Technology Digest

Bulletin of telecom technology

Issue, February 2018

Web services

In this paper

What are Web Services

Web services are collection of open protocols and standards used to exchange data between applications or systems. W3C (World Wide Web Consortium) defines web services as “a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL – Web Service Definition Language). Interaction with the web services are done in a manner prescribed by their description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.”

For data exchange over computer networks, software applications written in various programming languages and running on various platforms can use web services in a manner like inter-process communication within a single computer. Web services are an open standard (XML – eXtensible Markup Language, SOAP – Simple Object Access Protocol, JSON – JavaScript Object Notation, REST – REpresentational State Transfer, HTTP – HyperText Transfer Protocol, etc.) based web applications that interact with other web applications for exchanging data.

In this issue of technology digest, the brief of web services and there uses are given in the simple and understandable form.

Background

Web services evolved from the RPC (Remote Procedure Call) mechanism in DCE (Distributed Computing Environment), a software development framework from the early 1990s. DCE includes a distributed file system (DCE/DFS) and a Kerberos-based authentication system. DCE/RPC has the client-server architecture in which a client invokes a procedure to be executed on the server. Arguments are passed from the client to the server and values are returned from the server to the client. In principle, the framework is platform- and language- neutral.

Web services are, especially, useful in distributed computing. In a web service, the requesting client and the service need not be coded in the same programming language or even in the same style of language. Clients and services can be implemented in any language style e.g., object-oriented, procedural, functional, etc. The languages on either end may be statically typed (for instance, C# and Java) or dynamically typed (for example, JavaScript and Python). The complexities of stubs and skeletons, the serializing and deserializing of objects encoded in proprietary formats, give way to relatively simple text-based representations of messages to be exchanged over standard transport layer protocols such as HTTP.

Why Web Services

Web Services offers multiple benefits while exchanging data between applications or systems:

1. Interoperability

Web services allow various applications to talk to each other and share data and services among themselves. Web services can be used by other applications. e.g., a VB.NET application can talk to Java-based web services and vice versa. To make the application platform and technology independent, web-services can be used.

2. Standardized Protocol

Standardized industry standard protocols are used for the communication by web services. Web services are open standards (XML, SOAP, JSON, REST, HTTP, etc.) based web applications. Such 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 the quality.

3. Exposing the Existing Function on the network

Web services are unit of managed code that can be remotely activated using HTTP requests. Web services allow one to expose the functionality of their existing code over the network. Once exposed network, the functionality of program can be used by other applications.

4. Low-Cost Communication

As Web services use 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 Business to Business (B2B). Besides SOAP/REST over HTTP, web services can also be implemented on other reliable transport mechanisms like SMTP (Simple Mail Transfer Protocol).

Characteristics of Web Services

Web services have the following special behavioral characteristics: -

1. XML/JSON – Based

Web services use SOAP (XML format) /REST (JSON format) based standards over HTTP protocol at data representation and data transportation layers. Using XML or JSON eliminates any networking, operating system, or platform binding. Web Services based applications are highly interoperable at their core level.

