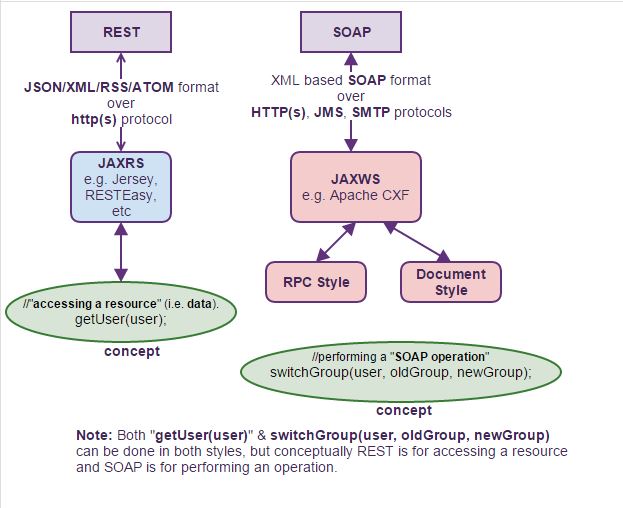
<https://www.java-success.com/java-web-services-interview-questions-and-answers/>

Q1. What are the different styles of Web Services used for application integration? and What are the differences between both approaches?  
A1. SOAP WS and RESTful Web Service. Web services are very popular and widely used to integrate similar (i.e. Java applications) and disparate systems (i.e. legacy applications and applications written in .Net etc) as they are **language neutral**.

[](https://www.java-success.com/wp-content/uploads/2014/08/REST-Vs-SOAP-comparison.jpg)

Java Web Service styles comparison

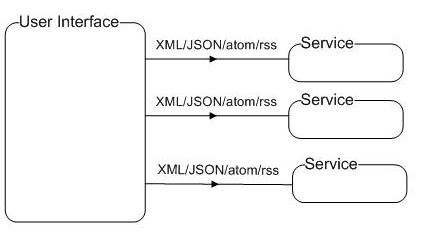
**SOAP vs. REST comparison**

|  |  |
| --- | --- |
| **SOAP Web service** | **RESTful Web service** |
| SOAP (Simple Object Access Protocol) is a standard **communication protocol** on top of transport protocols such as HTTP, SMTP, Messaging, TCP, UDP, etc. | REST is an **architectural style** by which data can be transmitted over transport protocol such as HTTP(S).  Each unique URL is a some representation of a resource (i.e Object like Account, Customer, etc), and you can get the contents of the resources (i.e Objects) via HTTP verb “GET” and to modify via “DELETE”,”POST”, or “PUT”. |
| SOAP Layers  SOAP Layers | [REST Layers](https://www.java-success.com/wp-content/uploads/2014/08/RESTful-layers1.jpg)  REST Layers |
| SOAP uses its own protocol and focuses on exposing pieces of **application logic (not data)** as services. SOAP exposes operations. SOAP is focused on accessing named operations, which implement some business logic through different interfaces. | REST is about exposing a public API over the internet to handle CRUD (Create, Read, Update, and Delete) operations on data. REST is focused on accessing named resources through a single consistent interface. |
| SOAP only permits XML data formats.  [SOAP](https://www.java-success.com/wp-content/uploads/2014/08/SOAP-paylaod.jpg)  SOAP | REST permits many different data formats like XML, JSON data, text, HTML, atom, RSS, etc. JSON is less verbose than XML and is a better fit for data and parses much faster.  **URL:** http://localhost:8080/myapp/createEmptyCase  [REST content](https://www.java-success.com/wp-content/uploads/2014/08/REST-body-content-xml-json-more.jpg)  REST content XML, JSON, RSS, etc |
| SOAP based reads **cannot be cached**. The application that uses SOAP needs to provide cacheing. | REST based **reads can be cached**. Performs and scales better. |
| Supports both **SSL security**and **WS-security**, which adds some enterprise security features. Supports identity through intermediaries, not just point to point SSL.  — WS-Security maintains its encryption right up to the point where the request is being processed.  — WS-Security allows you to secure parts (e.g. only credit card details) of the message that needs to be secured. Given that encryption/decryption is not a cheap operation, this can be a performance boost for larger messages.  — It is also possible with WS-Security to secure different parts of the message using different keys or encryption algorithms. This allows separate parts of the message to be read by different people without exposing other, unneeded information.  — SSL security can only be used with HTTP. WS-Security can be used with other protocols like UDP, SMTP, etc. | Supports **only point-to-point SSL security**.  — The basic mechanism behind SSL is that the client encrypts all of the requests based on a key retrieved from a third party. When the request is received at the destination, it is decrypted and presented to the service. This means the request is only encrypted while it is traveling between the client and the server. Once it hits the server (or a proxy which has a valid certificate), it is decrypted from that moment on.  — The SSL encrypts the whole message, whether all of it is sensitive or not. |
| Has comprehensive support for both **ACID** based transaction management for short-lived transactions and **compensation** based transaction management for long-running transactions. It also supports two-phase commit across distributed resources. | REST supports transactions, but it is **neither ACID compliant nor can provide two phase commit** across distributed transactional resources as it is limited by its HTTP protocol. |
| SOAP has success or retry logic built in and provides end-to-end reliability even through SOAP intermediaries. | REST does not have a standard messaging system, and expects clients invoking the service to deal with communication failures by retrying. |

**Which one to favor?** In general, a REST based web service is preferred due to its simplicity, performance, scalability, and support for multiple data formats. SOAP is favored where service requires comprehensive support for security and transactional reliability.

