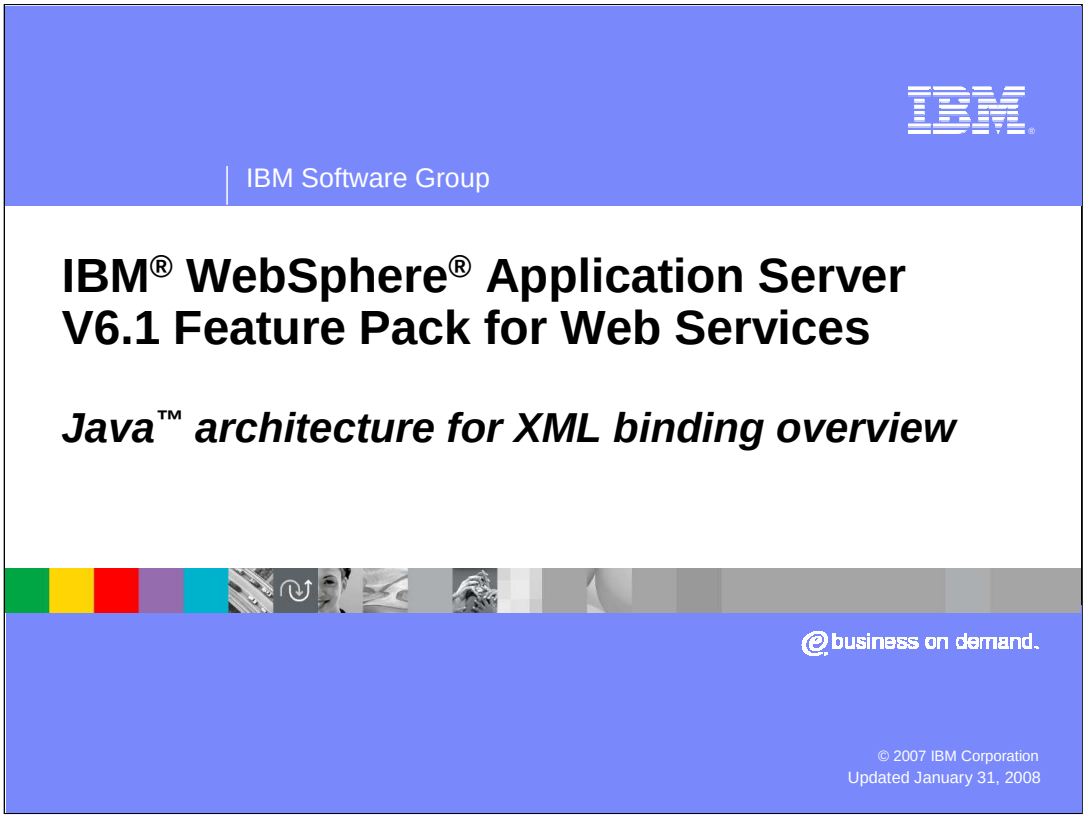
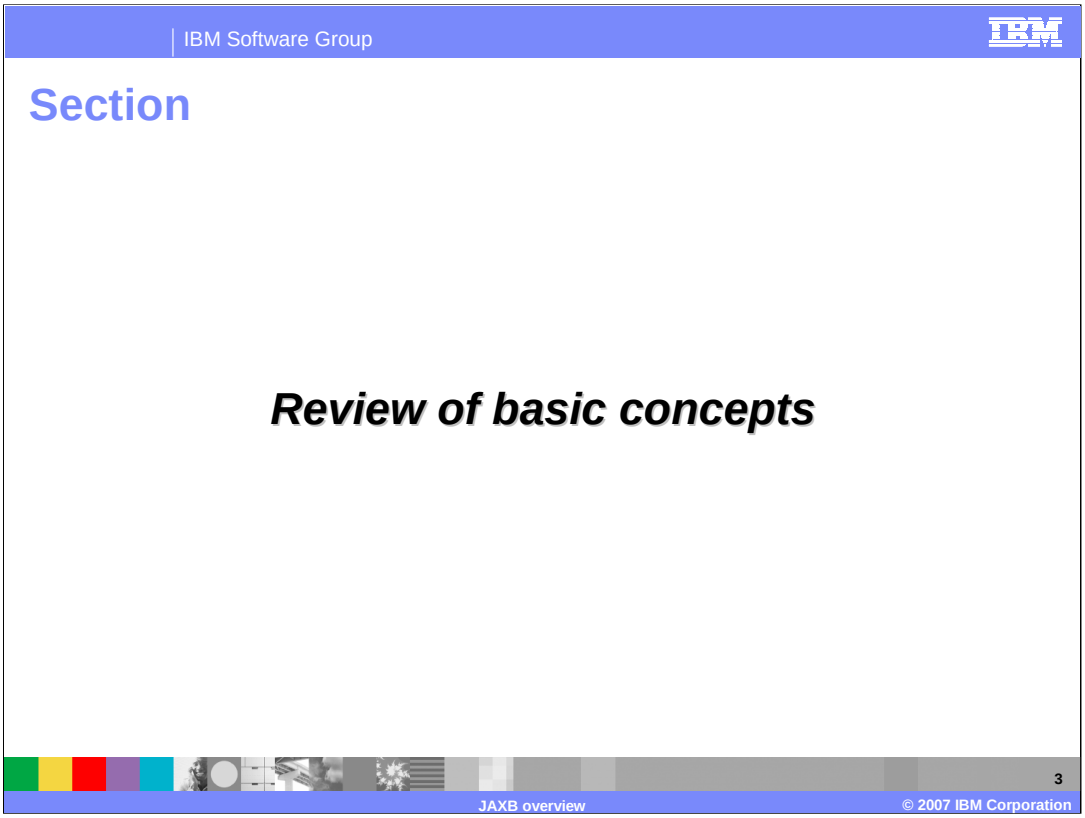


