RESEARCH ARTICLE

SAMRIDDHI Volume 14, Special Issue 2, 2022 Print ISSN : 2229-7111 Online ISSN : 2454-5767

Growth of Individualizing Web Services using APIs: REST and SOAP

Abhijit Banubakode[1*], Priya Chore[2]

[1*,2] MET Institute of Computer Science, Mumbai, India; e-mail*: abhijitsiu@gmail.com

ABSTRACT

Web Services are a collection of open protocols and standards that enable clients and servers to connect with one another. It allows various programmes to communicate with one another. Representational State Transfer (REST) and Simple Object Access Protocol (SOAP) are the two most extensively utilized web services today (SOAP). REST is an architectural approach, whereas SOAP is an underlying protocol. Both services are used to manage internet communication (www). Both services have advantages and downsides, and it is up to the web developer to decide which the better option is for their purposes. The purpose of this research is to construct a REST API and a SOAP API, respectively, using JAX-RS and JAX-WS, and to compare the two APIs.

Keywords: Postman, Protocol, REST, SOAP, Tomcat, Web Service.

SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology, (2022); DOI : 10.18090/samriddhi.v14spli02.15

INTRODUCTION

A web service is a client-server programme that allows numerous programmes to communicate with each other via the internet. Because XML can be used to encrypt all data, it's a type of server or programme that sends XML messages in a standard format via the Internet. TCP/IP, HTTP, Java, HTML, and XML can all be used to create it. It allows for interoperability across different apps as well as language freedom. It is a networked software object that provides services through the Internet. The virtual representation of a web service is shown in Figure 1. A web service receives a client's request and responds to the client's applications. SOAP, WSDL, and UDDI are the three main components of web services. SOAP (Simple Object Access Protocol) is an XML-based protocol.

It can communicate using the CORBA, SOAP, and Java RMI protocols. The use of a web service allows you to discover the functionality of code over the internet. There are primarily two types of web services: REST and SOAP. REST is a web service development architectural style. The client submits a request to the server, which the server responds to using resources. Nouns and verbs make up the REST language. Despite the fact that REST web services are not protocol-agnostic, almost every REST service uses HTTP verbs to govern resource activities. Resources include videos, web pages, photographs, and anything else that can be allowed in a computer-based system. There are no limitations to what you can do with REST.

Both services are excellent for communication, but there is a distinction between them. REST is an architectural style for constructing web services, whereas SOAP is a protocol. The Java APIs for REST and SOAP are JAX-RS and JAX-WS, respectively. In some

Corresponding Author : Abhijit Banubakode, MET Institute of Computer Science, Mumbai, India; e-mail : abhijitsiu@gmail.com

How to cite this article : Banubakode, A., Chore, P. (2022). Growth of Individualizing Web Services using APIs: REST and SOAP.

SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology, Volume 14, Special Issue (2), 284-290.

Source of support : Nil

Conflict of interest : None

areas of concern, both services have advantages and disadvantages. As a result, it is preferable to determine which service can be employed in particular case to achieve the best outcomes. The purpose of this research paper is to provide such a judgement between these two internet services based on our thorough investigation. The rest of the paper is organised as follows: The second section looks at the research that has been done in this area. In part III, the background research is briefly presented. In Section IV, we present our primary planned work. In section V, state-of-the-art parameters are used to compare REST and SOAP.

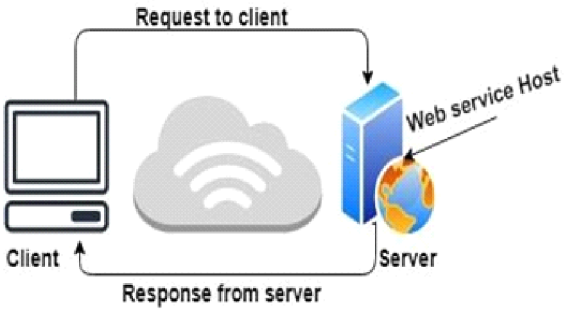


Figure 1: Virtual representation of web service

- Researching various web service requirements. In today's corporate environment, web-based apps are created using a variety of programming platforms. Angular JS, Node.js, and other frameworks are used in some apps, while others are created in Java. To work together, these diverse programmes almost always require some type of communication. It's tough to ensure accurate communication between them because they're written in different programming languages
- Web services have a role here. Web services provide a common framework for connecting several applications developed in different computer languages.

