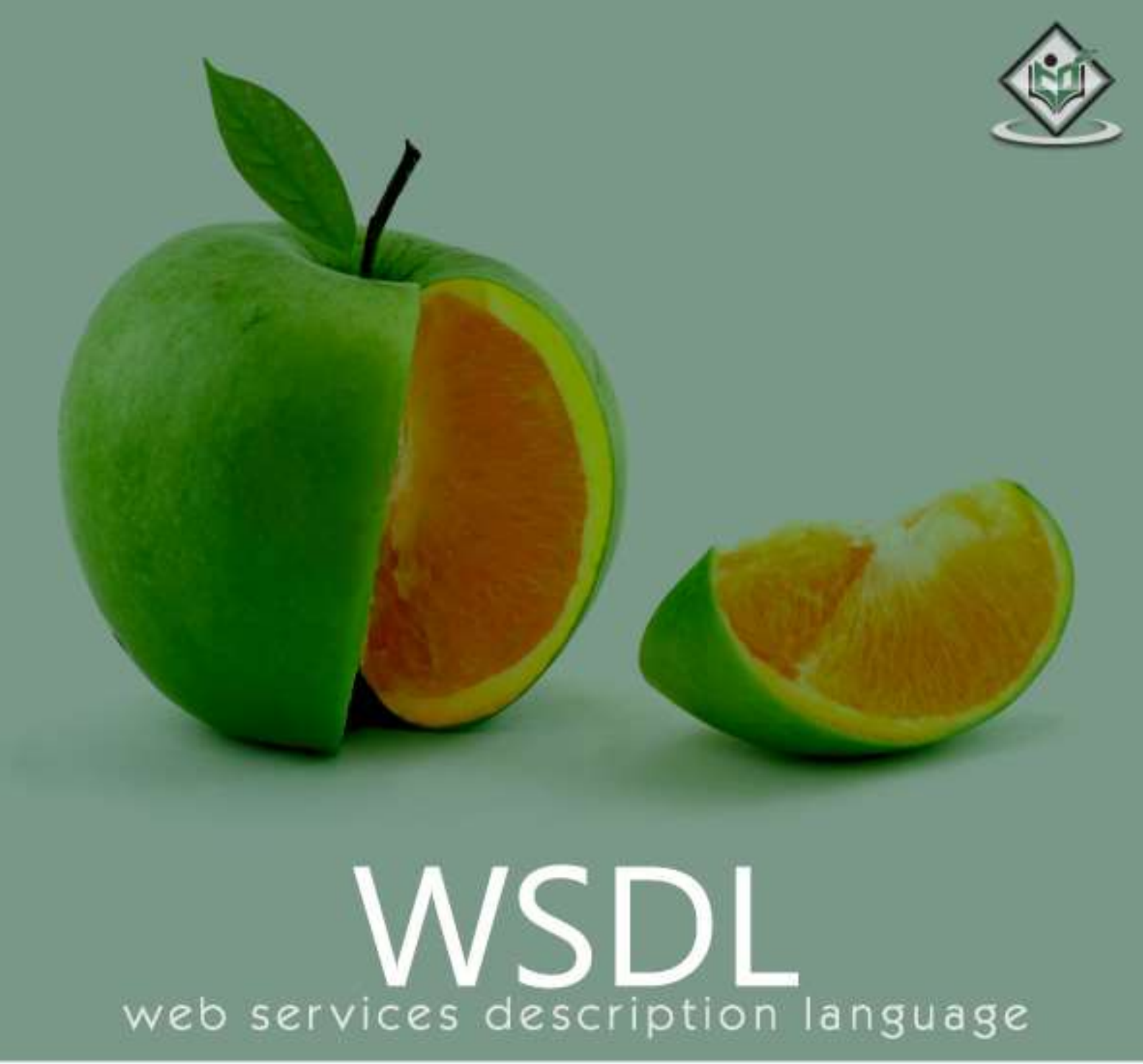

WSDL

web services description language

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

This is a brief tutorial that explains how to use WSDL to exchange information in a distributed environment. It uses plenty of examples to show the functionalities of the elements used in a WSDL file such as definitions, types, message, port type, binding, port, and service.

Audience

This tutorial is going to help all those readers who want to learn the basics of WSDL and use its features to interface with XML-based services.

Prerequisites

WSDL is often used in combination with SOAP and XML Schema. Hence, you need to have a basic understanding of XML Schema, XML namespace, and web services in order to make the most of this tutorial.

Copyright & Disclaimer

Table of Contents

1. WSDL – Introduction

WSDL stands for Web Services Description Language. It is the standard format for describing a web service. WSDL was developed jointly by Microsoft and IBM.

Features of WSDL

- 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 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 released a draft for version 2.0 that will be a recommendation (an official standard), and thus endorsed by the W3C.

2. WSDL – Elements

WSDL breaks down web services into three specific, identifiable elements that can be combined or reused once defined.

