International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 06 | June 2021 www.irjet.net p-ISSN: 2395-0072

REST API Framework: Designing and Developing Web Services

Utkarsh Singh

Dept of Information Science and Engineering, R V College of Engineering, Bengaluru-560059

---***---

Abstract - *REpresentational State Transfer also known as REST is an architectural style and an approach for communicating the data that is used for web service development. It is a stateless client server model. The client requests the server for required information via the API and then the web server sends back the response to the client. REST is used so extensively nowadays that it is also known as language of the internet. REST API's are used in various places such as cloud computing, Internet of things, Microservices etc. This paper talks about the framework, its architecture, conventions, security aspects related to REST framework.*

Key Words: **RESTful web services, RESTful, API, HTTP, CRUD, Web**

1. INTRODUCTION

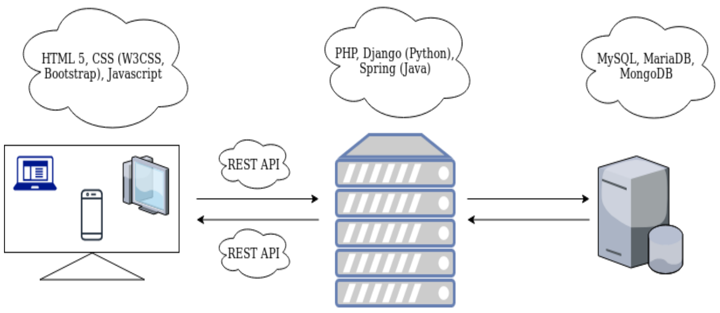
Computers are really good at doing what we tell them to do provided we explicitly state how to do it. It holds true for HTTP protocol as well. This suggests that HTTP is a stateless protocol. It essentially means that we need to provide information to the protocol inorder to get what is required. REST API's are simple, easy maintainable APIs built on REST architecture. The client does certain operations and requests the data from server via REST APIs. The web server having the access to the Database query the required data and returns to the client the response in JSON/XML format.

The API is referred to as a contract between consumer/client (the call) and producer/server (the response). When a request is made by a client via RESTful API, it transfers the state of the resource to the endpoint. This representation is delivered in several formats via HTTP such as JSON, HTML, PHP, Python XML etc. Various other information such as metadata, authorization, URI, cookies etc is delivered via HTTP headers

REST API is the chosen standard in today's world because of:

- Flexibility: It is simple to do a migration from one server to another or switch between different databases at any time.
- Scalability: Because of separation between client and server, it can be scaled up at individual component level without any difficulty.
- Easy to build and uses less resources.
- Reliability - It is more resistant to failure.

A RESTful system consists of


HTML 5, CSS (W3CSS, Bootstrap), Javascript
PHP, Django (Python), Spring (Java)
MySQL, MariaDB, MongoDB
REST API
REST API

2. TERMINOLOGIES AND KEY ELEMENTS

2.1 Principles of RESTful API

There are namely 6 architectural constraints that makes a web service as a RESTful API:

- **Uniform Interface**: It defines a uniform interface unit between a client and a server by simplifying and decoupling the whole business architecture which enables each part to build on its own. The responses and error codes should have self-descriptive messages.
- **Stateless:** It means that the necessary state to process a request is sent within the request as a part of URI, body, headers or parameters.
- **Cacheable**: Since on the web, the clients can cache responses locally hence responses must explicitly define themselves cacheable or not to prevent clients from using the older data to further requests and transaction speed could be also faster.
- **Client-Server**: The client requests from server and the server queries from database and return the response back to the client.
- **Layered System**: The client may not be connected directly to server of intermediary server. This

layered approach makes sure each layer doesn't directly affect the other layer. The REST framework comes in the application layer of OSI model

- **Code OnDemand**: This is an optional feature to add. According to the constraint, server can provide executable code to the client and they can execute it. Examples could be client-side scripts such as JS or compiled components like Java applets.

