معايير خدمات الويب

Standards of Web Services

د. عبدالناصر ضياف

XML: eXtensible Markup Language (short)

- XML is a markup language like HTML
- XML was designed to store and transport data
- XML was designed to be both human-and-machine readable
- XML was designed to be self-descriptive
- XML plays an important role in many different IT systems
- XML is often used for distributing data over the Internet
- All types of software developers need to have a good understanding of XML

XML: Sample Document

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>أحمد</to>
  <from>جمال</from>
  <heading>تذكير</heading>
  <body>لا تنساني هذه الجمعة!</body>
</note>
```

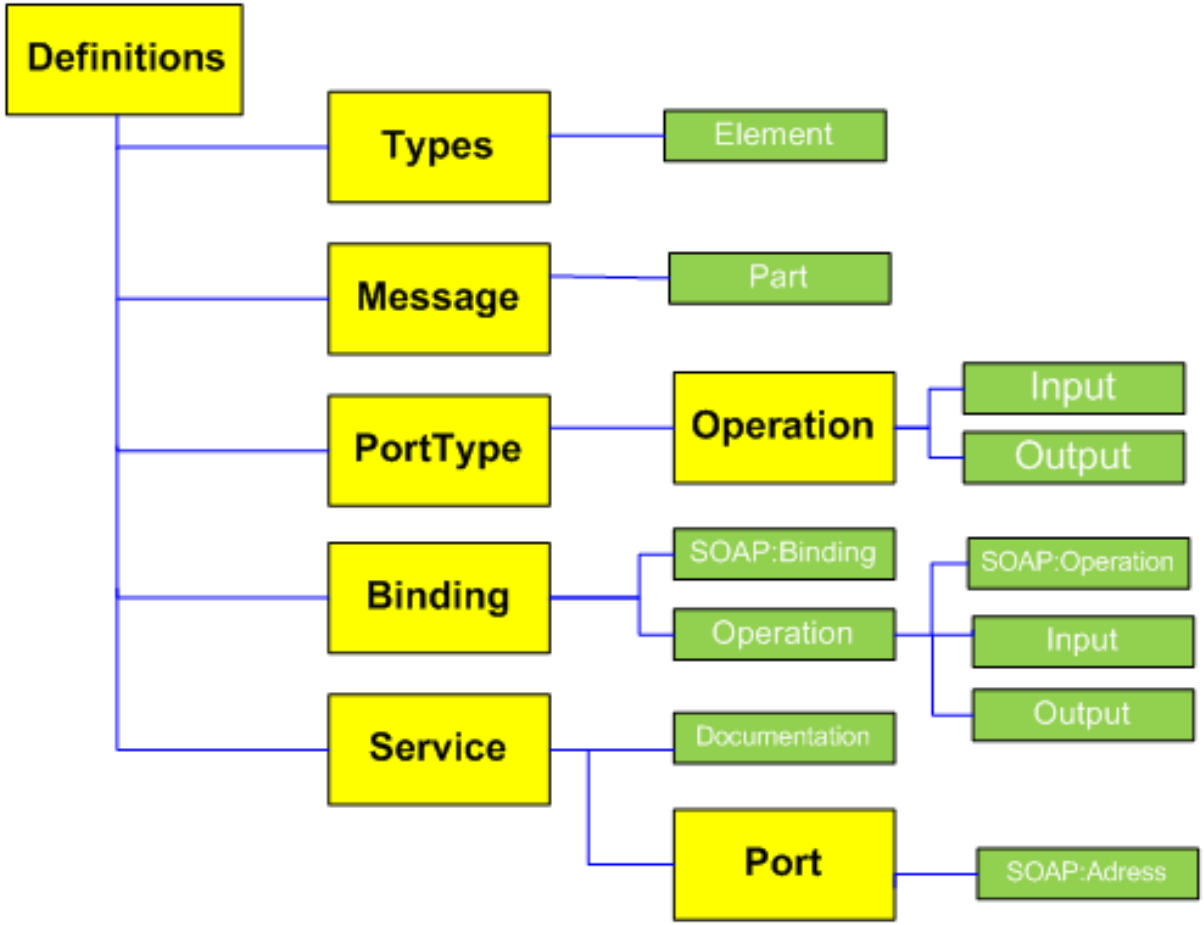
WSDL: Web Service Description Language

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

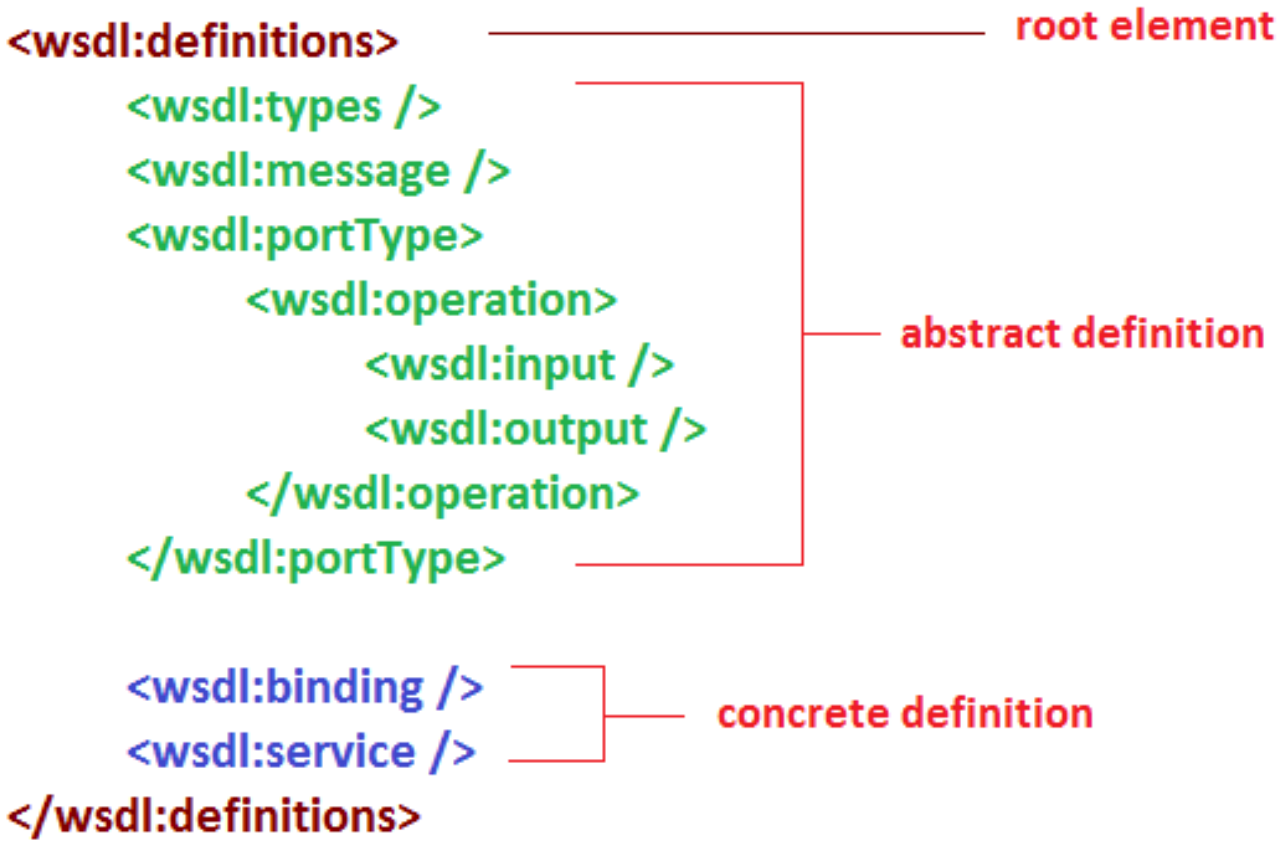
WSDL: Web Service Description Language (cont.)

- WSDL specifies what a request message must contain and what the response message will look like in a clear notation
- You can think of it as a contract between the WS and the client who wishes to use this service
- WSDL not only describes WSs and message contents, but also defines where the service is available and what communications protocol is used to talk to the service.
- **In general**, WSDL allows developers to describe WSs and their capabilities and locations in a standard manner.

WSDL Document Structure



WSDL Document Structure (tags)



WSDL Document Elements

- A WSDL document is a simple XML document describes a WS and its methods, and specifies the location of the service, using the following elements:
- **definitions**: It is the root element of all WSDL documents. It defines the name of the WS, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.
- **types**: a container for data type definitions used by the WS messages in the form of XML Schema systems (xs).
- **message**: an abstract definition of the data elements for each operation in the form of a message.
- **operation**: an abstract description of an action or method that accepts and processes messages.

WSDL Document Elements (cont.)

- **portType**: an abstract set of operations, and their messages, mapped to one or more end-points
- **binding**: a concrete protocol and data formats for the operations and messages defined for a particular port type.
- **port**: a single endpoint defined as a combination of a binding and a network address, providing the target address of the service communication.
- **service**: a collection of related end-points
- **Additional WSDL elements:**
- **documentation:** it is used to provide human-readable documentation inside any other WSDL element;
- **import:** it is used to import other WSDL documents or XML schemas.

WSDL Document Sample

