XML in the Java™ Platform

Larry Cable, Senior Staff Engineer
Mark Reinhold, Senior Staff Engineer
Java Software, Sun Microsystems, Inc.

<Contents>

- XML Overview
- XML usage in the Java™ Platform
- XML “Basic Plumbing” Standard Extension
- XML Data Binding Extension
- Resources
- Q&A

What's the Problem?

- **Fundamental "issues":**
  - **How do I represent my "application" data?**
    - **Performance (speed/time)**
    - **Persistence (short/long lived)**
    - **Mutability**
    - **Composition**
    - **Security (encryption/identity)**
    - **…**

What's the Problem?

- **Open Information Management:**
  - **Interpretation**
  - **Presentation**
  - **Interoperation**
  - **Portability**
  - **Composition**
  - **Interrogation**
  - **…**

XML Evolution

- SGML
  - Complete but complex language for describing the structure of (large) documents
- HTML
  - Derived from SGML but focused on rendering/presentation of documents
  - Divergence, incompatibilities, rendering centric design, and lack of extensibility are an issue

What Is XML?

- eXtensible Markup metaLanguage and a panacea?
- Universal data description mechanism
- Application areas:
  - Presentation Oriented Publishing
  - Enterprise Message Oriented Middleware
  - ...

XML Applications

- Web “Compound” Documents:
  - General and domain specific
- Dynamic publishing
- E-Commerce/EDI
- Enterprise Application Integration
- Structured Information Management and Retrieval

XML Technologies

- **XML 1.0:**
  - **Describes XML syntax/semantics**
- **Some related technologies:**
  - **SAX: event driven parser**
  - **Document Object Model: tree based XML parser**
  - **XLink/XPointer: hyperlinks for XML documents**
  - **XSL/XSLT: Style Sheet/Transformation**
  - **Schema: More flexible XML DTD mechanism**
  - **X Query Language: Queries over XML documents**

XML and the Java™ Platform

- XML provides a universal syntax
  - But what about behavior (semantics)?

```
<transmogrify>
This text has semantics associated with it, and
conferred upon it by the enclosing element, but
how do we discover what those semantics are in
a portable fashion?
</transmogrify>
```

- Complementary technologies
  - Java™ Platform provides universal semantics
- Portable reusable data;
  Portable behavior

What Is Sun Doing?

- Already using XML in the platform
- Participating in the W3C, XML.ORG, OASIS, …
- Developing XML Standard Extension
- Java™ 2 Platform, Enterprise Edition (J2EE)
- XML Data Binding Standard Extension

XML Applications in the Java 2 Platform

- **JavaHelp™ API:**
  - **Used to describe help meta-information**
- **Enterprise JavaBeans™ technology (EJB):**
  - **Used to describe deployment descriptor in 1.1**
- **J2EE Application Programming Model:**
  - **Application integration**
- **JavaServer Pages™ software (JSP):**
  - **Alternative XML based syntax**
  - **Generate/consume XML**

Standard Extension

- What are our goals?
  - First-class XML support in the Java 2 platform
  - Work with W3C and other XML technology creators to develop APIs
  - Explore other uses of XML in the Platform

Industry Support

- Ariba
- Bluestone Software
- BEA/Web Logic
- Commerce One
- Fujitsu
- Megginson Technologies Ltd.
- NetPost
- Object Design
- Oracle
- Sybase
- Vervet Logic
- Webmethods
- ...

What Do We Need?