Associated Work

Chuangwei Zhang et al. [1] have developed a design for web services management that makes it easier, decreases system installation time, and increases efficiency. Paul Adamczyk [3] differentiates two architectural styles, REST and SOAP, and succinctly explains Restful web services as displaying four REST principles proposed by Roy Fielding, based on a study of current web services.

Chia Hung Kao et al. [4] describe and assess a REST- based web application testing framework. The software tester follows a consistent method for developing test cases and test scripts. The proposed framework cuts down on the amount of time and effort required to comprehend application design and implementation.

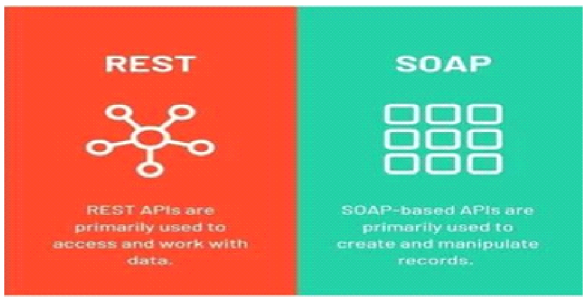
Florian Haupt et al. [5] provide a framework for REST API structural analysis that includes metrics and graphical API representations. They also convert the rest API description, which is available in a variety of languages, into a model that has been authorised and clearly illustrates the rest API design. to investigate the many types of RESTful Web Services.

Web services can be divided into two categories. Web services based on SOAP.

Web services that are Restful**.**

There are certain components that must be present in order for a web service to be fully functional. Regardless of the development language used to programme the web service, these components must be present.

Let's take a closer look at these elements.



Representational State Transfer (REST) was suggested by Roy Fielding in the year 2000 as a design methodology for the evolution of web services. It's a collection of rules designed to encourage people to use web services. In their PhD dissertation, Roy Fielding [2] lists seven limitations:

- **Start by using the NULL style**: It's a list of restrictions that hasn't been addressed yet. It lays the ground work for explaining REST.
- **Client-Server Architecture:** Client and server architectural styles are separated so that they can evolve independently. you may improve consumer interface flexibility and scalability by simplifying server components. Stateless:The customer and server ought to speak in a stateless way. The client's request will not be saved. A client request should not be recorded in a session or in a history. Resource allocation and management.
- **Cache:** Cache limitation is paired with client and server communication to increase network efficiency. This requirement specifies that the catchability (or non-catchability) of the response is expressed clearly

(or implicitly). The client must reuse the response in order for it to be cacheable. Client-side caching enhances performance, while server-side caching promotes scalability.

- **Uniform Interface:** The uniform Interface constraint distinguishes REST from other network styles. It simplifies and decouples the architecture, allowing it to evolve independently.

Assets identity, sources manipulation with the aid of representations, self-descriptive messages, and hypermedia because the engine of application country are the 4 interface regulations that have been diagnosed (HATEOAS).

- **Code-on-Demand:** REST functionality for downloading and executing code could be introduced. It allows the customer to restriction the quantity of functionalities that ought to be pre-implemented.

This constraint promotes the extensibility of the machine. API is a communication interface that allows two software programmes to communicate with each other. APIs are web services that use the HTTP protocol to communicate with one another (REST or SOAP). API interaction is defined as the sort of request and response sent between a client and a server. REST API refers to an API that builds web services using REST rules or standards. REST is not specified as an HTTP protocol, although it does adhere to limitations when designing APIs. REST leverages HTTP to define the desired action for requests and responses. There are several resource methods or request kinds that can be used to facilitate communication:

- **GET:** Get information about a resource.
- **POST:** The development of a new subordinate resource has been completed.
- **PUT:** There will be an upgrade to the present resource.
- **DELETE:** Remove any existing resources indicated by Request URI.

For Example,

POST /users: It create a user.

GET /users/ {id}: It retrieves the details of a user. GET /users: It retrieve the details of all users.

DELETE/users: It delete all users.

DELETE /users/ {id}: It delete a user.

GET/users/ {id}/posts/post_id: It retrieve the details of a specific post.

POST/ users/ {id}/posts: It create a post of the user.

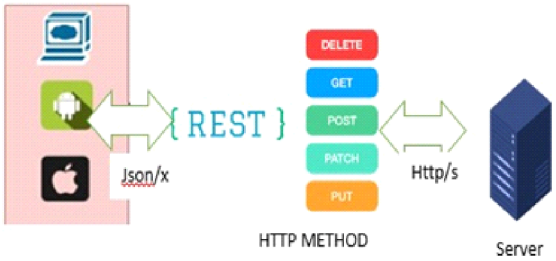


Figure 2: Represents REST API work flow following HTTP methods and return response to clients in JSON/XML format

The aforementioned HTTP techniques can be both safe and idempotent. Table-1 shows the features of idempotent and secure HTTP methods. Idempotent methods are HTTP methods that can be called several times without having any effect on the data stored, whereas safe these are HTTP methods that do not update the state of the server. These methods and resources are read-only.

