**Web Service**

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. Because all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications.

**Web Services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains. These applications can be** local, distributed, or Web-based**.** Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.

Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. **Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.**

To summarize, a complete web service is, therefore, any service that:

Is available over the Internet or private (intranet) networks

Uses a standardized XML messaging system

Is not tied to any one operating system or programming language

Is self-describing via a common XML

Is discoverable via a simple find mechanism

Components of Web Services:

The basic Web services platform is XML + HTTP. All the standard Web Services works using following components

WSDL (Web Services Description Language)

UDDI (Universal Description, Discovery and Integration)

SOAP (Simple Object Access Protocol)

Benefits of using Web Services

Exposing the existing function on to network:

A Web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. So, Web Services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.

Connecting Different Applications ie Interoperability:

Web Services allows different applications to share data and services each other or among themselves. Other applications can also use the services of the web services. For example VB or .NET application can talk to java web services and vice versa. So, Web services is used to make the application platform and technology independent.

Standardized Protocol:

Web Services uses standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) uses the well defined protocol in the Web Services protocol stack. This standardization of protocol stack gives the business many advantages like wide range of choices, reduction in the cost due to competition and increase in the quality.

Low Cost of communication:

Web Services uses SOAP over HTTP protocol for the communication, so you can use your existing low cost internet for implementing Web Services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Beside SOAP over HTTP, Web Services can also be implemented on other reliable transport mechanisms like FTP etc.

Web services have special behavioral characteristics

XML-based

Web Services uses XML based data representation and data transportation layers. Using XML eliminates any networking, operating system, or platform binding. So Web Services based applications are highly interoperable application at their core level.

Loosely coupled

A consumer of a web service is not tied to that web service directly. The web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must also be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

Coarse-grained

Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service. Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.

Ability to be synchronous or asynchronous

Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions. Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

Supports Remote Procedure Calls (RPCs)

Web services allow 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. Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms. A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

Supports document exchange

One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be simple, such as when representing a current address, or they can be complex, representing an entire book or request for quotation, RFQ. Web services support the transparent exchange of documents to facilitate business integration.

<http://www.tutorialspoint.com/webservices>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**