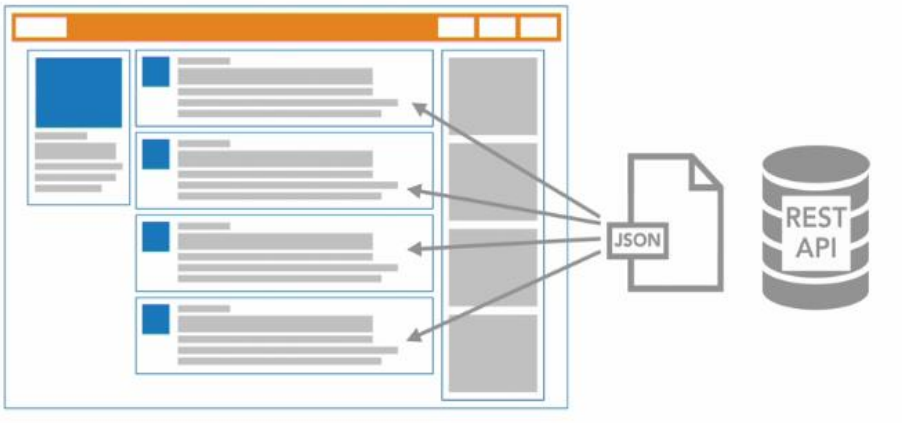
REST API - Representational state transfer application programming interface

Clients requests data from REST API. The client can be a website,app,Iot device, a program etc.

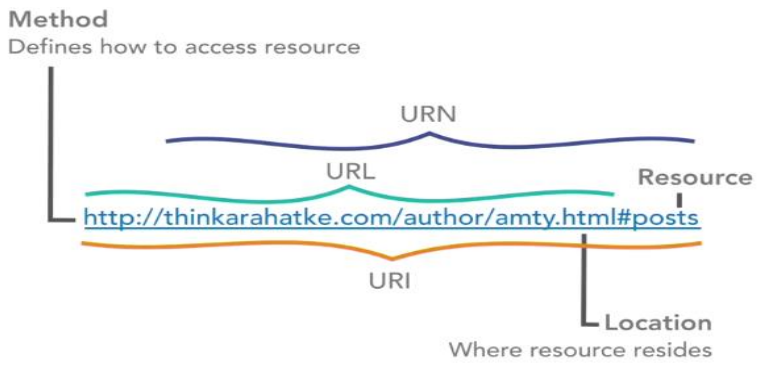
Ex: Access tweeter API to get latest tweets. The API is a set of programs and rules written to provide data to requested resources (clients) from the servers.



1)It's an interface between data source and websites.

2)Websites consists of various information which is in the form of stream of data. The templete remains same but everytime when refreshed, the data object is requested from website to API and API sends the data object to websitewhich fits into the templetes of website. Everytime when a new feed is requested by website, the original state of website is replaced with new state (with new data objects) representing the data on the website.That's why it is called REST.

API is a set of rules which works with the REST framework and understands the GET,POST,DELETE requests made by the website.



There are 3 terms URL, URI and URN. From the above picture it shows that URL is the location of data, URL is URL+particular resource containing data and URN is the name of the resource with location. URN does not include protocols like https, ftp etc.

REST and HTTP are not linked, they're just a convenient pairing. When a REST service runs on the web over HTTP to give us access to a web resource, we call it a RESTful API. The web platform is what makes it RESTful. In other words, if you send a request through HTTP to a REST service that service is a RESTful API. REST API can also get requests from client over FTP, SMTP or other protocols but when it gets request from HTTP then it becomes Restful API.

Parts of Rest API:

1) **Methods:** They are standard HTTP operators.
   GET, POST, PUT, PATCH, DELETE, OPTIONS, HEAD

2) **URI:** The resource we want to interact with.

   The above method works on the URI when a client access API.

   Get request uses the URI details along with some authentication, data format and settings to get data from API.

   Post request is sent to API to create a new data on the server. In post request we use post keyword with the data in xml or json format. The data format is also passed along with post request to make API understand the format of data.

   Normally we use java script to get and post requests. For post request we use Ajax call which consists of data as well while sending request to server via API.

The option method of rest API tells what are the methods available in API and the arguments which can be passed while using methods. This helps to identify the functionality provided by API.

The term resource is an important keyword which is used while working with API. Resource is a data which can be text, video, audio, img etc. and kept in the server which client wants to access. This resource has its own conceptual mapping.

