**WSDL**

|  |  |  |
| --- | --- | --- |
| WSDL |  | **Web Services Description Language is the standard format for describing a web service in XML format.**  **In this tutorial you will learn what is WSDL and Why and How to use it.**  **WSDL is very easy to learn and very important for Web Services** |

## WSDL Abstract:

* WSDL stands for Web Services Description Language
* WSDL is an XML based protocol for information exchange in decentralized and distributed environments.
* WSDL is the standard format for describing a web service.
* WSDL definition describes 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 UDDI, an XML-based worldwide business registry.
* WSDL is the language that UDDI uses.
* WSDL was developed jointly by Microsoft and IBM.
* WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'

## WSDL Usage:

WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

## History of WSDL

WSDL 1.1 was submitted as a W3C Note by Ariba, IBM and Microsoft for describing services for the W3C XML Activity on XML Protocols in March 2001.

WSDL 1.1 has not been endorsed by the World Wide Web Consortium (W3C), however it has just (May 11th, 2005) released a draft for version 2.0, that will be a recommendation (an official standard), and thus endorsed by the W3C.

## Our Recomedation:

We recommend to study the following subjects alongwith WSDL

* XML, XML Namespaces and XML Schema
* [UDDI](http://www.tutorialspoint.com/uddi)
* [Web Services](http://www.tutorialspoint.com/webservices)
* [SOAP](http://www.tutorialspoint.com/soap)

|  |  |
| --- | --- |
| WSDL breaks down Web services into three specific, identifiable elements that can be combined or reused once defined.  Three major elements of WSDL that can be defined separately and they are:   * Types * Operations * Binding   A WSDL document has various elements, but they are contained within these three main elements, which can be developed as separate documents and then they can be combined or reused to form complete WSDL files.  Following are the elements of WSDL document. Within these elements are further subelements, or parts:   * **Definition:** 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. * **Data types:** the data types - in the form of XML schemas or possibly some other mechanism - to be used in the messages * **Message:** an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation. * **Operation:** the abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message * **Port type :** an abstract set of operations mapped to one or more end points, defining the collection of operations for a binding; the collection of operations, because it is abstract, can be mapped to multiple transports through various bindings. * **Binding:** the concrete protocol and data formats for the operations and messages defined for a particular port type. * **Port:** a combination of a binding and a network address, providing the target address of the service communication. * **Service:** a collection of related end points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.   In addition to these major elements, the WSDL specification also defines the following utility elements:   * **Documentation:** element is used to provide human-readable documentation and can be included inside any other WSDL element. * **Import:** element is used to import other WSDL documents or XML Schemas.   **NOTE:** WSDL parts usually are generated automatically using Web services-aware tools. The WSDL Document Structure The main structure of a WSDL document looks like this:   |  | | --- | | <definitions>  <types>  **definition of types........**  </types>  <message>  **definition of a message....**  </message>  <portType>  <operation>  **definition of a operation.......**  </operation>  </portType>  <binding>  **definition of a binding....**  </binding>  <service>  **definition of a service....**  </service>  </definitions> |   A WSDL document can also contain other elements, like extension elements and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.  Proceed further to analyze an example of WSDL Document. |

Following is the WSDL file that is provided to demonstrate a simple WSDL program.

Assuming the service provides a single publicly available function, called *sayHello*. This function expects a single string parameter and returns a single string greeting. For example if you pass the parameter *world* then service function *sayHello* returns the greeting, "Hello, world!".

## Content of HelloService.wsdl file

|  |
| --- |
| <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">    <message name="SayHelloRequest">  <part name="firstName" type="xsd:string"/>  </message>  <message name="SayHelloResponse">  <part name="greeting" type="xsd:string"/>  </message>  <portType name="Hello\_PortType">  <operation name="sayHello">  <input message="tns:SayHelloRequest"/>  <output message="tns:SayHelloResponse"/>  </operation>  </portType>  <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>  <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>  </definitions> |

# Analysis of the Example

* **Definition :** HelloService
* **Type :** Using built-in data types and they are defined in XMLSchema.
* **Message :**
  1. sayHelloRequest : firstName parameter
  2. sayHelloresponse: greeting return value
* **Port Type:** sayHello **operation** that consists of a request and response service.
* **Binding:** Direction to use the SOAP HTTP transport protocol.
* **Service:** Service available at http://www.examples.com/SayHello/.
* **Port:** Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed.