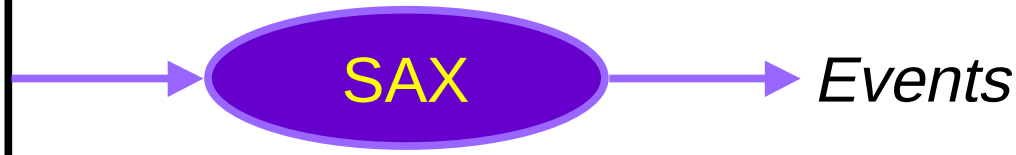
- XML “Basic Plumbing”
  - Phase 1
    - Ability to parse XML
      - Parser(s)
      - SAX (1 or 2?)
      - DOM (level 1 or level 2?)
    - Namespaces
      - Support in parser(s)

SAX

- Event driven interface to XML Parser(s)
- Developed by XML-DEV experts
- org.xml.sax.* package
- www.megginson.com/SAX/javadoc

SAX ...

```
<?xml version="1.0" ?>
<!DOCTYPE stuff SYSTEM "…" >
<stuff>
this is some stuff!
</stuff>
```

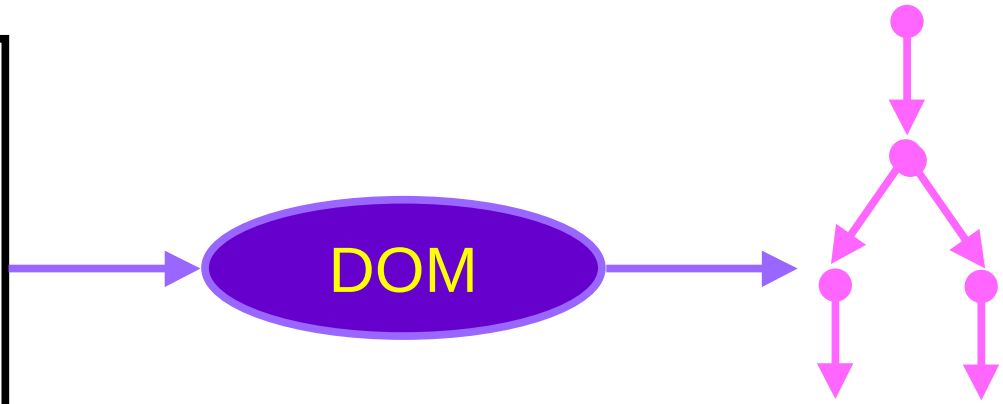


DOM

- "Tree" data structure interface to XML parser
- Models document structure as objects
- Defines a Java programming language binding:
  - org.w3c.dom.* package
- W3C specification:
  - www.w3c.org/TR/1998/REC-DOM-Level-1-1998001

DOM …

```
<?xml version="1.0" ?>
<!DOCTYPE stuff SYSTEM "…" >
<stuff>
this is some stuff!
</stuff>
```



Namespaces

- A Namespace defines a distinct set of XML markup elements (DTD)
- Allows multiple vocabularies to be combined in a single document instance
- Requires parser support
- Defined by:
  - www.w3c.org/TR/REC-xml-names
- Schema has implications …

