



JAXB 2.0

Sang Shin

www.javapassion.com

Java Technology Architect



**UNLOCK
OPPORTUNITY**

What will you open?

SUN TECH DAYS 2006-2007

A Worldwide Developer Conference

Agenda

- Improvements of JAXB 2.0 over JAXB 1.0
- Architecture
- Default data type mapping
- Customized mapping: Schema to Java

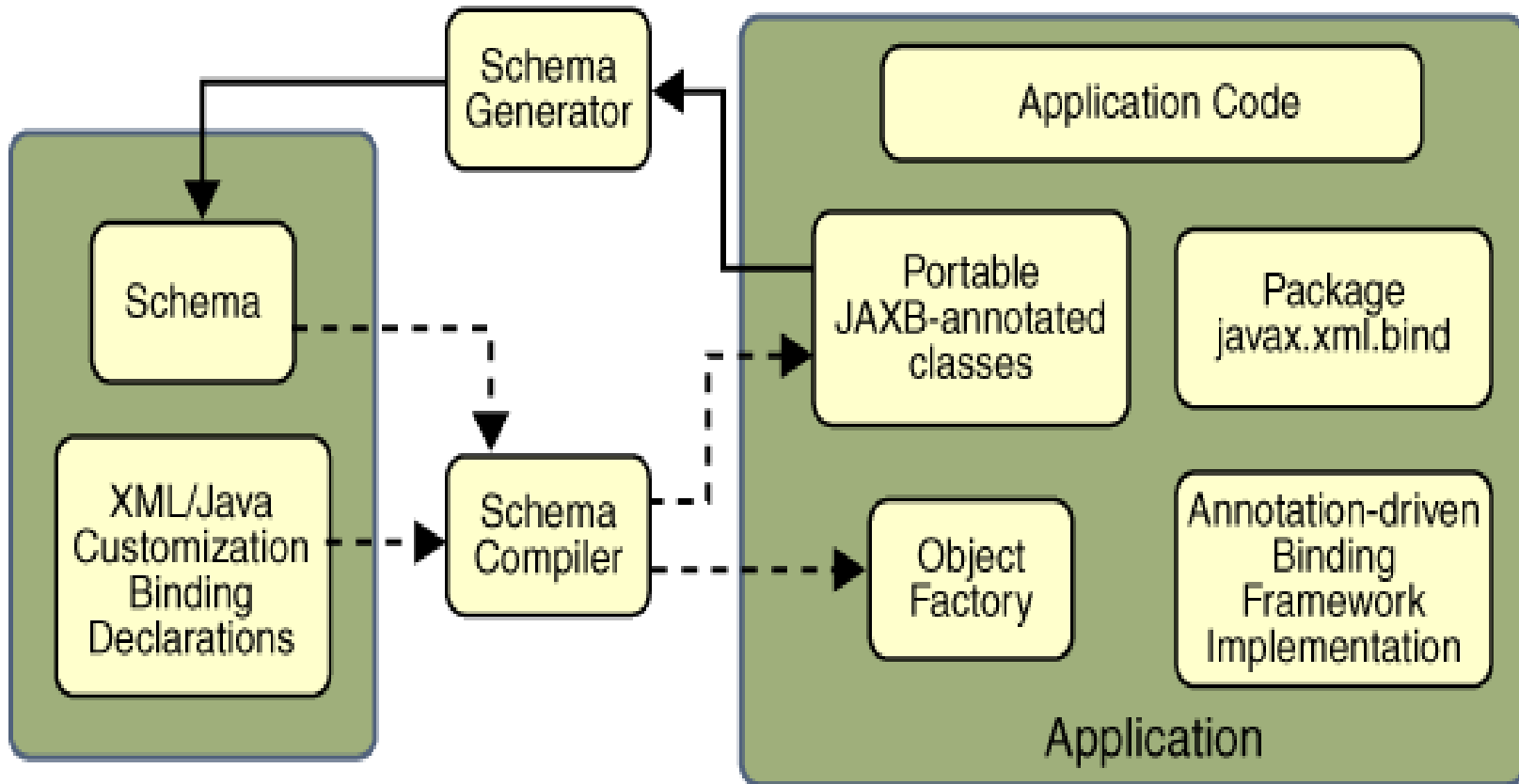
JAXB 2.0 Improvements over JAXB 1.0

JAXB 2.0 Improvements over JAXB 1.0

- Support for all W3C XML Schema features. (JAXB 1.0 did not specify bindings for some of the W3C XML Schema features.)
- Support for binding Java-to-XML Schema, with the addition of the *javax.xml.bind.annotation* package to control this binding. (JAXB 1.0 specified the mapping of XML Schema-to-Java, but not Java-to-XML Schema.)
- A significant reduction in the number of generated schema-derived classes.
- Additional validation capabilities through the JAXP 1.3 validation APIs.
- Smaller runtime libraries.