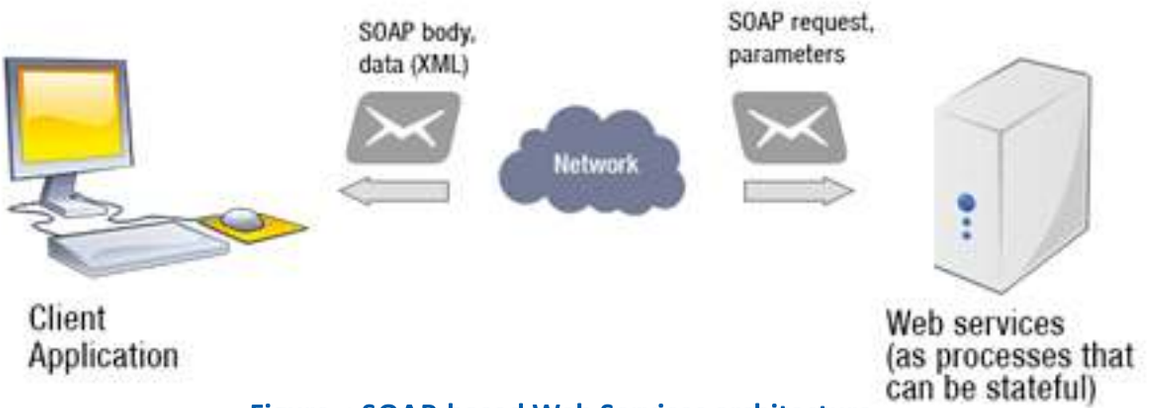

Figure : SOAP-based Web Services architecture

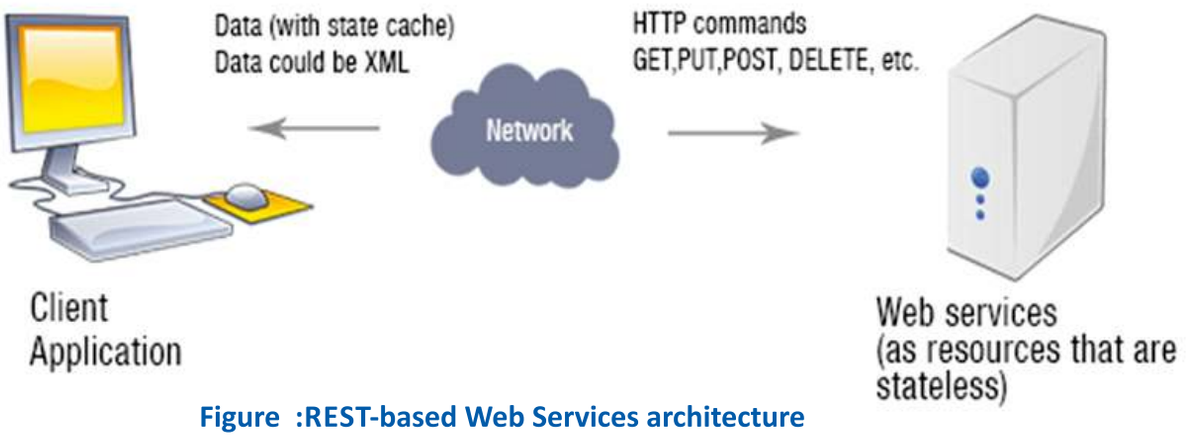

Figure :REST-based Web Services architecture

1. Loosely Coupled

Consumers of web services are 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 simply means closely tied client and server logic to one another, implying that if one of the interface changes, the other should also be updated. Adoption of loosely coupled architecture makes software systems more manageable and provides for simpler integration between different systems.

2. Coarse-Grained

Object-oriented technologies expose their services through individual methods. Individual methods are too fine an operation for any useful capability at a corporate level. Building a program from scratch requires the creation of several fine-grained methods that are composed of a coarse-grained service which is consumed by either a client or another service.

Businesses and the interfaces exposed should be coarse-grained. Coarse-grained services can be defined using web-services that access the adequate amount of business logic.

3. Ability to be Synchronous or Asynchronous

Binding of the client to the execution of the service is referred as synchronicity. Synchronous invocations happen when the client blocks and waits for the service to complete operations for further processing. Asynchronous operations allow clients to invoke services and start executing other functions simultaneously.

For asynchronous clients, results are retrieved at a later point in time, while synchronous clients receive their result after the completion of service. The asynchronous capability is a key factor in enabling loosely coupled systems.

4. Supports Remote Procedure Calls (RPCs)

Using web-services, procedures, functions, and methods on remote objects can be invoked by clients using XML-based protocol. Remote procedures expose input and output parameters that are supported by web service.

Development of component through Enterprise JavaBeans (EJBs) and .NET are distributed and accessible through a variety of RPC mechanisms which are increasingly becoming a part of architectures and enterprise deployments.

Web-services support RPC by providing services of its own, which are equivalent to those of a traditional component, or by translation of incoming invocations into invocations of EJB or .NET component.

5. Supports Document Exchange

One of the key advantages of XML is its generic way of representation of not only data but also complex documents that can be as simple as representing a current address, or they can be as complex as representation of an entire book. Web services facilitate business integration by supporting the transparent exchange of documents.

Web Services Security

Although security is critical to web services, no explicit security or authentication requirements are available for either XML-RPC or SOAP.

There are three specific security issues with web services: -

- Confidentiality
- Authentication
- Network Security

1. Confidentiality

If a client sends an XML request to a server, confidentiality of the communication can be ensured as below:

- XML-RPC and SOAP run primarily on top of HTTP.
- HTTP supports Secure Sockets Layer (SSL).
- SSL can be used to encrypt data for communication.
- SSL is a widely accepted and deployed encryption technology.

Individual web services may consist of a chain of applications. For example, more than one applications might be tied together with one large service. In such cases, SSL may not be adequate; the messages need to be encrypted at each node along the service path, and each node may represent a potential weak link in the chain. Although there is no agreed-upon solution to this issue, the W3C XML Encryption Standard can be used to solve it. It provides a framework for encryption and decryption of entire XML documents or just portions of an XML document. It can be checked at https://www.w3.org/Encryption

2. Authentication

If a client connects to a web service, following option can be considered to identify the user and whether the user is authorized to use the service or not.

- Built-in support for Basic and Digest authentication by HTTP can be used to protect services in a similar manner as HTML documents are protected
- For public key encryption, to digitally sign SOAP messages, SOAP Digital Signature (SOAP-DSIG) can be used which validates the identity of another party by enabling client or server. It can be checked at https://www.w3.org/TR/SOAP-dsig.
- The Organization for the Advancement of Structured Information Standards (OASIS) is working on the Security Assertion Markup Language (SAML).

3. Network Security

For now, if one is truly intent on filtering out SOAP or XML-RPC messages, there is a possibility to filter out all HTTP POST requests with content type set to text/XML.

Another alternative is to filter the SOAP Action HTTP header attribute. Some tools are also being designed and developed to filter web services traffic.