Ex: In a library if we want to get red book named C programming, the below mapping will be used:

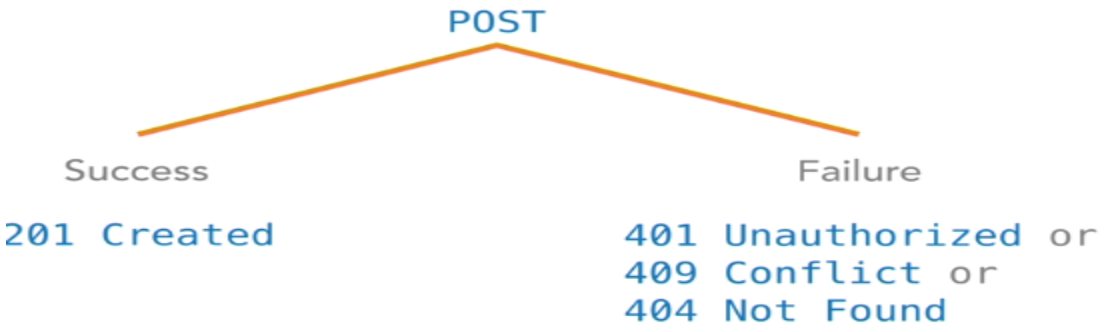
Library/bookshelf/redbooks/cprogramming. Here the path is the conceptual mapping and cprogramming is the resource. Resource can also be the group of other resources like Library/bookshelf/redbooks. Here resource is collection of all the red books.

When a client access the data via API, the API creates the copy of the resource or resources and the client doesn't access the actual resource. It access the copy of resource. Because of this it is possible to access same resource by multiple clients at the same time and any changes in the resource is then merged with the actual resource.

Below are some of the important methods which are mostly used while accessing API. They are:

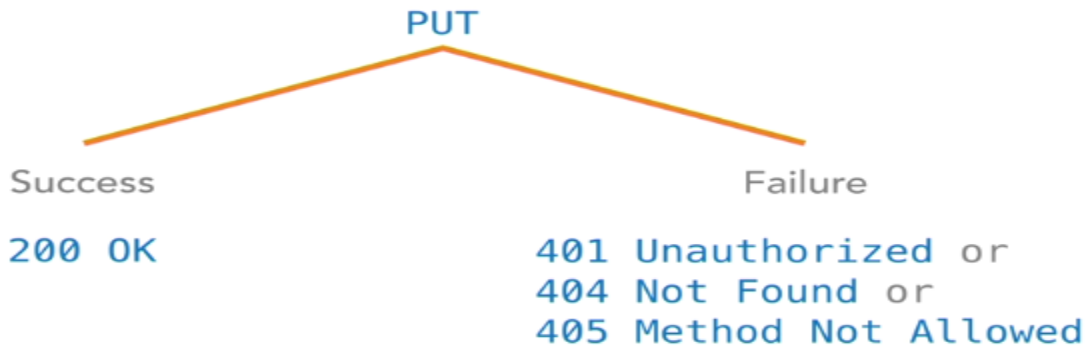
1) GET – To get the data from server using API we use get request instruction to API. It either returns **success 200 OK** with the requested data or **failure 404 Not Found**

2) To send the data from client to server there are 3 different methods which are sent as instructions with the data to server. They are POST, PUT, and PATCH. The POST method is used to add a data in the server.



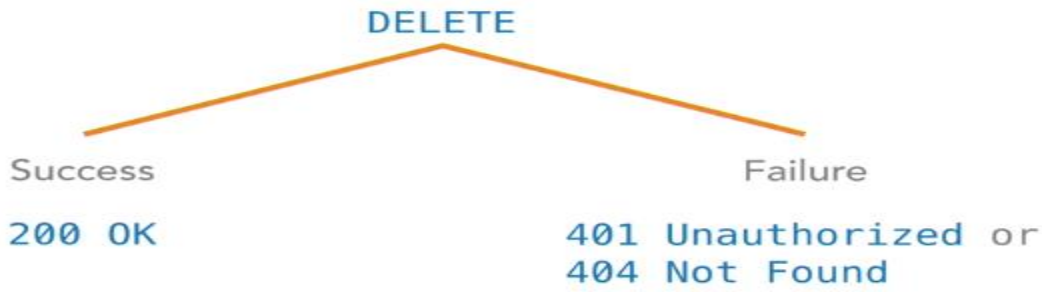
Ex: submit a form containing details. When a post is successful it return 201 code else it returns 401 which means client doesn't have access to add data to server or 409 meaning the data client

is trying to add already exist or 404 meaning the resource in which data is requested to be added does not exist.(URI does not exist).