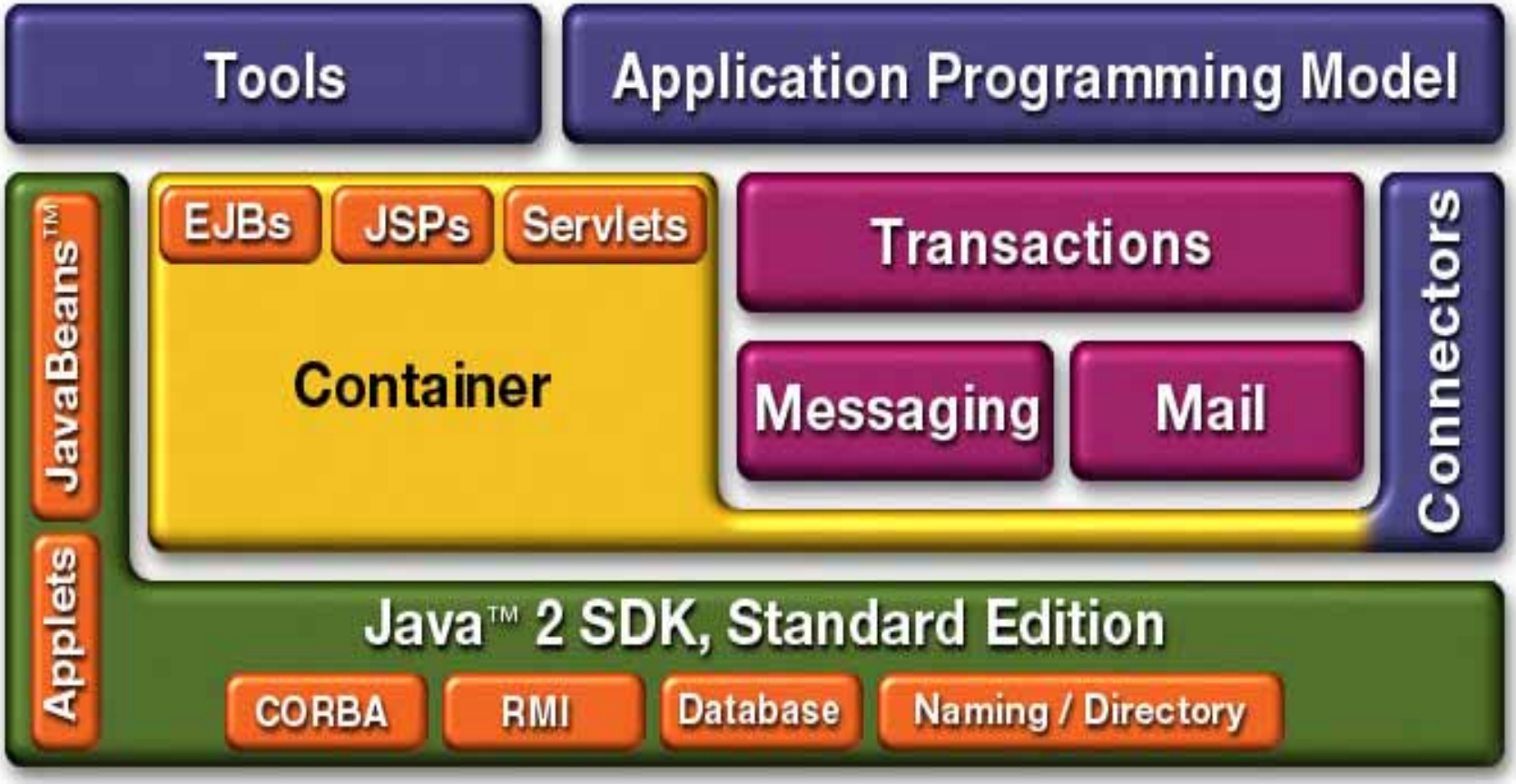
Roadmap

- **XML Standard Extension:**
  - **Public draft of spec and alpha RI** **Q399**
  - **Public release of spec and RI** **Q499**

