mood-book

"XML and Web Services"

by

M. Naresh Choudary

Associate Professor
Dept. of CSE (AI & ML)
Sreyas Institute of Engg. and Tech.
Hyderabad

Introduction

- XML(eXtensible Markup Language) is a text based markup language that is fast becoming a standard of data interchange
  - An open standard from W3C
  - A direct descendant from SGML (Standard Generalized Markup Language)
- XML describes data in a way that humans can understand and computers can process

Example: Product Inventory Data

```
<Product>
    <Name>Refrigerator</Name>
    <Model Number>R3456d2h</Model Number>
    <Manufacturer>General Electric</Manufacturer>
    <Price>25000.00</Price>
    <Quantity>100</Quantity>
</Product>
```

Data Interchange

- XMLs key role is data interchange
- Two business partners want to exchange customer data
  - Agree on a set of tags
  - Exchange data without having to change internal databases
- Other business partners can join in the exchange by using the tagset
  - New tags can be added to extend the functionality

XML = Universal Data

- TCP/IP ➔ Universal Networking
- HTML ➔ Universal Rendering
- Java ➔ Universal Code
- XML ➔ Universal Data

- Numerous standard bodies are set up for standardization of tags in different domains
  - ebXML (Electronic Business eXtensible Markup Language)
  - XBRL (eXtensible Business Reporting Language)
  - MML (Medical Markup Language)
  - CML (Chemical Markup Language)

HTML vs. XML

- Both are markup languages
  - HTML has fixed set of tags
  - XML allows user to specify the tags based on requirements
- Usage
  - HTML tags specify how to display data
  - XML tags specify semantics of the data
- Tag Interpretation
  - HTML specifies what each tag and attribute means
  - XML tags delimit data & leave interpretation to the parsing application
- Well formedness
  - HTML very tolerant of rule violations (nesting, matching tags)
  - XML very strictly follows rules of well formedness

..contd

Parameter	XML	HTML
Type of language	XML is a framework for specifying markup languages.	HTML is predefined markup language.
Language type	Case sensitive	Case insensitive
Structural details	It is provided	It is not provided.
Purpose	Transfer of data	Presentation of the data
Coding Errors	No coding errors are allowed.	Small errors are ignored.
Whitespace	You can use whitespaces in your code.	You can't use white spaces in your code.
Nesting	Should be done appropriately.	Does not have any effect on the code.
Driven by	XML is content driven	HTML is format driven
End of tags	The closing tag is essential in a well-formed XML document.	The closing tag is not always required. <HTML> tag needs an equivalent </HTML> tag but tag does not require </br> tag
Quotes	Quotes required around XML attribute values?.	Quotes are not required for the values of attributes.
Object support	Objects have to be expressed by conventions. Mostly using attributes and elements.	Offers native object support

..contd

Null support	Need to use xsi:nil on elements in an XML instance document and also need to import the corresponding namespace.	Natively recognizes the null value.
Namespaces	XML provides support for namespaces. It helps you to remove the risk of name collisions when combining with other documents.	Does not support the concept of namespaces. Naming collisions can be avoided either using a prefix in an object member name or by nesting objects.
Formatting decisions	Require more significant effort to map application types to XML elements and attributes.	Provides direct mapping for application data.
Size	Documents are mostly lengthy in size, especially when an element-centric approach used in formatting.	The syntax is very brief and yields formatted text.
Parsing in Javascript	Requires an XML DOM implementation and application code to map text back into JavaScript objects.	No extra application code required to parse text. For this purpose, you can use the eval function of JavaScript.
Learning curve	Very hard as you need to learn technologies like XPath, XML Schema, DOM, etc.	HTML is a simple technology stack that is familiar to developers.

XML Document

- Prolog
  - Instructs the parser as to what it it parsing
  - Contains processing instructions for processor
- Body
  - Tags – Entities
  - Attributes – Properties of Entities
  - Comments – Statements for clarification in the document

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<contact>
        <name>
                <first name>Naresh</first name>
                <last name>Choudary</last name>
        </name>
        <address>
                <street>Sreyas Street</street>
                <city>Hyderabad</city>
                <state>Telangana</state>
                <zip>500068</zip>
        </address>
</contact>
```

← Prolog

← Body

Prolog

<?xml version="1.0" encoding="UTF-8"?>

- Contains declaration that identifies a document as xml
- Version
  - Version of XML markup language used in the data
  - Not optional
- Encoding
  - Identifies the character set used to encode the data
  - Default compressed Unicode: UTF-8
- May contain entity definitions and tag specifications

XML Syntax: Elements & Attributes

- Uses less-than and greater-than characters (<…>) as delimiters
- Every opening tag must having an accompanying closing tag
  - <First Name>Naresh</First Name>
  - Empty tags do not require an accompanying closing tag.
  - Empty tags have a forward slash before the greater-than sign e.g. <Name/>
- Tags can have attributes which must be enclosed in double quotes
  - <name first="Naresh" last="Choudary">
- Elements should be properly nested
  - The nesting can not be interleaved
  - Each document must have one single root element
- Elements and attribute names are case sensitive

Tree Structure – Elements

- XML documents have a tree structure containing multiple levels of nested tags.
  - Root element is a single XML element which encloses all of the other XML elements and data in the document
  - All other elements are children of the root element

```
<?xml version="1.0" encoding="UTF-8"?>
<contact>                                              ← Root Element
    <name>
        <first name>Naresh</first name>
        <last name>Choudary</last name>
    </name>
    <address>
        <street>Sreyas Street</street>                 ← Child Elements
        <city>Hyderabad</city>
        <state>Telangana</state>
        <zip>500068</zip>
    </address>
</contact>
```

Attributes

- Attributes are properties associated with an element
- Each attribute is a name value pair
  - No element may contain two attributes with same name
  - Name and value are strings

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<contact>
    <name first="Naresh" last="Choudary"></name>        ← Attributes
    <address>
        <street>Sreyas Street</street>        ← Nested Elements
        <city>Hyderabad</city>
        <state>Telangana</state>
        <zip>500068</zip>
    </address>
</contact>
```