This presentation covers the JAXB programming model supported by the IBM WebSphere Application Server V6.1 Feature Pack for Web Services.

Agenda

- Review of basic concepts
  - XML schema document
  - Binding
  - Marshalling
  - Unmarshalling
- Java architecture for XML binding (JAXB)

This presentation begins with a review of some basic concepts related to working with XML and Web services, such as describing XML schema and instance documents, and the concepts behind turning XML into Java. Following that is a discussion of the Java architecture for XML binding, or JAXB.

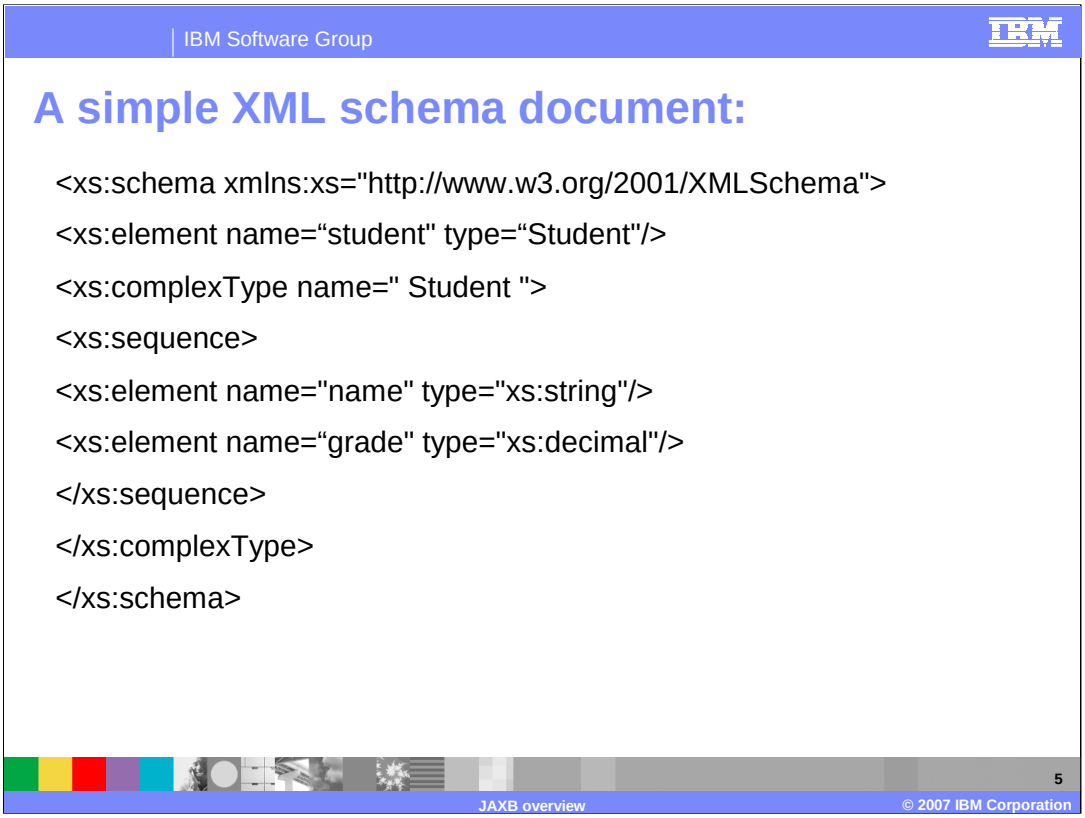


This section explains some of the basic concepts behind how Java Web services work with XML using JAXB.

