Deep Dive Into Java™ Architecture for XML Binding 2.0

Sekhar Vajjhala

Hans Hrasna

Kohsuke Kawaguchi

TS-1607

Goal of This Talk

Use Java™ Architecture for XML Databinding (JAXB) for simplified access to XML content

Agenda

JAXB 2.0 Overview

Annotation Driven Programming Model

Java and .NET Databinding Interoperability

JAXB NetBeans™ Module

JAXB on java.net

JAXB Performance

Summary

JAXB 2.0

Major Features

- Ease of development
  - Typed access to XML content
  - Leverage Java SE 5 features
- 100% XML Schema support
- Java classes to XML Schema binding
- Databinding for Java API for XML Web Services (JAX-WS)
- Schema evolution
- Partial Updateable binding
- Part of Java SE 6 and Java EE 5 platforms

Java EE 5

Focus is on Ease of Development
Major Revamp of Programming Model
EJB 3.0 support for POJOs means less to learn, less to code and less to maintain
New Java Persistence API makes object / relational mapping cleaner and easier
New and updated Web Services (JAX-WS 2.0 and JAXB 2.0) simplifies SOA implementation
JavaServer Faces 1.2 facilitates building Web 2.0 Applications with AJAX
Annotations eliminate the need for deployment descriptors, in most cases
Supported by NetBeans Enterprise Developer Pack 5.5 Preview today
Get the SDK: http://java.sun.com/javaee/downloads/

Project GlassFish

- Simplifying Java application development with **Java EE 5 technologies**
- Includes JWSDP, EJB 3.0, JSF 1.2, JAX-WS and JAXB 2.0
- Supports > **20** frameworks and apps
- **Open source** CDDL license
- Basis for the **Sun Java System Application Server PE 9**
- **Free** to download and **free** to deploy
- Over **1200** members & **200,000** downloads
- Integrated with NetBeans
- **News:** blogs.sun.com/theaquarium

Building a Java EE 5 open source application server

Java.sun.com/javaee/GlassFish

Source: Sun 2/06—See website for latest stats

Agenda

JAXB 2.0 Overview

Annotation Driven Programming Model

Java and .NET Databinding Interoperability

JAXB NetBeans Module

JAXB on java.net

JAXB Performance

Summary

Annotation Driven Programming Model

Overview

- Metadata for Java based classes ↔ XML mapping
  - e.g. @XmlAttribute—map to XML attribute
  - Package: javax.xml.bind.annotation
- Start from XML Schema
  - Generate portable, annotated code
- Start from Java based classes
  - Annotate code with JAXB annotations
  - Generate XML Schema
- Use JAXB Runtime to process XML instances
  - Marshal, Unmarshal, Validate

Java Based Classes ↔ XML Schema

Example: Annotation Driven Programming Model

```
@XmlType(name="Employee", propOrder="name, id")

public class Employee {
   public String getName(){ ... }
   public void setName(String ) {...}

   public long getId() {...}
   public void setId(long ) {...}

}
```

```
<xs:complexType name="Employee">
   <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="0"/>
    <xs:element name="id" type="xs:long"/>

    ...
```

XML Schema → Java Based Classes

Ease of development

```
// derived from XML Namespace
package com.example;

// value class - derived from XML Schema complex type
// Use new PurchaseOrder()
public class PurchaseOrder {

    // JavaBean properties
    public Address getShipTo() {...}
    public void setShipTo( Address ) { ... }

    // JAXP 1.3 datatype
    public XMLGregorianCalendar getShipDate() {...}
    public void setShipDate(XMLGregorianCalendar ) {...}
}

// derived from simple type with enumeration facets
public enum USState {...}
```

XML Schema → Java Based Classes

100% XML Schema support

```
// derived from XML Namespace
package com.example;

public class ObjectFactory {
    // JAXBElement wraps value with element name..
    // e.g. Substitution groups
    JAXBElement<AutoType> createAuto(AutoType value);
}
public class PurchaseOrder {
    // <xs:any maxOccurs="unbounded"/>
    public List<Object> getAny() { ... }
    // <xs:anyAttribute/>
    public Map<QName, String> getOtherAttributes {...}

    // roundtrip element names, absence of element, xsi:nil
    public JAXBElement<FooType> getFoo() {...}
}
```

