Copyright Notice

Vishwasoft Technologies

Web Service Description standard : WSDL

Prakash Badhe

prakash.badhe@vishwasoft.in

What is wsdl ?

- WSDL means Web Service Description Language
- WSWSDL is a specification defining how to describe web services in a common XML grammar.
- WSDL represents a contract between the service requestor and the service provider
- WSDL describes four pieces of data:
  - Interface information describing all publicly available functions
  - Data type information for all message requests and message responses
  - Binding information about the transport protocol to be used
  - Address information for locating the specified service

WSDL

- WSDL enables a service provider to specify the following characteristics of a Web service:
  - Name of the Web service and addressing information
  - Protocol and encoding style to be used when accessing the public operations of the Web service
  - Type information: Operations, parameters, and data types comprising the interface of the Web service, plus a name for this interface

WSDL an XML

- A WSDL specification uses XML syntax; therefore, there is an XML Schema for it.
- A valid WSDL document consists of one or more files. If there is more than one file, the use of the import element is required. This import element creates the needed references to locate the different files of the WSDL document.

WSDL Standard Format

- The Web Services Description Language (WSDL) is a standardized XML format for describing network services.
- The description includes the name of the service, the location of the service, and ways to communicate with the service.
- WSDL service descriptions can be stored in UDDI registries or published on the web (or both).

WSDL usage

- WSDL is platform- and language-independent and is used to describe web services.
- Using WSDL, a client can locate a web service and invoke any of its publicly available functions.
- WSDL uses the W3C XML which specifies basic type system for encoding most data types.
- This type system includes a list of built-in simple types, including strings, floats, doubles, integers etc.
- More complex types can be specified.

WSDL Structure

<definitions>
  <types> **definition of types.....**
  </types>
  <message> **definition of a message....**
  </message>
  <portType> **definition of a port.....**
  </portType>
  <binding> **definition of a binding....**
  </binding>
 </definitions

WSDL Structure

The WSDL document contains the following main elements

- **Types -** A container for data type definitions using some type system, such as XML Schema.
- **Message -** An abstract, typed definition of the data being communicated. A message can have one or more typed parts.
- **Port type -** An abstract set of one or more operations supported by one or more ports.
- **Operation -** An abstract description of an action supported by the service that defines the input and output message and optional fault message.

WSDL Structure-2

- **Binding -** A concrete protocol and data format specification for a particular port type. The binding information contains the protocol name, the invocation style, a service ID, and the encoding for each operation.
- **Service -** A collection of related ports.
- **Port -** A single endpoint, which is defined as an aggregation of a binding and a network address

WSDL part1

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

<wsdl:definitions targetNamespace=http://greeting
    xmlns:impl=http://greeting
    xmlns:intf="http://greeting"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsi="http://ws-i.org/profiles/basic/1.1/xsd"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <wsdl:types>
        <schema targetNamespace="http://greeting"
                xmlns="http://www.w3.org/2001/XMLSchema"
                xmlns:impl=http://greeting
                xmlns:intf=http://greeting
                xmlns:wsdl=http://schemas.xmlsoap.org/wsdl/

        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

WSDL Part2

```
<element name="sayHelloResponse">
    <complexType>
                <sequence>
                        <element name="sayHelloReturn"
                                                nillable="true"
                                                type="xsd:string"/>
                </sequence>
    </complexType>
  </element>
  <element name="sayHello">
    <complexType>
                <sequence/>
    </complexType>
  </element>
 </schema>
</wsdl:types>
```

WSDL Part3


```
<wsdl:message name="sayHelloRequest">
    <wsdl:part element="impl:sayHello" name="parameters"/>
</wsdl:message>
<wsdl:message name="sayHelloResponse">
    <wsdl:part element="impl:sayHelloResponse"
        name="parameters"/>
</wsdl:message>
<wsdl:portType name="Hello">
    <wsdl:operation name="sayHello">
        <wsdl:input message="impl:sayHelloRequest"
            name="sayHelloRequest"/>
        <wsdl:output message="impl:sayHelloResponse"
            name="sayHelloResponse"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="HelloSoapBinding" type="impl:Hello">
```