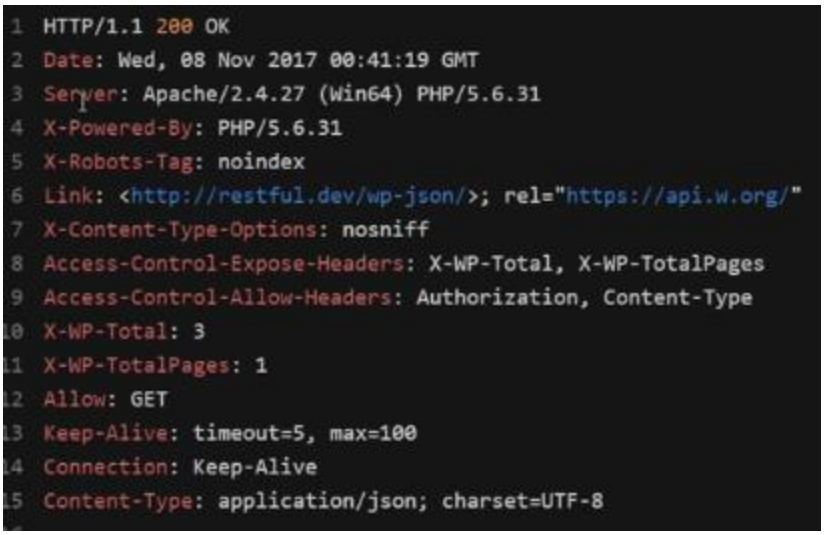
PUT request is used to update the data of the resource which is already present. It overwrites the content of resource. Example updating user details. When success it returns 200, when not able to update data due to access issue it returns 401, when resource is not present in the server it returns 404 and when PUT method wants to update the collection of resource then 405 comes which is because PUT method is designed to update singleton resource not collection of resources.

Patch is used to modify an existing resource. Where Put updates the resource by replacing content, Patch can carry along instructions on how to modify the existing resource without necessarily replacing everything. Patch returns the same status as Put.



Delete is used to delete a singleton resource not the collection of resources. It return 200 on successfully deleting the resources, 401 when not authorize to delete a resource ,404 when resource not found on server and 405 when method not allowed to delete collection of resources when passed with delete command to server via API.

OPTION is used to describe the details of methods present in API which can be used to perform manipulations (DELETE, PUT, PATCH, and POST) or get data (GET) from API.



HEAD: When we send a request to API then response from API contains a head section. Head section is used by clients to handle the returned data. In the head section the details are like the protocol used, response from server, server details, authentication details and type of content with encoding which is handled by API. Every response from an API will have head section and the details mentioned in HEAD section will vary based on the instruction fired to API (like GET, POST, PATCH etc.).

HTTP response code:

In the header of every request fired to API, the header returned from API contains the HTTP response code. Client can use these codes to identify the success or failure of response and act accordingly. Each code consists of different meanings.

1xx	Information
2xx	Success
3xx	Redirection
4xx	Client error
5xx	Server error

Status code group 1xx is used to provide information to the client like the server is ready to process request, waiting to process etc.

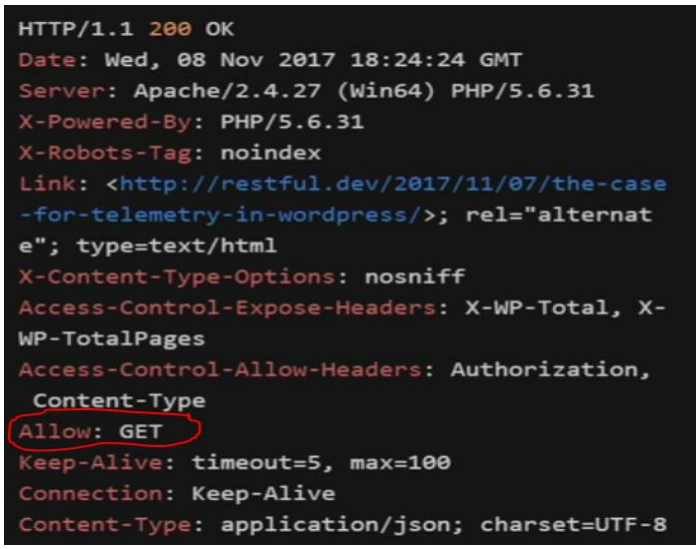
Status code 2xx is used to provide success information like 200: the request was successful, 201: The request was successful and the resource was created (ex: PUT request) and 204: server processed the request and returned no content.

