International Journal of Research in Engineering, Science and Management
Volume 4, Issue 7, July 2021
https://www.ijresm.com | ISSN (Online): 2581-5792

Evolution of Modern Web Services – REST API with its Architecture and Design

Shweta Uttam Meshram*
Application Developer, EAI, IBM, Pune, India

***Abstract*: Web services are the modern enhanced forms of the web applications that are typically based on web APIs that are accessed through Hyper Text Transfer Protocol (HTTP) to execute on a far off system hosting [1]. Web services that are the RPCs over the www have a brief history of events that eventually gave rise to the world of these services. The paper presents an overview of RESTful Web API development, design and explores its full capabilities in developing these APIs. It aims to summarize modern APIs and the rising of REST and JSON from the previously built web services using XML over HTTP. Later, the paper manifolds the RESTful service mandates, giving a brief explanation of each of these constraints. To express the purpose of the measure of being RESTful or not by an architecture, the Richardson Maturity Model (RMM) is addressed. They specify the various design rules in the RESTful API creation process.**

***Keywords*: HTTP, JSON, OAuth 2.0, REST, RMM, RPC, SOA, URI, Web API, XML.**

1. Introduction

Even before the adoption of REST services, the internet just started to arise and was becoming popular. That was when Yahoo and Hotmail were some popular mail and social messaging apps, and the integration with web applications was difficult. APIs are the pragmatic access to the data and systems. It is an interface defined by the user to data and system that is consumed by the applications [3]. The web services that was the RPC over the internet or world wide web. In 2000, in the Architectural Styles and the Design of Network-Based Software Architectures by Roy Fielding [4], he came up with the concept of REST. The building of web services using XML over HTTP (plain old XML) were some earlier concepts.

In the REST makes the work easy on top of HTTP, which enables it to be used all over the web and in internal networks.

It is basically an architectural style and not a programming language or technology. It also provides guidelines for distributed systems to communicate directly using the existing principles. The protocols to create web services and APIs, with no need for SOAP or others is used to integrate the business logic around operating systems and servers.

The API value chain comprises assets, API, application development, applications and ultimately the end users that are the customers. It is the three types of value chains- customer, partner and company. Internet of Things also commonly called as IoT solutions also fall within this group of APIs. By creating internal API catalogs, organizations have a leverage of documenting applications and service interface leading to better API consumption and API integration [5].

The motivation of this paper is to study the modern architecture, development and design of the RESTful APIs. To expose their assets to their partners and/ or the public domain developers by building internal applications for either a web server or a web browser. There are many categories of APIs- SOAP, XML-RPC, JSON-RPC, REST and so on. The paper discusses the designing of the REST API. It explores more on the design rulebook and the formatting and versioning. API is the basis of all the development of multi-platform applications that run over the web. Prior research questions the study of REST and its evolution since the time being. The goal of this research paper is to develop and analyze web services using the RESTful architectural style.

2. Evolution of REST/JSON API

The service oriented architectural style is a set of principles, methods and procedures, not a technology or a programming language. Its principal aim is to develop a software application [2]. The architecture comprises service broker, service provider and service requester which are arranged in a cyclic format. The Web Service Description Language (WSDL) is the language that connects in between the service broker and service provider. It is written in XML and used in describing the web services. These descriptions thereby include location of services and its methodologies. The Universal Description, Discovery, and Integration or UDDI is used for specification for a distributed registry of web services. UDDI is a platform for independent language and an open framework. It can also communicates through various protocols like CORBA, SOAP, Java RMI Protocol, etc. UDDI uses WSDL to describe interfaces to web services [5].

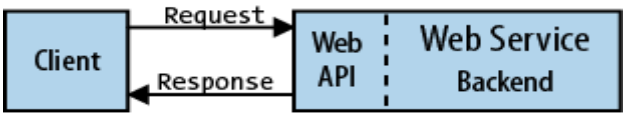

Fig. 1. Web API [2]

The RPC developed in 1991 used Common Object Broker Architecture (CORBA), and the mechanism supported multi

*Corresponding author: shwetameshram056@gmail.com

languages. They were used for class applications and not for the Internet. Later in 1998, the Simple Object Access Protocol (SOAP) strengthened, which only used XML. It was very complex protocol and difficult to use. It heavily depended on message standard formats.

The Simple Object Access Protocol (SOAP) is based on the Service Oriented Architecture. It supports the XML format, while the protocol used here is SOAP/HTTP or SOAP/JMS. For every operation here, a separate name is used. It is heavily weighted and has a complicated setup. It uses both types of communications-asynchronous and synchronous.

