

# DEVELOPING WEB SERVICES WITH JAVA

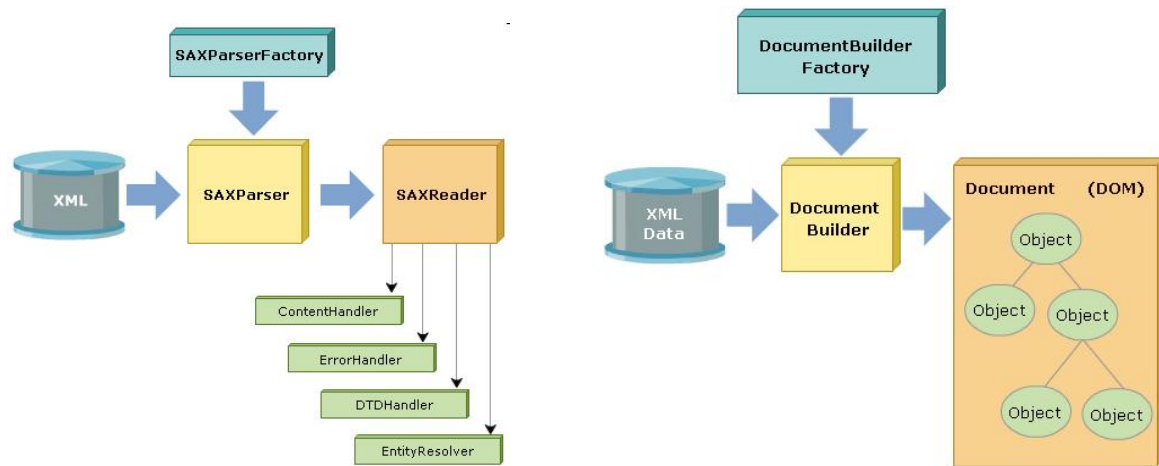
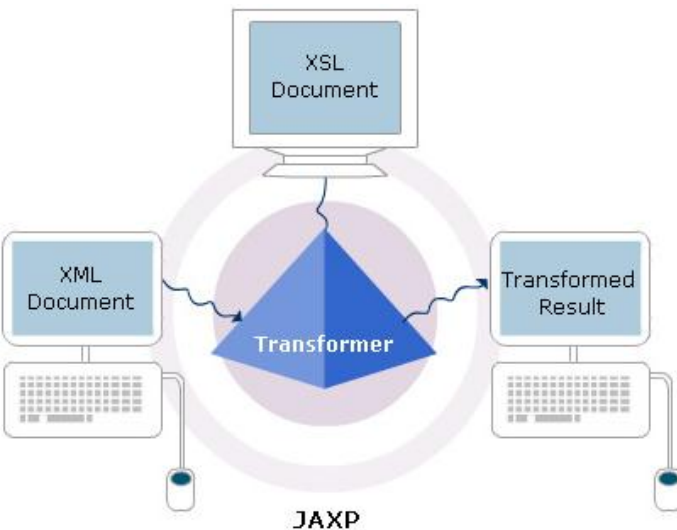
## **WEB SERVICES**

# CONTENTS

- JAXP
- JAX-RPC
- JAXR
- SAAJ
- JAXB
- Steps to consumes Web Services with Web application & Swing application using NetBeans
- Workshops
- Exercises

# JAXP

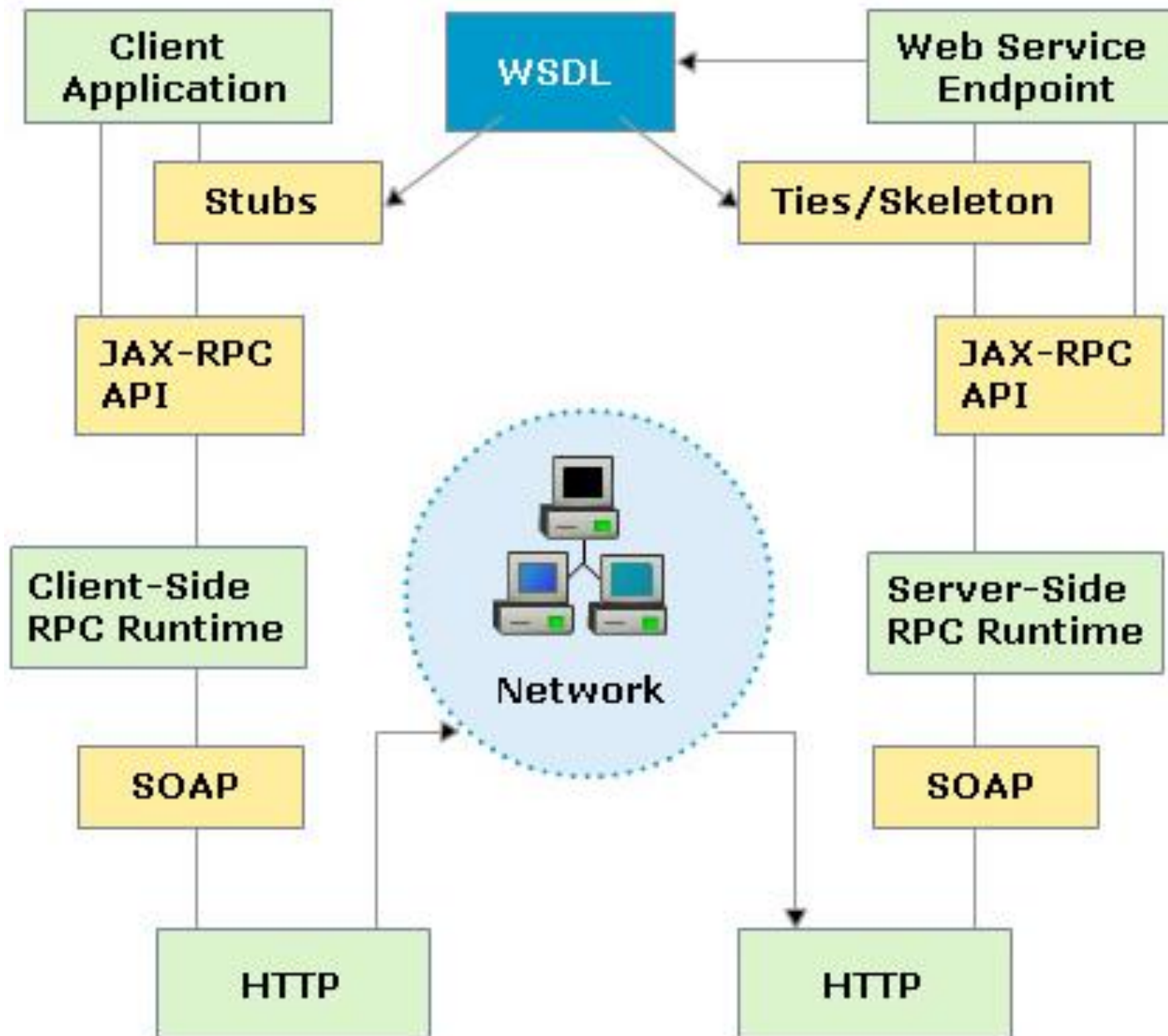
- XML parser is used to process/parse the XML documents that are exchanged among Web Services
- JAXP
  - Provides the capability of validating and parsing the XML document using applications written in Java programming language.
  - Provides a framework for using SAX2 and DOM2 standard Java APIs that read, write, and modify XML documents
  - Enables the applications to parse the XML documents with the help of SAX or DOM models, and it permits the transformation of XML documents from one format to another using XSLT engines.