2.2 Basic Components of REST Request

There are two basic steps to making a request a REST API:

- The client accesses a specific location on the database via REST API and states the method that should be executed. This is basically known as a **request.**
- The server runs the method, returns the data to the client. This is known as a **response.**

There are four major components that constitutes the request-response process:

Endpoint: The location of the specific resource. REST APIs use URI (Uniform Resource Identifiers) that creates addresses for the resources. The endpoint (or route) is the URL that's requested. In this case it is typically a URL for example:

http://api.reddit.com/create

Headers: Meta-data or information about the request/response that contains important things like authorization, data type contained, caching information and related cookies etc.
Some common headers which might be encountered while working with REST APIs are:

- Content-Type
- Authorization
- Cache-Control

Method: The type of interaction that is being performed on the resource. REST APIs help us to develop web applications which have all possible CRUD (create, read, update, delete) operations.

Data (or Body): Contains the data payload that is either sent to or received from the server. A REST body can contain virtually any type of media, but most popular is application/json

2.3 HTTP Response Status Codes

HTTP response status codes tell us if an HTTP request has been successfully processed with appropriate response of the REST API. Different responses can be majorly grouped in five different classes:

- **Information response (100 to 199)**
  103(Early Hints), 101(Switching Protocol), 100(Continue)
- **Successful response (200 to 299)**
  200(OK), 203(Authorization error), 201(Created), 204(No content)
- **Redirect (300 to 399)**
  301(Moved Permanently),302(Found), 306(Unused), 308(Permanent Redirect)
- **Client errors (400 to 499)**
  401(Unauthorized), 400(Bad Request), 404(Not Found), 403(Forbidden), 408(Request Timeout)
- **Server errors (500 to 599)**

500(Server Error), 502(Bad Gateway), 501(Not Implemented)

2.4 Conventions for REST API

Table -1: Conventions to follows

Endpoint	HTTP verbs	Description about the rule
api/client	GET	Get all clients
api/client/new	GET	Display a form to add a new user
api/client	POST	Adds a client to the database
api/client/tom	GET	Get tom's information
api/client/tom/edit	GET	Show a form/input to edit client tom's information
api/client/tom	PUT	Update tom's information
api/client/tom	DELETE	Delete tom's data

3. AUTHENTICATION OF REST-API

Authentication is critical for internet security. REST API lets client's access and modify the internal data but only after an authentication is in place.

HTTP Authentication Schemes:

3.1 BASIC Authentication

HTTP Basic Authentication is the most basic authentication and straightforward method. Using this sender puts username and password into the request header. The data is encoded with Base64 to ensure safe transmission. It doesn't require cookies, session IDs etc.

3.2 Bearer Authentication

It is also known as token authentication which involves security tokens also known as bearer tokens. The bearer token gives access to a specific resource/URL and is most likely to be a cryptographic string, usually generated by server in response to login request. Client would then send the token in the authorization header whenever it makes requests to protect resources from external attack.

3.3 API Keys

Here, a unique generated value is assigned to each user for the very first time, which is sort of a fingerprint for that user. When a user tries to login again into the system the same Api key is used to prove the proper identity. For security purposes the Api keys or any sensitive information should not be shared within the query string as parameters. A much better approach is to put the API Key inside the authorization header.

3.4 OAuth

This is the most common authentication used. It uses an access token and refresh token. Access token is shared just like an API Key, it permits the application to have access to the user's data. Refresh tokens are optional. They fetch a new access token if the older ones are expired.

4. CONCLUSION

In this paper, the author explains in depth about RESTful APIs and the importance of it in designing and developing web services and making it more secure. REST APIs services are best for public URLs only if it is made secure while sharing confidential and sensitive information between client and server.In this paper, the author has introduced various security methods such as OAuth, OpenID etc for securing the developed web service. REST is very efficient compared to SOAP architecture and hence used as an industry standard currently.

5. REFERENCES

[1] Noureddine, M., & Bashroush, R. (2011). A provisioning model towards OAuth 2.0 performance optimization. 2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS).

[2] Haupt, F., Leymann, F., Scherer, A., & Vukojevic-Haupt, K. (2017). A Framework for the Structural Analysis of REST APIs. 2017 IEEE International Conference on Software Architecture (ICSA).

[3] Solapurkar, P. (2016). Building secure healthcare services using OAuth 2.0 and JSON web token in IOT cloud scenario. 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I).

[4] Kesinee Boonchuay, Youppadee Intasorn, Kritwara Rattanaopas(2017) Design and implementation of a REST API for association rule mining(IEEE-2017 14th International Conference).