Elements vs. Attributes

- Data should be stored in Elements
- Information about data (meta-data) should be stored in attributes
- Rules of thumb to use elements:
  - Elements should have information which some one may want to read.
  - Attributes are appropriate for information about document that has nothing to do with content of document

    e.g. URLs, units, references, ids belong to attributes
  - What is your meta-data may be some ones data

Comments

- XML comments begin with “<!--”and end with “-->”
  - All data between these delimiters is discarded
  - <!-- This is a list of names of people -->
- Comments should not come before XML declaration
- Comments can not be placed inside a tag
- Comments may be used to hide and surround tags

```
<Name>
    <first>Naresh</first>
    <!-- <last>Choudary</last> -->
</Name>
```

← last tag is ignored

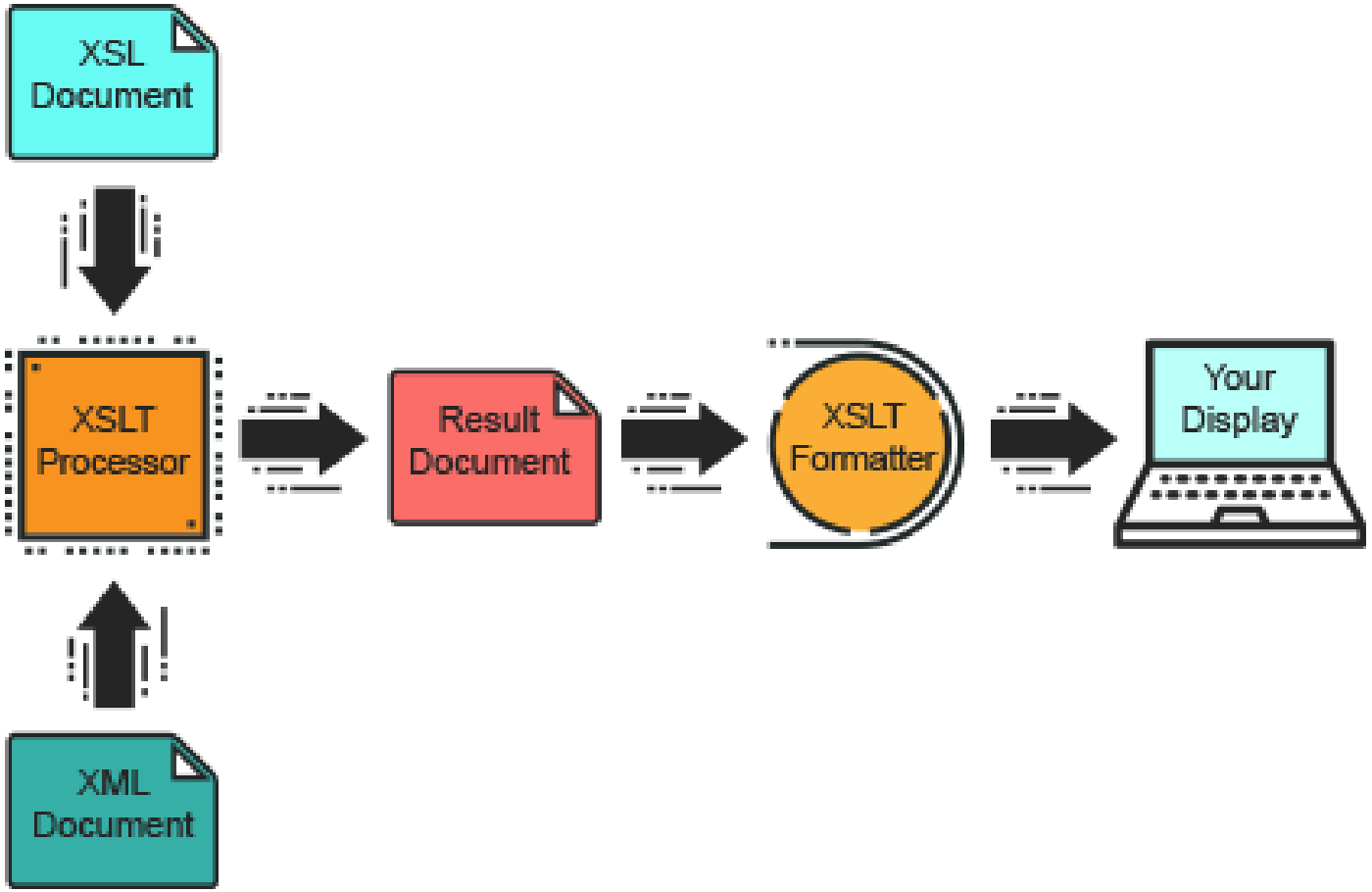
- “--” string may not occur inside a comment except as part of its opening and closing tag
  - <!- this is not - a valid comment -->

← Illegal

XSL

- eXtensible Stylesheet Language (XSL), it is used for expressing style sheets
- An XSL style sheet describes how to display an XML document of a given type
- XSL consists of 3 parts:
  - XPath (navigation in documents)
  - XSLT (transformation of documents)
    - Converting XML to another form
  - XSLFO (for formatting the objects)
    - Layout of XML document
- XSL uses XSLT which uses XPath

XSL Architecture



XPath

- XPath uses path expressions to select nodes or node-sets in an XML document

Expression	Description
nodename	Selects all nodes with the name "nodename"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

XSLT

- A transformation in the XSLT language is expressed in the form of a stylesheet
- The root element that declares the document to be an XSL style sheet is:
  `<xsl:stylesheet>` *or* `<xsl:transform>`
- Declaring an XSL style sheet

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                                version="1.0">
```

or

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                                version="1.0">
```

NOTE: *XML NameSpace is a mechanism to avoid name conflicts by differentiating* `elements` *or* `attributes`*, can have identical names, but different definitions.*

The Simplest Stylesheet

- Empty XSL/XSLT document

```
<?xml version='1.0' encoding='UTF-8'?>
<xsl:stylesheet
      xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
      version='1.0'>



</xsl:stylesheet>
```