Table-1: Characteristics of Http Methods

HTTP Method	Idempotent	Safe
GET	Yes	Yes
POST	No	NO
PUT	Yes	No
DELETE	Yes	No

SOAP Web Services API

The Simple Object Access Protocol (SOAP) is a network communication protocol that uses the XML messaging language. Service-Oriented Architecture (SOA) and SOA-related web services rely heavily on SOAP. Mentor, User Land Company, and Microsoft built it for the first time in 1998. In 1999, the initial version of the programme was released. The most recent version of SOAP is 1.2.

The service was enthusiastically utilised. SOAP's primary objective is to transport data across a network. It sent and received data over the internet via HTTP. The operations are started through SOAP messages. The SOAP service is also independent of platform and language.

Figure 3 depicts the format of a SOAP message sent between a server and a client.

- Envelope - A key component of the SOAP message format that identifies a document as SOAP and

determines the message format's start and end points. It is necessary for the SOAP message format to function.

- Header - This element provides optional information such as authorisation and verification information.
- Body - This is a required element that holds XML data. It comprises data that is sent to the programme, as well as request and response metadata.
- Fault - This element contains information about message transmission issues such as Version Mismatch, Client Fault (when the message was created incorrectly), and Server Fault (when there was a server difficulty). Errors and status data are also provided.

SOAP Binding is a transport layer mechanism for exchanging messages. Two SOAP communications requests with different binding styles are RPC (Remote Procedural Call) and Document Style.

- RPC - It's essentially a client-server request-and-response communication. Both the request and response formats are XML, with the message's general architecture determined by the root element Envelope. RPC used the representation and design of the message that was used for request and response. In RPC, synchronous communication occurs when a response is received before a message request is made. Because it is intended to deliver smaller messages, it is a simpler and less competent style.
- Document Style - It has a lot of substance and is also quite sophisticated. It's also mentioned.

In a SOAP request, two headers are defined: Content-Type and Content-Length. Content-Type is an example of a MIME type for message request and response, and character encoding is also provided for the XML body. The Content-Length header, on the other hand, indicates the number of bytes required in the request and response for the message body.

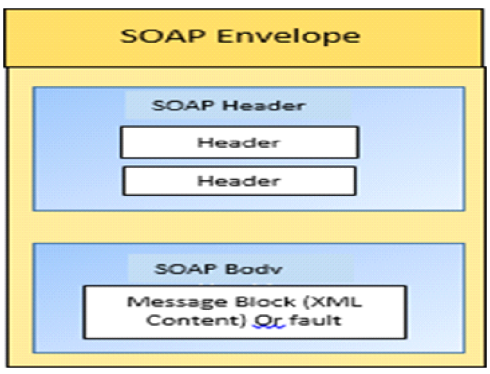


Figure 3: SOAP Messaging format

PROPOSED APPROACH

This study compares REST and SOAP web services in terms of response time, architecture, security, dependability, and efficiency, as well as their development and architectural styles. Also calculates which service is the best to choose depending on the criteria. It is also determined which service is the best appropriate for client-server communication. In a summary, this research article builds a REST API with JAX- RS and a SOAP API with JAX-WS, as well as additional visual aids, and tests them in Postman 7.1.1 to evaluate web service performance and response time.

Technologies used to create REST API with JAX-RS and SOAP with JAX-WS:

- Jersey version - 2.25
- JAX-WS - 2.1
- Apache Maven - 3.8.0
- Tomcat 7.0
- Eclipse Java IDE Mars 2.0
- Postman 7.7.1

Jersey [25] is a Java-based open source framework that supports JAX-RS APIs for constructing REST services. JAX-RS was created to enable the creation of RESTful web services in Java easier.

When creating a REST service with the URL Pattern "rest," as shown in Fig.4, these changes should be replicated in web.xml. Some of the JAX-RS annotations are listed in Table II. JAX- RS is a Java API that aids in the creation of Restful web services. It has annotations that export the java package javax.ws.rs, which makes creating web services easier.

Figure 4: Description of web.xml

SOAP with JAX-WS [26] with two styles: RPC style and Document Style (default) annotated with @SOAP Binding. As seen in Figure 5, @Web Service is utilized for interface implementation.