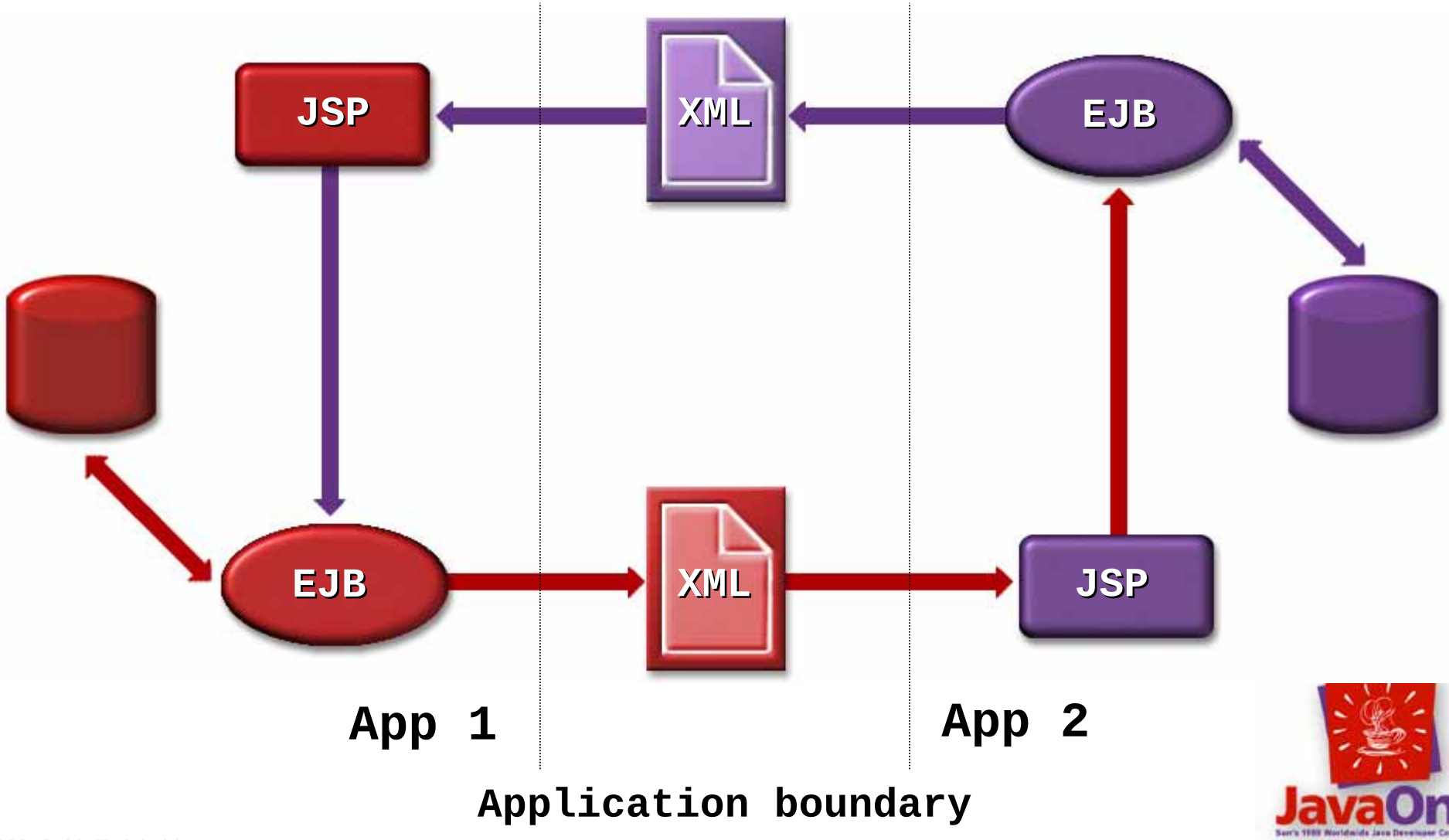
XSL/XSLT & Query

- Under consideration for Phase 2:
  - XSL/XSLT:
    - Transformation language and stylesheet support
  - XML Query Language:
    - Query language that enables searches of XML document

J2EE Architecture



J2EE Web App Model



JavaServer Pages℠ Model

J2EE XML Futures

- **Application Programming Model:**
  - **EJB:**
    - **XML/Query usage**
  - **Java Message Service (JMS):**
    - **Produce/consume XML messages**
  - **JSP:**
    - **XML syntax**
    - **Better support for producing/consuming XML**
    - **Pre/post filtering XSL/XSLT support**

XML Data Binding

- XML = Portable data
  … Data without meaning
- Schemas add meaning to XML
- Working with XML in programs
  … Binding makes XML easy to use
- Archiving

XML = Portable Data

XML = Portable Data

Elements

```
<ShoeOrder>
</ShoeOrder>
```

XML = Portable Data

Elements

Text

```
<ShoeOrder>
  <color>Brown</color>
  <size>9 1/2</size>
  <width>AA</width>
</ShoeOrder>
```

XML = Portable Data

Elements

Text

Attributes

```
<ShoeOrder id="4040458">
  <color>Brown</color>
  <size>9 1/2</size>
  <width>AA</width>
</ShoeOrder>
```

XML = Portable Data

- Elements
- Text
- Attributes
- Links

```
<ShoeOrder id="4040458"
           style="Sandal">
  <color>Brown</color>
  <size>9 1/2</size>
  <width>AA</width>
</ShoeOrder>

<ShoeStyle id="Sandal">
  <gender>F</gender>
  <colors>Brown Black</colors>
</ShoeStyle>
```

Terminological Digression

Is XML for documents, or for data?

