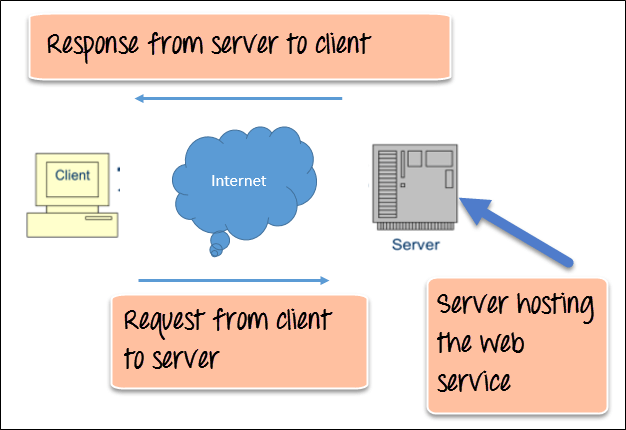
## **What are Web Services? Architecture, Types, Example:**

## **What is Web Service?**

**Web service** is a standardized medium to propagate communication between the client and server applications on the WWW (World Wide Web). A web service is a software module that is designed to perform a certain set of tasks.

* Web services in cloud computing can be searched for over the network and can also be invoked accordingly.
* When invoked, the web service would be able to provide the functionality to the client, which invokes that web service.



The above diagram shows a very simplistic view of how a web service would actually work. The client would invoke a series of web service calls via requests to a server which would host the actual web service.

These requests are made through what is known as remote procedure calls. Remote Procedure Calls(RPC) are calls made to methods which are hosted by the relevant web service.

As an example, Amazon provides a web service that provides prices for products sold online via amazon.com. The front end or presentation layer can be in .Net or[Java](https://www.guru99.com/java-tutorial.html)but either programming language would have the ability to communicate with the web service.

The main component of a web service design is the data which is transferred between the client and the server, and that is XML. XML (Extensible markup language) is a counterpart to HTML and easy to understand the intermediate language that is understood by many programming languages.

So when applications talk to each other, they actually talk in XML. This provides a common platform for application developed in various programming languages to talk to each other.

Web services use something known as SOAP (Simple Object Access Protocol) for sending the XML data between applications. The data is sent over normal HTTP. The data which is sent from the web service to the application is called a SOAP message. The SOAP message is nothing but an XML document. Since the document is written in XML, the client application calling the web service can be written in any programming language.

## Why do you need a Web Service?

Modern day business applications use variety of programming platforms to develop web-based applications. Some applications may be developed in Java, others in .Net, while some other in Angular JS, Node.js, etc.

Most often than not, these heterogeneous applications need some sort of communication to happen between them. Since they are built using different development languages, it becomes really difficult to ensure accurate communication between applications.

Here is where web services come in. Web services provide a common platform that allows multiple applications built on various [programming languages](https://www.guru99.com/best-programming-language.html) to have the ability to communicate with each other.

## Type of Web Service

There are mainly two types of web services.

1. SOAP web services.
2. RESTful web services.

In order for a web service to be fully functional, there are certain components that need to be in place. These components need to be present irrespective of whatever development language is used for programming the web service.

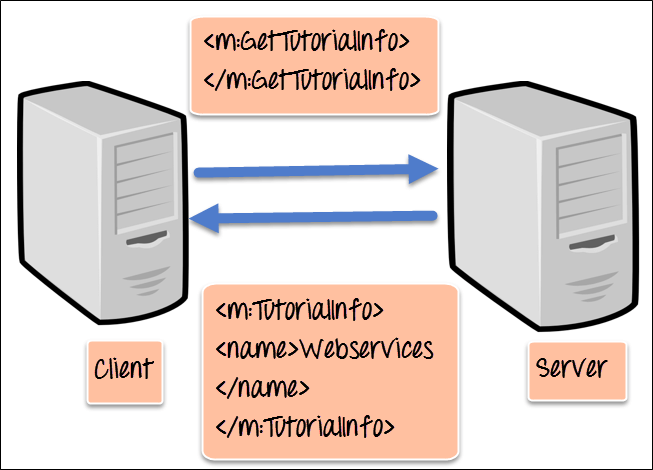
Let’s look at these components in more detail.

**SOAP (Simple Object Access Protocol)**

SOAP is known as a transport-independent messaging protocol. SOAP is based on transferring XML data as SOAP Messages. Each message has something which is known as an XML document. Only the structure of the XML document follows a specific pattern, but not the content. The best part of Web services and SOAP is that its all sent via HTTP, which is the standard web protocol.

Here is what a SOAP message consists of

* Each SOAP document needs to have a root element known as the <Envelope> element. The root element is the first element in an XML document.
* The “envelope” is in turn divided into 2 parts. The first is the header, and the next is the body.
* The header contains the routing data which is basically the information which tells the XML document to which client it needs to be sent to.
* The body will contain the actual message.



**WSDL (Web services description language)**

**A web service cannot be used if it cannot be found**. The client invoking the web service should know where the web service actually resides.

Secondly, the client application needs to know what the web service actually does, so that it can invoke the right web service. This is done with the help of the WSDL, known as the Web services description language. The WSDL file is again an XML-based file which basically tells the client application what the web service does. By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.

#### Web Service Example

A Web services example of a WSDL file is given below.