WSDL Bindings

```
<wsdlsoap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="sayHello">
                <wsdlsoap:operation soapAction=""/>
                <wsdl:input name="sayHelloRequest">
                        <wsdlsoap:body use="literal"/>
                </wsdl:input>
                <wsdl:output name="sayHelloResponse">
                        <wsdlsoap:body use="literal"/>
                </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
```

WSDL service Settings

```
<wsdl:service name="HelloService">
        <wsdl:port binding="impl:HelloSoapBinding"
                                name="Hello">
                <wsdlsoap:address

        location="http://localhost:9080/HelloService/services
   /Hello"/>
        </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

WSDL definitions

- The **definitions** element specifies the name of the WebService. It also specifies numerous namespaces that will be used throughout the remainder of the document.
- <definitions name="HelloService"
  targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
  xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

The use of namespaces is important for differentiating elements and it enables the document to reference multiple external specifications, including the WSDL specification, the SOAP specification, and the XML Schema specification.

Definition Namespaces

- The definitions element specifies a **targetNamespace** attribute. The targetNamespace is a convention of XML Schema that enables the WSDL document to refer to itself. E.g. targetNamespace = “http://www.ecerami.com/wsdl/HelloService.wsdl”
- However namespace specification does not require that the document actually exist at this location.The important point is that you specify a value that is unique, different from all other namespaces that are defined.
- The definitions element also specifies a **default** namespace: **xmlns=http://schemas.xmlsoap.org/wsdl/**.which is the default WSDL namespace.
- All elements without a namespace prefix, such as message or portType, are therefore assumed to be part of the default WSDL namespace.

WSDL Elements -1

- definitions

  The definitions element must be the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.

- types

  The types element describes all the data types used between the client and server. If the service uses only XML Schema built-in simple types, such as strings and integers, the types element is not required.

WSDL Elements - 2

- message

  The message element describes a one-way message, whether it is a single message request or a single message response. It defines the name of the message and contains zero or more message part elements, which can refer to message parameters or message return values.

- portType
  - The portType element combines multiple message elements to form a complete one-way or round-trip operation. For example, a portType can combine one request and one response message into a single request/response operation, most commonly used in SOAP services. Note that a portType can define multiple operations.

WSDL Elements - 3

- binding

  The binding element describes the concrete specifics of how the service will be implemented on the wire. WSDL includes built-in extensions for defining SOAP services, and SOAP-specific information therefore goes here.

- service

  The service element defines the address for invoking the specified service. Most commonly, this includes a URL for invoking the SOAP service.

WSDL Message Elements

- The **<message>** element defines the data elements of an operation.
- Two message elements are defined. The first represents a request message and the second represents a response message.
- Each message can consist of one or more parts. The parts are analogous to the parameters of a function call in a programming language.
- These parts have name and type attributes to specify the xml schema types.
- The value of the type attribute must be specified as an XML Schema QName--this means that the *value* of the attribute must be namespace-qualified. For example, the firstName type attribute is set to **xsd:string**, the **xsd** prefix references the namespace for XML Schema, defined earlier within the definitions element.
- If the function expects multiple arguments or returns multiple values, you can specify multiple part elements.

WSDL Types

- The **<types>** element defines the data type that are used by the web service.
- For maximum platform neutrality, WSDL uses XML Schema syntax to define data types.

WSDL portTypes

- A WSDL port describes the interfaces (legal operations) exposed by a web service.
- It is defined by using the <portType> element.
- It defines a web service, the operations that can be performed, and the messages that are involved.
- The portType element defines a single operation. The operation itself consists of a single input message and a single output message.
- The portType defines the connection point to a web service. It is analogous to a function library (or a module, or a class) where groups of functions are defined. Each operation can be compared to a function in a the library.

Messages with PortTypes

- The message attribute must be specified as an XML Schema QName inside the portType
- This means that the value of the attribute must be namespace-qualified.
- For example, the input element specifies a message attribute of *tns:SayHelloRequest*; the tns prefix references the ***targetNamespace*** defined earlier within the definitions element.

WSDL Operation types

- WSDL defines four types of operations.
- **One-way**: The operation can receive a message but will not return a response. The operation has a single input element
- **Request-response**: The operation can receive a request and will return a response. The service receives a message and sends a response. The operation has one input element, followed by one output element. To encapsulate errors, an optional fault element can also be specified.
- **Solicit-response**: The operation can send a request and will wait for a response. The service sends a message and receives a response. The operation has one output element , followed by one input element. To encapsulate errors, an optional fault element can also be specified.
- **Notification** :The operation can send a message but will not wait for a response. The service sends a message. The operation therefore has a single output element.

One way operation