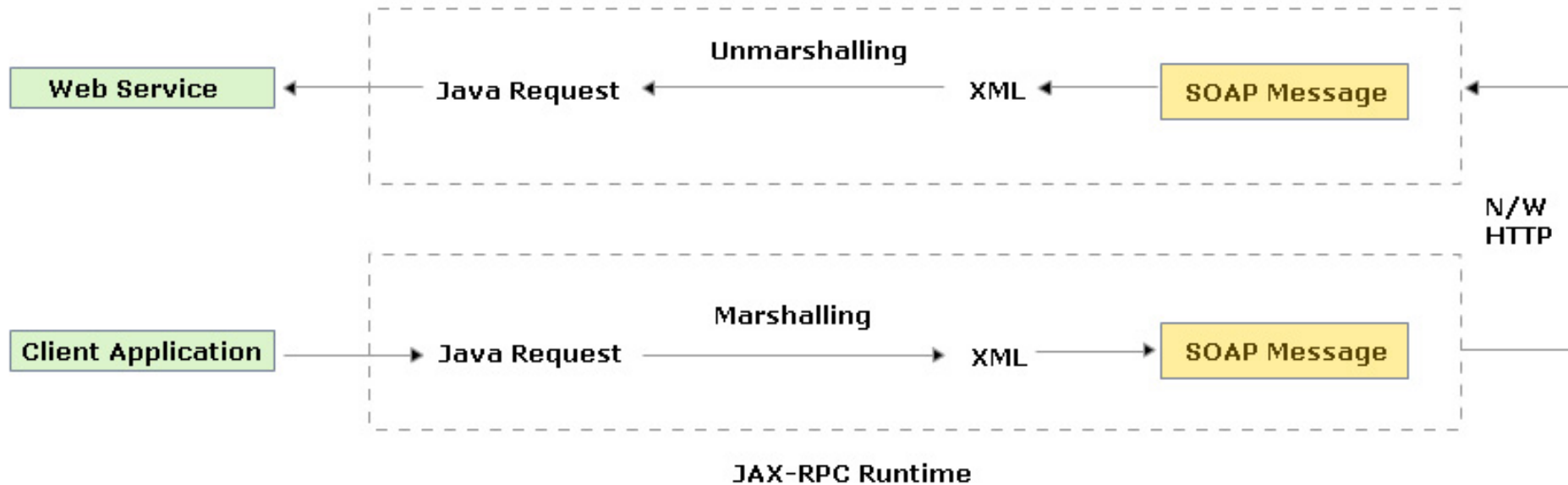
# ***JAX – RPC***

- Defines an API framework and runtime environment for the creation and execution of XML-based remote procedural calls.
- Hides all the complexities of the SOAP packaging and messaging by providing the required mapping between Java and XML/WSDL.
- Features
  - Enables interoperability with any SOAP based Web Services using WSDL.
  - Provides mapping of data types between XML and Java.
  - Invokes methods on the generated stubs.
  - Supports standard Internet protocol such as HTTP.
  - Provides portability of Service endpoints and service clients across JAX-RPC implementations
- Benefits
  - The creation of SOAP requests and responses are standardized.
  - The marshalling and unmarshalling of parameters and other runtime are standardized.
  - The developers' responsibility is reduced by providing the SOAP creation and marshalling/unmarshalling tasks in a library or a tool.

# *JAX – RPC (cont)*



# *CLIENT REQUEST TO JAX-RPC SERVICES*



- The client application's request passes through the client-side JAX-RPC runtime.
- The runtime maps the Java type request to XML and converts it into a corresponding SOAP form.
- The SOAP message is sent across the network to the server.
- On the server side, the JAX-RPC runtime receives the SOAP message. The server-side runtime applies the XML to Java mapping, and then maps the request to the corresponding Java method call, along with its parameters.
- A client application can make a request to Web Service by using pre-created static classes or by using classes or interfaces generated at runtime

# ***MODES OF OPERATION OF JAX – RPC SERVICES***

- There are 3 modes
  - **Synchronous request-response mode**
    - A client invokes a remote method from a thread which gets blocked, until a return value or exception is returned.
  - **One-way RPC mode**
    - A client invokes a remote method, and continues in its thread without blocking. No return value or exception is expected on this call.
  - **Non-blocking RPC invocation mode**
    - A client invokes a remote procedure and continues in its thread without blocking. Later, the client processes the remote method return call or by periodically requesting for the return value.

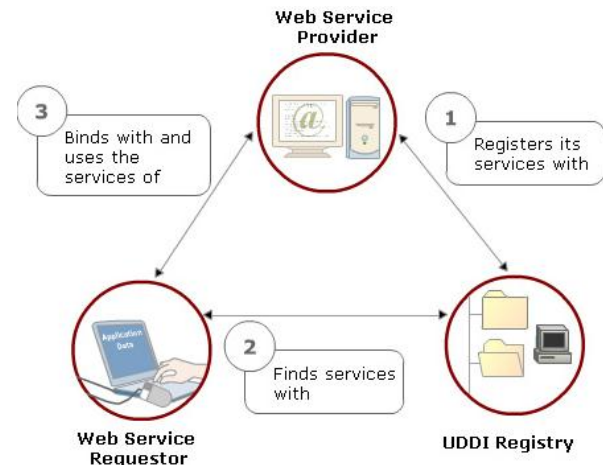
# REGISTRY

- Is a shared resource which is present in the form of a Web-based Service that makes dynamic business-to-business interactions easier to use
- Allows organizations to publish, search, and utilize Web Services
- Is a place where the service publishes its interfaces to enable clients to discover the service.
- Is a key component in Web Services architecture
- An XML-based registry provides capabilities that take the advantage of emerging e-business and Web Services standards.
- Requires an expanded information model and query capabilities in order to support more complex queries, and therefore make the search for Web Services as accurate as possible
- Technical component keep to ebXML and UDDI standard