XML Schema → Java Based Classes

Customization

Defined in namespace: http://java.sun.com/xml/ns/jaxb

```
<!-- Inline: Ease of development  -->

<xs:complexType name=”PurchaseOrderType>
   <xs:annotation><xs:appinfo>
      <jaxb:classname=”POType”/>
   </xs:appinfo></xs:annotation>
   ...

// Binding for <xs:PurchaseOrderType>

public class PoType {...}

<!-- External: No modification of schema-->

<jaxb:bindings node="//xs:complexType[@name='PurchaseOrderType']">

   <jaxb:class name="POType"/>

</jaxb:bindings>
```

XML Schema→Java Based Classes

Using Customizations

```
<!-- use global customizations  -->
<jaxb:globalBindings collectionProperty =”indexed”/>

<!-- use fine grained customizations for control -->
<xs:complexType name=”PurchaseOrder”>
  <xs:sequence>
    <xs:element name=”items” type=”ItemType”
                maxOccurs=”unbounded”/>
        <xs:annotation><xs:appinfo>
           <jaxb:property collectionType=”indexed”/>
        ....

<!-- Download and use JAXB samples -->
inline-customize - Inline JAXB customizations sample
external-customize - External JAXB customizations sample
```

Java Based Classes → XML Schema

Classes

```
// no arg constructor - rely on default
public class PurchaseOrder { ... }

@XmlType(factoryClass=POFactory.class,factoryMethod="getPO")
public class PurchaseOrder {...}

// Use XmlJavaTypeAdapter for non JavaBean classes
public class POAdapter extends XmlAdapter<..> {...}
@XmlJavaTypeAdapter(POAdapter.class)
public class PurchaseOrder {...}

// map class hierarchy to schema complex type hierarchy
public class I18NPurchaseOrder extends PurchaseOrder {...}
```

Java Based Classes → XML Schema

Properties/Fields

```
// public fields or properties - rely on default
public class PurchaseOrder {
    public String name ;
    public List<Item> getItems() {...}
}
@XmlAccessorType(XmlAccessType.FIELD)
public class PurchaseOrder {
    private String name; // persist field
}
@XmlAccessorType(XmlAccessType.FIELD)
package com.acme
// inherit from package level annotation @XmlAccessorType
public class PurchaseOrder {...}
```

Java Based Classes →XML Schema

Types

```
public class PurchaseOrder {
    public XMLGregorianCalendar f1;   // JAXP 1.3 datatype
    public List<Items> f2;            // generics
    public Items[]     f3;            // arrays
    public int         f4;            // primitives
    public Integer     f5;            // wrapper classes
    public USState     f6;            // enum type
    public Address     f7;            // type reference
    ...                               // ...
}
public enum USState { NH, MA }
public Address {...}
public USAddress extends Address {...}
public InternationalAddress extends Address {...}
```

Java Based Classes → XML Schema

XML Schema constructs

```
@XmlSchema(xmlns = {XmlNs(prefix = "po",
                          namespaceURI="http://www.example.com/myPO1")})
package primer.po


@XmlRootElement
public class PurchaseOrder {
    @XmlAttribute public XMLGregorianCalendar shipDate;

    // wrap collection: <wrap> <a> 1 </a> <a> 2 </a> </wrap>
    @XmlElementWrapper(name="wrap") public List<Integer> a;

    // marshal a list of values not elements : <a> 1 2 3 </a>
    @XmlList public List<Integer> a;
}
```

JAXB 2.0 Runtime

Marshal/Unmarshal

```
public class MyApplication {

    // typically useful for XML Schema to Java based classes
    JAXBContext jc = JAXBContext.newInstance("com.foo");

    // typically useful for Java based classes To XML Schema
    JAXBContext jc = JAXBContext.newInstance(classes-to-bind);

    Marshaller m = jc.createMarshaller(...);
    m.marshal(..);

    Unmarshaller u = jc.createUnmarshaller(...);
    Foo f = (Foo) u.unmarshal(...);
}
```