NOTE: *This will simply copy the text content of the input document to the output.*

<xsl:template> Element

- An XSL style sheet consists of one or more set of rules that are called templates
- A template contains rules to apply when a specified node is matched
- The `<xsl:template>` element is used to build templates.
- An XSL style sheet is an XML document, it always begins with the XML declaration:
- The content inside the `<xsl:template>` element defines some HTML to write to the output
- `<xsl:template match='/'>`, the value of the match attribute is an XPath expression (i.e. match="/" defines the whole document)

<xsl:value-of> Element

- It is used to extract the value of a selected node
- It can be used to extract the value of an XML element and add it to the output stream of the transformation
- The `select` attribute contains an XPath expression
- An XPath expression works like navigating a file system; a forward slash (/) selects subdirectories

Example:

```
<xsl:value-of select = "firstname"/>
```

<xsl:for-each> Element

- It allows you to do looping in XSLT
- It can be used to select every XML element of a specified node-set
- The select attribute works like same that of <value-of> element

Example:

```
<xsl:for-each select="class/student">
```

<xsl:if> Element

- It is used to put a conditional test against the content of the XML file
- Syntax:

```
<xsl:if test="boolean_expression">
        some output if the expression is true
</xsl:if>
```

Example:

```
<xsl:if test = "marks > 90">
```

<xsl:apply-templates> Element

- It applies a template rule to the current element or to the current element's child nodes
- If we add a `select` attribute, it will process only the child elements that matches the value of the attribute
- We can also use the `select` attribute to specify in which order the child nodes are to be processed
- Syntax:

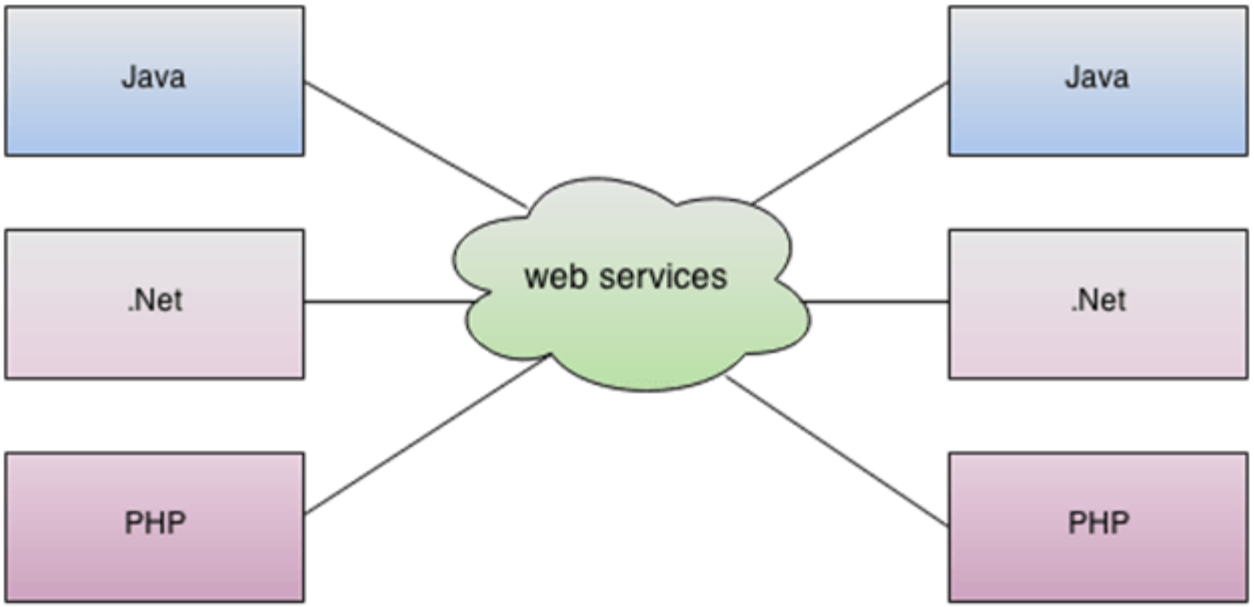
```
<xsl:appy-templates select="expression">
```

Example:

```
<xsl:apply-templates select = "class/student" />
```

Web Services

- Web Service is a software service that allows applications to communicate with each other in a standard format



..contd

- A Web Service exposes an interface that can be accessed through XML messaging
- A Web service uses XML based protocol (SOAP) to describe an operation or the data exchange with another web service
- A group of web services collaborating accomplish the tasks of an application, then that architecture is called as Service-Oriented Architecture (SOA)

Benefits of Web Services

- **Exposing Business Functionality on the network**
  - A web service is a unit of managed code that provides some sort of functionality to client applications or end users
  - This functionality can be invoked over the HTTP protocol which means that it can also be invoked over the internet
- **Interoperability amongst applications**
  - All types of applications can talk to each other
  - So instead of writing specific code which can only be understood by specific applications, you can now write generic code that can be understood by all applications

..contd

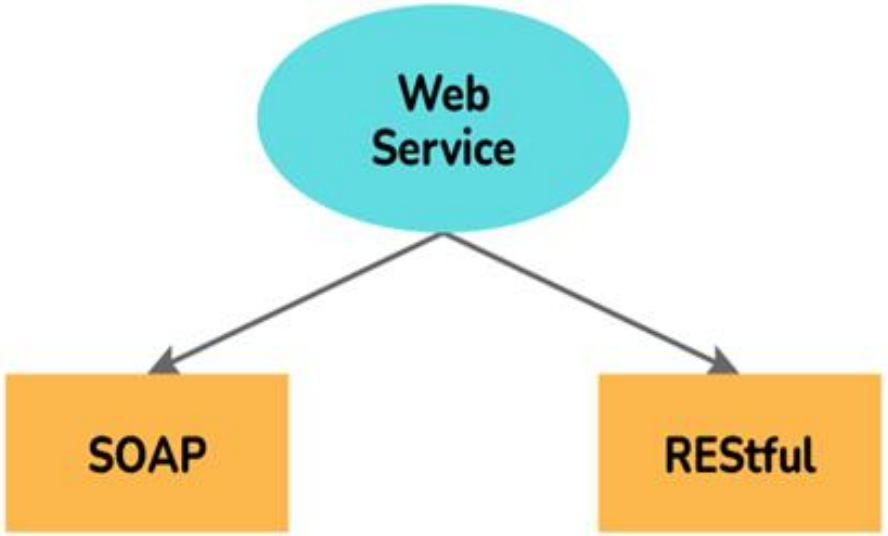
- A Standardized Protocol which everybody understands
  - It uses standardized industry protocol for the communication
  - All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) uses well-defined protocols in the web services protocol stack
- Loosely Coupled
  - Each service exists independently of the other services that make up the application
  - Individual pieces of the application to be modified without impacting unrelated areas