# UDDI

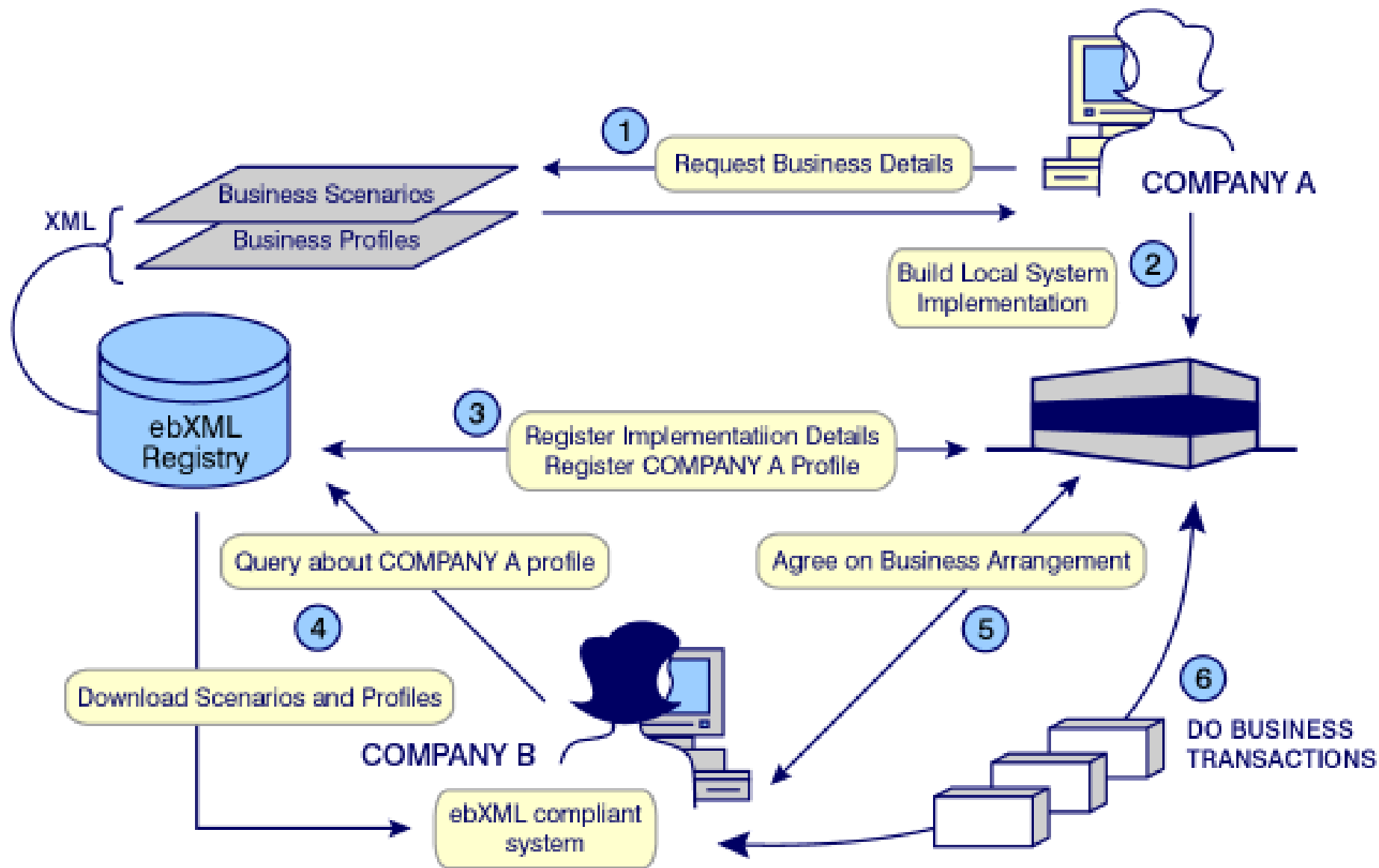
- Is briefly Universal Description, Discovery, and Integration
- Is an XML registry that allows remote clients to access, and search the Web Services.
- Specification provides a standard mechanism for storing information about organization and their Web Services.
- There are four core elements in the UDDI information model
  - **Business Information:** explained using the businessEntity element (a name, description, and contact about the organization), which represents a physical organization.
  - **Service Information:** explained using the businessService element, which groups together related services offered by an organization.
  - **Binding Information:** explained using the bindingTemplate element, which provides instruction on how to invoke a remote Web Service.
  - **Information about specifications for services:** explained using the tModel element, which contains information such as name, publishing organization, and URL pointers to the actual specification themselves.



# ebXML

- Is briefly the Electronic business XML
- Is a business-to-business XML framework that enables business to be conducted electronically over the Internet.
- Enables organizations to advertise and discover information about businesses.
- Is both a metadata registry as well as a repository that can hold arbitrary content. In contact, a Web Service description may be published in an ebXML Registry and repository is include all metadata as well as technical specifications and related artifacts.
- Stores information about Collaboration-Protocol Profile (CPP) and Collaboration-Protocol Agreement (CPA). The CPP is an XML document that contains information about a business. The CPA is also an XML document that describes the specific capabilities that two businesses have agreed to use in business collaboration
- The features in the ebXML information model:
  - Supports data validation feature improves integrity of registry data.
  - Supports packaging (or grouping) of related registry objects.
  - Supports both synchronous and asynchronous communication.
  - Provides digital-signature-based authentication, validation and authorization.