Both!

- **Roots are in the document world (SGML)**
- **It is also useful for data messages**

XML = Data Without Meaning

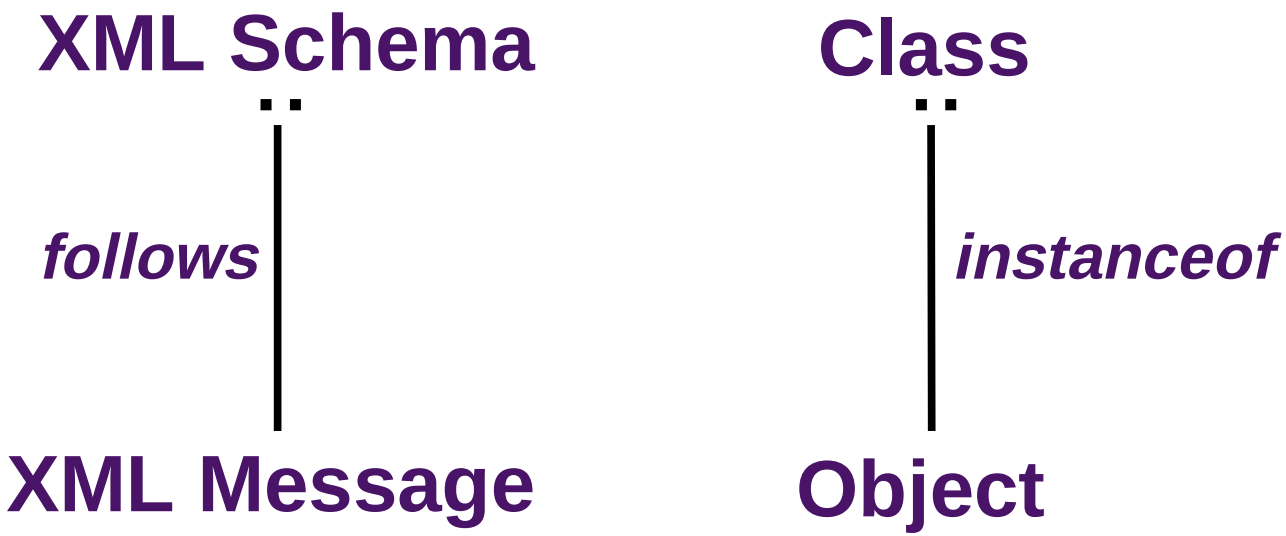
```
<ShoeOrder id="4040458"
           style="Sandal">
  <color>Brown</color>
  <size>9 1/2</size>
</ShoeOrder>
```

Are these identical?

```
<ShoeOrder id="4040458"
           style="Sandal"
           colour="Brown"
           size="42">
</ShoeOrder>
```

Schemas Add Meaning to XML

Schema = Description of syntax & semantics of an XML message



Schemas Add Meaning to XML

Benefits of schemas

- **Constrain structure and content**
  - **Enables automatic validation**
- **Describe intended conceptual meaning**
  - **Aids human understanding**

Schemas Add Meaning to XML

There are many XML schema languages

- **DCD**
- **DDML**
- **SOX**
- **XML-Data**
- **XML Schema (W3C proposed recommendation)**

Schemas Add Meaning to XML

```
<schema name="ShoeOrder">
  <elementType name="ShoeOrder">
```

Schemas Add Meaning to XML

```
<schema name="ShoeOrder">
  <elementType name="ShoeOrder">

    <attrDecl name="id" required="true">
      <datatypeRef name="ID"/>
    </attrDecl>

    <attrDecl name="style" required="true">
      <datatypeRef name="IDREF"/>
    </attrDecl>
```