Asynchronous messaging is the callback without acknowledgement. SOAP can be any of these two- stateful or stateless. But mostly they are SOAP web services. Stateful are the services in which server stores the data from the client side and uses over a series of requests. Stateless is all about self-contained state that does not depend on any external frame of reference.

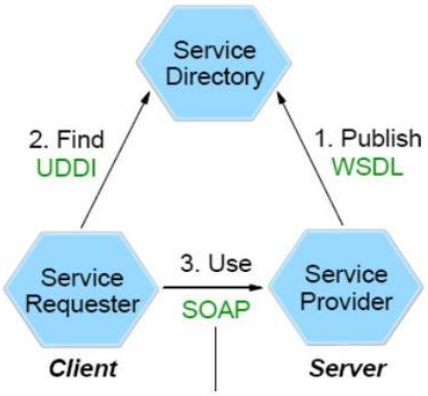

Fig. 2. Relationship of Web Services and SOA [6]

API are available on the client side and the server side as well. At the client-side, they exposed the interfaces as browser plugins or JavaScript. While at server-side, they exposed the interfaces through the web as JSON or XML. In 2000, Jason developed APIs and they standardized it in the year 2013. Soon it became the first choice for formatting among API developers that. They used mostly JSON. But they can use overall XML and JSON. They used these for either a web server or for a web browser [2].

HTTP REST follows restful design patterns. The architecture style of REST allows any server to communicate with any other server over the network. It simplified communication and made integration easier. They made REST to work on top of HTTP, which enabled it to be used all over the web and in internal networks [5].

eBay was the organization first to bring out REST-based APIs. Later, it introduced it with the selected partners in around November 2000. Then, Amazon, Delicious and Flickr provided REST-based APIs.

After that, social media platforms like Facebook, Twitter, Google, and others started using it. Now, you will hardly find any web application which is not developed using a REST API, they are widely used in mobile applications and software services.

3. RESTful Service Mandates

Given below is the figure showing the basic building blocks of Resource Oriented Architecture (ROA) like resource, resource provider, URI and much more.

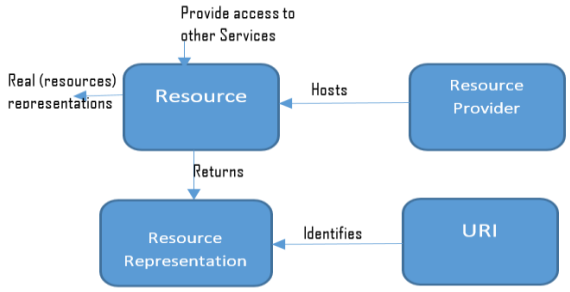

Fig. 3. Resource Oriented Architecture [5]

The Representational State Transfer also called as is based upon Resource Oriented Architecture. It supports many formats like- XML, JSON, CSV, Plain, YMAL, etc. and follows a simple set of design rules. It can use single URL for multiple operations. The HTTP and URL collectively form a REST service, but it is not the only way to make a service RESTful. It is light weighted and easy to setup. It uses synchronous communication and is stateless [7]. The architectural properties of REST are as given: Testability, Modifiability, Performance, Portability, Reliability, Simplicity, Visibility and Scalability [10].

The question arises- What makes an API restful? As the founder of the REST style, Roy enforced some REST constraints that must be mandatory for any web service to be exemplified as RESTful. These mandatory constraints are as follows:

Client- server: The separation of concerns is the core theme of the Web's client-server constraints. The Web is nothing but a client- server based system. The RESTful web services using any language or technology, can be implemented and deployed independently so that they can be conformed to the Web's uniform interface.

Statelessness: The interaction between the client and the server side is stateless. No information is stored in the server. This restriction is being stateless.

Cache: Caching is one in every of net architecture's maximum vital constraints. The cache constraints teach the internet server of every reaction's information. Caching reaction information can assist to lessen client-perceived latency, boom the general availability and reliability of an application, and manage an internet server's load. In a word, caching reduces the general value of the Web.

Interface/uniform contract: Uniform Interface is the most important constraint of all. It ensures that the internal implementation details in the server for managing the resources should not be visible to the clients. These must be hidden from them.

Layered system: This constraint states that- without letting the client know about it, the architecture of the server can be layered. The server only interacts with client having deployed

only APIs on it. So, basically the client has no idea regarding the storage, or its manipulation and authentication of the data. This is used necessarily for the purpose of balancing the load.