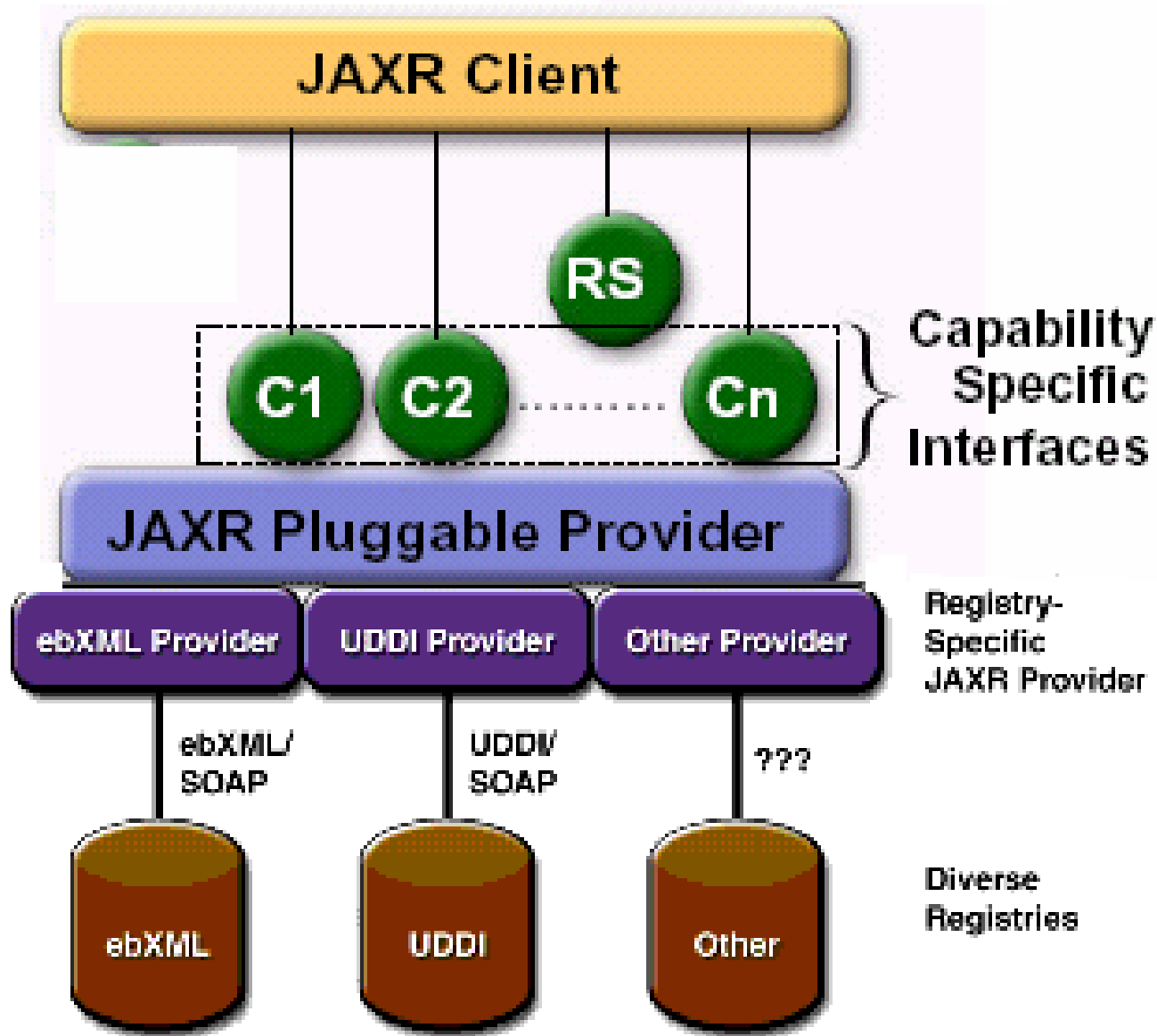
# *ebXML (cont)*



# JAXR

- Is briefly Java API for XML Registries
- Is also called as an abstract uniform Java API
- Provides a single set of APIs with has a flexible architecture to access a variety of XML registries, such as UDDI and the ebXML Registry.
- Simplifiers the process of publishing and searching for Web Service endpoints.
- JAXR information model provides a more intuitive and natural interface to most developers
- A JAXR client
  - Uses an implementation of the JAXR API provided by a JAXR provider to access business registries.
  - Invokes life-cycle and query management methods on the JAXR provider
- A JAXR Provider has two parts
  - Registry-specific JAXR provider: provides a registry-specific implementation of the API.
  - JAXR pluggable provider: implements the API features that are independent of the type of registry. The pluggable provider hides the details registry-specific provides from clients.

# ***JAXR ARCHITECTURE***



# ***JAXR ARCHITECTURE (cont)***

- There are 3 main parts
  - Registry Provider: provides an implement of a registry specification or standard
  - JAXR Provider:
    - Is implemented to access an existing registry provider.
    - Is a proxy for JAXR Client.
    - Implement the JAXR API (provides a simple API for the Java platform to access diverse registries that vary significantly in their capabilities and underlying information model)
      - javax.xml.registry.infomodel: define the types of objects that reside in registry and how they relate to each other such as Connection, RegistryService, and infomodel.
      - javax.xml.registry: define how those objects are submitted to the registry and managed. 02 interface is usually used as (Business)QueryManage or DeclarativeQueryManary, and (Business)LifeCycleManage
      - The JAXRException interface is provided to handle errors in using JAXR
      - The JAXR methods use Collection type which is dataType of parameter and returned value
  - JAXR Client: is a Java program that uses the JAXR API to access the services provided by a JAXR provider

# ***JAXR ARCHITECTURE (cont)***

- The other objects
  - JAXR Pluggable Provider: is a single abstract providing the Pluggable Connection Factory to support connection with JAXR Provider
  - Registry Specific implement the JAXR Provider, then JAXR Provider is plugged into Pluggable Provider
  - RegistryService Interface: is the principle interface implemented by a JAXR provider. RS provides the methods for getting objects, and the getCapabilityProfile method that allows the JAXR client access to the capability profile.
  - Capacity specific Interface: provide specific capabilities such as cycle management and query management.
  - Connection Interface: represent a client session with a registry provider using a JAXR provider (maintain dynamic state information for a JAXR provider)
  - JAXR API defines 02 capacity profiles: level 0 (Business Interface - use UDDI) và level 1 (Generic Interface - use ebXML registry)