Schemas Add Meaning to XML

```
<schema name="ShoeOrder">
  <elementType name="ShoeOrder">
    <attrDecl .../>

    <model>
      <sequence>
        <elementTypeRef name="color"/>
        <elementTypeRef name="size"/>
        <elementTypeRef name="width"/>
      </sequence>
    </model>
```

Schemas Add Meaning to XML

```
<schema name="ShoeOrder">
  <elementType name="ShoeOrder">
    <attrDecl .../>
    <model .../>

    <elementType name="color">
      <datatypeRef name="Colors"/>
    </elementType>

    <elementType name="size">
      <datatypeRef name="Sizes"/>
    </elementType>

    <elementType name="width" .../>
```

Schemas Add Meaning to XML

```
<schema name="ShoeOrder">
  <elementType name="ShoeOrder">
    <attrDecl .../>
    <model .../>
    <elementType .../>

    <datatype name="Colors">
      <basetype name="string"/>
      <enumeration>
        <literal>Black</literal>
        <literal>Brown</literal>
        <literal>Tan</literal>
      </enumeration>
    </datatype>
```

Schemas Add Meaning to XML

```
<schema name="ShoeOrder">
  <elementType name="ShoeOrder">
    <attrDecl .../>
    <model .../>
    <elementType .../>

    <datatype name="Sizes">
      <basetype name="string"/>
      <lexicalRepresentation>
        <lexical>[1-9][0-9]?( 1/2)?</lexical>
      </lexicalRepresentation>
      <minInclusive>3 1/2</minInclusive>
      <maxInclusive>13</maxInclusive>
    </datatype>
```

Schemas Add Meaning to XML

```
<schema name="ShoeOrder">
  <elementType name="ShoeOrder">
    <attrDecl .../>
    <model .../>
    <elementType .../>

    <datatype name="Widths">
      <basetype name="string"/>
      <lexicalRepresentation>
        <lexical>AAA|AA|[A-E]|EE|EEE</lexical>
      </lexicalRepresentation>
    </datatype>
```

Schemas Add Meaning to XML

```
<schema name="ShoeOrder">
  <elementType name="ShoeOrder">
    <attrDecl .../>
    <model .../>
    <elementType .../>
    <datatype .../>

  </elementType>
</schema>
```

Working with XML in Programs

Idea: Map message components to, and from, objects

Need: Classes for message components
Marshalling/unmarshalling code

Working with XML in Programs

Classes for message components

```
public class ShoeOrder {
    public ShoeOrder(String id, Style style,
                     String color, String size);
    public String  getId();
    public void    setId(String id);
    public Style   getStyle();
    public void    setStyle(Style style);
    public String  getColor();
    public void    setColor(String color);
    public String  getSize();
    public void    setSize(String size);
}
```

Working with XML in Programs

Marshalling/unmarshalling code

```
public void acceptOrder(InputStream in) {
    ShoeOrder so = unmarshal(in);
    WarehouseDB.submit(so);
}
```

How do you write `unmarshal`?

Working with XML in Programs

How do you write `unmarshal`? Use SAX!

```
private static ShoeOrder newOrder = null;

static class DocHandler implements DocumentHandler {

  public void setDocumentLocator(Locator l) { }
  public void startDocument() { }
  public void endDocument() { }
  public void ignorableWhitespace(char[] cbuf, int offset, int len) { }
  public void processingInstruction(String target, String data) { }

  ShoeOrder so = null;
  String cur = null;

  public void startElement(String name, AttributeList al) {
    if (name.equals("ShoeOrder")) {
      so = new ShoeOrder();
      for (int i = 0, n = al.getLength(); i < n; i++) {
               String an = al.getName(i);
               if (an.equals("id")) {
                 so.setId(al.getValue(i));
               } else if (an.equals("style")) {
                 so.setStyle(al.getValue(i));
               } else {
                 throw new RuntimeException("Unknown attribute: "
                                            + an);
               }
      }
    } else {
      cur = name;
    }
  }

  public void characters(char[] cbuf, int offset, int len) {
    if (cur == null) return;
    String val = new String(cbuf, offset, len);
    if (cur.equals("color")) {
      so.setColor(val);
    } else if (cur.equals("size")) {
      so.setSize(Integer.parseInt(val));
    } else if (cur.equals("width")) {
      so.setWidth(val);
    } else {
      throw new RuntimeException("Unknown element: " + cur);
    }
  }

  public void endElement(String name) {
    if (name.equals("ShoeOrder")) {
      newOrder = so;
    } else {
      cur = null;
    }
  }

}

public static ShoeOrder unmarshal(InputStream in)
        throws SAXException
{
    InputSource is = new InputSource(in);
    Parser p = ParserFactory.makeParser();
    p.setDocumentHandler(new DocHandler());
    p.parse(is);
    return newOrder;
}
```