..contd

- **Ease of Integration**
  - Data is isolated between applications by creating 'silos'
  - Web Services act as glue between these and enable easier communications within and across organizations
- **Service Reuse**
  - A specific function within the domain is only ever coded once and used over and over again by consuming applications
- **Reduction in cost of communication**
  - Using low-cost internet connection, SOAP over HTTP protocol can reduce cost for implementing web services

Types of Web Services

- There are mainly two types of web services:
  - SOAP web services
  - RESTful web services



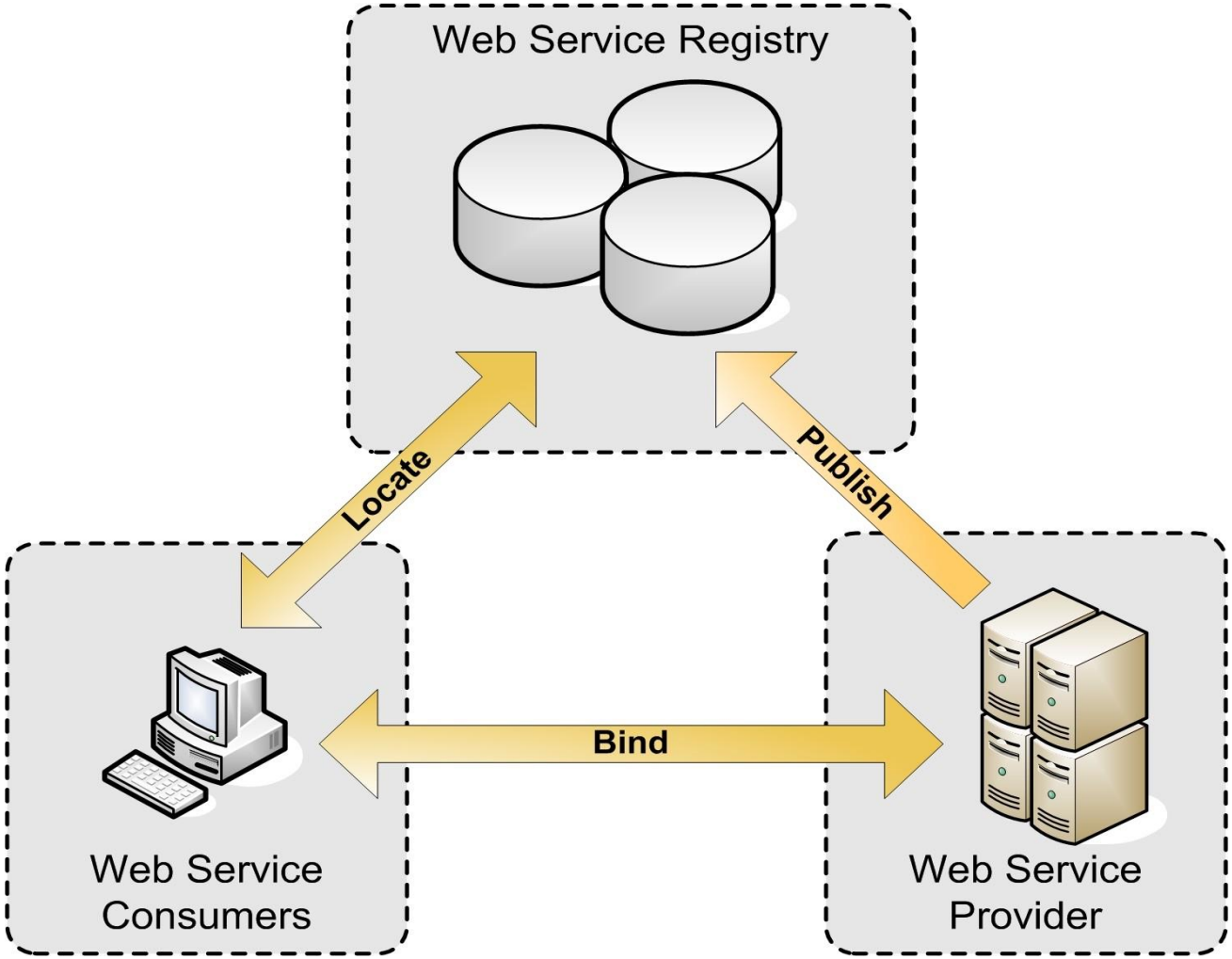
SOAP vs. REST

SOAP	REST
SOAP is a protocol	REST is an architectural style
SOAP stands for Simple Object Access Protocol	REST stands for REpresentational State Transfer
SOAP can't use REST because it is a protocol	REST can use SOAP web services because it is a concept and can use any protocol like HTTP, SOAP
SOAP defines standards to be strictly followed	REST does not define too much standards like SOAP
SOAP requires more bandwidth and resource than REST	REST requires less bandwidth and resource than SOAP
SOAP defines its own security	RESTful web services inherits security measures from the underlying transport
SOAP permits XML data format only	REST permits different data format such as Plain text, HTML, XML, JSON etc

Service Oriented Architecture (SOA)