# JAXR INTERFACES

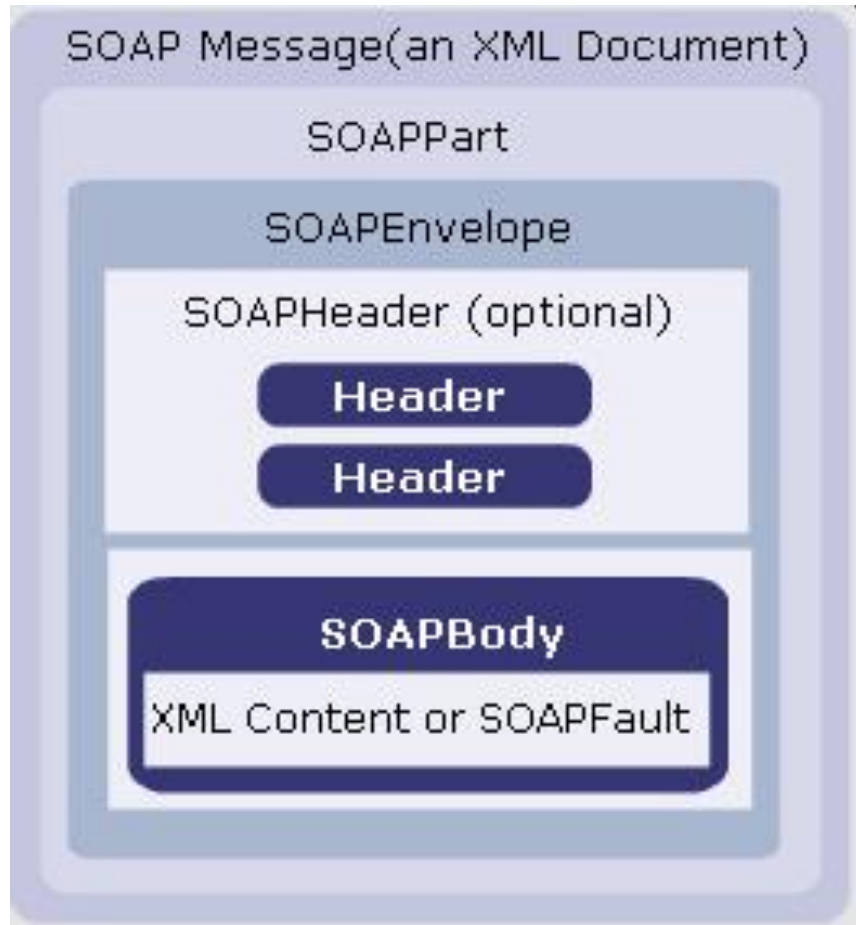
Interface	Description
Organization	- Are RegistryObjects that provide information on an organization that has been published to the underlying registry.
Service	- Are RegistryObjects that provide information on services offered by an Organization.
ServiceBinding	ServiceBinding are RegistryObjects that represent technical information on how to access a specific interface offered by a Service instance.
Concept	Concept instance represent an arbitrary notion, or concept. It can be virtually anything.
ClassificationScheme	ClassificationScheme instance represent a taxonomy that can be used to classify or categorize RegistryObjects instance.
ExternalLink	ExternalLink instances provide a link to content managed outside the registry using a URI.
ExternalIdentifier	ExternalIdentifier instance provide identification information to a RegistryObject.
Classification	Classification instances classify a RegistryObject instance using a classification scheme.
PostalAddress	PostalAddress instances provide address information for user and an Organization.



# SAAJ

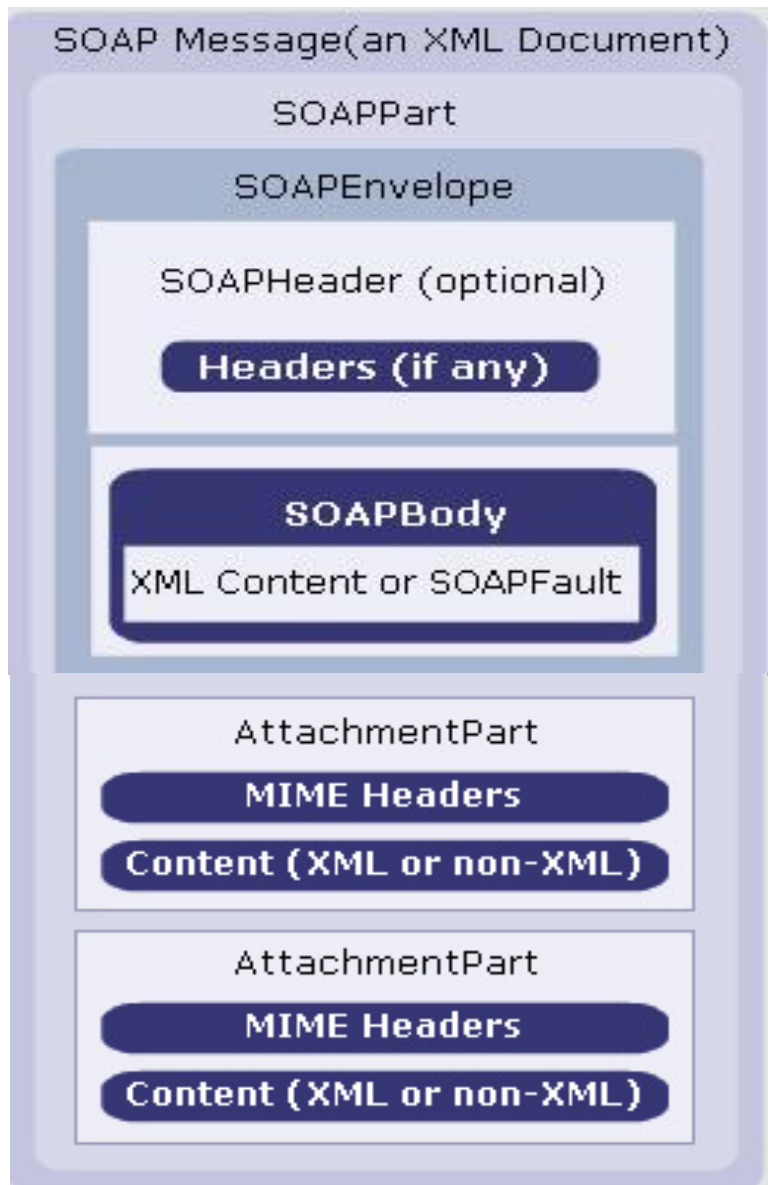
- Is briefly SOAP with Attachments API for Java
- Is an API that allows users to create and send SOAP messages with attachments by means of the javax.xml.soap package.
- SAAJ messages follow SOAP standards, which prescribe the format for messages and also specify some things that are required.
- SAAJ API allows to create XML messages that adapt to the SOAP 1.1 and WebService-I Basic Profile 1.0 specifications.
- There are two perspectives of SAAJ, namely Messages and Connections
- Simple Object Access Protocol (SOAP)
  - Provides a common message format for Web Services.
  - Enables developers to produce and consume messages conforming to the SOAP 1.1 specification and SOAP with Attachments note. Attachments may be in the form of complete XML documents, XML fragments, or MIME-type attachments.
  - Allows developers use it to write SOAP messaging applications directly instead of using JAX-RPC.
  - The two types of SOAP messages are those that have attachments and those without attachments

# ***SOAP MESSAGES WITHOUT ATTACHMENTS***



- The SAAJ API uses the `SOAPMessage` class for the SOAP messages, the `SOAPPart` class for the SOAP part and the `SOAPEnvelope` interface for the SOAP envelope.
- When a new `SOAPMessage` object is created, it has a `SOAPPart` object that contains a `SOAPEnvelope` object.
- The `SOAPEnvelope` object in turn automatically contains an empty `SOAPHeader` object followed by an empty `SOAPBody` object.
- The `SOAPHeader` object can include one more headers that contain metadata about the message, such as information about the sending and receiving parties.
- The `SOAPBody` object contains the message content.

# SOAP MESSAGES WITH ATTACHMENTS



- A SOAP message can have one or more attachment parts along with the SOAP part.
- The SOAP part must contain only XML content.
- The SAAJ API provides the AttachmentPart class to represent an attachment part of a SOAP message.
- A SOAPMessage object automatically contains a SOAPPart object and its required sub elements. Since the AttachmentPart Objects are optional, it has to be created and added manually.
- If a SOAPMessage object has one or more attachments, the MIME header of each AttachmentPart object indicates the type of data contained in it. It may also have additional MIME headers to identify it or to give its location. When a SOAPMessage object has one or more AttachmentPart objects, its SOAPPart object may or may not contain message content.

