jsr_031_xmld.html
  - Submitted via JCP (Java Community Process)
  - Approved 23.08.1999
- Sun implementation called JAXB
  - Java API for XML Data Binding
    - Formally project Adelard
    - Expected to ship Q1 2001
    - Target Java platform is J2SE (Java 2 Standard Edition)

XML Schema's Making a Connection!

- XML Schemas are to XML what Classes are to Java!
  - Although XML Schemas don't contain logic ☺
- An XML document can be thought of as an 'instance' of a given XML Schema
- What more do XML Schemas and Java classes have in common?
  - This depends on your schema?

XML Schema's

- DTD (Document Type Definition)
- RDF (Resource Definition Language)
- XML-D (XML Data)
- XML-DR (XML Data - Reduced)
- DCD (Document Content Definition)
- SOX (Schema for Object-Oriented XML)
- DDML (Document Definition Markup Language)

Choosing an XML Schema for XML Data Binding

- Programmatic Access
- Scope
  - Namespace
  - Element
- Data Types
  - Constraints (Validation)
- Inheritance / Derivation
- Content Models
- Reuse

W3C XML Schema

- Requirements published 15th February 1999
- Usage Scenarios
  - “Open and uniform transfer of data between applications, including databases”
    - “When the exchange data model is represented by the more expressive XML Schema definitions, the task of mapping the exchange data model to and from application internal data models will be simplified. "
- Early Implementers…
  - Parsers
    - Oracle and Xerces
  - Editors / IDE’s
    - XML Spy, Microsoft .NET
- Current Status

XML Schema Namespaces

- XML Schema
  - http://www.w3.org/2000/10/XMLSchema
    - Typically uses the ‘xsd’ prefix
- XML Schema Instance
  - http://www.w3.org/1999/XMLSchema-instance
    - Typically uses the ‘xsi’ prefix
  - Used by XML documents that represent ‘instances’ of the XML Schema
    - Associate an XML Schema for validation purposes
    - Implement Polymorphism!
    - Indicate ‘null’ Elements

XML Schema Types and Derivation

- An XML Schema type is either…
  - Simple Type
  - Complex Type
- Type derivation by Extension
  - Can extend based type
  - Carn't modify base type
  - Permits only Complex base types
- Type derivation by Restriction
  - Carn't extend based type
  - Can modify base type
  - Permits both Complex and Simple base types

XML Schema 'schema' Element

- xmlns Attribute/s
- targetNamepace Attribute
- version Attribute
- "default" Attributes
  - elementForm, attributeForm

```
<xsd:schema
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
  xmlns:inv="http://www.sprokets.com/schemas/invoice"
  xmlns:whs="http://www.sprokets.com/schemas/whouse"
  targetNamespace="http://www.sprokets.com/schemas/invoice"
  elementFormDefault="qualified"
  version="1.0">
```

XML Schema <include> Element

- schemaLocation Attribute

```
<xsd:schema
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
  xmlns:inv="http://www.sprokets.com/schemas/invoice"
  targetNamespace="http://www.sprokets.com/schemas/invoice"
  elementFormDefault="qualified"
  version="1.0">

  <xsd:include schemaLocation="types.xsd"/>

</xsd:schema>
```

XML Schema <import> Element

- namespace Attribute
- schemaLocation Attribute

```
<xsd:schema
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
  xmlns:inv="http://www.sprokets.com/schemas/invoice"
  xmlns:whs="http://www.sprokets.com/schemas/whouse"
  targetNamespace="http://www.sprokets.com/schemas/invoice"
  elementFormDefault="qualified"
  version="1.0">

  <xsd:import
    namespace="http://www.sprokets.com/schemas/whouse"
    schemaLocation="whouse.xsd"/>

</xsd:schema>
```

XML Schema Simple Types

- Describes the text value of an Element
  - <Location>**WAB10**</Location>
- Describes the value of an Attribute
  - <Item reason="**DA**">
- Built-In W3C XML Schema Simple Types
  - string, boolean, short, int, long, decimal, time**I**nstant, binary, float, double …
  - Also all DTD types (ID, IDREF …)
  - List types NMTOKENS, IDREFS, ENTITIES
    - <Items>A1 B2 C3 D4</Items>
- Derived (Custom) Simple Types
  - <simpleType> Element