Challenges in Web Services

As such, no technology is perfect. Although Web services solve certain problems, it has certain issues which are described briefly as below:

1. Availability

Normally no internet site is available for 100% of the time, the same is the case with web services, as they use the same architecture. Even if the server is up and running between the entities, their ISPs network may not be available all the time. Accordingly, it is necessary to build mechanisms that will retry the transactions when this occurs. Though some of the newer protocols supported by Web services (JMS – Java Message Service, for instance) handles this automatically, the majority built on HTTP may not.

2. Matching Requirements

Any time one creates a general service that will handle a variety of customers; it runs into specialized requirements. There might be a requirement of some customers of feature that nobody else needs. Web services are envisioned as a "one size fits many customers" technology. If one's business can't fit into that model, other solutions should be considered.

3. Immutable Interfaces

If one invests in creating a Web service for its customers, he has to avoid changing method(s) that is provided and the parameters that one's customers expect. One can create new methods and add them to the service, but if he changes the existing ones, his customers' programs will break.

While this sort of problem occurs in almost all systems, it is especially true in Web services. One might not know who is using his service, and as a result, one has no way to inform those users of the change. In most other systems, due to the tight coupling, usually more verbal or written coordination between producers of a service and consumers of it happens.

4. Guaranteed Execution

The whole idea of having a computer program instead of doing a job manually is that the automation. HTTP is not a reliable protocol; it doesn't guarantee delivery or a response. If one needs this kind of guarantee, either write code(s) to retry requests itself or arrange to send one's requests through an intermediary who will perform these retries for him. Revised versions of the specification allow for using protocols such as JMS to resolve this issue, but most of the services still utilize HTTP, which does not.

Applications of Web Services in TRAI

Application integration is one of the most prominent issues that information systems are facing nowadays, and web-services help application integration. TRAI also uses web services in different mobile apps and portals to facilitate Indian telecom subscribers with efficient grievance redress mechanism and get insights into the state of telecom in India that may be helpful in the formulation of effective regulations, thereafter.

DND 2.0 mobile app launched by TRAI uses web services in the form of different APIs to achieve desired functionalities of the app. For example, DND status API is used to get the DND status of the mobile number from the National Consumer Preference Register (NCPR) database, Complaint's status API is used to get the status of the

complaint registered in the portal. Similarly, CDAC APIs and SMS gateway are being used for OTP verification by the app.

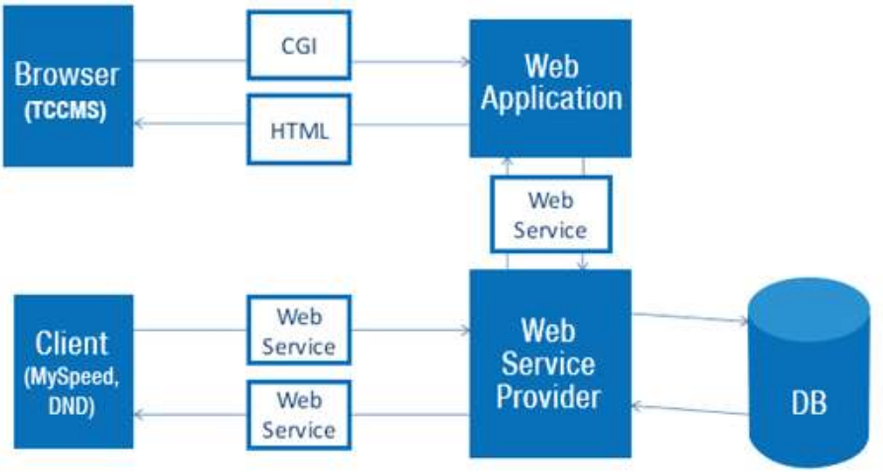

Figure : High-level architecture of TRAI Apps/Portals with Web Services

TRAI MySpeed App also uses different APIs for communication between the server and the app. For example, initially, at the beginning of the test, an API is used to check the version and get URL info. Similarly, there are different APIs used at various stages of the individual tests conducted on the app. There is also an API that is used to trigger autotests on user's device.

To fetch the data from TSPs for QoS analytics, TRAI proposes to use web services in which web services will be developed by TSPs and TRAI will develop an application for consuming web services for fetching data from their servers. The architecture is designed so as to avoid any performance issues as TRAI will have control to fetch data as and when required.

CONCLUSION

Web services are one of the key elements of the programmable Web. They are extremely versatile software elements that really have the potential to open up a new era in software: the age of interoperability. Business-to-Business (B2B) transactions can be effectively set-up and participated in using web services. Integration of heterogeneous platforms and exposing software functionality to customers can easily be done using web services.

Web services are exclusively based on open and commonly accepted Internet protocols. This has advantage and disadvantage both. The advantage is that it enables true interoperability, however it comes at the price of reduced speed, which may be seriously affected by the bandwidth. Web services are not good at everything but certainly represent a category of software agents that are being looked for by many.

Published by: Telecom Regulatory Authority of India
Editorial responsibility: TD Division, TRAI
Contributions, comments and suggestion: sroit@trai.gov.in