```
<message name=“myMessage">
   <part name="term" type="xs:string"/>
   <part name="value" type="xs:string"/> </message>
<portType name=“myPort">
  <operation name="setData">
   <input name=“Info“ message=" myMessage "/>
  </operation>
</portType >
```

The " **setData** " operation allows input of new **myPort** messages using a " **myMessage** " message with the input parameters "term" and "value". However, no output is defined for the operation.

Request-Response Operation


```
<message name="getRequest">
   <part name="term" type="xs:string"/>
 </message>
 <message name="getResponse">
   <part name="value" type="xs:string"/> </message>
<portType name="Terms">
    <operation name="getTermData">
      <input message=" getRequest "/>
      <output message="getResponse"/>
    </operation>
</portType>
```

The port "Terms" defines a request-response operation called "getTermData".

The "getTermData" operation requires an input message called "get Request" with a parameter called "term", and will return an output message called "getResponse" with a parameter called "value".

WSDL Bindings

- WSDL bindings defines the message format and protocol details for a web service.
- The binding element provides specific details on how a portType operation will actually be transmitted over the wire. Bindings can be made available via multiple transports, including HTTP GET, HTTP POST, or SOAP. You can specify multiple bindings for a single portType.
- The binding element has two attributes - the name and the type attribute.
- The name attribute (any name can be used) defines the name of the binding, and The type attribute references the portType defined earlier in the document.

SOAP Binding -1

- **soap:binding** : This element indicates that the binding will be made available via SOAP. The style attribute indicates the overall style of the SOAP message format.
- A style value of *rpc* specifies an RPC format. This means that the body of the SOAP request will include a wrapper XML element indicating the function name. Function parameters are then embedded inside the wrapper element. Likewise, the body of the SOAP response will include a wrapper XML element that mirrors the function request. Return values are then embedded inside the response wrapper element. The services follow the SOAP RPC and encoding rules
- A style value of **document** specifies an XML document call format. This means that the request and response messages will consist simply of XML documents. The document style is flatter than the rpc style and does not require the use of wrapper elements.

  (most Microsoft .NET WSDL files use the document style) ,
- The transport attribute indicates the transport of the SOAP messages. The value http://schemas.xmlsoap.org/soap/http indicates the SOAP HTTP transport, whereas http://schemas.xmlsoap.org/soap/smtp indicates the SOAP SMTP transport.

SOAP Bindings -2

- **soap:operation** : This element indicates the binding of a specific operation to a specific SOAP implementation. The operation element refers the operation that the port exposes.
- For each operation the corresponding SOAP action has to be defined which is the request URL for operation.
- The soapAction attribute specifies that the **SOAPAction** HTTP header be used for identifying the service.
- **soap:body :**This element enables to specify the details of the input and output messages. The body element specifies the SOAP encoding style and the namespace URN associated with the specified service. These specify how the input and output are encoded. The options are "literal“ and other encoding styles.

Binding with operation


```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
   <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
   </operation>
</portType>
<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" /> <operation>
  <soap: operation soapAction="http://example.com/getTerm"/>
  <input> <soap: body use="literal"/>
   </input>
    <output>
   <soap: body use="literal"/>
   </output> </operation>
</binding>
```

WSDL Service element

- The service element specifies the location of the service specifying the host address for the service. http://localhost:8080/data/servlet/rpcrouter.

WSDL with Arrays