A detailed description of these elements is given in subsequent sections of the tutorial

|  |  |
| --- | --- |
| The **<definition>** element must be the root element of all WSDL documents. It defines the name of the web service.  Here is the example piece of code from last session which uses *definition* element.   |  | | --- | | <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> |   From the above example we can conclude the followings points:   * The definitions element is a container of all the other elements. * The definitions element specifies that this document is the *HelloService*. * The definitions element specifies a *targetNamespace* attribute. The *targetNamespace* is a convention of XML Schema that enables the WSDL document to refer to itself. In this example we have specified a *targetNamespace* of http://www.examples.com/wsdl/HelloService.wsdl. * The definition element specifies a default namespace: xmlns=http://schemas.xmlsoap.org/wsdl/. All elements without a namespace prefix, such as *message* or *portType*, are therefore assumed to be part of the default WSDL namespace. * It also specifies numerous namespaces that will be used throughout the remainder of the document.   **NOTE:** The namespace specification does not require that the document actually exist at the given location. The important point is that you specify a value that is unique, different from all other namespaces that are defined. |
| A Web service needs to define its inputs and outputs and how they are mapped into and out of services. WSDL <types> element take care of defining the data types that are used by the web service. Types are XML documents, or document parts.  Here is a piece of code taken from W3C specification. This code depicts how a types element can be used within a WSDL.   * The types element describes all the data types used between the client and 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. * 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.  |  | | --- | | <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> |   Data types address the problem of how to identify the data types and formats you intend to use with your Web services. Type information is shared between sender and receiver. The recipients of messages therefore need access to the information you used to encode your data and must understand how to decode the data. |

* The **<message>** element describes the data being exchanged between the Web service providers and consumers.
* Each Web Service has two messages: input and output.
* The input describes the parameters for the Web Service and the output describes the return data from the Web Service.
* Each message contains zero or more **<part>** parameters, one for each parameter of the Web Service's function.
* Each **<part>** parameter associates with a concrete type defined in the **<types>** container element.

Lets take a piece of code from the Example Session:

|  |
| --- |
| <message name="SayHelloRequest">  <part name="firstName" type="xsd:string"/>  </message>  <message name="SayHelloResponse">  <part name="greeting" type="xsd:string"/>  </message> |

Here, two message elements are defined. The first represents a request message *SayHelloRequest*, and the second represents a response message *SayHelloResponse*.

Each of these messages contains a single part element. For the request, the part specifies the function parameters; in this case, we specify a single firstName parameter. For the response, the part specifies the function return values; in this case, we specify a single greeting return value.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| The **<portType>** element combines multiple message elements to form a complete oneway or round-trip operation.  For example, a **<portType>** can combine one request and one response message into a single request/response operation. This is most commonly used in SOAP services. A portType can define multiple operations.  Lets take a piece of code from the Example Session:   |  | | --- | | <portType name="Hello\_PortType">  <operation name="sayHello">  <input message="tns:SayHelloRequest"/>  <output message="tns:SayHelloResponse"/>  </operation>  </portType> |  * The portType element defines a single operation, called *sayHello*. * The operation itself consists of a single input message *SayHelloRequest* * The operation itself consists of a single output message *SayHelloResponse*  Patterns of Operation WSDL supports four basic patterns of operation: One-way : The service receives a message. The operation therefore has a single *input* element. The grammar for a one-way operation is:   |  | | --- | | <wsdl:definitions .... > <wsdl:portType .... > \*  <wsdl:operation name="nmtoken">  <wsdl:input name="nmtoken"? message="qname"/>  </wsdl:operation>  </wsdl:portType >  </wsdl:definitions> |  Request-response: The service receives a message and sends 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. The grammar for a request-response operation is:   |  | | --- | | <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> |  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. The grammar for a solicit-response operation is:   |  | | --- | | <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> |  Notification : The service sends a message. The operation therefore has a single *output* element. Following is the grammer for a notification operation:   |  | | --- | | <wsdl:definitions .... >  <wsdl:portType .... > \*  <wsdl:operation name="nmtoken">  <wsdl:output name="nmtoken"? message="qname"/>  </wsdl:operation>  </wsdl:portType >  </wsdl:definitions> | |

* The **<service>** element defines the ports supported by the Web service. For each of the supported protocols, there is one port element. The service element is a collection of ports.
* Web service clients can learn from the service element where to access the service, through which port to access the Web service, and how the communication messages are defined.
* The service element includes a documentation element to provide human-readable documentation.

Here is a pice of code from Example Session:

|  |
| --- |
| <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 binding attributes of por element associate the address of the service with a binding element defined in the Web service. In this example this is *Hello\_Binding*

|  |
| --- |
| <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> |

You have leanred about WSDL, now the next step is to learn about SOAP, UDDI and Web Services.

**Web Services**

Web services are open standard ( XML, SOAP, HTTP etc.) based Web applications that interact with other web applications for the purpose of exchanging data.

To learn more about Web Services visit [Web Services Tutorial](http://www.tutorialspoint.com/webservices/index.htm)

**UDDI**

UDDI is an XML-based standard for describing, publishing, and finding Web services.

To learn more about UDDI visit [UDDI Tutorial](http://www.tutorialspoint.com/uddi/index.htm)

**SOAP**

SOAP is a simple XML-based protocol that allows applications to exchange information over HTTP.

To learn more about SOAP visit [SOAP](http://www.tutorialspoint.com/soap/index.htm)