JAXB 2.0 Runtime

Validation

```
public class MyApplication {
    // Use JAXP 1.3 API for validation
    javax.xml.validation.Schema schema;
    ...
    Unmarshaller u = jc.createUnmarshaller(...);
    u.setSchema(schema);
    Marshaller m = jc.createMarshaller(...);
    m.setSchema(schema);

    // Validate against versioned schemas
    u.setSchema(schema_v1); // validate against version 1
    u.setSchema(schema_v2); // or against version 2

    // customize validation event handling
    ValidationEventHandler myhandler;
    m.setEventHandler(myhandler);
    u.setEventHandler(myHanlder);
}
```

JAXB 2.0 Runtime

Event Callbacks

```
// Class defined for access to non public final fields/methods
public class PurchaseOrder {

    // e.g. Initialize default value - element defaulting
    void beforeUnmarshal(Unmarshaller, Object parent) {...}
    void afterUnmarshal(Unmarshaller, Object parent) {...}

    void beforeMarshal(Marshaller) {...}
    void afterMarshal(Marshaller) {...}
    ...
}

// External listeners for centralized processing
Unmarshaller.setListener(UnmarshallerListener);
Marshaller.setListener(MarshallerListener);

// or both - they are ordered
```

JAXB JAX-WS Integration

Databinding support

- 100 % XML Schema Support
- Attachment Support (MTOM/XOP)
- Start from WSDL/XML Schema
  - Use <jaxb:globalBindings serializable ../> to pass JAXB objects as Enterprise JavaBeans™ (EJB™) specification method parameters
  - Mix JAXB and JAX-WS customizations
- Start from Java classes
  - Use JAXB annotated classes

Agenda

JAXB 2.0 Overview

Annotation Driven Programming Model

Java and .NET Databinding Interoperability

JAXB NetBeans Module

JAXB on java.net

JAXB Performance

Summary

Project Tango—Background

- Goal
  - Deliver next generation Web Services technologies enabling first class interoperability between Sun's Java Products and Windows Operating environments supporting WCF[1]
- Implementation strategy
  - Build on JAX-WS and JAXB technologies
  - Work closely with Microsoft and perform product level testing
  - Build an active Open Source community centered around the GlassFish community

[1] Windows Communications Foundation aka “Indigo”

Java Technology and .NET Databinding Interoperability

WCF client -> Java web service

- Start from Java
  - Use JAXB and JAX-WS annotated classes
  - Create service WSDL
  - Deploy Java web service
- Start from WSDL/XML Schema
  - Use simple XML Schema constructs and concepts
  - Generate Java web service
  - Deploy Java web service

Java Technology and .NET Databinding Interoperability

Java client -> WCF web service

- Start from WCF service WSDL
  - Generate Java client
  - Use JAXB and JAX-WS annotated classes

Agenda

JAXB 2.0 Overview

Annotation Driven Programming Model

Java and .NET Databinding Interoperability

JAXB NetBean Modules

JAXB on java.net

JAXB Performance

Summary

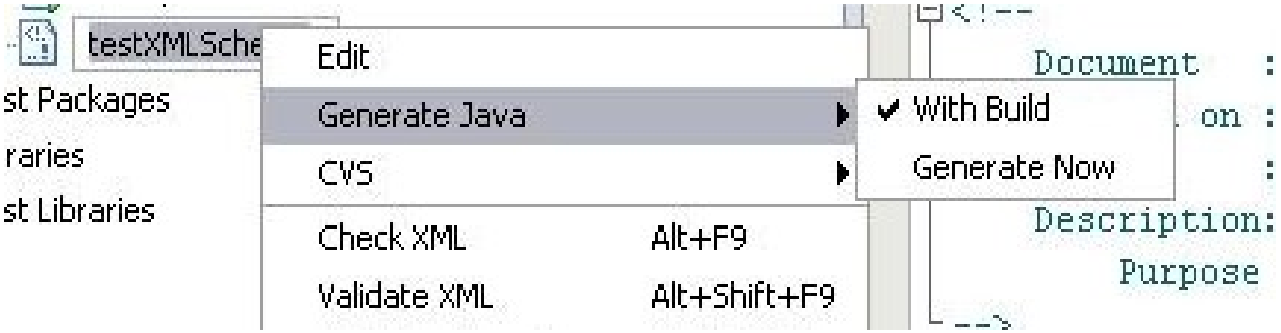
JAXB: Easier with NetBeans!