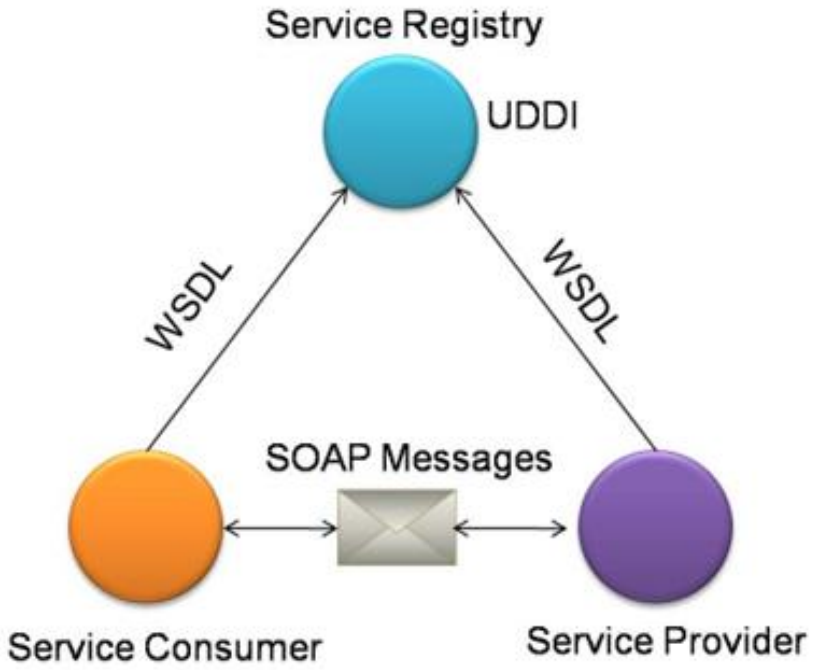
- A more sophisticated system:
  - A registry, acts as a mediator for Web services
  - A provider, can publish services to the registry
  - A consumer, can then discover services in the registry
- The provider presents the interface and implementation of the service, and the requester uses the Web service.

..contd



Components of Web Services

- There are three major web service components:
  - SOAP
  - WSDL
  - UDDI



Simple Object Access Protocol (SOAP)

- It is a transport-independent messaging protocol
- SOAP is built on sending XML data in the form of SOAP Messages
- A document known as an XML document is attached to each message
- The best thing about Web services and SOAP is that everything is sent through HTTP, the standard web protocol

..contd

- In an XML document, the root element is the first element
- The “envelope” is separated into two halves
  - header
  - body
- The routing data, or information that directs the XML document to which client it should be sent to, is contained in the header
- The real message will be in the body

Web Services Description Language (WSDL)

- WSDL is an XML format for describing all the information needed to invoke and communicate with a Web Service
- It gives the answers to the questions Who? What? Where? Why? How?
- A service description has two major components:
  - **Functional Description**

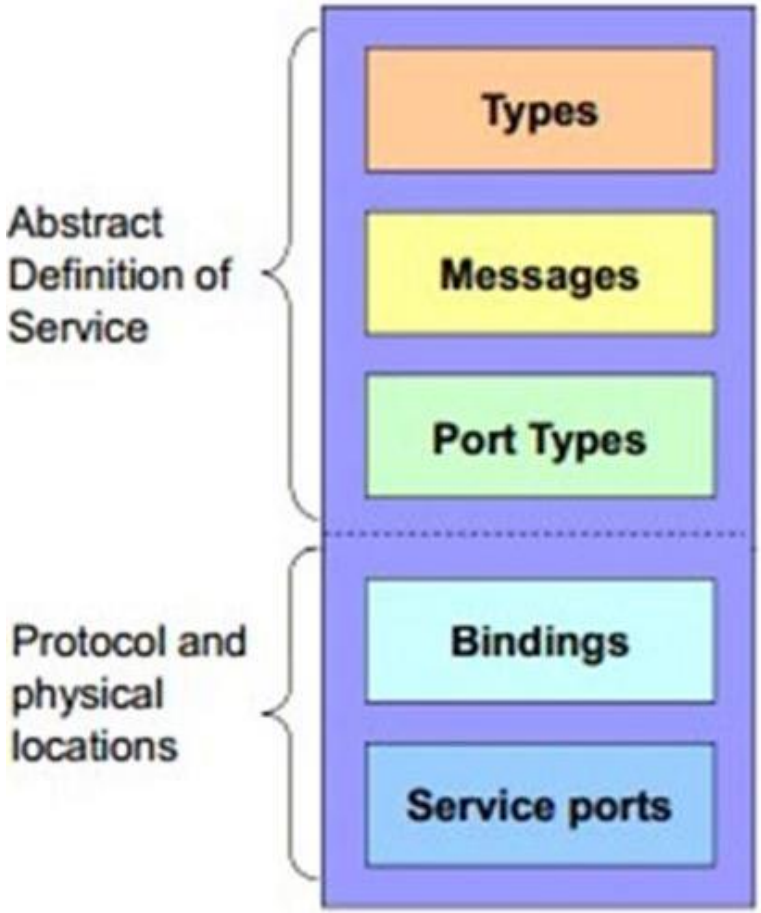
    Defines details of how the Web Service is invoked, where it's invoked. Focuses on the details of the syntax of the message and how to configure the network protocols to deliver the message.
  - **Nonfunctional Description**

    Provides other details that are secondary to the message (such as security policy) but instruct the requestor's runtime environment to include additional SOAP headers.