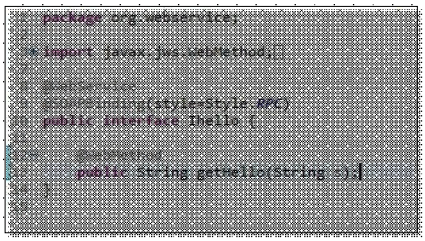


Figure 5: SOAP with JAX-WS

TABLE-2: JAX-RS Annotations

Annotations	Description
@Path	Specify the URI path
@GET	Annoate HTTP GET request method
@POST	Annotate HTTP POST request method
@DELETE	Annotate HTTP DELETE
@Path Param	Extract URI Path Parameters
@ Query Param	Extract URI Path Query Parameters
@ Products	Specify Media Type to be produced
@ Consumes	Specify Media Type to be produced by REST

Table-3 lists some of the JAX-WS annotations that make developing web services easier. The Java API for XML-based web services, primarily SOAP, is known as JAX-WS. It exports the javax.jws package, which contains Java to WSDL annotations. Javax.xml.ws is the basic JAX-WS package. The JAX-WS annotations are listed in Table -3.

Table-2: JAX-RS Annotations

Annotations	Description
@ WebService	Used for implementing web service over class or over an Interface.
@SOAPBinding	Specify SOAP Messaging Style
@Web Method	Only apply over a method specify web service operation
@Web Result	Specify how WSDL file looks.

These internet services are tested using Postman [27], an API testing tool. It's an API development tool for testing, building, and modifying APIs. HTTP requests can be made with Postman (GET, POST, PUT, and Patch). When an API request is performed, Postman may perform integration tests to confirm that APIs work as intended, as well as return response time and response size.

Figure 6 shows a flow chart illustrating the differences between REST and SOAP development techniques. We can infer that REST is less difficult to build than SOAP since REST uses HTTP methods that are easier to understand than SOAP WSDL files. Data is added using the SOAP standard, which uses an envelope style that makes the payload bigger when utilizing the SOAP service. REST, on the other hand, sends data without a payload through the URI Interface. As a result, REST is lighter and consumes less bandwidth than SOAP, which consumes more.

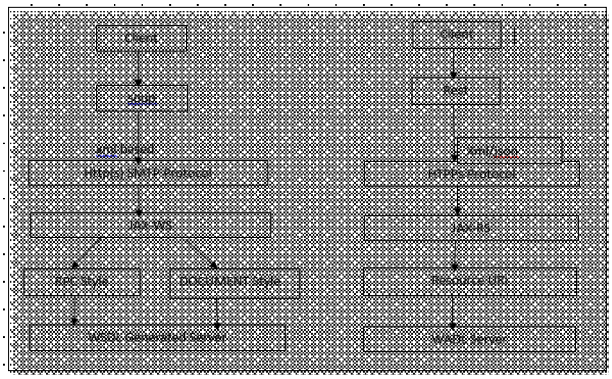


Figure 6: Flow chart of REST and SOAP

Process a big volume of credit card transactions in [7] real time - A large credit card corporation situated in the United States

Objective: Process billions of API calls in real time with a latency of less than 70 milliseconds.

Problem: The Company's current API management solution contributed 500ms of latency to every API call, resulting in a direct revenue loss.

When income is at stake, performance overcomes feature richness in API management systems.

Background Information and The Difficulty

A big credit card company was having problems with transaction delays. When paying with a credit card, most point-of-sale (POS) systems will time out once a specified limit has been reached. Cards will need to be processed again, however transactions will fail automatically to avoid the card from being charged twice.

At the same time, the company was moving to Open Banking standards, which provide API specifications that allow third- party developers and corporations to build applications and services based on customer-permissioned data and analytics, resulting in hundreds of billions of API calls.

As a result, the organization began looking for a scalable solution that could manage billions of API calls.

How real-time API management made a difference

The old API solution introduced 500ms of latency to each API call, causing a tiny proportion of transactions to fail. Even a modest percentage of billions of transactions, however, is a significant proportion, especially when money is lost. Users frequently try paying with a different credit card when a transaction fails. Millions of dollars could be lost due to timed-out API requests if the second card they use isn't issued by the same company as the first.