- Schema to Java with one click!
- Java to Schema with one click!

JAXB NetBeans Module

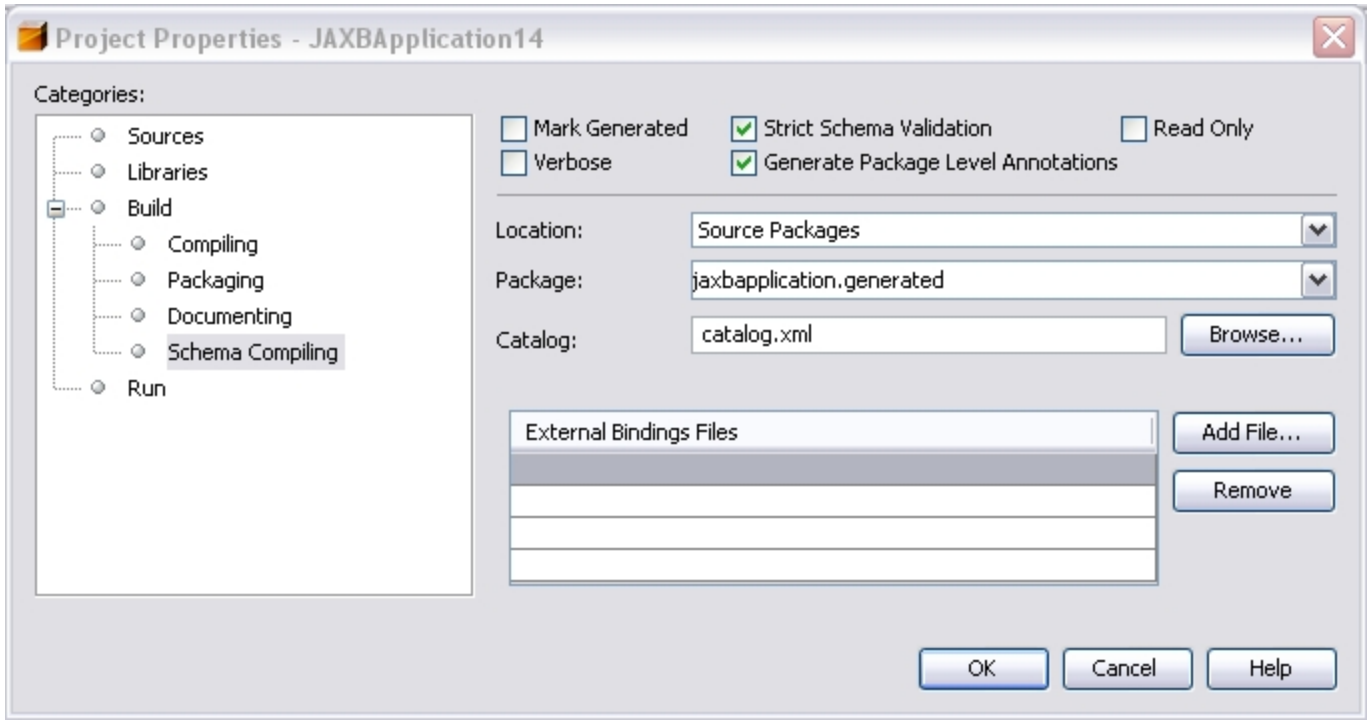
- Integrated with standard Java project module
- Customizers for XJC and SchemaGen options
- No need to write or edit ant scripts

JAXB NetBeans Module Features



- Modes
  - Generate with build—mark files to be generated in-line
    - Multiple files
    - Ant task generated for each file
    - (Clean) →Generate → Compile
  - Generate Now—single or multiple Java files
    - Single or Multiple files
    - No ant task created