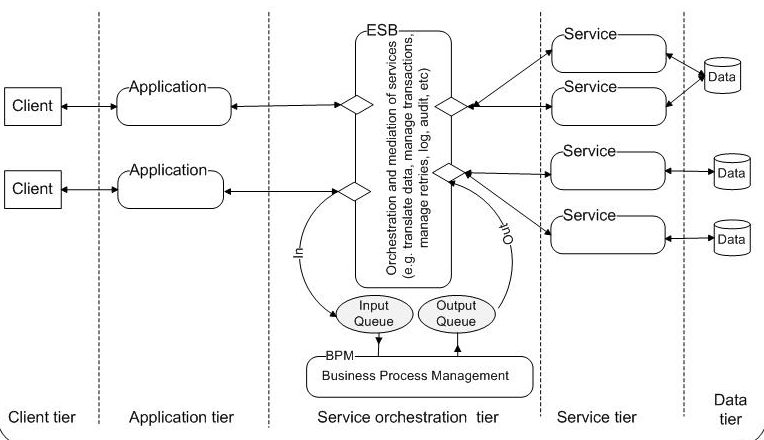
SOA done right is more about RESTFul + JSON, favoring lighter weight approaches to moving messages around than the heavyweight ESBs using WSDL+XML that gave SOA a bad name.

Q2. Differentiate between SOA (Service Oriented Architecture) versus WOA (Web Oriented Architecture)?  
A2. **WOA extends SOA** to be a light-weight architecture using technologies such as REST and POX (Plain Old XML). POX compliments REST. JSON is a variant for data returned by REST Web Services. It consumes less bandwidth and is easily handled by web developers mastering the Javascript language

[](https://www.java-success.com/wp-content/uploads/2014/08/JudgingExperience-WS.jpg)

WOA – RESTFul Service Calls via AJAX to populate different sections of a UI

SOA and WOA differ in terms of the layers of abstraction.**SOA is a system-level architectural style** that tries to expose business capabilities so that they can be consumed by many applications. **WOA is an interface-level architectural style** that focuses on the means by which these service capabilities are exposed to consumers. You can start out with a WOA and then grow into SOA.

[](https://www.java-success.com/wp-content/uploads/2014/08/Screen-shot-2014-08-28-at-9.25.07-PM.png)

SOA (when SOAP WS in service tier) or WOA (when REST WS in service tier)

According to Nick Gall, “**WOA**= **SOA**+ **REST**+ **WWW**“. In the above diagram from the **Service Orchestration tier**, which is responsible for loosely coupling services,

For the **SOA**=> you will be making **SOAP** style web services in the “**Service Tier**“.

For the **WOA**=> you will be making more lighter **REST**style web services in the “**Service Tier**“.

Q3. How would you decide what style of Web Service to use? SOAP WS or REST?  
A3. In general, a REST based Web service is preferred due to its simplicity, performance, scalability, and support for multiple data formats. SOAP is favored where service requires comprehensive support for security and transactional reliability.

The answer really depends on the functional and non-functional requirements. Asking the questions listed below will help you choose.

**1)** Does the service expose data or business logic? (REST is a better choice for exposing data, SOAP WS might be a better choice for logic).

**2)** Do consumers and the service providers require a formal contract? (SOAP has a formal contract via WSDL)

**3)** Do we need to support multiple data formats?

**4)** Do we need to make AJAX calls? (REST can use the XMLHttpRequest)

**5)** Is the call synchronous or asynchronous?

**6)** Is the call stateful or stateless? (REST is suited for statless CRUD operations)

**7)** What level of security is required? (SOAP WS has better support for security)

**8)** What level of transaction support is required? (SOAP WS has better support for transaction management)

**9)** Do we have limited band width? (SOAP is more verbose)

**10)** What’s best for the developers who will build clients for the service? (REST is easier to implement, test, and maintain)

Q4. What tools do you use to test your Web Services?  
A4. SoapUI tool for SOAP WS & RESTFul web service testing and on the browser the Firefox “**poster**” plugin or Google Chrome “**Postman**” extension for RESTFul services.

Q5. Why not favor traditional style middle-ware such as RPC, CORBA, RMI and DCOM as opposed to Web services?  
A5. The**traditional middle-wares tightly couple** connections to the applications. Tightly coupled applications are hard to maintain and less reusable. Generally do not support heterogeneity. Do not work across Internet and can be more expensive and hard to use.

**Web Services support loosely coupled connections**. The interface of the Web service provides a layer of abstraction between the client and the server. The loosely coupled applications reduce the cost of maintenance and increases re-usability. Web Services present a new form of middle-ware based on XML and Web. Web services are language and platform independent. You can develop a Web service using any language and deploy it on to any platform, from small device to the largest supercomputer. Web service uses language neutral protocols such as HTTP and communicates between disparate applications by passing XML or JSON messages to each other via a Web API. Do work across internet, less expensive and easier to use.

Q6. What is the difference between SOA and a Web service?  
A6. **SOA** is a software design principle and an architectural pattern for implementing loosely coupled, reusable and coarse grained services. You can implement SOA using any protocols such as HTTP, HTTPS, JMS, SMTP, RMI, IIOP (i.e. EJB uses IIOP), RPC etc. Messages can be in XML or Data Transfer Objects (DTOs).

**Web service** is an implementation technology and one of the ways to implement SOA. You can build SOA based applications without using Web services – for example by using other traditional technologies like Java RMI, EJB, JMS based messaging, etc. But what Web services offer is the standards based and platform-independent service via HTTP, XML, SOAP, WSDL and UDDI, thus allowing interoperability between heterogeneous technologies such as J2EE and .NET.