XML Schema <simpleType> Element

- Used to…
  - Create new simple types from built-in simple types
  - Create new simple types from other simple types
- <simpleType>
  - Attributes
    - name (optional)
  - Sub-Elements
    - <restriction>
      - Derive from other simple types by restriction
      - Legal values can be constrained via facet elements
    - <union>
    - <list>
      - Create new list types from existing atomic types

XML Schema <simpleType> Element

```
<xsd:simpleType name="TypeLocation">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="W[A-Z][A-Z][0-9][0-9]"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="TypeReason">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="DA"/>
    <xsd:enumeration value="EX"/>
    ...
  </xsd:restriction>
</xsd:simpleType>


<Location>WAB10</Location>
<Item reason="DA">...</Item>
```

XML Schema Complex Types

- Describe the Attributes of an Element
  - <Cost **currency="GBP"**>**25.50**</Cost>
- Describe the 'element' content of an Element
  - <Item **reason="DA"**>
    <Description/>
    <Product/>
    </Item>
- Describe the 'mixed' content of an Element
  - <Name>**<Mr/>Andrew Fawcett**</Name>

XML Schema <complexType> Element

- Attributes (optional)
  - name, abstract, final, mixed
- Sub-Elements
  - Building Content Models
    - <sequence> e.g. <Alphabet>**<A/><B/><C/>…**</Alphabet>
    - <all> e.g. <Alphabet>**<X/><D/><K/>…**</Alphabet>
    - <choice> e.g. <Colour>**<Black/>**</Colour> or <Colour>**<White/>**</Colour>
  - <simpleContent>
    - <restriction> Derive a new complexType from a simpleType by restriction.
  - <complexContent>
    - <extension> Derive a new complexType by extension
    - <restriction> Derive a new complexType by restriction

XML Schema <complexType> Element

```
<xsd:complexType name="Product">
  <xsd:sequence>
    <xsd:element name="Code" type="xsd:string"/>
    <xsd:element name="Description"
                 type="whs:typeDescription"/>
    <xsd:element name="WarehouseLocation"
                 type="whs:typeLocation"/>
    <xsd:element name="UnitPrice" type="xsd:decimal"/>
  </xsd:sequence>
</xsd:complexType>


<Product>
  <Code>W2000</Code>
  <Description>Medium Widget</Description>
  <WarehouseLocation>WAB11</WarehouseLocation>
  <UnitPrice>15.75</UnitPrice>
</Product>
```

XML Schema <complexType> Element

```
<xsd:complexType name="InvoiceItem">
  <xsd:complexContent>
    <xsd:extension base="whs:Product">
      <xsd:sequence>
        <xsd:element name="Quantity" type="xsd:int"/>
        <xsd:element name="ExtPrice" type="xsd:decimal"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<Item>
  <whs:Code>W1000</whs:Code>
  <whs:Description>Small Widget</whs:Description>
  <whs:WarehouseLocation>WAB10</whs:WarehouseLocation>
  <whs:UnitPrice>10.50</whs:UnitPrice>
  <Quantity>10</Quantity>
  <ExtPrice>105.00</ExtPrice>
</Item>
```

XML Schema <element> Element

- Describes an Element who's content and attributes must match that describe by the associated type
- As a sub-element of <schema>
  - The element is a valid root node of the document
- As sub-element of <complexType>
  - Element's specified at this level are 'local' to the type
- Attributes
  - name Attribute
  - type Attribute (optional?)
  - minOccurs / maxOccurs Attributes

XML Schema Anonymous Types

- Could also be described as ‘inline’ types
  - <element name=“Product”>
    <complexType>…</complexType>
    </element>
  - <element name=“Location”>
    <simpleType>…</simpleType>
    </element>
- <element>
  - The ‘type’ attribute is not required
- <complexType> and <simpleType>
  - The ‘name’ attribute is not required

XML Schema Polymorphism!