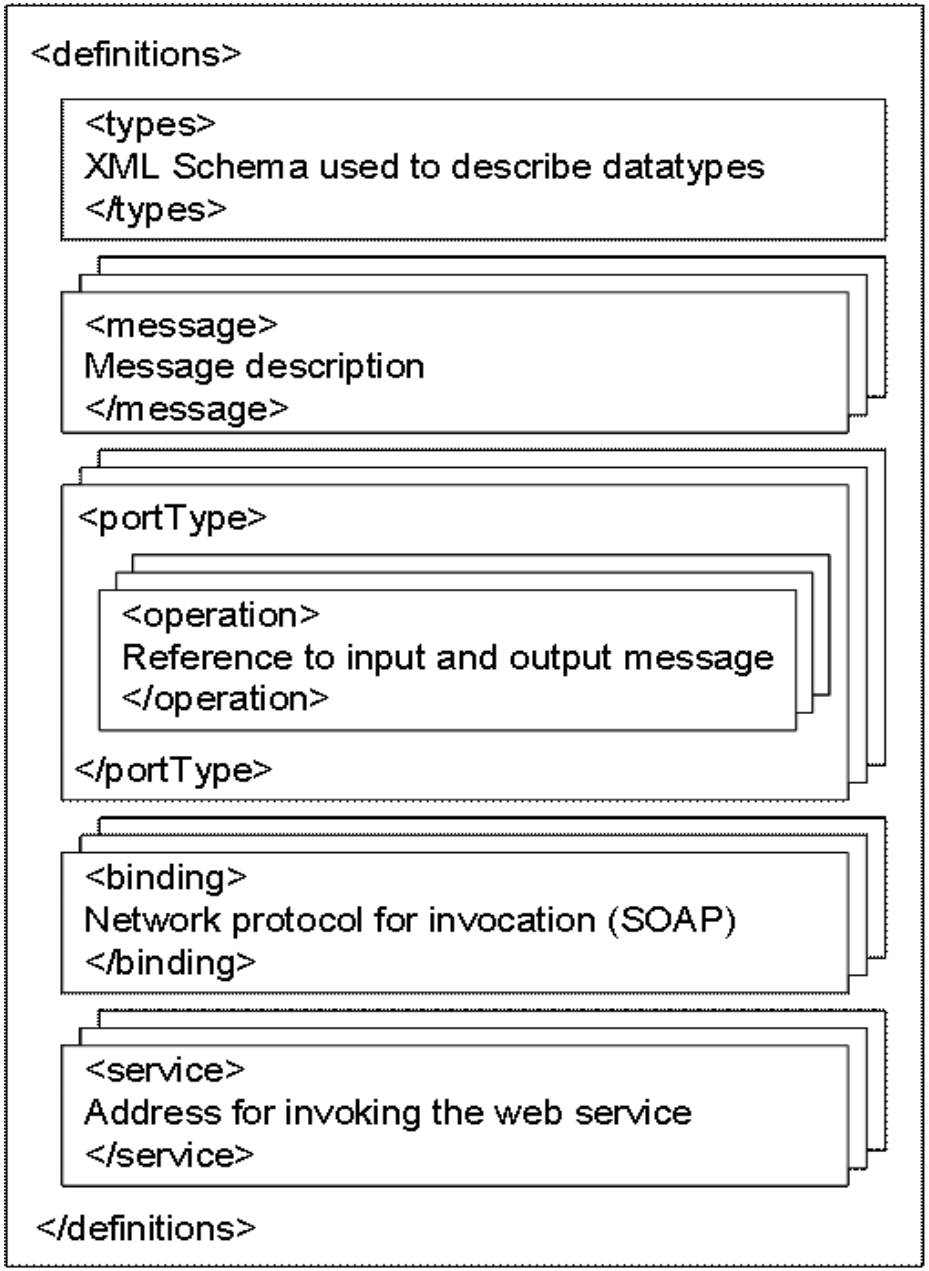
WSDL Document Structure

- A WSDL Document is a set of definitions with a single root element
- Services can be defined using the following XML elements:
  - **Types**, think Data Type
  - **Message**, think Methods
  - **PortType**, think Interfaces
  - **Binding**, think Encoding Scheme
  - **Service**, many URLs

..contd



- Types
  - What data types will be transmitted
- Messages
  - What messages will be transmitted
- Port Types
  - What business operations (functions) will be supported
- Bindings
  - How will the messages be transmitted on the wire?
  - What message protocol (e.g. SOAP) specific details are there?
- Service ports
  - Where is the service located?



..contd

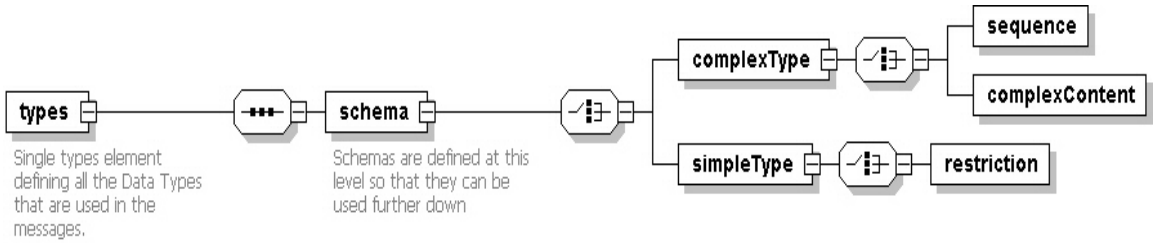
- types: Provides information about any complex data types used in the WSDL document. When simple types are used the document does not need to have a types section
- message: An abstract definition of the data being communicated
- portType: An abstract set of operations supported by one or more endpoints
- operation: An abstract description of the action supported by the service

..contd

- binding: Describes how the operation is invoked by specifying concrete protocol and data format specifications for the operations and messages
- port: Specifies a single endpoint as an address for the binding, thus defining a single communication endpoint
- service: Specifies the port address(es) of the binding. The service is a collection of network endpoints or ports

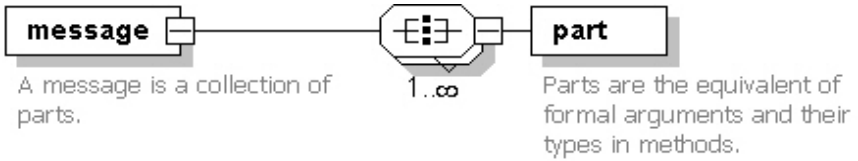
<types> Element

- The default type system in WSDL is the XML Schema (XSD)
- A WSDL document can have at most one `types` element
- The `types` element can contain `simpleType` or `complexType`
- At the lowest level elements intuitively named (again!) element are defined with a `name` and a `type` attribute