```
<complexType name="ArrayOfString">
   <complexContent>
    <restriction base="soapenc:Array">
      <attribute ref="soapenc:arrayType"
       wsdl:arrayType="string[]"/>
     </restriction>
   </complexContent>
</complexType>
```

WSDL Types - 1

- In order for a SOAP client to communicate effectively with a SOAP server, the client and server must agree on a data type system.
- The XML 1.0 does not provide a data type system.
- WSDL uses the W3C XML Schema specification for data type declarations.
- The XML Schema specification is the most widely used specification for data typing.
- The XML Schema specification includes a basic type system for encoding most data types. This type system includes a long list of built-in simple types, including strings, floats, doubles, integers, time, and date.

Simple Types

string
Boolean
float ,double
decimal
binary
integer : values 126789, -1, 0, 1, 126789 and nonPositiveInteger
negativeInteger and long
int : values -1, 126789675
short : values 1, 12678
byte : values -1, 126
nonNegativeInteger
unsignedLong
unsignedInt
unsignedShort
unsignedByte
positiveInteger
date 1999-05-31
time 13:20:00.000, 13:20:00.000-05:00

WSDL Types -2

- The XML Schema specification also provides a facility for creating *new* data types. This is important if you want to create data types that go beyond what is already defined within the Schema.
- For example, a service might return an array of floats or a more complex stock quote object containing the high, low, and volume figures for a specific stock. Whenever your service goes beyond the simple XML Schema data types, you must declare these new data types within the WSDL types element.
- We can define arrays or more complex data types.

Complex types


```
<types>
 <xsd:schema
   targetNamespace="http://www.bajaj.com/schema"
  xmlns="http://www.w3.org/2001/XMLSchema">
 <xsd:complexType name="product">
  <xsd:sequence>
     <xsd:element name="name" type="xsd:string"/>
     <xsd:element name="description"
   type="xsd:string"/>
     <xsd:element name="price" type="xsd:double"/>
     <xsd:element name="SKU" type="xsd:string"/>
     </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
</types>
```

Complex Type Definition

- The sequence element specifies a list of sub elements and requires that these elements appear in the order specified.
- Simple types cannot have element children or attributes, whereas complex types can have element children and attributes.
- XML Schemas also enable to specify cardinality via the *minOccurs* and *maxOccurs* attributes. If these attributes are absent, they default to 1, requiring that each sub element must occur exactly one time.
- Each sub element can also have its own data type and can be mixed and matched string data types with double data types.

Using Complex Types

```
<message name="ProductRequest">
   <part name="sku" type="xsd:string"/>
 </message>
<message name=“ProductResponse">
  <part name="product" type="xsp:product"/>
</message>
 <portType name="Product_Port">
    <operation name="getProduct">
      <input message="tns:ProductRequest"/>
      <output message="tns:ProductResponse"/>
   </operation>
 </portType>
```

Creating new Complex Types

- The XML Schema requires that any new type you create be based on some existing data type.This existing base type is specified via the base attribute.
- You can then choose to modify this base type using one of two main methods: *extension* or *restriction*.
- Extension means that new data type will have all the properties of the base type plus some extra functionality. Restriction means that new data type will have all the properties of the base data type, but may have additional restrictions placed on the data.

Extension Mode


```
<xs:element name="employee" type="fullpersoninfo"/>
  <xs:complexType name="personinfo">
   <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/> </xs:sequence>
 </xs:complexType>
<xs:complexType name="fullpersoninfo"> <xs:complexContent>
  <xs:extension base="personinfo">
     <xs:sequence>
     <xs:element name="address" type="xs:string"/>
     <xs:element name="city" type="xs:string"/>
     <xs:element name="country" type="xs:string"/>
     </xs:sequence> </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Complex Types Issues