JAXB NetBeans—Schema Compiling



DEMO

JAXB NetBeans Module

JAXB Module for NetBeans

- Available this summer
- Open source
- http://xml.netbeans.org

Agenda

JAXB 2.0 Overview

Annotation Driven Programming Model

Java and .NET Databinding Interoperability

JAXB NetBeans Module

JAXB on java.net

JAXB Performance

Summary

Plugins for XJC

Going beyond customizations

- What is it?
  - Java code that participates in the code generation
  - Can change the generated code
- Two aspects
  - Using existing plugins
  - Writing it

Plugins for XJC

Covering spectrum

- One end: customizations
  - Maybe JAXB doesn't offer customizations you want
  - Maybe you need to put 100 customizations
  - You can only do within what we thought you'll do
- The other end: plugin
  - Harder, but more flexible
  - Won't work in other JAXB impls
- To use one, download a jar and run:

```
xjc -cp plugin.jar -help
```

Plugin Showcase: Fluent API

By Hanson Char

- Build a tree like StringBuffer

```
// BEFORE
USAddress address = new USAddress();
address.setName(name);
address.setStreet(street);
address.setCity(city);
address.setState(state);
```

```
// AFTER
USAddress address = new USAddress()
                    .withName(name)
                    .withStreet(street)
                    .withCity(city)
                    .withState(state);
```

Plugin Showcase: CamelCase-Always

By Nicola Fagnani

- Change XML-to-Java name conversion rules
- Very small (2 classes) so you can change it

XML name	NAME	SSN_CODE
Class name		
spec	NAME	SSNCODE
this plugin	Name	SsnCode
Method name		
spec	getNAME	getSSNCODE
this plugin	getName	getSsnCode

Using XJC Plugins

- And there are more
  - Value constructor plugin
  - Hash code plugin
  - Cloneable plugin
  - EJB3 plugin (AKA hyperjaxb3)
  - ...
- Find out more at http://jaxb2-commons.dev.java.net/

Writing Plugins

- Extend the Plugin class
- Put a service file in META-INF/services/com.sun.tools.xjc.Plugin
- Package everything in a JAR

- Plugins can
  - Replace/modify some compilation strategies
  - Add more code to the generated code

Simpler and Better Binding Mode

If you want it to “just work”

- Adjust default binding for simple use
  - Makes aggressive, more restricted assumptions
  - Based on user feedback in the late stage
- Benefits
  - Optimize away JAXBElement more aggressively
  - Generate better set of properties
  - Reduce need for customizations
  - Still runs on any JAXB 2.0 runtime
  - Repeated properties get plural forms like “getChildren”

Simpler and Better Binding Mode

Example

```
// Translation: at least one must be present
<element name="foo" type="foo"/>

<complexType name="foo">
  <choice>
    <sequence>
      <element name="A" type="..." />
      <element name="B" type="..." minOccurs="0" />
      <element name="C" type="..." minOccurs="0" />
    <sequence>
      <element name="B" type="..." />
      <element name="C" type="..." minOccurs="0" />
    <element name="C" type="..." />
```

Simpler and Better Binding Mode

Default Binding

```
class ObjectFactory {
  JAXBElement<Foo> createFoo(Foo);
  JAXBElement<String> createFooA(String);
  JAXBElement<String> createFooB(String);
  JAXBElement<String> createFooC(String);
}

class Foo {
  List<JAXBElement<String>> getContent();
}
```