```
<message name="GetStockPriceRequest">
      <part name="stock" type="xs:string"/>
 </message>
<message name="GetStockPriceResponse">
      <part name="value" type="xs:string"/>
 </message>

<portType name="StocksRates">
        <operation name="GetStockPrice">
              <input message="GetStockPriceRequest"/>
              <output message="GetStockPriceResponse"/>
        </operation>
</portType>
```

WSDL <definitions>

```
<definitions name="HelloService"
   targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   ..........................................
</definitions>
```

- This document is called *HelloService*
- The *targetNamespace* is an agreement of XML Schema that enables the WSDL document to refer to itself.
- Default namespace: (xmlns=http://schemas.xmlsoap.org/wsdl/)
- Numerous namespaces are specified to be used throughout the document.

WSDL <types>

- Type information is shared between the sender and the receiver.
- The *types* element describes all the data types used between the client and the server.
- WSDL is not tied exclusively to a specific typing system.
- WSDL uses the W3C XML Schema specification as its default choice to define data types.
  - W3C: The World Wide Web Consortium is the main international standards organization for the World Wide Web.
- If the service uses only XML Schema built-in simple types, such as strings and integers, then *types* element is not required.
- WSDL allows the types to be defined in separate elements so that the types are reusable with multiple web services.

WSDL <types> (a piece of code)

```
<types>
   <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">

      <element name="TradePriceRequest">
         <complexType>
            <all>
               <element name="tickerSymbol" type="string"/>
            </all>
         </complexType>
      </element>

      <element name="TradePrice">
         <complexType>
            <all>
               <element name="price" type="float"/>
            </all>
         </complexType>
      </element>

   </schema>
</types>
```

WSDL <message>

```
<message name="SayHelloRequest">
   <part name="firstName" type="xsd:string"/>
</message>

<message name="SayHelloResponse">
   <part name="greeting" type="xsd:string"/>
</message>
```

- It describes the data being exchanged between the WS provider and the consumers.
- Each WS has two messages: ***input*** and ***output***.
- Each message may contain ***<part>*** parameters, one for each parameter of the WS function.
- Each ***<part>*** parameter associates with a concrete type defined in the ***<types>*** element.

WSDL <portType>

```
<portType name="Hello_PortType">
   <operation name="sayHello">
      <input message="tns:SayHelloRequest"/>
      <output message="tns:SayHelloResponse"/>
   </operation>
</portType>
```

- The ***<portType>*** element combines multiple message elements to form a complete one-way or round-trip operation.
- This *portType* element defines a single operation, called *sayHello*.
- The ***input*** describes the received parameters and the ***output*** describes the return data.
- This operation consists of a single input message *SayHelloRequest* and an output message *SayHelloResponse*.

WSDL <portType> Patterns of Operations

WSDL supports four basic patterns of operation:

- **One-way**
  - The service receives a message.
  - The operation therefore has a single *input* element.

```
<wsdl:definitions .... >
   <wsdl:portType .... > *
      <wsdl:operation name="nmtoken">
         <wsdl:input name="nmtoken"? message="qname"/>
      </wsdl:operation>
   </wsdl:portType >
</wsdl:definitions>
```

WSDL <portType> Patterns of Operations

- Request-response
  - The service receives a message and sends back a response.
  - The operation therefore has one ***input*** element, followed by one ***output*** element.
  - To encapsulate errors, an optional ***fault*** element can also be specified.

