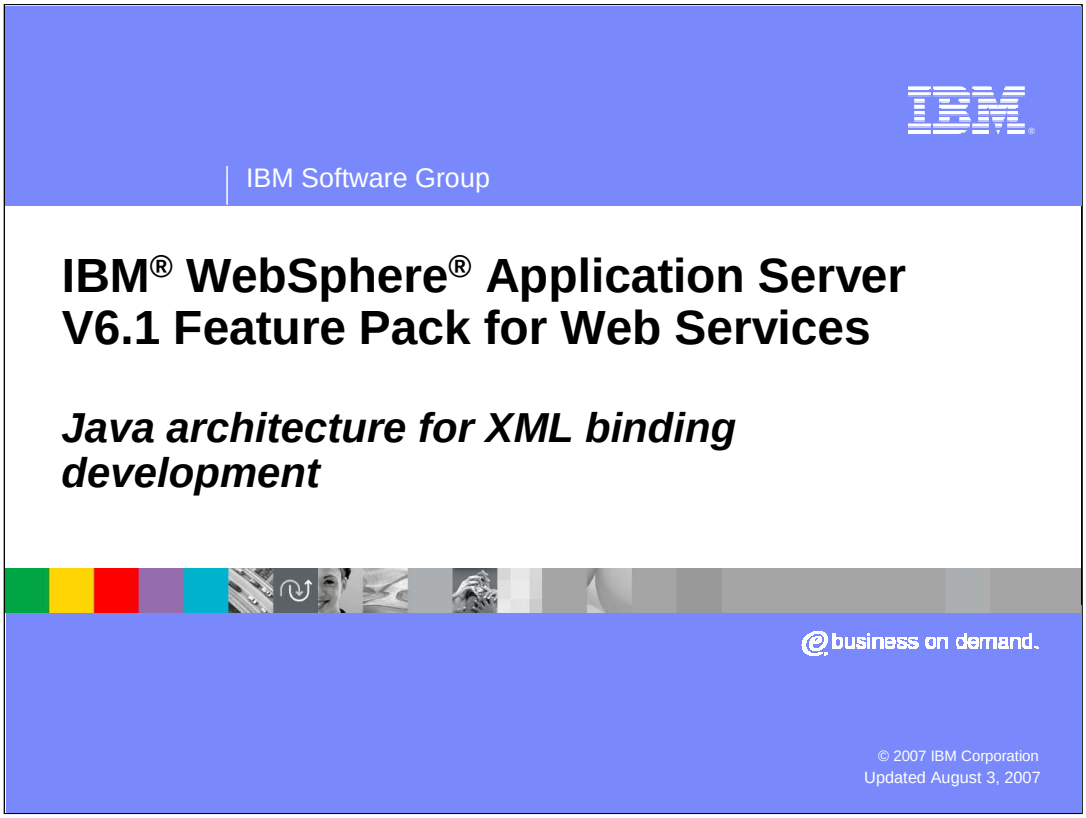


This presentation will provide an understanding of the JAXB programming model supported by the IBM WebSphere Application Server V6.1 Feature Pack for Web Services and the tools used to develop JAXB applications.