Simpler and Better Binding Mode

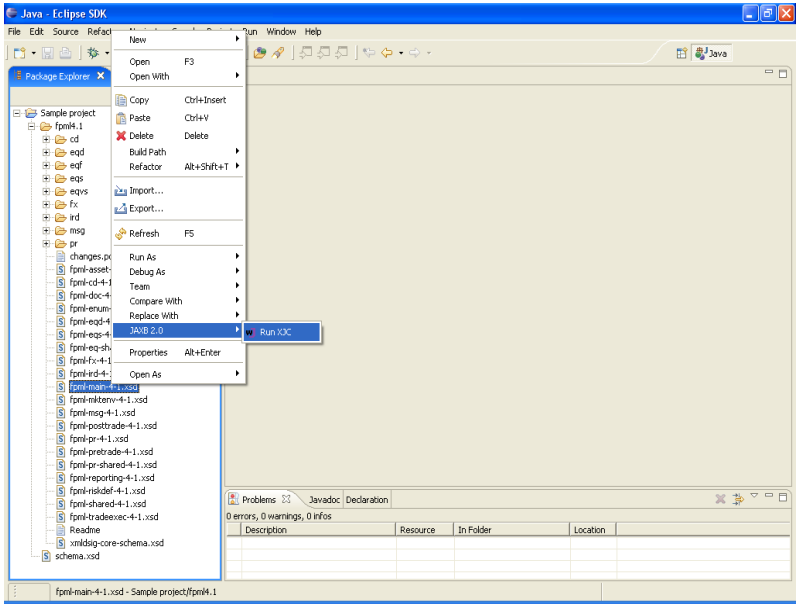
But with this mode…

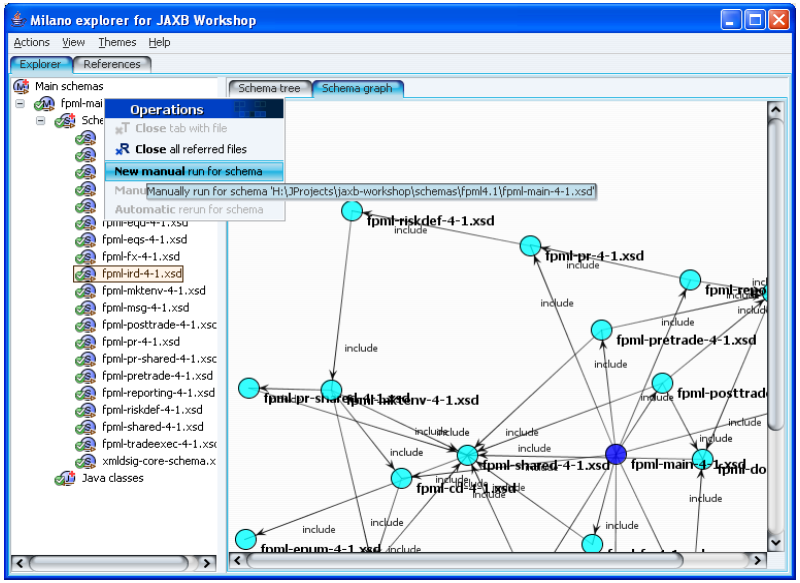
```
$ xjc -extension foo.xsd simpleMode.schemalet
```

```
class Foo {
  String getA/setA;
  String getB/setB;
  String getC/setC;
}
```

Tools Around XJC

- IDE plugins
  - NetBeans, Eclipse, and IntelliJ IDEA (Kirill)
- Build tool plugins
  - Maven1 (SourceForge)
  - Maven2 (Jonathan)
  - Ant
- JAXB Workshop (Kirill)





Agenda

JAXB 2.0 Overview

Annotation Driven Programming Model

Java and .NET Databinding Interoperability

JAXB NetBeans Modules

JAXB on java.net

JAXB Performance

Summary

Performance

What we did

Optimized marshaller for UTF-8 (220% faster)

Unmarshaller performance (17% faster)

Reduced jar file size (18% smaller)

Performance

What you can do

- Reuse and share JAXBContext

```
Foo load(File f) {
    u = JAXBContext.newInstance(...).createUnmarshaller();
    return (Foo)u.unmarshal(f);
}
```

```
static final JAXBContext context =
                        JAXBContext.newInstance(...);
Foo load(File f) {
   return (Foo)context.createUnmarshaller().unmarshal(f);
}
```

Summary

- Supports 100 % XML Schema
- Supports Java classes to XML Schema
- There are great tools to make JAXB even easier
- JAXB RI is fast

For More Information

Links

http://www.jcp.org/en/jsr/detail?id=222

http://jaxb.dev.java.net

http://glassfish.dev.java.net

Q&A

Deep Dive Into Java™ Architecture for XML Binding 2.0

Sekhar Vajjhala

Hans Hrasna

Kohsuke Kawaguchi

TS-1607