- xsi:type Attribute
  - Defined by XML Schema Instance namespace
  - Tells the application (parser) that the type used in the instance document differs from that defined in the XML Schema document
- Polymorphism is only allowed if…
  - The type indicated by the xsi:type attribute is derived from the type specified in the XML Schema for the given element
  - The schema, type or element has not explicitly blocked derived types in the instance document

XML Schema Polymorphism!

```
<Returns>
  ...
  <Items>
    <Item reason="DA">
      <Description>A cog was missing!</Description>
      <Product>
        <whs:Code>W1000</whs:Code>
        <whs:Description>Small Widget</whs:Description>
        <whs:WarehouseLocation>WAB10</whs:WarehouseLocation>
        <whs:UnitPrice>10.50</whs:UnitPrice>
      </Product>
      <Quantity>5</Quantity>
    </Item>
  </Items>
  ...
</Returns>
```

XML Schema Polymorphism!

```
<Returns>
  ...
  <Items>
    <Item reason="DA">
      <Description>A cog was missing!</Description>
      <Product xsi:type="inv:InvoiceItem">
        <whs:Code>W1000</whs:Code>
        <whs:Description>Small Widget</whs:Description>
        <whs:WarehouseLocation>WAB10</whs:WarehouseLocation>
        <whs:UnitPrice>10.50</whs:UnitPrice>
        <inv:Quantity>10</inv:Quantity>
        <inv:ExtPrice>105.00</inv:ExtPrice>
      </Product>
      <Quantity>5</Quantity>
    </Item>
  </Items>
  ...
</Returns>
```

Mapping XML Schema and Java

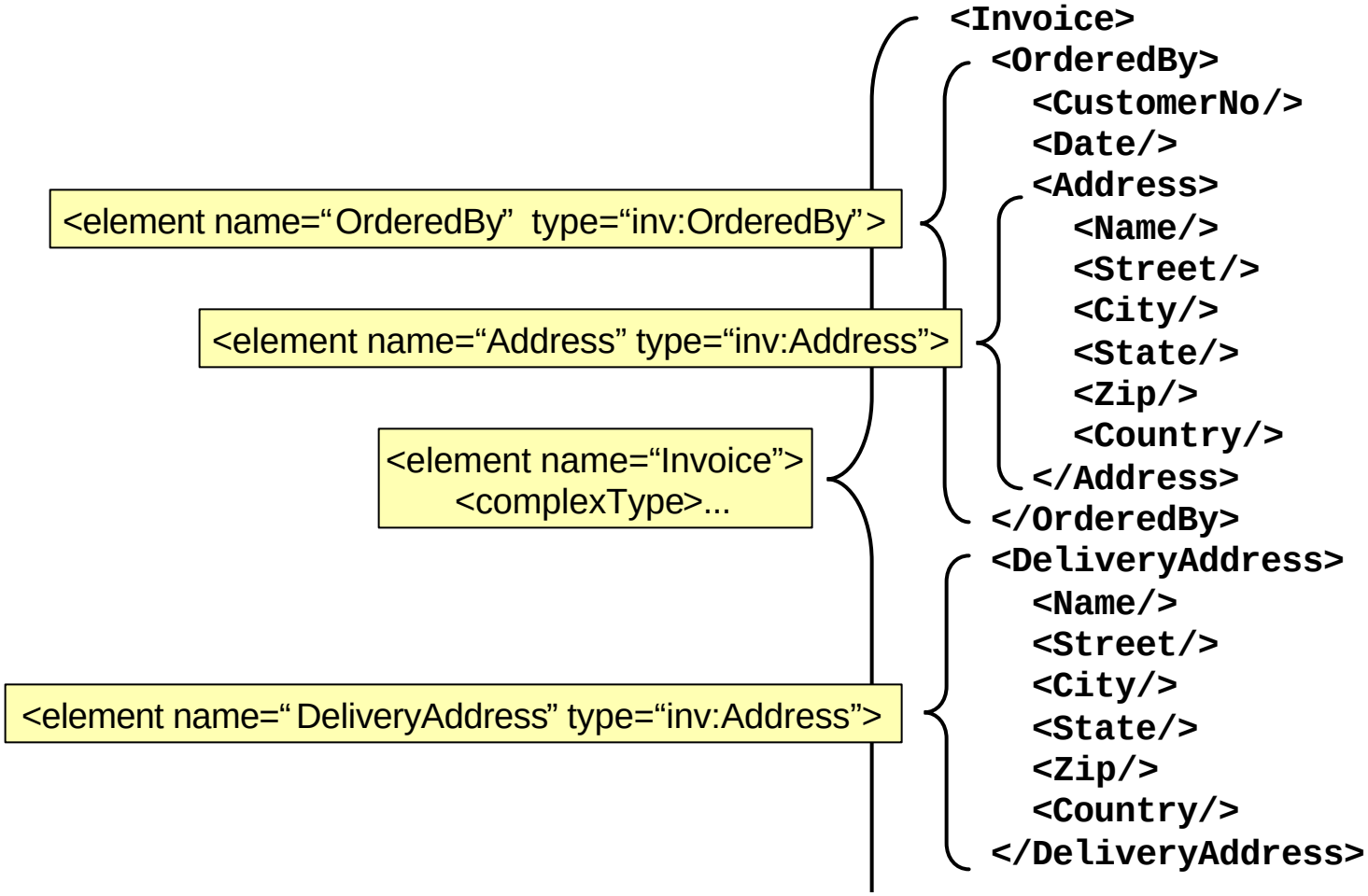
- Namespace to Package
- <complexType> to Class
- <simpleType> to Type Class
- <element> to Object
- Derivation
- Polymorphism
- Content Models
- Scope