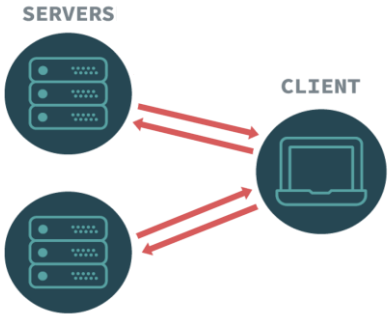

Fig. 4. Layering [8]

Code on demand: This constraint is optional-- It is mostly on you, whether to or not apply it or not. A REST constraint that optionally allows a web server to exchange executable packages to its clients on an as-needed foundation is known as the code on demand.

The combination of an URL and HTTP method can be RESTful or not, but it is always REST like. If these above rules follow, then the web service is an HTTP REST API.

4. Richardson Maturity Model

The Richardson maturity model usually known as RMM is a used as a measure of RESTful ness of architecture.

A score is assigned among zero and three. An API that follows the rest constraints is at level 3, while the ones that don't are at level 0.

According to these 4 levels, level 0 use RPC. It is also known as the swamps of POX. The API makes use of XML and HTTP, however mostly HTTP GET method.

Level 1 comprises of only resources and URL, while it represents the real-world objects.

Level 2 incorporates of a larger picture, together with resources, URL and HTTP verbs or methods like GET, PUT, POST, PATCH, etc. These methods are known as the CRUD operations.

Lastly, level 3 comprises of the hypermedia control and makes use of concept of HATEOAS [2].

The REST specification does now, no longer put in force a excessive degree of restrict in phrases of ways developers and designers build and design REST APIs. Speaking at Q Con in 2008, Leonard Richardson supplied his studies of reviewing many REST APIs. Martin Fowler additionally included Leonard Richardson's adulthood version on his blog. The version offers degrees of adoption that groups observe to constructing APIs.

Hypermedia as the engine of Application State having an acronym of HATEOAS, is one of the constraints within side the REST API which maintains the fashion of the structure specific from the opposite architectures of the community applications. The 'H' in the HATEOAS denotes Hypermedia, may be any content material that hyperlinks to the outside kinds of media which include text, pictures, movies, etc.

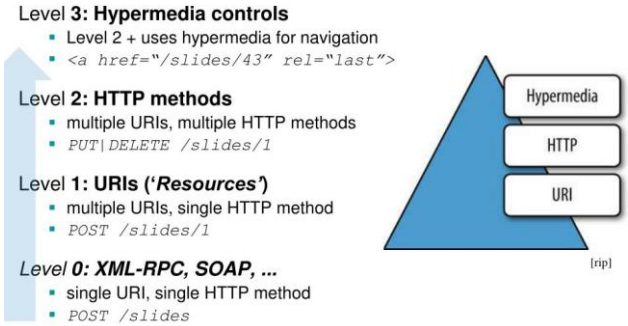

Fig. 5. RMM [9]

While we attempt to visit any of the website, we are often greeted by good enough variety of severe links that guides us to dynamically discover exquisite moves and get proper access to the links. Because of that, it makes easy for clients to now no longer crack the hard code that the URI structures for these resources. Thus, HATEOAS make the API interplay feasible without tons of issues or any additional complications.

5. REST API Designing

The REST APIs are widely spread and are much popular, but many of them follow a fixed design technique. This is due to the fact that- these web services rival with competition and so much of attention. For the sake of following the best practices, these rules were proposed. There are certain rules to design these API web services. Along with these rules for URI design, there are guidelines provided for certain representational forms too. These are made, keeping in consideration some web standards. Mark Masse created a framework for these implementations of REST APIs, which he called it as Web Modeling Language, also called as WRML [2].

Implementing these designs is not as easy as the way it seems. The tools and frameworks are continuously developing and there is always a scope of improvement. It was suggested that the frameworks for the APIs work in the similar fashion as the web applications. Therefore, these could be casted from the similar molds.

Firstly, we should not keep the base of the URL very complex. The API endpoint here is the URL that comprise of the base URL along with the grouping name i.e., package, version, rest resource, and the resource id. Versions can be one or multiple. They can also be switched and can be used on our own accord.

We should not use the Web URL. The API, developer are the ones that are commonly used for that.

Example 1: https://domain/product/version/resource/{id}

Where root URL - https://domain/product/version

Practices for resource name, actions, associations: The resource names should be nouns and plurals are mostly suggested. The actions can be given verbs [2].

Several HTTP operations that can be used for the RESTful

APIs. We must use the right method of HTTP operations. Create, Retrieve, Update, Delete also called as CRUD practices, are followed. The POST operation is used for Create, GET for Retrieve, PUT for Update and DELETE is used for Delete in CRUD.

