INTRODUCTION TO WEB SERVICES

We can’t mention the term “web services” nowadays without immediately evoking references to Amazon Web Services or Google’s Web service, Google Cloud Platform. There’s a reason for that, though. These tech giants have raised the bar by addressing the need for application development. And scale from the likes of Amazon and Google is exactly what makes modern web services possible.

However, despite all of its technological advancements, web service testing and processes are still complex animals. Since they depend on operating systems to manage applications, the smallest of variances can result in multifaceted workflows when trying to move data between servers and the cloud.

But with APIs, web developers can integrate advanced functions and features into apps, allowing for much better customization and flexibility. And the end result is a better experience for users and customers. They reap the benefits of well-built web APIs, increasing efficiency and practicality of services and apps. At the end of the day, the goal is to offer an enjoyable experience.

What are Web Services?

There is more than one way to answer, “What is a web service?” But, essentially, web services include any software, application, or cloud technology that provides standardized web protocols (HTTP or HTTPS) to interoperate, communicate, and exchange data messaging – usually XML (Extensible Markup Language) – throughout the internet.

In other words, web services are XML-centered data exchange systems that use the internet for A2A (application-to-application) communication and interfacing. These processes involve programs, messages, documents, and/or objects.

A key feature of web services is that applications can be written in various languages and are still able to communicate by exchanging data with one another via a web service between clients and servers. A client summons a web service by sending a request via XML, and the service then responses with an XML response. Web services are also often associated with SOA (Services Oriented Architecture).

To break that down, a web service comprises these essential functions:

- Available over the internet or intranet networks
- Standardized XML messaging system
- Independent of a single operating system or programming language
- Self-describing via standard XML language
- Discoverable through a simple location method

A web service supports communication among numerous apps with HTML, XML, WSDL, SOAP, and other open standards. XML tags the data, SOAP transfers the message, and WSDL describe the service's accessibility.

What are the Different Types of Web Services?
There are a few central types of web services: XML-RPC, UDDI, SOAP, and REST

XML-RPC (Remote Procedure Call) is the most basic XML protocol to exchange data between a wide variety of devices on a network. It uses HTTP to quickly and easily transfer data and communication other information from client to server.

UDDI (Universal Description, Discovery, and Integration) is an XML-based standard for detailing, publishing, and discovering web services. It's basically an internet registry for businesses around the world. The goal is to streamline digital transactions and e-commerce among company systems.

SOAP, is an XML-based Web service protocol to exchange data and documents over HTTP or SMTP (Simple Mail Transfer Protocol). It allows independent processes operating on disparate systems to communicate using XML.

REST, provides communication and connectivity between devices and the internet for API-based tasks. Most RESTful services use HTTP as the supporting protocol.

Here are some well-known web services that use markup languages:

- Web template
- JSON-RPC
- JSON-WSP
- Web Services Description Language (WSDL)
- Web Services Conversation Language (WSCL)
- Web Services Flow Language (WSFL)
- Web Services Metadata Exchange (WS-MetadataExchange)
- XML Interface for Network Services (XINS)

RESTful Web Services
What is a RESTful web service? The acronym REST, or sometimes ReST, stands for Representational State Transfer and is an architectural style, meaning each unique URL represents an individual object of some sort. A REST web service uses HTTP and supports/repurposes several HTTP methods: GET, POST, PUT or DELETE. It also offers simple CRUD-oriented services. Fun fact: The original RESTful architecture was designed by one of the leading authors of HTTP, Roy Fielding.

Pros: Lightweight, human readable, easier to build

Cons: Point-to-point communication, lack of standards

SOAP Web Services
SOAP is defined as Simple Object Access Protocol. This web service protocol exchanges structured data using XML and generally HTTP and SMTP for transmission. SOAP also uses WSDL (Web Services Description Language) documents to distribute a web service description model. This describes how the SOAP requests (client-side) and responses (server-side) must appear. Additionally, SOAP web Services have standards for security and addressing.

Pros: Usually easier to consume, more standards (WSDL, etc.), distributed computing

Cons: Difficult set-up, more convoluted coding, harder to develop

API vs. Web Services

Web services and APIs are often mistaken for each other, which isn't all that surprising since there is some distinct common ground.

Most web services provide an API, which, with its set of commands and functions, is used to retrieve data. Here's one example: Twitter delivers an API that authorizes a developer access tweets from a server and then collects data in JSON format.

But here's something to keep in mind: All web services can be APIs, but not all APIs can be web services. Now, if that syllogism makes your head spin, maybe these distinctions will clear up the API vs. web services confusion:

Differences between APIs and Web Services

1. APIs can be hosted within an app or IIS (Internet Information Services), but a web service can only be hosted on IIS.
2. Web services are not an open source and are used to understand JSON (JavaScript Object Notation) or XML, whereas APIs are an open source and only used for XML
3. API is a light-weight architecture (best for limited bandwidth devices (e.g. smartphone). Web services are not lightweight architectures since they require SOAP to send and receive network data
4. APIs can use any form of communication, but a Web service only uses SOAP, REST, and XML-RPC
5. APIs support URL, request/response headers, caching, versioning, content formats. Web services only support HTTP

Similarities between APIs and Web Services

1. Both are accessed through HTTP/HTTPS to enable communication between services providers and customers.
2. Both call a function, process data, and receive a response