Sproket's XML Schema

- Warehouse Schema (whouse.xsd)
  - targetNamespace
    - http://www.sprokets.com/schemas/warehouse
    - Namespace prefix 'whs'
- Invoice Schema (invoice.xsd)
  - targetNamespace
    - http://www.sprokets.com/schemas/invoice
    - Namespace prefix 'inv'
- Returns Schema (returns.xsd)
  - targetNamespace
    - http://www.sprokets.com/schemas/returns
    - Namespace prefix 'rtn'

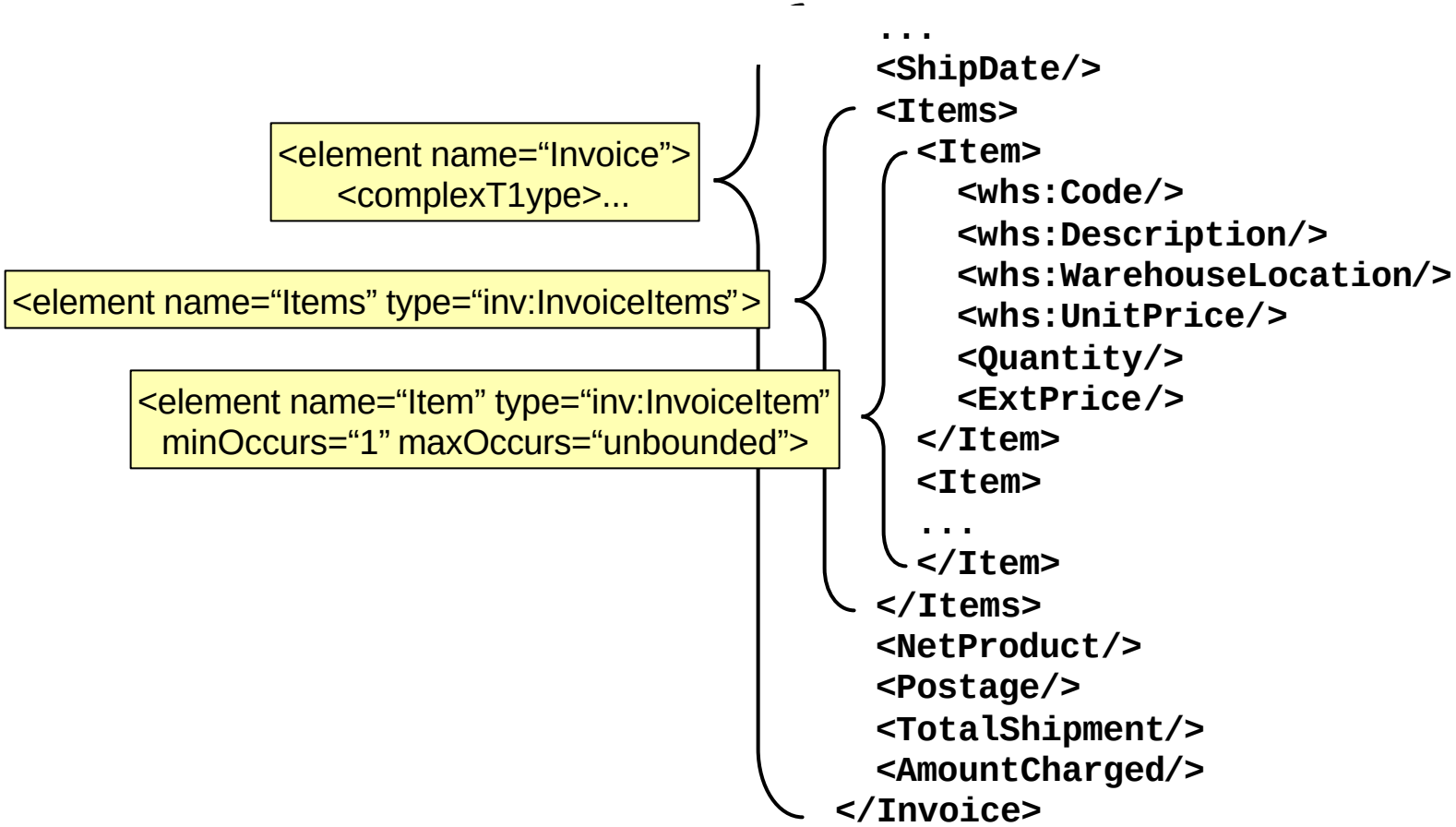
XML Schema

http://www.sprokets.com/schemas/invoice



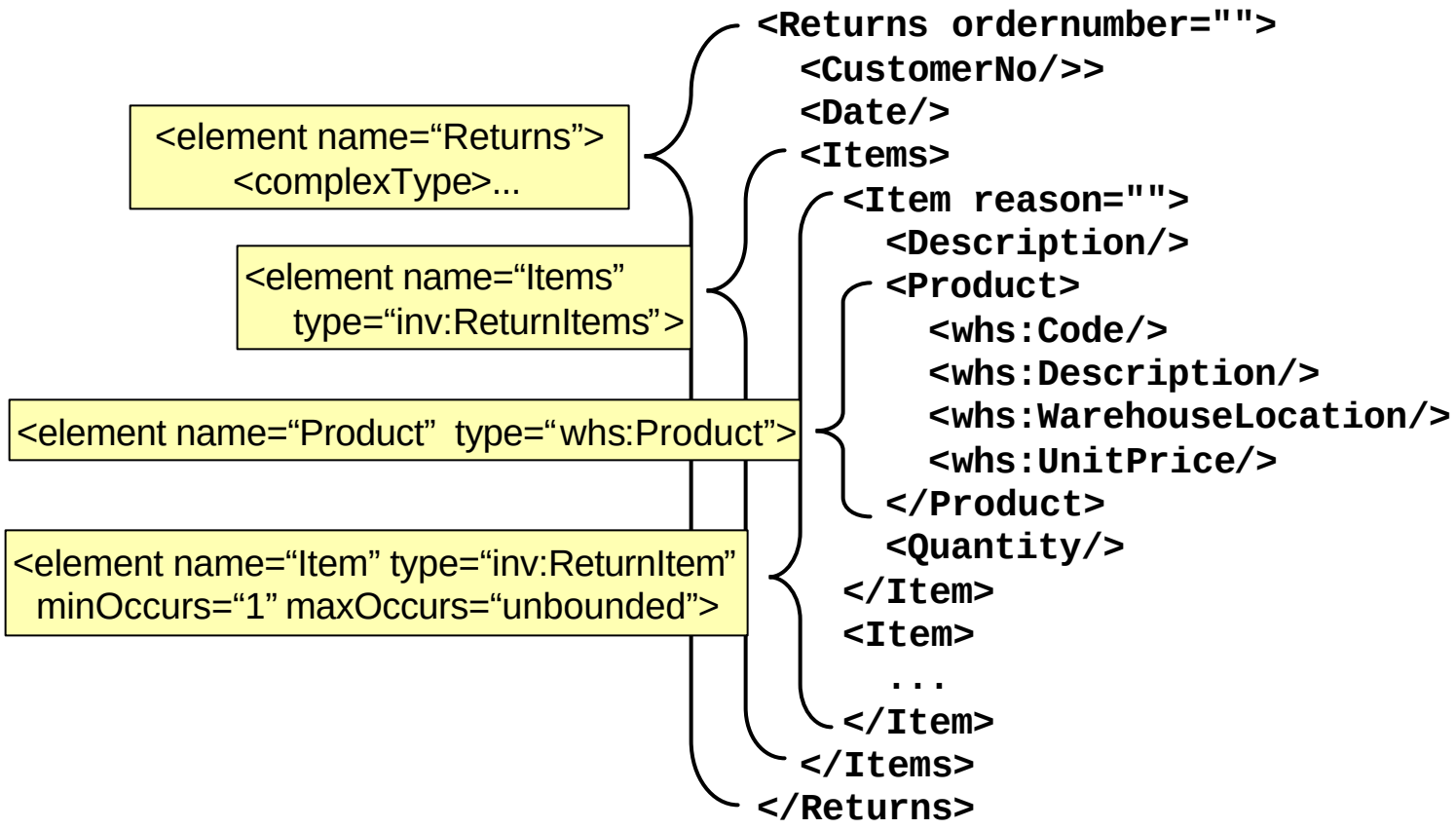
XML Schema

http://www.sprokets.com/schemas/invoice



XML Schema

http://www.sprokets.com/schemas/returns



Validating XML with XML Schema

- XML Document

```
<Invoice
  xmlns="http://www.sprokets.com/schemas/invoice"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xsi:schemaLocation=
   "http://www.sprokets.com/schemas/invoice
    file:///c:/schemas/invoice.xsd"
  ordernumber="1001"
  rep="Fred Blogs">
...
</Invoice>
```

- Java and Xerces

```
SAXParser parser = new SAXParser();
parser.setErrorHandler(new ErrorHandler() { ... });
parser.setFeature(
   "http://xml.org/sax/features/validation", true);