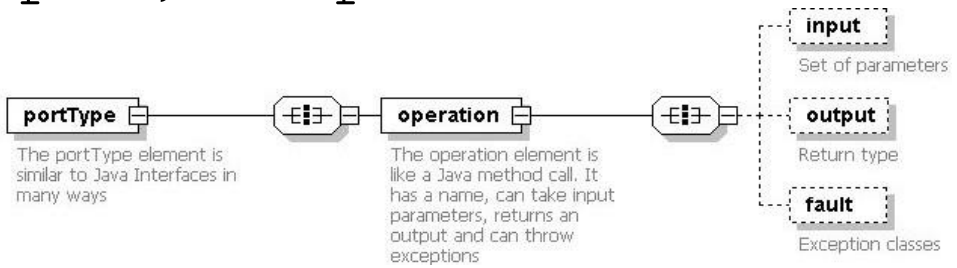
<message> Element

- A `message` is a collection of parts; intuitively a `part` is a named argument with its type
- A WSDL document can contain zero or more `message` elements
- Each `message` element can be used as an input, output or fault message within an `operation`
- The `type` attribute of `part` can be any standard data type from the XSD Schema or a user defined one



<portType> Element

- The `portType` element describes the interface to a Web Service
- A WSDL Document can contain zero or more `portType`
- A `portType` element contains a single `name` attribute.
- A `portType` contains one or more `operation` elements, with a `name` attribute can contain `input`, `output` and `fault` elements

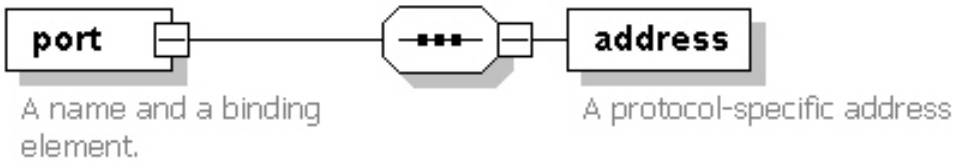


<binding> Element

- The `binding` element specifies to the service requester how to format the message in a protocol-specific manner
- Each `portType` can have one or more `binding` elements associated with it
- For a given `portType` the `binding` element has to specify an messaging and transport pair. (SOAP/HTTP, SOAP/SMTP, etc)

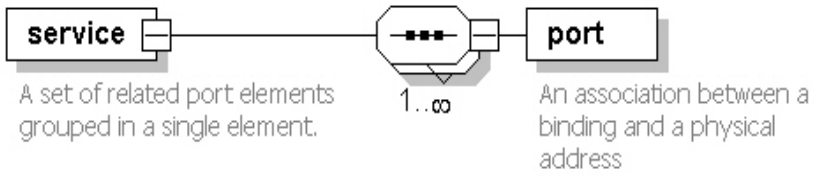
<port> Element

- The `port` element specifies the network address of the endpoint hosting the Web Service
- It associates a single protocol-specific address to an individual `binding` element
- Ports are named and must be unique within the document



<service> Element

- The `service` element is a collection of related port elements identified by a single service name
- A WSDL Document is allowed to contain multiple service elements, but conventionally contains a single one.
- Each service must be uniquely named



Universal Description, Discovery, and Integration(UDDI)

- An XML-based lookup service for locating web services in the Internet
- UDDI provides a platform-independent way of describing and discovering web services and web service providers
- The UDDI data structures provide a framework for the description of basic service information, and an extensible mechanism to specify detailed service access information using any standard description language

How UDDI can be used?

- If the industry published an UDDI standard for flight rate checking and reservation, airlines could register their services into an UDDI directory.
- Travel agencies could then search the UDDI directory to find the airline's reservation interface.
- When the interface is found, the travel agency can communicate with the service immediately because it uses a well-defined reservation interface. (by WSDL)