Architecture

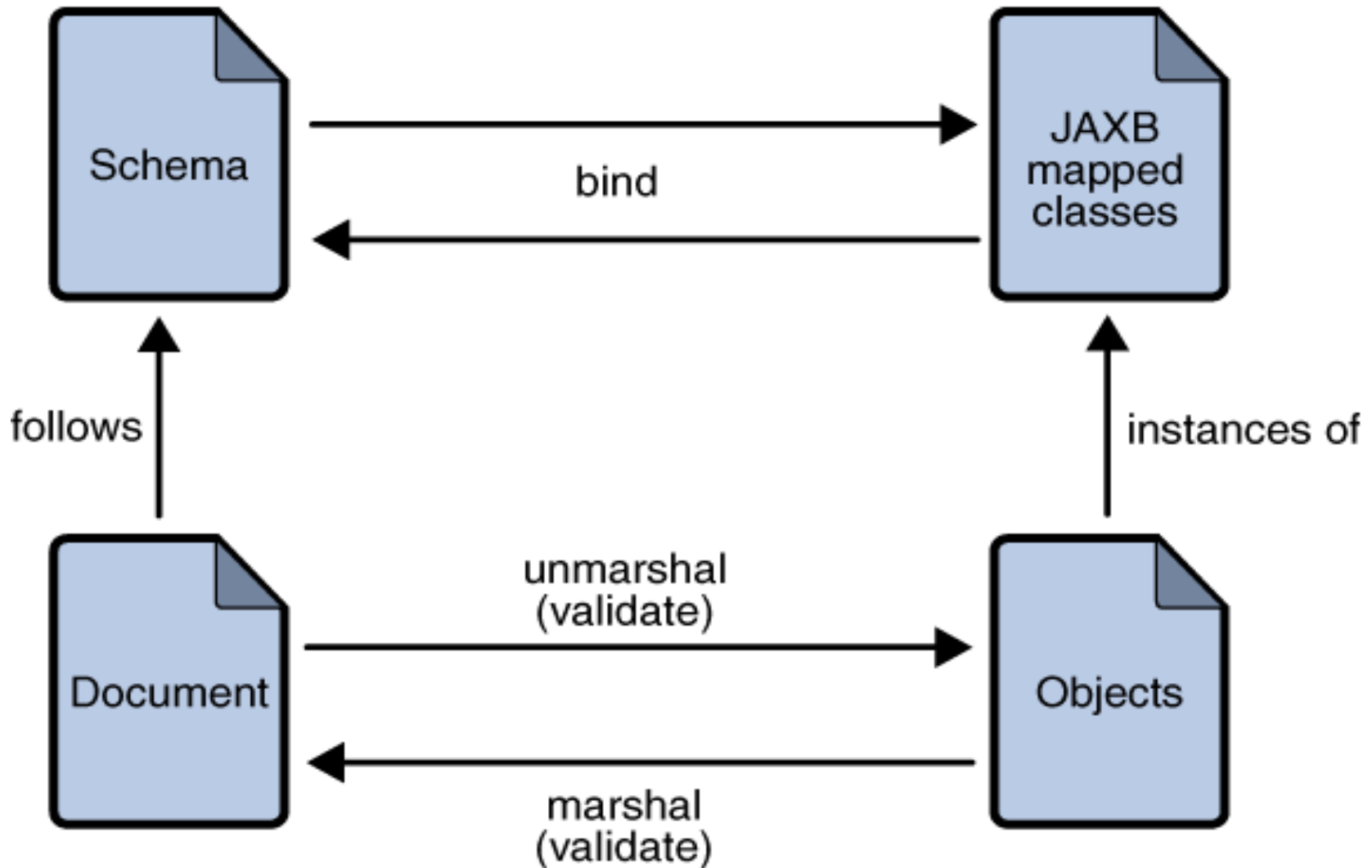
Architecture



Architectural Components

- Schema compiler
 - > Binds a source schema to a set of schema-derived program elements. The binding is described by an XML-based binding language.
- Schema generator
 - > Maps a set of existing program elements to a derived schema. The mapping is described by program annotations.
- Binding runtime framework
 - > Provides unmarshalling (reading) and marshalling (writing) operations for accessing, manipulating, and validating XML content using either schema-derived or existing program elements

JAXB Binding Process



Unmarshalling

- Provides a client application the ability to convert XML data into JAXB-derived Java objects.