<definitions>

<message name="TutorialRequest">

<part name="TutorialID" type="xsd:string"/>

</message>

<message name="TutorialResponse">

<part name="TutorialName" type="xsd:string"/>

</message>

<portType name="Tutorial\_PortType">

<operation name="Tutorial">

<input message="tns:TutorialRequest"/>

<output message="tns:TutorialResponse"/>

</operation>

</portType>

<binding name="Tutorial\_Binding" type="tns:Tutorial\_PortType">

<soap:binding style="rpc"

transport="http://schemas.xmlsoap.org/soap/http"/>

<operation name="Tutorial">

<soap:operation soapAction="Tutorial"/>

<input>

<soap:body

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"

namespace="urn:examples:Tutorialservice"

use="encoded"/>

</input>

<output>

<soap:body

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"

namespace="urn:examples:Tutorialservice"

use="encoded"/>

</output>

</operation>

</binding>

</definitions>

The important aspects to note about the above WSDL declaration examples of web services are as follows:

1. **<message>** – The message parameter in the WSDL definition is used to define the different data elements for each operation performed by the web service. So in the web services examples above, we have 2 messages which can be exchanged between the web service and the client application, one is the “TutorialRequest”, and the other is the “TutorialResponse” operation. The TutorialRequest contains an element called “TutorialID” which is of the type string. Similarly, the TutorialResponse operation contains an element called “TutorialName” which is also a type string.
2. **<portType>** – This actually describes the operation which can be performed by the web service, which in our case is called Tutorial. This operation can take 2 messages; one is an input message, and the other is the output message.
3. **<binding>** – This element contains the protocol which is used. So in our case, we are defining it to use http (**http://schemas.xmlsoap.org/soap/http**). We also specify other details for the body of the operation, like the namespace and whether the message should be encoded.

We will discuss “WDSL” in detail in this [tutorial](https://www.guru99.com/wsdl-web-services-description-language.html).

**Universal Description, Discovery, and Integration (UDDI)**

UDDI is a standard for describing, publishing, and discovering the web services that are provided by a particular service provider. It provides a specification which helps in hosting the information on web services.

Now we discussed in the previous topic about WSDL and how it contains information on what the Web service actually does. But how can a client application locate a WSDL file to understand the various operations offered by a web service? So UDDI is the answer to this and provides a repository on which WSDL files can be hosted. So the client application will have complete access to the UDDI, which acts as a database containing all the WSDL files.

**Just as a telephone directory has the name, address and telephone number of a particular person, the same way the UDDI registry will have the relevant information for the web service**. So that a client application knows, where it can be found.

## Web Services Advantages

We already understand why web services came about in the first place, which was to provide a platform which could allow different applications to talk to each other.

But let’s look at the list of web services advantages for why it is important to use web services.

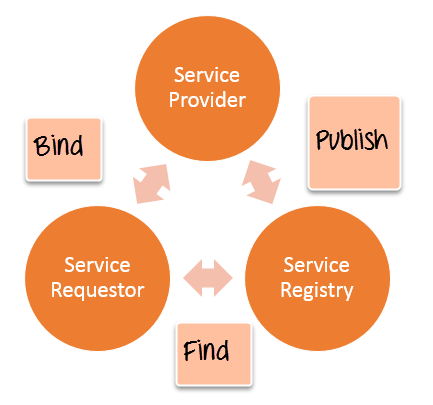
1. **Exposing Business Functionality on the network** – A web service is a unit of managed code that provides some sort of functionality to client applications or end users. This functionality can be invoked over the HTTP protocol which means that it can also be invoked over the internet. Nowadays all applications are on the internet which makes the purpose of Web services more useful. That means the web service can be anywhere on the internet and provide the necessary functionality as required.
2. **Interoperability amongst applications** – Web services allow various applications to talk to each other and share data and services among themselves. 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.
3. **A Standardized Protocol which everybody understands** – Web services use 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.
4. **Reduction in cost of communication** – Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services.

## Web Services Architecture

Every framework needs some sort of architecture to make sure the entire framework works as desired, similarly, in web services. The **Web Services Architecture** consists of three distinct roles as given below:

1. **Provider** – The provider creates the web service and makes it available to client application who want to use it.
2. **Requestor** – A requestor is nothing but the client application that needs to contact a web service. The client application can be a .Net, Java, or any other language based application which looks for some sort of functionality via a web service.
3. **Broker** – The broker is nothing but the application which provides access to the UDDI. The UDDI, as discussed in the earlier topic enables the client application to locate the web service.

The diagram below showcases how the Service provider, the Service requestor and Service registry interact with each other.

Web Services Architecture

1. **Publish** – A provider informs the broker (service registry) about the existence of the web service by using the broker’s publish interface to make the service accessible to clients
2. **Find** – The requestor consults the broker to locate a published web service
3. **Bind** – With the information it gained from the broker(service registry) about the web service, the requestor is able to bind, or invoke, the web service.

## Web service Characteristics

Web services have the following special behavioral characteristics:

1. **They are XML-Based** – Web Services uses XML to represent the data at the representation and data transportation layers. Using XML eliminates any networking, operating system, or platform sort of dependency since XML is the common language understood by all.
2. **Loosely Coupled** – Loosely coupled means that the client and the web service are not bound to each other, which means that even if the web service changes over time, it should not change the way the client calls the web service. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.
3. **Synchronous or Asynchronous functionality** – Synchronicity refers to the binding of the client to the execution of the service. In synchronous operations, the client will actually wait for the web service to complete an operation. An example of this is probably a scenario wherein a database read and write operation are being performed. If data is read from one database and subsequently written to another, then the operations have to be done in a sequential manner. Asynchronous operations allow a client to invoke a service and then execute other functions in parallel. This is one of the common and probably the most preferred techniques for ensuring that other services are not stopped when a particular operation is being carried out.
4. **Ability to support Remote Procedure Calls (RPCs)** – Web services enable clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.
5. **Supports Document Exchange** – One of the key benefits of XML is its generic way of representing not only data but also complex documents. These documents can be as simple as representing a current address, or they can be as complex as representing an entire book.