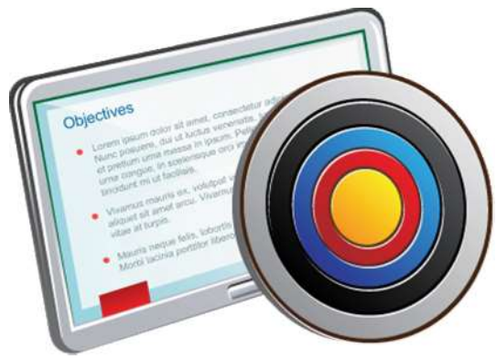
XML Parsing Using JAXB

Objectives

After completing this lesson, you should be able to do the following:

- List the different Java XML APIs
- Explain the benefits of JAXB
- Unmarshall XML data with JAXB
- Marshall XML data with JAXB
- Compile XML Schema to Java
- Generate XML Schema from Java classes
- Apply JAXB binding annotations
- Create external binding configuration files



Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- Lesson 1: Introduction to Web Services
- Lesson 2: Creating XML Documents
- **Lesson 3: Processing XML with JAXB** — **You are here!**
- Lesson 4: SOAP Web Services Overview
- Lesson 5: Creating JAX-WS Clients

Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- Lesson 6: Exploring REST Services
- Lesson 7: Creating REST Clients
- Lesson 8: Bottom Up JAX Web Services
- Lesson 9: Top Down JAX Web Services
- Lesson 10: Implementing JAX RS Web Services

Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- Lesson 11: Web Service Error Handling
- Lesson 12: Java EE Security and Securing JAX WS

Java XML APIs

There are several ways to process XML in Java.

- Java API for XML Processing (JAXP):
  - SAX – An event-based parser framework. Developers create event handlers that fire when reading a document.
  - DOM – An object model. An XML document is converted into a tree of objects comprised of types such as org.w3c.dom.Element.
- Streaming API for XML (StAX): Implements a pull-parser API
- Java Architecture for XML Binding (JAXB): Binds an XML document to a tree of objects similar to DOM. Unlike DOM, the object types are custom types.

JAXB: Overview

- Allows reading and writing of XML documents
- Is an object-based model of XML document structure similar to DOM
- Binds developer-supplied object types to XML with no need to read XML into memory and then insert the data into domain objects
- Is an annotation-based configuration of Java to XML mapping
- Supports XML Schema to Java class generation and Java class to XML Schema generation
- Is used by JAX-WS and JAX-RS
  - Return values and method parameters that are JAXB-annotated class types are automatically converted for you.

XML and Java Class Comparison

Given the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <name>Matt</name>
</person>
```

An equivalent Java class is:

```
@XmlRootElement
public class Person {
    private String name;
    public String getName() {/*…*/}
    public void setName(String name) {/*…*/}
}
```

Reading XML with JAXB

Reading XML is accomplished by using a JAXBContext, one or more JAXB annotated classes, and an Unmarshaller.

```
try {
    JAXBContext jc =
        JAXBContext.newInstance(Person.class);
    Unmarshaller u = jc.createUnmarshaller();
    InputStream in =
        new FileInputStream("src/simple-read.xml");
    Person p = (Person)u.unmarshal(in);
    System.out.println("Name: " + p.getName());
} catch (JAXBException | IOException ex) {
    ex.printStackTrace();
}
```

Writing XML with JAXB

Writing XML is accomplished by using a JAXBContext, one or more JAXB annotated classes, and a Marshaller.

```
try {
    Person p = new Person();
    p.setName("tom");
    JAXBContext jc =
        JAXBContext.newInstance(Person.class);
    Marshaller m = jc.createMarshaller();
    OutputStream out =
      new FileOutputStream("src/simple-write.xml");
    m.marshal(p, out);
} catch (JAXBException | IOException ex) {
    ex.printStackTrace(); }
```

xjc

`xjc` is the JAXB Binding Compiler. `xjc` takes XML Schema input and produces a Java package containing Java classes.

```
xjc -p ou.xjc person.xsd
```

Produces:

```
ou/
  xjc/
    ObjectFactory.java
    package-info.java
    Person.java
```

Java Package

Factory methods JAXB bound classes

A Java class for each complex type

A place for annotations that apply to all classes in a package

schemagen

`schemagen` is the JAXB Java to XML Schema generator. `schemagen` takes one or more Java files as input and produces XML Schemas.

```
schemagen ou\simple\Person.java
```

Produces `schema1.xsd` and `ou\simple\Person.class`

JAXBContext

The `JAXBContext` class is the entry point into the JAXB API. It is used to obtain:

- An Unmarshaller that can read XML
- A Marshaller that can write XML

`JAXBContext` can be passed a `var-args` class listing.

```
JAXBContext jc = JAXBContext.newInstance(Person.class);
```

`JAXBContext` can be passed a string of packages names.

```
JAXBContext jc = JAXBContext.newInstance("ou.schema");
```

Errors and Validation

JAXB does not perform strict validation checking by default.

- When reading XML, unexpected elements and attributes that are not mapped to Java elements are ignored.
- When reading XML, malformed XML will cause a `javax.xml.bind.UnmarshalException` when calling unmarshall.
- If you want to keep track of validation failures, you can attach a ValidationEventHandler to the Unmarshaller.