- The root schema element must include a namespace declaration for the SOAP encoding specification (http://schemas.xmlsoap.org/soap/encoding/). This is required because new data types extend the array definition specified by SOAP.
- The root schema element must specify a targetNamespace attribute. Any newly defined elements, such as our new array data types, will belong to the specified targetNamespace. To reference these data types later in the document, you must refer back to the same targetNamespace. :

Complex Elements

- A complex element is an XML element that contains other elements and/or attributes.
- There are four kinds of complex elements:
  - empty elements : <product pid="1345"/>
  - elements that contain only other elements

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

  - elements that contain only text
  - elements that contain both other elements and text

Enumerations Types

- The enumeration is used to constrain the values of almost every simple type, except the boolean type. It limits a simple type to a set of distinct values.
- For example, we can use the enumeration to define a new simple type called USState, derived from string, whose value must be one of the standard US state abbreviations.

Define Enumeration

```
<xsd:simpleType name="USState">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="AK"/>
      <xsd:enumeration value="AL"/>
      <xsd:enumeration value="AR"/>
   </xsd:restriction>
</xsd:simpleType>
```

Miscellaneous Types

- Miscellaneous data types are boolean, base64Binary, hexBinary, float, double, anyURI, QName, and NOTATION.
- The anyURI data type is used to specify a URI.
- NOTATION : Describes the format of non-XML data within an XML document.
- QName denotes XML Namespace

WSDL utility Elements

- WSDL specification defines the following utility elements:
- **documentation** : The documentation element is used to provide human-readable documentation and can be included inside any other WSDL element.
- **import** :The import element is used to import other WSDL documents or XML Schemas. This enables more modular WSDL documents. For example, two WSDL documents can import the same basic elements and yet include their own service elements to make the same service available at two physical addresses.

Include Element

- The ***include*** element allows to assemble the contents of a given WSDL 2.0 namespace from several WSDL 2.0 documents that define components for that namespace. The components defined by a given WSDL 2.0 document consist of those whose definitions are contained in the document and those that are defined by any WSDL 2.0 documents that are included in it via the include element.
- The effect of the include element is cumulative so that if document A includes document B and document B includes document C, then the components defined by document A consist of those whose definitions are contained in documents A, B, and C.

Import and Include

- In contrast, the *import* element does not define any components. Instead, the import element declares that the components whose definitions are contained in a WSDL 2.0 document for a given WSDL 2.0 namespace refer to components that belong to a different WSDL 2.0 namespace.
- If a WSDL 2.0 document contains definitions of components that refer to other namespaces, then those namespaces must be declared via an import element.
- The import element also has an optional location attribute that is a hint to the processor where the definitions of the imported namespace can be found. However, the processor may find the definitions by other means, for example, by using a catalog.

WSDL 2.0 Interface

- In WSDL 2.0 interface is basically a set of operations.
- <description targetNamespace="*xs:anyURI*" > . . .

<interface name="*xs:NCName*" extends="*list of xs:QName*"? styleDefault="*list of xs:anyURI*"? >

<fault name="*xs:NCName*" element="*xs:QName*"? > </fault>* <operation name="*xs:NCName*" pattern="*xs:anyURI*" style="*list of xs:anyURI*"? wsdlx:safe="*xs:boolean*"? >

<input messageLabel="*xs:NCName*"? element="*union of xs:QName, xs:Token*"? >

</input>*

<output messageLabel="*xs:NCName*"? element="*union of xs:QName, xs:Token*"? >

</output>*

<infault ref="*xs:QName*" messageLabel="*xs:NCName*"? > </infault>* <outfault ref="*xs:QName*" messageLabel="*xs:NCName*"? >

</outfault>*

</operation>*

</interface>* . . .

</description>

Interface Attributes

- The interface element has two optional attributes: *styleDefault* and *extends*.
- The styleDefault attribute can be used to define a default value for the style attributes of all operations under this interface
- The extends attribute is for inheritance.
- The *fault* element is used to declare faults that may occur during execution of operations of an interface. They are declared directly under interface, and referenced from operations where they apply, in order to permit reuse across multiple operations.