Working with XML in Programs

How do you write `unmarshal`? ~~Use SAX!~~ Use DOM!

```
public static ShoeOrder unmarshal(InputStream in)
    throws IOException, SAXException
{
    XmlDocument xd = XmlDocument.createXmlDocument(in, false);
    Element r = xd.getDocumentElement();
    ShoeOrder so = new ShoeOrder();
    so.setId(r.getAttribute("id"));
    so.setStyle(r.getAttribute("style"));
    for (Node n = r.getFirstChild(); n != null; n = n.getNextSibling()) {
        if (n instanceof Element) {
            Element e = (Element)n;
            String tn = e.getTagName();
            if (tn.equals("color")) {
                String val = ((CharacterData)e.getFirstChild()).getData();
                so.setColor(val);
            } else if (tn.equals("size")) {
                String val = ((CharacterData)e.getFirstChild()).getData();
                so.setSize(Integer.parseInt(val));
            } else if (tn.equals("width")) {
                String val = ((CharacterData)e.getFirstChild()).getData();
                so.setWidth(val);
            } else {
                throw new RuntimeException("Unknown element: " + tn);
            }
        }
    }
    return so;
}
```

Working with XML in Programs

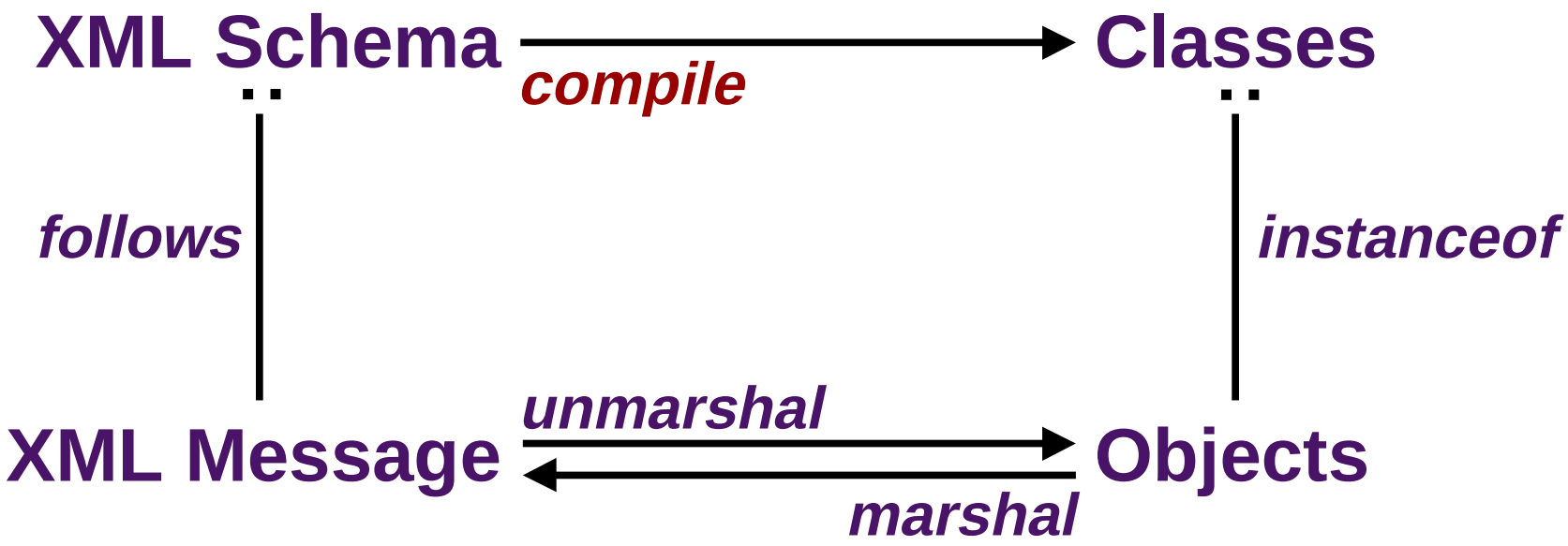
How do you write `unmarshal`? ~~**Use SAX!**~~ ~~**Use DOM!**~~