```
Unmarshaller u = jc.createUnmarshaller();
ValidationEventCollector vec =
    new ValidationEventCollector();
u.setEventHandler(vec);
```

XmlRootElement

The `@XmlRootElement` annotation is used to indicate that a class is used as a global (root) XML element.

```
@XmlRootElement(name="human")
public class Person { /* … */
```

Corresponds to an XML Schema of:

```
<xs:element name="human" type="person"/>
<xs:complexType name="person">
  <!-- ... -->
</xs:complexType>
```

XmlType

The `@XmlType` annotation is used to:

- Specify the name of the `complextType`
- Specify the order of child elements

```
@XmlRootElement(name="human")
@XmlType(name="individual",
         propOrder={"name", "address"})
public class Person { /* … */ }
```

XmlAccessorType

The `@XmlAccessorType` annotation on a class controls which members are bound to XML. The default value is:

```
@XmlAccessorType(XmlAccessType.PUBLIC_MEMBER)
```

Possible values are:

- `PUBLIC_MEMBER` – All public fields and public getter/setter method pairs are bound to XML elements.
- `FIELD` – All fields, unless static or transient, are bound to XML elements.
- `PROPERTY` – All getter/setter method pairs are bound to XML elements.
- `NONE` – No members are bound to XML elements.

XmlElement

The `@XmlElement` annotation is used to control binding of class members to XML.

```
@XmlRootElement(name="human")
@XmlAccessorType(XmlAccessType.NONE)
public class Person {
    @XmlElement(name="first-name", required=true)
     private String name;
```

- Map the field to XML even though the XML accessor type is `NONE`.
- The XML element name will be first-name instead of name.
- The `minOccurs` value is left at the default value of 1 instead of adding `minOccurs="0"`.

XmlAttribute

The `@XmlAttribute` annotation maps a class member to an XML attribute.

```
@XmlRootElement()
public class Person {
    @XmlAttribute
    public String name;
    public String address;
}
```

Corresponds to the following XML structure:

```
<person name="matt">
    <address>221B Baker Street</address>
</person>
```

XmlValue

The `@XmlValue` annotation maps a class member to simple content within a complex type or a simple type where possible.

```
@XmlRootElement()
public class Person {
    @XmlAttribute
    public String name;
    @XmlValue
    public String address;
}
```

- Corresponds to the following XML structure:

```
<person name="matt">221B Baker Street</person>
```

Java enums can be mapped to XML enumerated values by using the `@XmlEnum` annotation.

```
@XmlType
@XmlEnum
public enum ProjectState {
    @XmlEnumValue("0")
    LATE,
    @XmlEnumValue("1")
    REALLY_LATE;
}
```

Java identifiers cannot start with a number but XML enumerated values can.

XmlElements

An `@XmlElements` annotation is used to map a Java member to an XML Schema choice structure.

```
@XmlElements(value = {
    @XmlElement(name = "pie",
    type = Pie.class,
    required=true),
    @XmlElement(name = "ice-cream",
    type = IceCream.class,
    required=true)
})
public Object obj;
```

XML Schema Inline Binding Customization

When running `xjc`, you can customize the generated classes.

```
<xs:schema version="1.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
           …
           jaxb:version="2.1">
<xs:complexType name="personType">
   <xs:annotation>
        <xs:appinfo>
            <jaxb:class name="Person"/>
        </xs:appinfo>
    </xs:annotation>
    <!-- ... -->
</xs:complexType>
```

The resulting class would normally be named `PersonType`.

NetBeans JAXB Support

- NetBeans supports the Ant xjc task through an XML Binding file type located in the XML new file category.
- There is no support for `schemagen`. You may run it on the command line, modify your `build.xml`, or generate the schema programmatically.

```
JAXBContext context = JAXBContext.newInstance("mypackage");
context.generateSchema(new SchemaOutputResolver() {
  public Result createOutput(String namespaceUri,
                             String fileName) throws IOException {
    return new StreamResult(new File("src/myschema.xsd"));
  }
});
```

JAXB performs XML schema validation by default.

a. True

b. False

The default accessor type used by JAXB to obtain the state of an object is:

a. None – class element must be annotated with a JAXB annotation if they are to be read
b. Fields – all fields regardless of access level are read
c. Properties – getter and setter methods are used
d. Public members – public fields and properties are used

Resources

Topic	Website
Java API for XML Processing (JAXP)	http://docs.oracle.com/javase/tutorial/jaxp/index.html
JAXB Homepage	http://jaxb.java.net/
JSR 222: JavaTM Architecture for XML Binding (JAXB) 2.0	http://jcp.org/en/jsr/detail?id=222

Summary

In this lesson, you should have learned how to:

- List the different Java XML APIs
- Explain the benefits of JAXB
- Unmarshall XML data with JAXB
- Marshall XML data with JAXB
- Compile XML Schema to Java
- Generate XML Schema from Java classes
- Apply JAXB binding annotations
- Create external binding configuration files

Practice 3: Overview

This practice covers the following topics:

- Creating Java Classes From XML Schema
- Creating XML Schemas From JAXB Annotated Classes