The three major elements of WSDL that can be defined separately 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.

WSDL Elements

A WSDL document contains the following elements:

- **Definition:** It is 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 to be used in the messages are in the form of XML schemas.
- **Message**: It is 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**: It is 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:** It is an abstract set of operations mapped to one or more end-points, defining the collection of operations for a binding; the collection of operations, as it is abstract, can be mapped to multiple transports through various bindings.
- **Binding:** It is the concrete protocol and data formats for the operations and messages defined for a particular port type.
- **Port:** It is a combination of a binding and a network address, providing the target address of the service communication.
- **Service:** It is 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:** This element is used to provide human-readable documentation and can be included inside any other WSDL element.
- **Import:** This element is used to import other WSDL documents or XML Schemas.

NOTE: WSDL parts are usually 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.

3. WSDL – Example

Given below is a WSDL file that is provided to demonstrate a simple WSDL program.

Let us assume 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 the service function *sayHello* returns the greeting, "Hello, world!".

Example

Contents 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>
```

Example Analysis

- **Definitions** : HelloService
- **Type**: Using built-in data types and they are defined in XMLSchema.
- **Message** :
  - sayHelloRequest : firstName parameter
  - sayHelloresponse: greeting return value
- **Port Type**: sayHello operation that consists of a request and a 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.

4. WSDL – Definitions element

The **<definitions>** element must be the root element of all WSDL documents. It defines the name of the web service.

Here is the piece of code from the last chapter that uses the *definitions* 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 that *definitions*:

- is a container of all the other elements.
- specifies that this document is called *HelloService*.
- 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.
- 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 a part of the default WSDL namespace.
- specifies numerous namespaces that are used throughout the remainder of the document.

NOTE: The namespace specification does not require the document to be present at the given location. The important point is that you specify a value that is unique, different from all other namespaces that are defined.

5. WSDL – Types Element

A web service needs to define its inputs and outputs and how they are mapped into and out of the service. WSDL **<types>** element takes care of defining the data types that are used by the web service. Types are XML documents or document parts.

- 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.
- 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.

Here is a piece of code taken from W3C specification. This code depicts how a *types* element can be used within a WSDL.

```
<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 identifying the data types and the formats you intend to use with your web services. Type information is shared between the sender and the 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.

6. WSDL – Message element

The **<message>** element describes the data being exchanged between the web service providers and the 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 function.
- Each **<part>** parameter associates with a concrete type defined in the **<types>** container element.

Let us take a piece of code from the WSDL Example chapter:

```
<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.

7. WSDL – PortType Element

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. This is most commonly used in SOAP services. A portType can define multiple operations.

Let us take a piece of code from the WSDL Example chapter:

```
<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 consists of a single input message *SayHelloRequest* and an 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 grammar for a notification operation:

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

8. WSDL – Binding Element

The **<binding>** element provides specific details on how a *portType* operation will actually 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.
- The bindings provide information where the service is located.
- 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*.

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

The *name* attribute defines the name of the binding, and the *type* attribute points to the port for the binding, in this case the "tns:Hello_PortType" port.

SOAP Binding

WSDL 1.1 includes built-in extensions for SOAP 1.1. It allows you to specify SOAP-specific details including SOAP headers, SOAP encoding styles, and the SOAPAction HTTP header. The SOAP extension elements include the following:

- soap:binding
- soap:operation
- soap:body

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.

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:operation

This element indicates the binding of a specific operation to a specific SOAP implementation. The *soapAction* attribute specifies that the SOAPAction HTTP header be used for identifying the service.

soap:body

This element enables you to specify the details of the input and output messages. In the case of HelloWorld, the body element specifies the SOAP encoding style and the namespace URN associated with the specified service.

Here is the piece of code from the Example chapter:

```
<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>
```

9. WSDL – Ports

A **<port>** element defines an individual endpoint by specifying a single address for a binding.

Here is the grammar to specify a port:

```
<wsdl:definitions .... >
    <wsdl:service .... > *
        <wsdl:port name="nmtoken" binding="qname"> *
        <-- extensibility element (1) -->
       </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

- The port element has two attributes: *name* and *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.
- Binding extensibility elements are used to specify the address information for the port.
- A port MUST NOT specify more than one address.
- A port MUST NOT specify any binding information other than address information.

Here is a piece of code from the Example chapter:

```
<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>
```

10. WSDL – Service Element

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 the following 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 piece of code from the Example chapter:

```
<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 *port* 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>
```

11. WSDL – Summary

We have covered the basics of WSDL in this tutorial. The next step is to learn SOAP, UDDI, and Web Services.

Web Services

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

To learn more about Web Services, visit Web Services Tutorial.

UDDI

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

To learn more about UDDI, visit UDDI Tutorial.

SOAP

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

To learn more about SOAP, visit SOAP Tutorial.