parser.parse("invoice.xml");
```

XML Data Binding Implementations

- Sun's JAXB (Q1 2001)
  - Java
- Microsoft .NET (Public Beta 1)
  - Visual Basic 7.0
  - C#
- Exolab's Castor http://castor.exolab.org/
  - Java

Castor XML

- “ Castor is the shortest path between Java[tm] objects, XML documents, SQL tables and LDAP directories. It provides Java to XML binding, Java to SQL/LDAP persistence, and then some more. “
- Open Source Project
- Sponsored by Exolab
- Version 0.9

Castor Marshalling Framework

- Descriptors
  - Objects that contain XML meta data…
    - Element Names, Structure, Namespaces, Constraints ect.
- Options for XML Data Binding
  - Java Introspection
    - Runtime Descriptors
    - Assumes ‘bean’ type methods
  - Mapping Files
    - Runtime Descriptors
    - Allows more specific bindings
  - Source Generator
    - Compile time Descriptors!
    - Generates Classes and Descriptors from XML Schema
    - Generated Classes use ‘bean’ type methods

Castor XML Schema Source Generator

- org.exolab.castor.builder.SourceGenerator
  - Required Arguments
    - -i Sets the input XML Schema filename
  - Optional Arguments
    - -dest Sets the output directory
    - -package Sets the package name
    - -nodesc Disables generation of Class Descriptors
    - -nomarshall Disables generation of Marshall methods
    - -f Suppress non fatal warnings, such as overwriting files.
    - -line-separator Sets line separator style for desired platform unix, max or win
    - -types Sets the source generator types factory name (from castor.properties), j1, j2 or odmg

Castor XML ‘castorbuilder.properties’

- Java Class Mapping
  - org.exolab.castor.builder.javaclassmapping
    - ‘type’ maps <complexType>’s to Java classes
    - ‘element’ maps <element>’s to Java classes
  - Default is ‘element’, we use ‘type’ ☺
- XML Namespace to Java Package Mapping
  - org.exolab.castor.builder.nspackages
    - Comma delimited list of ns=package values
  - Default is no package

Castor
XML Schema Types

- Java types
  - xsd:short short
  - xsd:boolean boolean
  - xsd:string java.lang.String
  - xsd:decimal java.math.BigDecimal
  - xsd:timeInstant java.util.Date
- Castor types from org.exolab.castor.types
  - xsd:century Century
  - xsd:year Year
  - xsd:month Month
  - xsd:date Date
  - xsd:time Time
  - xsd:timePeriod TimePeriod