Marshalling

- Provides a client application the ability to convert a JAXB-derived Java object tree back into XML data.
 - > By default, the Marshaller uses UTF-8 encoding when generating XML data.
 - > Client applications are not required to validate the Java content tree before marshalling
 - > There is also no requirement that the Java content tree be valid with respect to its original schema to marshal it back into XML data.

Validation

- Validation is the process of verifying that an XML document meets all the constraints expressed in the schema
- JAXB 1.0 provided validation at unmarshal time and also enabled on-demand validation on a JAXB content tree
- JAXB 2.0 only allows validation at unmarshal and marshal time.
- A web service processing model is to be lax in reading in data and strict on writing it out. To meet that model, validation was added to marshal time so one could confirm that they did not invalidate the XML document when modifying the document in JAXB form

Default Data Type Mapping

Schema to Java

- The Java language provides a richer set of data type than XML schema

XML Schema Type	Java Data Type
xsd:string	java.lang.String
xsd:integer	java.math.BigInteger
xsd:int	int
xsd:long	long
xsd:short	short
xsd:decimal	java.math.BigDecimal
xsd:float	float
xsd:double	double
xsd:boolean	boolean
xsd:byte	byte
xsd:QName	javax.xml.namespace.QName
xsd:dateTime	javax.xml.datatype.XMLGregorianCalendar
xsd:base64Binary	byte []
xsd:hexBinary	byte []

Schema to Java

<code>xsd:unsignedInt</code>	<code>long</code>
<code>xsd:unsignedShort</code>	<code>int</code>
<code>xsd:unsignedByte</code>	<code>short</code>
<code>xsd:time</code>	<code>javax.xml.datatype.XMLGregorianCalendar</code>
<code>xsd:date</code>	<code>javax.xml.datatype.XMLGregorianCalendar</code>
<code>xsd:g</code>	<code>javax.xml.datatype.XMLGregorianCalendar</code>
<code>xsd:anySimpleType</code>	<code>java.lang.Object</code>
<code>xsd:anySimpleType</code>	<code>java.lang.String</code>
<code>xsd:duration</code>	<code>javax.xml.datatype.Duration</code>
<code>xsd:NOTATION</code>	<code>javax.xml.namespace.QName</code>

JAXBElement

- When XML element information can not be inferred by the derived Java representation of the XML content, a *JAXBElement* object is provided
- This object has methods for getting and setting the object name and object value.

Customized Mapping: Schema to Java

Schema-to-Java

- Two ways to customize an XML schema:
 - > As inline annotations in a source XML schema
 - > As declarations in an external binding customizations file that is passed to the JAXB binding compiler

Java to Schema: Annotations to Java Package

Annotation	Description and Default Setting
<p>@XmlSchema</p>	<p>Maps a package to an XML target namespace.</p> <p>Default Settings:</p> <pre>@XmlSchema (xmlns = {}, namespace = "", elementFormDefault = XmlNsForm.UNSET; attributeFormDefault = XmlNsForm.UNSET,)</pre>
<p>@XmlAccessorType</p>	<p>Controls default serialization of fields and properties.</p> <p>Default Settings:</p> <pre>@XmlAccessorType (value = AccessType.PUBLIC_MEMBER)</pre>
<p>@XmlAccessorOrder</p>	<p>Controls the default ordering of properties and fields mapped to XML elements.</p> <p>Default Settings:</p> <pre>@XmlAccessorOrder (value = AccessorOrder.UNDEFINED)</pre>
<p>@XmlSchemaType</p>	<p>Allows a customized mapping to a XML Schema built-in type.</p> <p>Default Settings:</p> <pre>@XmlSchemaType (namespace = "http://www.w3.org/2001/XMLSchema", type = DEFAULT.class)</pre>
<p>@XmlSchemaTypes</p>	<p>A container annotation for defining multiple @XmlSchemaType annotations.</p> <p>Default Settings: None</p>

JAXB Annotations to Java Class

Annotation	Description and Default Setting
<p>@XmlType</p>	<p>Maps a Java class to a schema type.</p> <p>Default Settings:</p> <pre>@XmlType (name = "##default", propOrder = {""}, namespace = "##default" , factoryClass = DEFAULT.class, factoryMethod = "")</pre>
<p>@XmlRootElement</p>	<p>Associates a global element with the schema type to which the class is mapped.</p> <p>Default Settings:</p> <pre>@XmlRootElement (name = "##default", namespace = "##default")</pre>

Thank you!