XML schema document

- An XML schema defines the allowable content and structure of an XML instance document
- An XML schema document is similar to a Java class definition:
  - an XML instance document may declare that it conforms to a specific XML schema document
  - a Java object conforms to a specific Java class
- Schema documents have largely replaced an earlier mechanism known as DTD (document type definition)
- XML schema documents are typically stored in files with ".xsd" extension
- There are tools that create an XML schema document from an XML instance document

An XML schema defines the allowable content and structure of an XML document. Schemas have largely replaced an earlier mechanism known as a Document Type Definition (DTD) for this purpose. This has occurred because most developers find working with schema documents to be easier since they also use XML to store data. An XML schema can be compared to a Java class definition and an instance document that conforms to that schema is similar to a Java object that is an instance or instantiation, of that Java class. XML schemas are typically associated with file name extensions of ".xsd".



An XML schema is, itself, an XML document. This is one of the reasons that schemas have largely replaced DTDs, which were not XML documents. An example of an XML schema is shown in this slide, notice how it describes the form of the data and the elements needed to create an instance of this document.

A conforming XML instance document

```
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="student.xsd">
 <name>Darby</name>
<grade>4</grade>
</student>
```


Here is an example of an instance document that conforms to the schema from the last slide. A “conforming” XML document references a schema and can be validated against that schema – that is, the content and structure of the XML document can be compared to what is allowed by the schema. Notice how the instance document provides actual data for elements specified in the schema.

XML to Java binding

- Defines a correspondence between an XML schema document and a collection of Java classes
- Programmer works with Java classes and objects instead of having to deal with complexities of XML
- Bindings are typically created with a tool called a “binding compiler”

It is simpler for programmers to work with Java classes and objects rather than dealing with the complexities of XML directly. To facilitate this, the notion of a “binding” is introduced. A binding is a correspondence between an XML schema and one or more Java classes. Bindings are created with a tool called a “binding compiler” and can be used to map XML documents to and from instances of Java classes.

Marshalling and unmarshalling

- Marshalling is the process of turning one or more Java objects into an XML instance document
  - This process was previously called “serializing”
- In the unmarshalling process you begin with a valid XML instance document and convert it to Java objects
  - This process was previously called “deserializing”

Marshalling, also called serializing, is the process of converting one or more Java objects into an XML document. Unmarshalling is the reverse of this process, turning XML documents into Java. This is often done for a Web service application by the runtime, allowing a developer to write a Web service that works with Java types, but communicates using XML.

Section

Java architecture for XML binding

This section provides an overview of the Java Architecture for XML Binding.

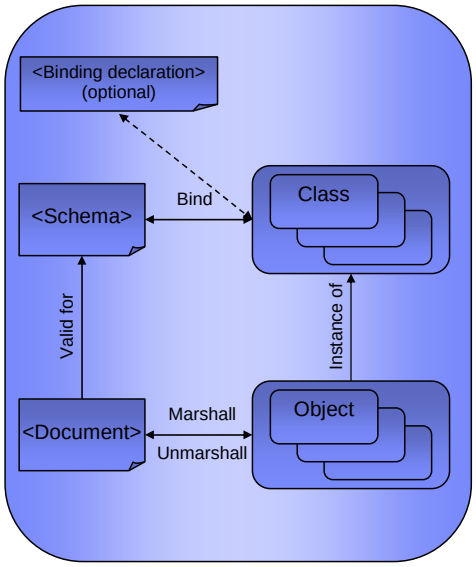
Java architecture for XML Binding

- Defines a set of tools and APIs to facilitate binding an XML schema document to or from a Java representation and for marshalling and unmarshalling data
- Specified in JSR 222

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 key 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. 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 exists 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; consult the specification for the specific mappings.

Section

Summary

This section summarizes 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*
- Allows the programmer to bypass the complexities of XML and, instead, 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 allows developers to work directly with Java objects 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 they send.

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_Overview.ppt

This module is also available in PDF format at: ../WASv61_WSFP_JAXB_Overview.pdf

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.