**What are Web Services?**

[**https://javapapers.com/web-service/web-service-introduction-tutorial/**](https://javapapers.com/web-service/web-service-introduction-tutorial/)

[**https://javapapers.com/web-service/java-web-service-using-eclipse/**](https://javapapers.com/web-service/java-web-service-using-eclipse/)

**how to invoke web services:**

[**https://docs.oracle.com/cd/E13222\_01/wls/docs103/webserv/client.html#wp211686**](https://docs.oracle.com/cd/E13222_01/wls/docs103/webserv/client.html#wp211686)

**creating a web service from java:**

[**https://docs.oracle.com/cd/E17802\_01/webservices/webservices/reference/tutorials/wsit/doc/Examples\_glassfish4.html#wp104810**](https://docs.oracle.com/cd/E17802_01/webservices/webservices/reference/tutorials/wsit/doc/Examples_glassfish4.html#wp104810)

[**https://www.educba.com/how-to-create-webservice-in-java/**](https://www.educba.com/how-to-create-webservice-in-java/)

[**https://www.tutorialspoint.com/restful/restful\_first\_application.tm**](https://www.tutorialspoint.com/restful/restful_first_application.tm)

[**https://javapapers.com/java/restful-web-services-with-java-jax-rs-using-jersey/**](https://javapapers.com/java/restful-web-services-with-java-jax-rs-using-jersey/)

<https://www.javaguides.net/2018/09/jax-rs-tutorial.html>

https://kinsta.com/blog/http-status-codes/

**https://www.geeksforgeeks.org/singleton-class-java/**

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 (Service-Oriented Architecture)](https://www.cleo.com/blog/knowledge-base-soa-service-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 describes the service’s accessibility.

Here’s an instance of how it works: A web service sits between two sets of java, .net, or PHP apps providing a way for these applications to communicate over a network. On one side, for example, a java app interacts with the java, .net, and PHP apps on the other end by way of the web service communicating an independent language.

Web services offer different benefits across business operations. The technology helps IT pros and web architects streamline connectivity by minimizing development time. And with this simplified infrastructure, company executives begin to see higher ROI (return on investment). In a B2B operation where both parties understand how the process works, web services provide efficient technology distribution throughout an entire network.

**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](https://www.cleo.com/white-paper/what-is-edi) 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**, which will be described in detail later in the blog, is an XML-based Web service protocol to exchange data and documents over [HTTP or SMTP](https://www.cleo.com/white-paper/secure-data-exchange-protocols) (Simple Mail Transfer Protocol). It allows independent processes operating on disparate systems to communicate using XML.

**REST**, which will also be described in great detail later in the blog, 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)

**SOAP vs. REST Web Services**

For years, IT pros and web developers have debated over [which web service is better](https://www.cleo.com/blog/soap-vs-rest-which-web-service-protocol-is-better) and why. Well, there isn’t a clear-cut winner – it all just depends. RESTful web services and SOAP offer different variations. For example, a REST web service is generally a better choice when time is a factor, but SOAP wins out when building a service with multiple, non-CRUD methods. A company’s specific requirements determine which type of web service a partner will implement unless already decided by the WS provider.

Then there are times when both options are the right answer. That’s the case for two of the world’s biggest e-commerce companies: Amazon and eBay use web services for both REST and SOAP. And as more organizations become service-focused and explore more functionality, they will have no choice but to support both types.

But what is the difference between REST web services and SOAP web services? Let’s break down each option while exploring some pros and cons.

**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

Modernized web services have changed the digital landscape with evolved [system integration](https://www.cleo.com/blog/system-integration) and interoperability. Before advancements to web services, limited and burdensome integration prevented streamlined data exchange among various technologies, formats, vendors, and B2B operations. Now, web services offer a level of modern functionality and less complexity.

“Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.”

How to use roll back:

try (Connection connection = DriverManager.getConnection(...)) {

connection.setAutoCommit(false);

try (Statement stmt = connection.createStatement()) {

stmt.executeUpdate(<your first update>);

stmt.executeUpdate(<your second update>);

connection.commit();

} catch (SQLException e) {

connection.rollback();

}

}