# SAAJ API

- The package `javax.xml.soap` provides the API for creating and building SOAP messages.
- Features
  - SAAJ allows two ways of exchanging messages
    - **Synchronous request-response messaging:** the client sends a message and then waits for the response.
    - **One-way asynchronous messaging:** the client sends a message and continues with its processing without waiting for a response.
  - SAAJ 1.2 provides some new features
    - Support for the Web Service-I Basic Profile 1.0, Document Object Model integration where the SAAJ APIs now extend Document Object Model API.
    - SOAPMessage properties are used for setting character set encoding and turning on the writing of an XML declaration at the start of the SOAP part of the message.

Class Name	Description
AttachmentPart	A single attachment to a SOAPMessage object.
MimeHeader	An object that stores a MIME header name and its value.
SOAPConnection	A point-to-point connection that a client can use for sending messages directly to a remote party.
SOAPConnectionFactory	A factory for creating SOAPConnection objects.
SOAPMessage	The root class for all SOAP messages.

# ***SAAJ (cont)***

- SOAP Connection

- In SAAJ API, the connection is represented by a SOAPConnection object, which goes from the one endpoint to another endpoint. Hence, this type of connection is called a point-to-point connection.
- The messages that are sent using the SAAJ API are called request-response messages. They are sent over a SOAPConnection object using the call() method by sending a request message and then blocking it, until it receives the response reply. The request parameter represents the message being sent, and the endpoint represents the location to which it is sent.
- The response received by the Web Service is a SOAPMessage object. The response is an acknowledgment that the update was received successfully.

- Comparison of SAAJ API with JAX-RPC API

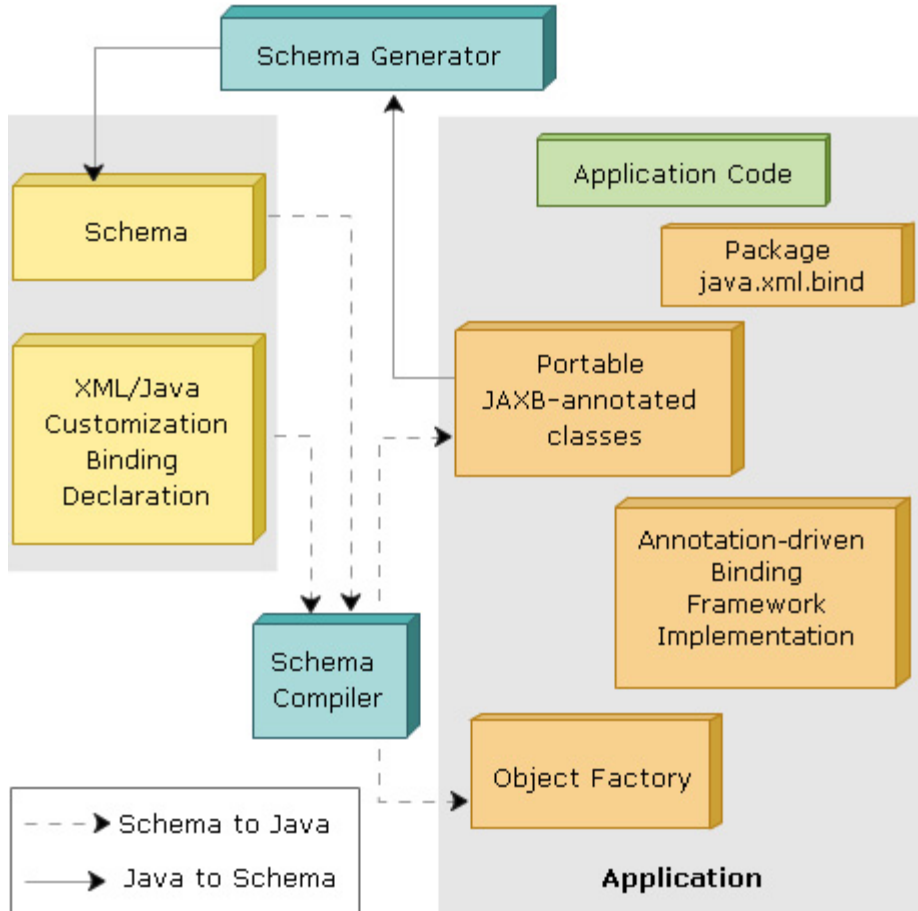
JAX-RPC API	SAAJ API
JAX-RPC is a high level API where the clients can make XML-based Remote Procedure calls over the internet.	SAAJ is a low-level API which supports JAX-RPC.
The JAX-RPC API widely uses the feature of interoperability.	SAAJ API is not used so widely.
JAX-RPC enables import and export of WSDL documents.	SAAJ enables the production and consumption of messages that conform to the SOAP 1.1 specifications.
JAX-RPC combines HTTP with the Java technology implementations of Secure Socket Layer and Transport Layer Security protocols for basic or mutual authentication.	This feature is not present in SAAJ.

# ***JAXB***

- Is briefly the Java Architecture for XML Binding
- Binds the XML schemas and Java representations in a fast and convenient way.
- Converts XML instance documents into Java content trees, and then converts Java content trees back into XML instance documents.
- Can make it easier to access XML documents from applications written in the Java programming language. To achieve this, first bind the schema for the XML document into a set of Java classes that represents the schema
- The JAXB API, defined in the `javax.xml.bind` package, consists of a set of interfaces through which client applications with code generated from a schema
- Provides a good quality XML data-binding facility for the J2EE platform.
- Limitations
  - JAXB requires a DTD and a subset of XML Schemas. Hence, it cannot be used to process generic XML, such as writing an XML editor or other tool.
  - Additional work is required to tell JAXB what kind of tree it should construct to simplify the application.
  - JAXB does not support the legal DTD constructs such as Internal subsets, NOTATIONs, ENTITY and ENTITIES, and Enumerated NOTATION types.
- Comprises of the following three components:
  - Binding compiler: creates Java classes from a given schema.
  - Binding framework: provides runtime services such as marshalling, unmarshalling, and validation to be performed on the contents classes.
  - Binding language: describes schema binding of Java classes which enables a developer to override the default binding rules.