Generated Source ‘whouse.xsd’

```
<xsd:complexType name="Product">
  <xsd:sequence>
    <xsd:element name="Code" type="xsd:string"/>
    <xsd:element name="Description" type="whs:typeDescription"/>
    <xsd:element name="WarehouseLocation" type="whs:typeLocation"/>
    <xsd:element name="UnitPrice" type="xsd:decimal"/>
  </xsd:sequence>
</xsd:complexType>
```

```
package com.sprokets.schemas.whouse;

public class Product implenents java.io.Serializable
{
  public String getCode()
  public String getDescription()
  public String getWarehouseLocation()
  public BigDecimal getUnitPrice()
  public void setCode(String code)
  public void setDescription(String description)
  public void setWarehouseLocation(String warehouseLocation)
  public void setUnitPrice(BigDecimal unitPrice)
}
```

Generated Source ‘invoice.xsd’

```
<xsd:complexType name="InvoiceItem">
  <xsd:complexContent>
    <xsd:extension base="whs:Product">
      <xsd:sequence>
        <xsd:element name="Quantity" type="xsd:int"/>
        <xsd:element name="ExtPrice" type="xsd:decimal"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

```
package com.sprokets.schemas.invoice;

public class InvoiceItem extends com.sprokets.schemas.whouse.Product
{
  public int getQuantity()
  public BigDecimal getExtPrice()
  public void setQuantity(int quantity)
  public void setExtPrice(BigDecimal unitPrice)
}
```

Generated Source ‘invoice.xsd’

```
<xsd:complexType name="InvoiceItems">
  <xsd:sequence>
    <xsd:element name="Item" type="inv:InvoiceItem"
       minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

package com.sprokets.schemas.invoice;

public class InvoiceItems implements java.io.Serializable
{
  public void addItem(InvoiceItem item)
  public void clearItem()
  public InvoiceItem getItem(int index)
  public InvoiceItem[] getItem()
  public int getItemCount()
  public boolean removeItem(InvoiceItem item)
  public void setItem(InvoiceItem item, int index)
  public void setItem(InvoiceItem[] items)
}
```

Generated Source ‘returns.xsd’

```
<xsd:complexType name="ReturnItem">
  <xsd:sequence>
    <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
    <xsd:element name="Product" type="whs:Product"/>
    <xsd:element name="Quantity" type="xsd:int"/>
  </xsd:sequence>
  <xsd:attribute name="reason" type="rtn:TypeReason"/>
</xsd:complexType>
```

```
package com.sprokets.schemas.returns;

public class ReturnItem implements java.io.Serializable
{
  public String getDescription()
  public Product getProduct()
  public int getQuantity()
  public TypeReason getReason()
  public void setDescription(String description)
  public void setProduct(Product product)
  public void setQuantity(int quantity)
  public void setReason(TypeReason reason)
}
```

Generated Source ‘returns.xsd’

```
<xsd:simpleType name="TypeReason">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="DA"/>
    <xsd:enumeration value="EX"/>
    <xsd:enumeration value="FA"/>
    <xsd:enumeration value="TS"/>
    ...
  </xsd:restriction>
</xsd:simpleType>
```

```
package com.sprokets.schemas.returns.types;

public class TypeReason implements java.io.Serializable
{
  public static final TypeReason DA ...
  public static final TypeReason EX ...
  public static final TypeReason FA ...
  public static final TypeReason TS ...
  ...
}
```

Marshalling with Castor

- org.exolab.castor.xml.Marshaller
  - Object to SAX
    `static void marshal(Object obj, DocumentHandler handler)`
  - Object to DOM
    `static void marshal(Object obj, Node node)`
  - Object to File (uses Xerces XML Serialiser by default)
    `static void marshal(Object obj, Writer writer)`
- Additional Options