API Data format: The format of the data should be anything like-JSON, CSV or XML. It has the ability to support multi formats, they mostly use query parameters, HTTP headers and/or resource format suffix. It is a common misconception that only JSON data format should be used, because today we use it. But in future, other formats will also be used [10].

Use of proper HTTP codes:

The Internet Assigned Numbers Authority is the organization that maintains the official registry of HTTP status codes.

Contract and Http status code: The 3-digit standard status codes denote: 1xx – informational, 2xx –success, 3xx – redirection, 4xx – client error and 5xx – Server error.

The status codes are categorized into five classes. The first digit of the three numbered code defines the class of response. The last two digits can be any number. It does not have any categorizing role.

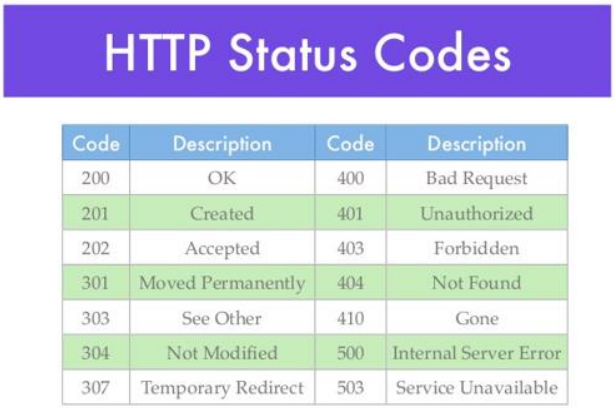

Fig. 6. HTTP status codes [12]

Versioning: Versioning is a crucial part in the development web services. Organizations develop different versions after a certain interval of time. These are named like-

One of the examples is- /v1/products; /v2/products, and so on. Also, versioning can be done by using dots such as, v2.3, v4.5, etc.

URIs: The Uniform Resource Identifiers (URIs) are used by the REST APIs to address resources. It is unique set of sequences of characters used by the various web technologies. It is basically a means of identifying a resource. URI designs can be the masterpieces that communicates with the API's resource model like:

http://api.example.restapi.org/India/Agra/Taj-Mahal/Shah-Jahan

Then people who understand like:

http://api.example.restapi.org/6578h-g6j6-j8i8-7h3g-7h7789e3245

The URI has a fixed format often called as the URI Format. The generic URI syntax defined by RFC 3986[19] as shown below:

URI = scheme"://"authority"/" path["?" query]["#"fragment]

Thus, a URI identifies a resource, impartial of the model of its representational shape and state. REST APIs must preserve a steady mapping of its URIs to its conceptually steady resources. A REST API must introduce a brand, new URI best if it intends to reveal a new concept [7].

6. Conclusion

Although the implementation REST API designs is harder, there is still room for improvement. Many developers use the REST API development frameworks that were originally created for building web applications. The selection of a particular APIs is solely personal or organizational preference depending upon the programming language and platform.

Unfortunately, many current REST API frameworks lack direct support for-- Natural separation of the resource model, Uniform, cross-format hypermedia structures, HATEOAS, schema validation and versioning, Integration and much more. The RESTful services are very useful because you do not tie your API to the client-side technology. It can be used over nearly any protocol, thus making it more portable and flexible.

References

[1] Pawan Kumar Bhat and Rajnish Kansal, "Development of RESTful WebAPI using Token based OAuth 2.0 Authorization".
[2] Harihara Subramanian and Pethuru Raj Harihara, "Hands-On RESTful API Design Patterns and Best Practices."
[3] James Gough, Daniel Bryant and Matthew Auburn, "Mastering API Architecture" in *O'Reilly Media, Inc.*, 2022
[4] "https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)"
[5] Sourabh Sharma, "Modern API Development with Spring and Spring Bootby".
[6] PPT - Web Services and Their Protocol Stack PowerPoint Presentation, free download - ID:409440 (slideserve.com)
[7] Leonard Richardson co-authored the milestone book, "*RESTful Web Services* (O'Reilly)".
[8] Aloi, "REST APIs Overview (for a non-technical and novice users) "
[9] PPT - Is your web API truly RESTful? Josef Hammer CERN PowerPoint Presentation - ID:2480632 (slideserve.com)
[10] Fernando Doglio, "REST API Development with Node: Manage and Understand the Full Capabilities of Successful REST Development."
[11] http://www.json.org
[12] HTTP status codes with explanation, Job and Interview Preparation (careerguroo.blogspot.com).