Agenda

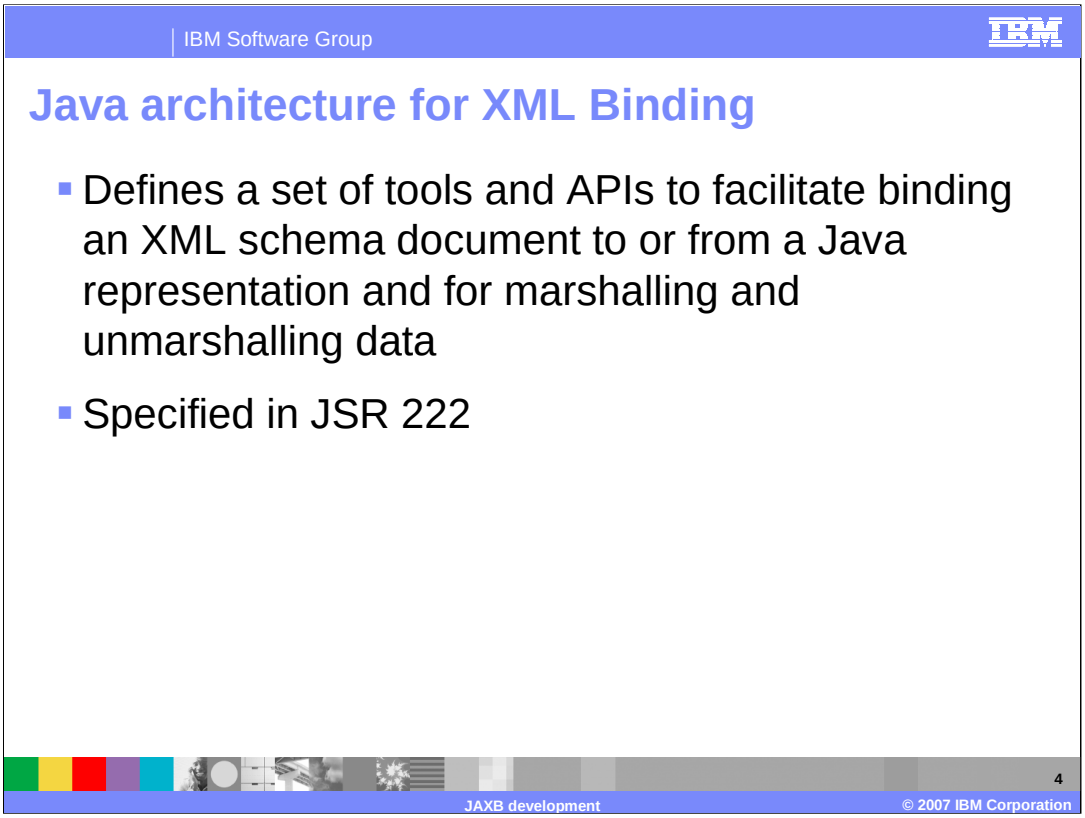
- Java architecture for XML binding (JAXB)
- The binding compiler
- The schema generator
- Tool limitations

This presentation begins with an explanation of the Java architecture for XML binding, or JAXB. Following that will be sections describing the binding compiler and the schema generator, two tools that are used for working with JAXB. Last is a section explaining some specific limitations in the tools.

Section

Java architecture for XML binding

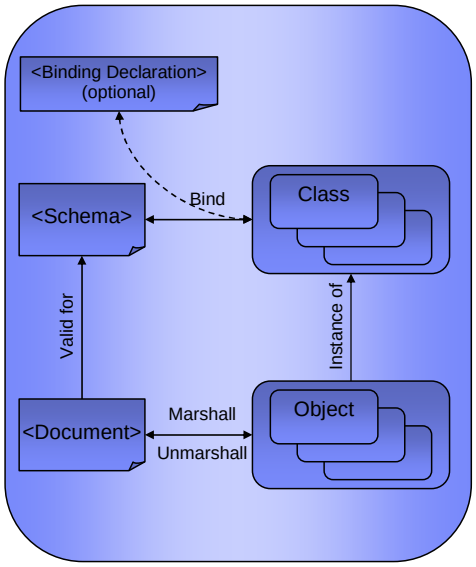
This section will provide an overview of the Java architecture for XML binding.



JAXB, specified by JSR 222, defines a set of tools and APIs to facilitate binding an XML schema to or from a Java representation and for marshalling and unmarshalling data. Formerly, this information was contained in the JAX-RPC specification and was specific to Web services. JAXB is used by Web services for binding, but JAXB can also be used by any Java application that works with XML. JAXB is much more flexible than keeping the binding rules specific to the JAX-RPC specification, and is one of the main differences in the JAX-WS programming model, defining many different binding rules.

JAXB

- Annotating XML schema
  - <class>
  - <method>
- MTOM mappings supported
  - Image
  - MIME DataHandler



The JAXB specification defines rules for binding XML schema documents to Java classes. This allows instance documents that conform to a schema to be marshalled into Java objects based on those Java classes. The feature pack for Web services provides tools for generating binding classes based on an XML schema; this is called a binding compiler. It also provides tools for creating an XML schema based on properly annotated binding classes. JAXB also defines mappings for working with the SOAP message transmission optimization mechanism, or MTOM.

JAXB mapping

```
@XmlType
public class Trade {
    @XmlElement(
        name="tickerSymbol")
    public String symbol;
    @XmlAttribute
    int getQuantity() {...}
    void setQuantity() {...}
}
```

```
<xs:complexType name="trade">
    <xs:sequence>
        <xs:element
            name="tickerSymbol"
            type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="quantity"
            type="xs:int"/>
</xs:complexType>
```


As an example, this slide shows the JAXB binding that would exist between the Java class named “Trade” and the XML element with name “trade”. It is worthwhile to look at additional bindings between Java and XML but is beyond the scope of this presentation. The details of what Java types are bound to which XML types is an important aspect to consider when developing applications using JAXB. The specification should be consulted for the specific mappings.

Section

The binding compiler

This section will provide an overview of the JAXB binding compiler, otherwise known as XJC.