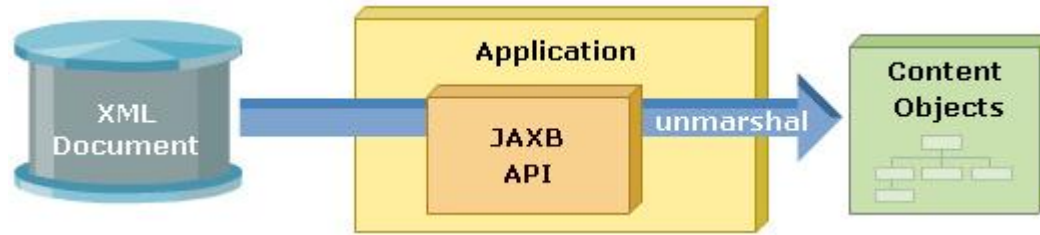
# JAXB ARCHITECTURE

- Consists of three components:



- **Schema compiler:** binds a source schema to a set of schema derived program elements using an XML-based binding language.
- **Schema generator:** Schema generator maps a set of existing program elements to a derived schema.
- **Binding runtime framework:** Binding runtime framework accesses, manipulates and validates XML content using the unmarshalling (reading) and marshalling (writing) operations.

# UNMARSHALLING



- Is the process of converting an XML document into a content tree (DOM).
- To unmarshal an XML document
  - First create a JAXBContext object that provides the entry point to the JAXB API.
  - Specify a context path which contains the list of one or more package names that contain interfaces generated by the binding compiler.
- To access XML data without unmarshalling it
  - Use the createCollection and createBookType methods, to create a content objects tree.
  - The program accesses the ObjectFactory class and uses the appropriate methods within it to create the required objects.
- Process
  - An object of JAXBContext class is created whose context path is Information.jaxb. The Information.jaxb is a package that contains the interfaces generated for the products.xsd schema.
  - An object of unmarshaller class is created that controls the process of unmarshalling. In particular, it contains methods that perform the actual unmarshalling operation.
  - The unmarshal method is invoked which performs the actual unmarshalling of the XML document.
  - The get method in the schema-derived classes is used to access the XML data.



# ***UNMARSHALLING (cont)***

- Using JAXB converts the xml schema to java class with the supported jar file as
  - jaxb-api.jar, jaxb-xjc.jar, jaxb-impl.jar, jaxb-libs.jar
  - xsdlib.jar, relaxngDatatype.jar
- Examples (customer.xsd)

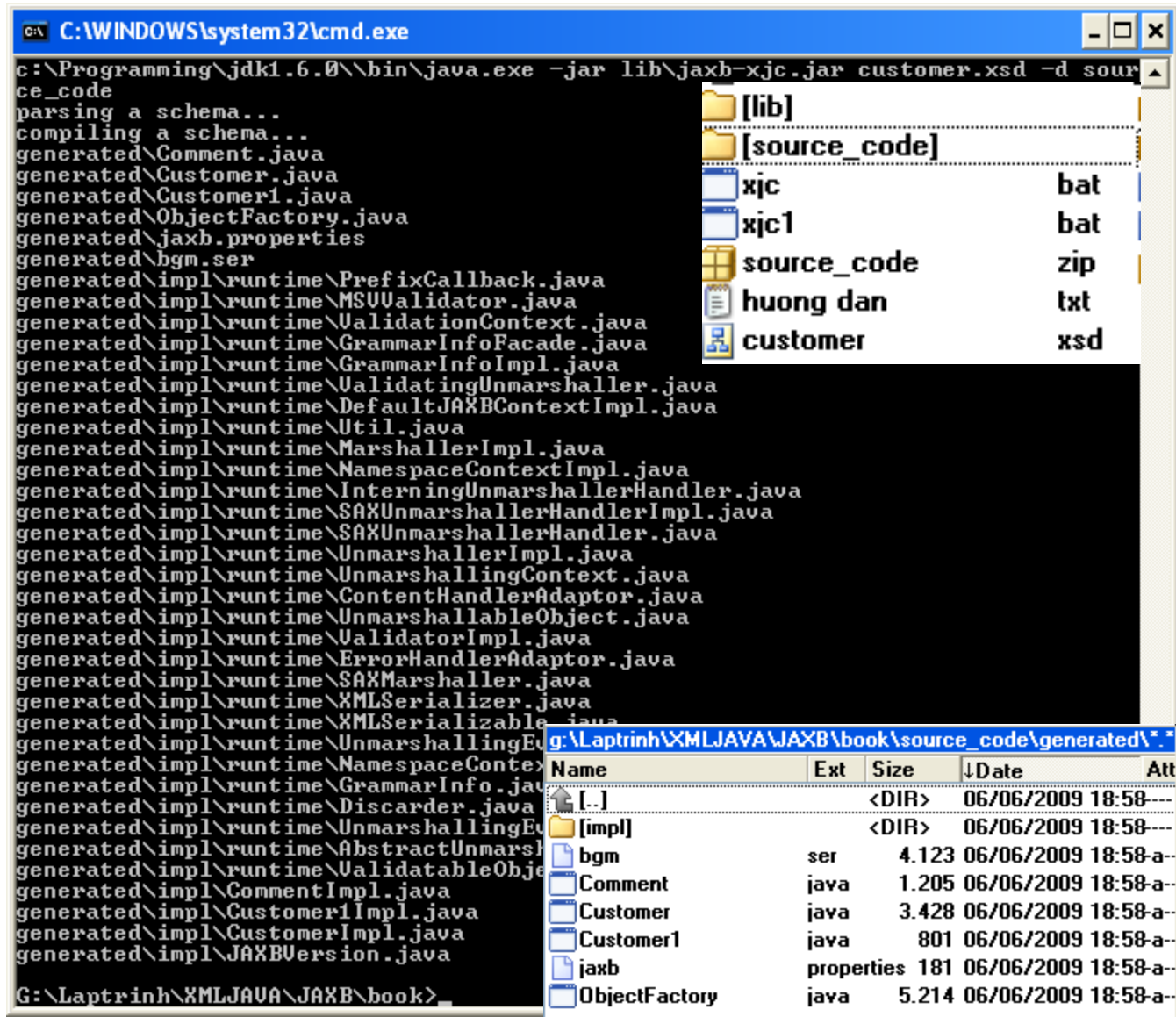
```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="customer1" type="Customer"/>
  <xsd:element name="comment" type="xsd:string"/>
  <xsd:complexType name="Customer">
    <xsd:sequence>
      <xsd:element name="customerid" type="xsd:string"/>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="address" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="phone" type="xsd:positiveInteger"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

# UNMARSHALLING (cont)

- bat file  
(xjc.bat)

