WEB SERVICE

WHAT IS WEB SERVICE?

- A web service is any piece of software that makes itself available over the internet and uses a standardized messaging system.
- 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.
- 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.

How Web Services Work

- Figure 1 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 that 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 that 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 but either programming language would have the ability to communicate with the web service.
- So when applications talk to each other, they actually talk in XML. This provides a common platform for applications developed in various programming languages to talk to each other.

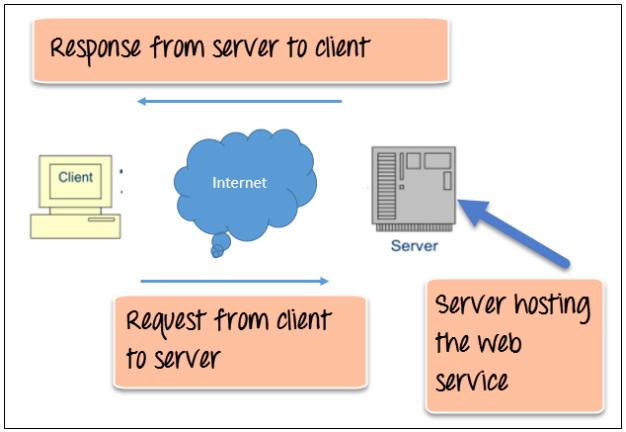

Figure 1. How web services work

Benefits of Using Web Services

- **Exposing the Existing Function on the 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. Web services allow exposing the functionality of existing code over the network. Once it is exposed on the network, other applications can use the functionality of your program.
- **Interoperability**
  Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use web services. For example, a VB or .NET application can talk to Java web services and vice versa. Web services are used to make the application platform and technology independent.
- **Standardized Protocol**
  Web services use standardized industry-standard protocol for communication. All four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack. This standardization of protocol stack gives the business many

advantages such as a wide range of choices, reduction in the cost due to competition, and increase in quality.

- **Low Cost Communication**
  Web services use SOAP over HTTP protocol, 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. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.

Characteristics of Web Services

- **XML-Based**

  Web services use XML at data representation and data transportation layers. Using XML eliminates any networking, operating system, or platform binding. Web services-based applications are highly interoperable 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 be updated.

- **Coarse-Grained**

  Object-oriented technologies such as Java expose their services through individual methods. An individual method is to find an operation to provide any useful capability at a corporate level. Businesses and the interfaces that Java 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 results at a later point in time, while synchronous clients receive their results when the service has been 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.

- **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 as simple as representing a current address, or they can be as complex as representing an entire book or Request for Quotation (RFQ). Web services support the transparent exchange of documents to facilitate business integration.

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 applications 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.

Figure 2 showcases how the Service provider, the Service requestor and Service registry interact with each other.

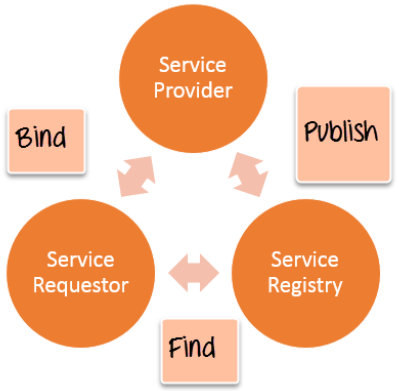

Figure 2. Web Services Architecture

4. **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
5. **Find** - The requestor consults the broker to locate a published web service
6. **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 SERVICES COMPONENTS

Over the past few years, three primary technologies have emerged as worldwide standards that make up the core of today's web services technology. SOAP, WSDL, and UDDI.

SOAP

SOAP is an XML-based protocol for exchanging information between computers.

- SOAP stands for Simple Object Access Protocol.
- It is a communication protocol that communicates between applications.
- It can run on any operating system.
- It describes how to encode an HTTP header and an XML file to make communication between two computers.
- It is a platform and language independent.
- It has the benefit to allow server firewalls.

WSDL

WSDL is an XML-based language for describing web services and how to access them.

- WSDL stands for Web Services Description Language.
- It is an XML-based language written in XML.
- It is used for describing the web services and the process of accessing them.
- It is an integral part of UDDI that facilitates businesses to be listed themselves and their services on the Internet.
- It navigates for individuals and other businesses to access the service.

UDDI

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

- It stands for Universal Description, Discovery and Integration.
- It is an open framework and platform-independent.
- It is seen with SOAP and WSDL as one of the three foundation standards of web services.
- It uses WSDL to describe interfaces to web services.
- It is a specification for a distributed registry of web services.

SOAP Web Services

- As discussed, SOAP is an XML-based protocol for application communication. Although it's definitely slower and more resource-heavy, it is similarly platform and language independent.

- In the Java ecosystem, Java EE provides the JAX-WS API to help you create SOAP-based web services.

- With JAX-WS, it can define a SOAP service in both an *RPC* or Document style. Both styles consist of a set of annotations to be applied to your classes, based on which the XML files are generated.

- This is an example of an RPC-style web service. First, create an interface or class with the proper annotations, which will declare the methods to be accessed by other applications:

```
@WebService

@SOAPBinding(style = SOAPBinding.Style.RPC)

public interface UserService {

    @WebMethod

    public void addUser(User user);

    @WebMethod

    public Users getUsers();

}
```

Two primary annotations here were used – *@WebService* to declare the service interface, and *@WebMethod* for each method to be exposed.

The *@SoapBinding* annotation specifies the style of web service. A Document-style service is declared in a similar manner**,** replacing the *@SoapBinding* annotation with:

```
@SOAPBinding(style = SOAPBinding.Style.Document)
```

The difference between the two styles is in the way the XML files are generated.

Finally, add an implementation class for the service interface**:**

```
@WebService(endpointInterface = "com.stackify.services.UserSe
rvice")

public class DefaultUserImpl implements UserService {

    ArrayList<User> usersList = new ArrayList<>();

    @Override

    public void addUser(User user) {

        usersList.add(user);

    }

    @Override

    public Users getUsers() {

        Users users = new Users();
```

```
        users.setUsers(usersList);

        return users;

    }

}
```

The implementing methods must be *public*, and must not be *static* or *final*. You can also make use of methods annotated with *@PostConstruct* and *@PreDestroy* for lifecycle event callbacks.

Note that creating the interface is optional for a JAX-WS web service implementation. The annotations can be added directly to the class, and JAX-WS will implicitly define a service endpoint interface.

Finally, to publish the web service, use the *Endpoint* class:

```
public class ServicePublisher {

    public static void main(String[] args) {

        Endpoint.publish("http://localhost:8080/users",new De
faultUserService());

    }

}
```

To run this application, the XML describes the endpoint, written in WSDL (Web Service Description Language) format, by accessing the URL:

```
http://localhost:8080/users?wsdl
```

REFERENCES:

Fastrack IT Academy (2019). *SAP Business One – Basic.* Fastrack IT Academy.

Java web services tutorial: Improve app communication and flexibility. Retrieved from: https://stackify.com/java-web-services/.

Lam, W. (2017). *Enterprise architecture and integration: Methods, implementation, and technologies.* Information Science Reference.

Learn web services. Retrieved from: https://www.tutorialspoint.com/webservices/web_services_components.htm.

What are web services? Architecture, type, example. Retrieved from: https://www.guru99.com/web-service-architecture.html.