Problems with these approaches

- **Need to write code**
- **Need to maintain code as schemas evolve**

We can do better…

Binding XML to Programs



Binding compiles schemas into classes

Binding XML to Programs

Binding generates classes that contain:

- **Marshalling/unmarshalling code**
  - **Full error and validity checking**
- **Component access methods (`get/set`)**
  - **Mutators ensure consistency with schema**

Big win: Simplifies creation & maintenance

Binding XML to Programs

```
public class ShoeOrder {
    public void                 marshal(OutputStream);
    public static ShoeOrder     unmarshal(InputStream);
    public ShoeOrder(String id, Style style,
                     String color, String size);
    public String  getId();
    public void    setId(String id);
    public Style   getStyle();
    public void    setStyle(Style style);
    public String  getColor();
    public void    setColor(String color);
    public String  getSize();
    public void    setSize(String size);
 }
```

Binding XML to Programs

How do you write `unmarshal`? ~~Use SAX!~~ ~~Use DOM!~~ You don't!

```
public void acceptOrder(InputStream in) {
    ShoeOrder so = ShoeOrder.unmarshal(in);
    WarehouseDB.submit(so);
}
```

Binding XML to Programs

Can you create an invalid ShoeOrder? No!

```
ShoeOrder so = ShoeOrder.unmarshal(in);

so.setColor("Red");    // Exception thrown
so.setSize("5 3/4");   // Exception thrown
so.setWidth("Z");      // Exception thrown
```

An XML Data-Binding Facility for the Java™ Platform

- What
  - Marshalling/unmarshalling framework
  - XML schema compiler
- How
  - Java℠ Community Process

Archiving

- **Binding is XML-centric**
  - **Classes derived from XML schema**
- **Archiving is program-centric**
  - **XML messages derived from classes**
  - **Can share marshalling/unmarshalling framework**

Archiving

Types of archiving

- General-purpose run-time archiving
  - Requires explicit metadata
  - Fast
- Archiving graphs of beans
  - Uses introspection to generate metadata
  - Slow to write, fast to read
  - Currently being explored by the Project Swing Team

XML Data Binding

- XML is portable, but meaningless, data
- Schemas add meaning to XML
- Binding leverages schemas to make XML easy to use
- A binding facility will be added to the Java Platform

Project X

- Technology Release 2 contains:
  - High performance XML parser(s) (incl. verifying)
  - SAX
  - DOM
  - Utility classes
- Free for commercial use
- Download from:
  - developer.java.sun.com/developer/products/xml/index.html

BOF

- **XML Advanced Topics**
  - **Thursday, 8:00pm–9:00pm, Room B1, Marriott**

Resources

- XML web site:
  - java.sun.com/xml
- Java Community Process (JCP):
  - java.sun.com/aboutJava/communityProcess
- Feedback:
  - xml-feedback@java.sun.com
- W3C:
  - www.w3c.org/XML
- XML.ORG:
  - www.xml.org

XML in the Java™ Platform

< QandA />

JavaOne℠

Sun's 1999 Worldwide Java Developer Conference℠