```
%JAVA_HOME%\bin\java.exe  
-jar lib\jaxb-xjc.jar %1 %2  
%3 %4 %5
```

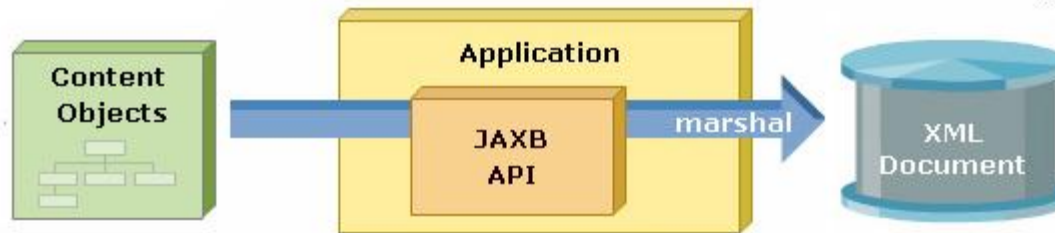
- Executing  
xjc.bat  
customer.xsd -d source\_code



```
C:\WINDOWS\system32\cmd.exe  
c:\Programing\jdk1.6.0\bin\java.exe -jar lib\jaxb-xjc.jar customer.xsd -d source_code  
parsing a schema...  
compiling a schema...  
generated\Comment.java  
generated\Customer.java  
generated\Customer1.java  
generated\ObjectFactory.java  
generated\jaxb.properties  
generated\bgm.ser  
generated\impl\runtime\PrefixCallback.java  
generated\impl\runtime\MSUValidator.java  
generated\impl\runtime\ValidationContext.java  
generated\impl\runtime\GrammarInfoFacade.java  
generated\impl\runtime\GrammarInfoImpl.java  
generated\impl\runtime\ValidatingUnmarshaller.java  
generated\impl\runtime\DefaultJAXBContextImpl.java  
generated\impl\runtime\Util.java  
generated\impl\runtime\MarshallerImpl.java  
generated\impl\runtime\NamespaceContextImpl.java  
generated\impl\runtime\InterningUnmarshallerHandler.java  
generated\impl\runtime\SAXUnmarshallerHandlerImpl.java  
generated\impl\runtime\SAXUnmarshallerHandler.java  
generated\impl\runtime\UnmarshallerImpl.java  
generated\impl\runtime\UnmarshallingContext.java  
generated\impl\runtime\ContentHandlerAdaptor.java  
generated\impl\runtime\UnmarshallableObject.java  
generated\impl\runtime\ValidatorImpl.java  
generated\impl\runtime\ErrorHandlerAdaptor.java  
generated\impl\runtime\SAXMarshaller.java  
generated\impl\runtime\XMLSerializer.java  
generated\impl\runtime\XMLSerializable.java  
generated\impl\runtime\UnmarshallingError.java  
generated\impl\runtime\NamespaceContextImpl.java  
generated\impl\runtime\GrammarInfoImpl.java  
generated\impl\runtime\Discarder.java  
generated\impl\runtime\UnmarshallingError.java  
generated\impl\runtime\AbstractUnmarshaller.java  
generated\impl\runtime\ValidatableObject.java  
generated\impl\CommentImpl.java  
generated\impl\CustomerImpl.java  
generated\impl\Customer1Impl.java  
generated\impl\JAXBVersion.java
```

Name	Ext	Size	Date	Att
[..]	<DIR>		06/06/2009 18:58---	
[impl]	<DIR>		06/06/2009 18:58---	
bgm	ser	4.123	06/06/2009 18:58-a	
Comment	java	1.205	06/06/2009 18:58-a	
Customer	java	3.428	06/06/2009 18:58-a	
Customer1	java	801	06/06/2009 18:58-a	
jaxb	properties	181	06/06/2009 18:58-a	
ObjectFactory	java	5.214	06/06/2009 18:58-a	

# MARSHALLING



- Is the process of building an XML document using the content (DOM) tree (can apply to use transformer)
- To marshalling
  - Modify the existing content tree, or create a new tree from the business logic output.
  - Optionally perform validation of the content tree in-memory, against the source schema.
  - Marshal the content tree into an XML document.
- Process
  - An object of JAXBContext class is created and the appropriate context path of the package that contains the classes and interfaces for the bound schema is specified.
  - An object of Marshaller class is created which controls the process of marshalling.
  - The Marshaller object sets its properties using the setProperty method. Here, the code turns the output format property on line breaks and indentation will appear in the output format.
  - The marshal method is invoked by specifying an object that contains the root of the content tree, and the output target. Here, the code marshals the content tree whose root is in the collection object and writes it as an output stream to the XML file Output.xml.

# ***JAXB BINDING PROCESS***

- Generate classes: first, the JAXB binding compiles the XML schema to generate JAXB classes based on that schema.
- Compile classes: all the generated classes, sources files, and application code should be compiled.
- Unmarshal: Unmarshal the XML documents with the help of JAXB binding framework.
- Generate content tree: the unmarshalling process generates tree of data objects which represents the structure and content of the source XML documents.
- Validate: the unmarshalling process optionally validates the source XML documents before generating the content tree.
- Process content: the client application modifies the XML data represented by the Java content tree.
- Marshal: the processed content tree is marshaled into XML documents which may be further validated before marshalling.


















# ***DATA VALIDATION***

- Validation is the process of verifying that an XML document meets all the constraints expressed in the schema.
- A Web Service processing model should not validate strictly while reading in data, but should validate strictly while writing out data.
- The source data can be validate against the associated schema as part of the unmarshalling operation using the `setValidating()` method.
- The statement requests JAXB to validate the source data against its schema. During validation, whenever the errors are encountered, a validation error should be reported but without stopping the data processing. Thus, it is possible for a JAXB implementation to successfully unmarshal an invalid XML document and built a Java content tree, although the result won't be valid.
- Since there is no `setValidating()` method for marshalling, the validation is not performed as part of the marshalling operation. Here, validation can be done at one point time, and marshalling at other time.
- The JAXB specification authorizes all providers to implement report validation errors when the errors are encountered, but the implementation does not have to stop processing the data. JAXB implementation successfully unmarshals an invalid XML document, and builds a Java content tree. However, the result is not valid.

# ***APPENDIX***

- Using Axis to create WS, then consume using Swing and Application

# WORKSHOP ACTIVITIES

XML Processing APIs <span>X</span>		
Creating an XML File	 Show Me	
Using "startDocument()" and "startElement()" Methods	 Show Me	 Let Me Try
Using "characters()", "endElement()", and "endDocument()" Methods	 Show Me	 Let Me Try
Creating "AirlineTree" Instance	 Show Me	
Building and Running the Application	 Show Me	 Let Me Try
Creating an XML File	 Show Me	
Defining a Bean Class	 Show Me	
Parsing XML Document	 Show Me	 Let Me Try
Retrieving DOM Tree Nodes	 Show Me	 Let Me Try
Creating "EmployeeClassTest" Instance	 Show Me	
Building and Running the Application	 Show Me	 Let Me Try

Building the Swings application can

- Building the tree presentation uses the JTree and SAX
- Building the XML document after using DOM Builder parsing original XML and storing in JavaBean