```
<wsdl:definitions .... >
   <wsdl:portType .... > *
      <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
         <wsdl:input name="nmtoken"? message="qname"/>
         <wsdl:output name="nmtoken"? message="qname"/>
         <wsdl:fault name="nmtoken" message="qname"/>*
      </wsdl:operation>
   </wsdl:portType >
</wsdl:definitions>
```

WSDL <portType> Patterns of Operations

- Solicit-response
  - The service sends a message and receives a response.
  - The operation therefore has one ***output*** element, followed by one ***input*** element.
  - To encapsulate errors, an optional ***fault*** element can also be specified.

```
<wsdl:definitions .... >
   <wsdl:portType .... > *
      <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
         <wsdl:output name="nmtoken"? message="qname"/>
         <wsdl:input name="nmtoken"? message="qname"/>
         <wsdl:fault name="nmtoken" message="qname"/>*
      </wsdl:operation>
   </wsdl:portType >
</wsdl:definitions>
```

WSDL <portType> Patterns of Operations

- Notification
  - The service sends a message.
  - The operation therefore has a single ***output*** element.

```
<wsdl:definitions .... >
   <wsdl:portType .... > *
      <wsdl:operation name="nmtoken">
         <wsdl:output name="nmtoken"? message="qname"/>
      </wsdl:operation>
   </wsdl:portType >
</wsdl:definitions>
```

WSDL <binding>

- The ***<binding>*** element provides specific details on how a *portType* operation will be transmitted over the wire.
- The bindings can be made available via multiple transports including HTTP GET, HTTP POST, or SOAP.
- The bindings provide concrete information on what protocol is being used to transfer *portType* operations.
- For SOAP protocol, the binding is ***<soap:binding>***, and the transport is SOAP messages on top of HTTP protocol.
- You can specify multiple bindings for a single *portType*.
- The binding element has two attributes : ***name*** and ***type*** attribute where the latter points to the port of the binding.

```
<binding name="Hello_Binding" type="tns:Hello_PortType">
```

WSDL <binding> (a piece of code)

```
<binding name="Hello_Binding" type="tns:Hello_PortType">
   <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/ >
   <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>

      <input>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice" use="encoded"/>
      </input>

      <output>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice" use="encoded"/>
      </output>
   </operation>
</binding>
```

WSDL <binding> SOAP Binding

- It allows you to specify SOAP specific details including *SOAP headers*, SOAP *encodingStyle*, and the *SOAPAction* HTTP header.
  - ***soap:binding***: indicates that the binding will be made available via SOAP. The *style* and *transport* attributes indicate the overall style of the SOAP message format and the transport of the SOAP messages, respectively.
  - ***soap:operation***: indicates the binding of a specific operation to a specific SOAP implementation. The *soapAction* attribute specifies the SOAPAction HTTP header be used for identifying the service.
  - ***soap:body***: specifies how the SOAP Body should be constructed in the SOAP message for an operation *input* and *output*. '*literal*' means no encoding.

WSDL <port>

- The **<*binding*>** element defines an individual endpoint by defining a single address for a binding.
- The ***name*** attribute provides a unique name among all ports defined within the enclosing WSDL document.
- The ***binding*** attribute refers to the binding using the linking rules defined by WSDL.
- A port MUST NOT specify more than one address.
- A port MUST NOT specify any binding information other than address information.

WSDL <service>

```
<service name="Hello_Service">
   <documentation>WSDL File for HelloService</documentation>
   <port binding="tns:Hello_Binding" name="Hello_Port">
      <soap:address
         location="http://www.examples.com/SayHello/">
   </port>
</service>
```

- The ***<service>*** element defines the ports supported by the WS.
- The service element is a collection of ports.

Making A Service Available

- In order for someone to use your WS, he/she has to know about it.
- To allow users to discover a WS, it is published to a registry (UDDI).
- To allow users to interact with a WS, you must publish a description of its interface (methods & arguments).
- This is done using WSDL.

Making A Service Available (cont.)

- Once you have published a description of your WS, you must have a host set up to serve it.
- A WS is often used to deliver services (although custom application-to-application communication is also possible).