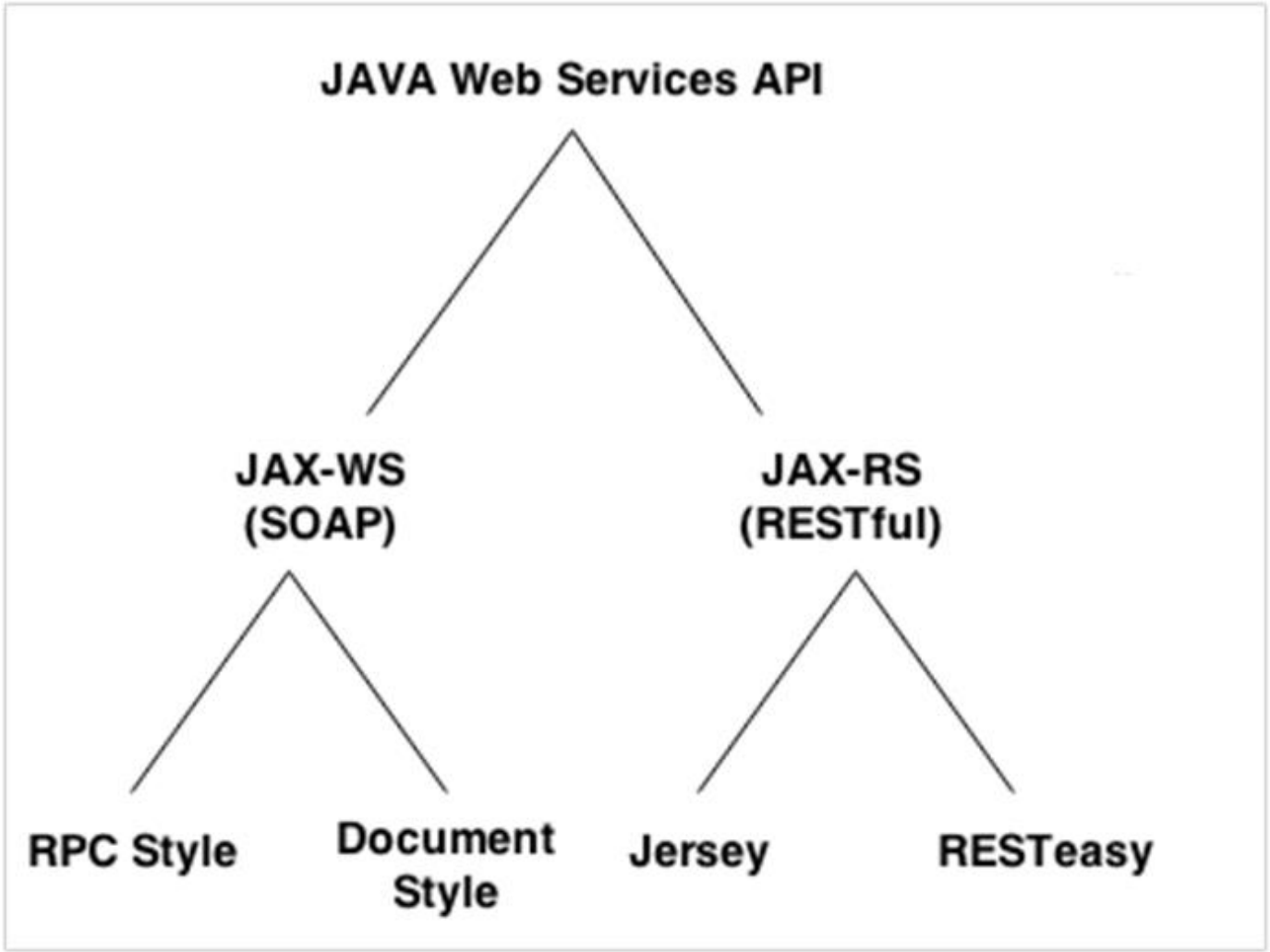
Java Web Services

- The java web service application can be accessed by other programming languages such as .Net and PHP etc.,
- Java provides it's own API to create both SOAP as well as REST web services:
  - JAX-WS (Java API for XML Web Services)
  - JAX-RS (Java API for RESTful Web Services)
- Java web service application perform communication through WSDL

Java Web Services API

- JAX-WS: for SOAP web services
- The are two ways to write JAX-WS application code:
  - RPC style
  - Document style
- JAX-RS: for RESTful web services
- There are two main implementations of JAX-RS:
  - Jersey
  - RESTeasy

..contd



JAX-WS

- There are two ways of building SOAP web services:
  - Top-down approach (contract-first)
  - Bottom-up approach(contract-last)
- In a Top-down approach, a WSDL document is created, and the necessary Java classes are generated from the WSDL
- In a Bottom-up approach, the Java classes are written, and the WSDL is generated from the Java classes

RPC Style

- RPC (Remote Procedure Call) style web services use method name and parameters to generate XML document structure
- The generated WSDL is difficult to be validated against predefined schema
- In RPC style, SOAP message is sent as many elements
- RPC style message is tightly coupled
- In RPC style, SOAP message keeps the operation name

..contd

- In RPC style, parameters are sent as discrete values
- In WSDL file, it doesn't specify the `types` section
- For message `part`, it defines `name` and `type` attributes
- For `soap:body`, it defines `use` and `namespace` attributes

Document Style

- Document style web services can be validated against predefined schema
- In document style, SOAP message is sent as a single document
- Document style message is loosely coupled
- In Document style, SOAP message has no operation name
- In Document style, parameters are sent in XML format

..contd

- In WSDL file, it specifies `types` details having `namespace` and `schemaLocation`
- For message `part`, it defines `name` and `element` attributes
- For `soap:body`, it defines `use` attribute only, no `namespace`

RPC vs. Document

RPC Style	Document Style
The generated WSDL is difficult to be validated against predefined schema	Document style web services can be validated against predefined schema
In RPC style, SOAP message is sent as many elements	In document style, SOAP message is sent as a single document
RPC style message is tightly coupled	Document style message is loosely coupled
In RPC style, SOAP message keeps the operation name	In Document style, SOAP message has no operation name
In RPC style, parameters are sent as discrete values	In Document style, parameters are sent in XML format
In WSDL file, it doesn't specify the types section	In WSDL file, it specifies types details having namespace and schemaLocation
For message part, it defines name and type attributes	For message part, it defines name and element attributes
For soap:body, it defines use and namespace attributes	For soap:body, it defines use attribute only, no namespace

Jersey Framework

- Jersey is open source framework for developing RESTful Web Services in Java
- It has advantages such as:
  - Using Java and the Java Virtual Machine, make it easy to build RESTful Web Services
  - Contains support for Web Application Description Language (WADL)
  - Contains Jersey Test Framework which lets run and test Jersey REST services inside JUnit
  - Supports for the REST MVC pattern, which would allow to return a View from Jersey services rather than just data

RESTeasy Framework

- RESTeasy is a JBoss implementation of JAX-RS
- It can run in any Servlet container
- It has benefits such as:
  - Portable to Tomcat and many other app-server
  - JAXB (Java Architecture for XML Binding) marshalling into XML, JSON etc.,
  - Digital Signature and encryption support with S/MIME and DOSETA (Domain Security Tagging)
  - Rich set of providers for: XML, JSON (JavaScript Object Notation), YAML (Yet Another Markup Language), XOP (Xml Binary Optimized Packaging) etc.,

JAX-WS vs. JAX-RS

JAX-WS	JAX-RS
It is a Java API for XML-Based Web Services	It is a Java API for RESTful Web Services
The are two ways to write JAX-WS application code: RPC style and Document style	There are two main implementations of JAX-RS: Jersey and RESTeasy
JAX-WS follows the SOAP protocol and interacts in XML messages	JAX-RS, as it has no fixed structure to it, can communicate through XML, HTML, JSON, and HTTP
It provides a standard way to develop a Web Services in SOAP notation	RESTful Web Services are represented as resources and can be identified by Uniform Resource Identifiers (URI)
Web Services are called/invoked through remote procedure calls	Remote procedure call in this case is represented a HTTP- request and the necessary data is passed as parameters of the query
SOAP define too much standards than REST	REST does not define too much standards like SOAP
JAX-WS is used for mainly building up web-services on an enterprise-level	JAX-RS is mostly used in smartphone apps and for purposes like Web Integration