The firm pioneered a real-time API reference architecture to ensure API calls were executed as close to real-time as feasible. They created clusters of two or more high availability API gateways to improve the stability and resiliency of their APIs. The organization also chose to provide dynamic authentication by pre-providing authentication information (through API keys and JSON Web Tokens, or JWT), allowing for near-instantaneous authentication. Additionally, they chose to delegate authorization to their back end's business-logic layer, allowing their API gateways to concentrate entirely on authentication, resulting in faster call response times.

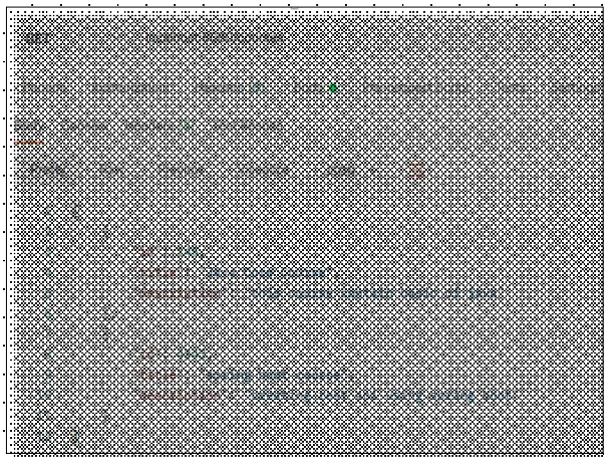
By using these best practices, the company was able to consistently achieve response times of less than 10 milliseconds, exceeding the performance criteria by 85 percent. This resulted in quick and measurable cost savings, allowing them to not only restore but also handle more transactions than previously. These speed gains were deemed more significant to the organization's business goals than some of the added features provided by the incumbent solution, such as a more robust developer site, API design tools, and API transformation capabilities.

The creation and consolidation of new revenue streams were also facilitated by improved transaction latency and dependability. As part of its open banking goals, the company may now make its core transactional engine available to ISVs and developers, allowing them to win more business by proving speed and reliability benefits over competitors.

IMPLEMENTATION, TESTING AND RESULT

Hear we implemented and tested rest API using spring boot:

Output:- When The GET Request Is perform. This is the client side output and we used postman to get the data post the data



FUTURE WORK

Several future work directions were identified that could build on the foundations. APIs are the foundation of online connectivity. They serve as a conduit for multiple applications, data, and devices to communicate with one another. Simply put, an API is a messenger that accepts requests, informs the system of what you want to do, and then returns the response to the user. For each API, documentation may create in future which includes specifications for how information is transferred between two systems. APIs can publicly interact with third-party applications. Finally, expanding an organization's business reach. So, when we book a ticket through Ticketnew.com, we include information about the movie we intend to watch.

Name of the Movie, Location, 3D/2D, Language.

These details are retrieved via API and then routed to servers associated with various movie theatres in order to return the aggregated response from multiple third-party servers. Giving the user the option of selecting the best theatre for them? This is how various applications interact with one another. And nowadays API is used all over word technology.

CONCLUSION

In our research work we conclude that REST is a type of data transfer that is based on the HTTP protocol's architecture. It enables you to send and retrieve data between two services using XML or JSON.It's usually a good idea to structure your web applications using RESTful architecture. This means that collections and resources can be easily identified and used to build a RESTful API. When developing an application that requires a Javascript-heavy front-end or integration with a smartphone app, a RESTful architecture is

required because it allows data to be transferred from the API to the client. It's often a good idea to plan out your RESTful collections and resources from the start.

REFERENCES

[1] C.Zhang and X.Yin, "Design and implementation of a single-service multifunction webservice," Nanjing, 2011 International Conference on Computer Science and Service System (CSSS), pp. 3912-3915.

[2] R.T.Fielding, DISSERTATION, 2000, "Architectural Styles and the Design of Network-based Software Architectures."

[3] P. Adamczyk, P.H. Smith, R.E. Johnson, and Munawar Hafiz, "REST and Web Services: In Theory and Practice," Springer Science+Business Media, 2011, DOI 10.1007/978-1-4419-8303-9 2.

[4] O'Reilly Media, "Designing Consistent RESTful Web Services Interface," ISBN: 978-1-449-31050-9.

[5] C.H.Kao, C.C.Lin, and J.Chen, "Performance Testing Framework for REST-basedWeb Applications," 13th

[6] International Conference on Quality Software, 2013, IEEE, DOI 10.1109/QSIC.2013.32, 2013. 6. F.Haupt, F.Leymann, A.Scherer, and K.Vukojevic-Haupt, "A Framework for the Structural Analysis of REST APIs," IEEE International

[7] Process billions of API calls in real time with a latency of less than 70 milliseconds.