The binding compiler

- The binding compiler is “xjc” and is normally invoked from script – xjc.sh or xjc.bat
- Input to xjc is:
  - XML schema documents
  - XML schema documents within WSDL documents
- Output is a set of files that are the Java classes that will map to parts of an XML document and ObjectFactory.java
  - ObjectFactory is an alternate way to create java Objects, and provides a way to create or instantiate element types

The JAXB binding compiler is named XJC and is normally invoked from a shell script; either xjc.sh or xjc.bat at the command line. It can also be used through IBM tools, such as the application server toolkit or Rational Application Developer. For more information, see the tool information in the Feature Pack for Web Services. XJC accepts one or more XML schemas as input and outputs a set of Java files that are the Java classes that will map to parts of an XML document and ObjectFactory.java, a factory to create objects.

xjc example

- xjc has the following command line format
  xjc [-options ...] <schema file/URL/dir>
- Invoking xjc with the ”-help” option will display usage information
- Some of the more commonly used options are:

```
-d <dir>            :  generated files will go into this directory
-p <pkg>            :  specifies the target package
-classpath <arg>  :  specify where to find user class files
-verbose             :  be extra verbose
-quiet               :  suppress compiler output
-help                :  display this help message
-version             :  display version information
```

- If the schema to use is named testSchema.xsd, use the following command line:
  xjc –verbose testSchema.xsd

This information shows how to use the XJC command. Use the -help option to display command syntax information on the command line. An example is given with a sample test schema.

Section

The schema generator

This section will provide an overview of the JAXB schema generator or the schemagen tool.

The schema generator

- The schema generator is “schemagen”
  - invoked from script – schemagen.sh or schemagen.bat
  - Java source files are used as input
- Command line options are available
  - For most situations the default values are fine
- Output is an XML schema document for each namespace referenced in the Java classes

The JAXB schema generator is named “schemagen” and is normally invoked as a shell script; either schemagen.sh or schemagen.bat at the command line. It can also be used through IBM tools; for more information see the tool information in the Feature Pack for Web services. The schemagen script accepts one or more annotated Java source files schemas as input and outputs an XML schema for each namespace referenced in the Java classes.

Example of schemagen

- schemagen has the following command line format

  schemagen [-options ...] <java files>

- Invoking schemagen with the ”-help” option will display usage information
- Some of the more commonly options are:

  -d <dir> : generated files will go into this directory
  -cp <path> : specify where to find user class files
  -classpath <path> : specify where to find user class files
  -version : display version information

- If the Java classes to use are named Obj1.java and Obj2.java, use the following command line:

  schemagen.bat Obj1.java Obj2.java

The information on this slide shows you how to use the schemagen command.

Use the -help option to display usage information on the command line. An example is given with sample test files.

Section

Tool limitations

This section will explain some limitations to be aware of when working with the JAXB tools.

Tool limitations

- Name collisions for generated artifacts are not handled automatically
  - JAXB provides you with JAXB customizations to handle any naming collisions
  - You can use XML schema customization by using JAXB binding files, or within the schema itself, or in Java classes, through JAXB class annotations
- XJC usage only officially supports usage of XSD files
  - Using WSDL under XJC is experimental and unsupported in the Feature Pack for Web Services

Many common problems related to JAXB are due to naming conflicts within the XML schemas. The tools, XJS and SCHEMAGEN, will not automatically handle naming collisions in the XML types and elements; you must handle this when defining the schema. JAXB defines specific customizations that can be used to avoid naming collisions. The XJC tool only supports XSD, or schema definition files, as input. Providing a WSDL document to XJC, while possible, is experimental and not supported in the Feature Pack for Web Services.

Section

Summary

This section will summarize the information provided.

Summary

- The JAXB specification provides a framework for:
  - The *unmarshalling* of an XML instance document into a content tree of interrelated Java objects; this may include validation of the document
  - The *marshalling* of such *content trees* back into XML instance documents
  - The *binding* between an XML schema document representation and *content tree*
- Main tools are a schema compiler and a schema generator
- Lets you bypass the complexities of XML and work with Java objects

The JAXB specification defines how XML instance documents can be unmarshalled into Java objects, and how Java objects can be marshaled into XML instance documents based on binding rules. This lets you work directly with Java objects in their applications and ignore many of the complexities of XML. JAXB is used by JAX-WS based Web services to describe the binding rules for those services and the SOAP messages that they send. The main tools provided for working with JAXB are the schema compiler, XJC, and the schema generator, SCHEMAGEN.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject= Feedback about WASv61_WSFP_JAXB_Development.ppt

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.