```
new Marshaller(DocumentHandler handler)
new Marshaller(Node node)
new Marshaller(Writer writer)
```

  - **`setValidation(boolean validation)`**
  - **`setNamespaceMapping(String prefix, String ns)`**
  - **`setNSPrefixAtRoot(boolean nsPrefixAtRoot)`**
  - **`setMapping(org.exolab.castor.mapping.Mapping mapping)`**
  - **`marshal(Object object)`**

Unmarshalling with Castor

- org.exolab.castor.xml.Unmarshaller
  - Object from SAX
    ```
    Object unmarshall(EventProducer events)
    ```
  - Object from DOM
    ```
    static Object unmarshall(Class cls, Node node)
    ```
  - Object from File
    ```
    static Object unmarshall(Class cls, Reader reader)
    static Object unmarshall(Class cls, InputSource)
    ```
- Additional Options
  ```
  new Unmarshaller(Class objectClass)
  new Unmarshaller(Class objectClass, ClassLoader loader)
  new Unmarshaller(org.exolab.castor.mapping.Mapping)
  ```
  - `setValidation(boolean validation)`
  - `Object unmarshall(Reader reader)`
  - `Object unmarshall(Node node)`
  - `Object unmarshall(EventProducer events)`

More to Castor...

- XML Schema Object Model
  - org.exolab.castor.xml.schema

```
SchemaUnmarshaller schemaHandler =
        new SchemaUnmarshaller();
Parser parser = ParserFactory.makeParser();
parser.setDocumentHandler(schemaHandler);
parser.parse("myschema.xsd");
Schema schema = schemaHandler.getSchema();
Enumeration enum = schema.getElementDecls();
Enumeration enum = schema.getComplexTypes();
```

- Java Source Object Model
  - org.exolab.javasource

Integrating Castor Source Generator with Ant

- The Ant Build System from jakarta.apache.org
- Ant build targets used by Sprokets.com
  - init
  - gen.whouse
  - gen.invoice
  - gen.returns
  - compile
- Using the <java> Task
  - To invoke the Source Generator
- Using the <uptodate> Task
  - To control when the Source Generator is invoked

Integrating Castor Source Generator with Ant

```
<project name="Sprokets" default="compile" basedir=".">

  <property name="schemas" value="."/>
  <property name="source" value="source"/>
  <property name="classes" value="classes"/>
  <property name="schemas.source"
            value="${source}/com/sprokets/schemas"/>
  <property name="schemas.classes"
            value="${classes}/com/sprokets/schemas"/>

  <path id="classpath">
    <pathelement location="."/>
    <pathelement location="lib\castor-0.9.jar"/>
    <pathelement location="lib\xerces.jar"/>
  </path>

  <target name="init">
    <mkdir dir="${schemas.source}/invoice"/>
    <uptodate property="gen.invoice"
              targetfile="${schemas}/invoice.xsd">
      <srcfiles dir="${schemas.source}/invoice"/>
    </uptodate>
  </target>
```

Integrating Castor Source Generator with Ant

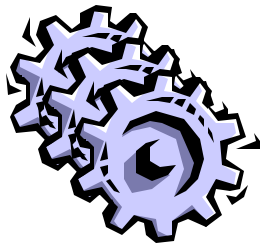
```
  <target name="gen.invoice" if="gen.invoice" depends="init">
    <delete dir="${schemas.source}/invoice"/>
    <delete dir="${schemas.classes}/invoice"/>
    <java classname="org.exolab.castor.builder.SourceGenerator">
      <arg value="-i"/>
      <arg value="${schemas}\invoice.xsd"/>
      <arg value="-dest"/>
      <arg value="${source}"/>
      <classpath refid="classpath"/>
    </java>
  </target>

  <target name="compile" depends="gen.invoice">
    <javac srcdir="${source}" destdir="${classes}">
      <classpath refid="classpath"/>
    </javac>
  </target>

</project>
```

Putting it all together...

- Produce Invoice Application
  - com.sprokets.apps.ProduceInvoice
  - Creates the ‘invoice.xml’ file



- Receive Invoice Application
  - com.sprokets.apps.ReceiveInvoice
  - Parses ‘invoice.xml’ and displays it’s content
- Product Returns Application
  - com.sprokets.apps.ProduceReturns
  - Uses information from the ‘invoice.xml’ file to produce the ‘returns.xml’ file.

Time for a closer look...

- Directory Structure
- XML Schema
- Generated Source
- Program Source
- Building the Example
- Running the Example