Status code 3xx indicated redirection i.e. the client is provided with the new URL/URI to access the data. The 301 return code indicates that the URL is moved permanently to other URL, 302/303: The URL requested is found in some other URL, 307: The URL requested is temporarily moved to other URL. The 3xx code is little confusing and depends on the logic implemented while building our own APIs.

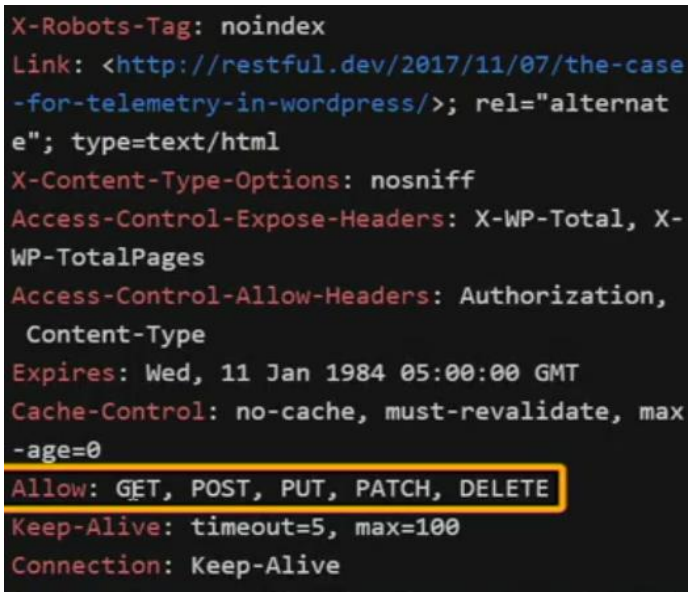
The 4xx codes are related to errors occurred at client end. The 5xx codes are related to errors occurred at server error i.e. server not available or issue with the server.

For every API, certain set of authentications are applied. There are few users who can only GET data, few who can POST and GET and very few who can PATCH, PUT and DELETE. Below are the details shared in the header which is received as a response from server. Here Allow keyword **(in red)** shows the authorization of different users.

Normal user:



Superuser with all authorization:



This shows the response changes with the change in the authorization of users.

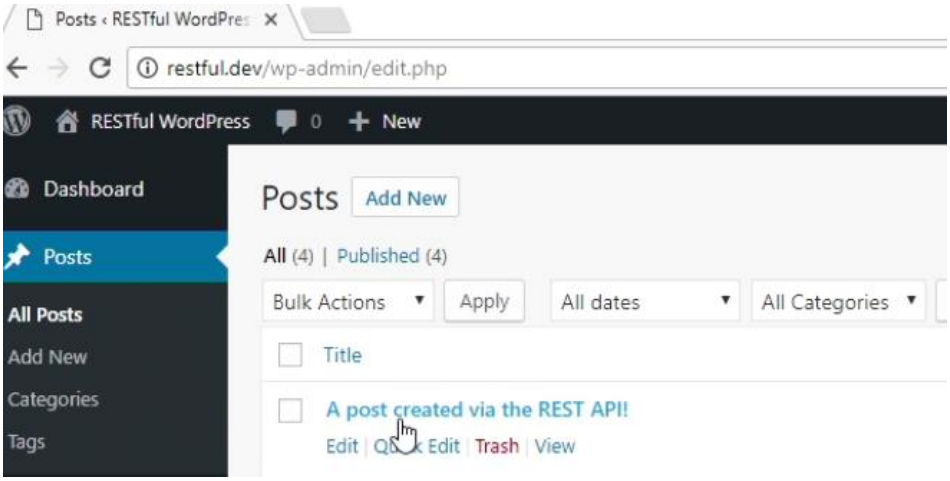
Below is the example of POST/DELETE

POST:

```
POST http://restful.dev/wp-json/wp/v2/posts
Authorization: Basic morten [redacted]
Content-Type: application/json

{
    "title": "A post created via the REST API!",
    "content": "This is the content of the post created with the REST API."
    "status": "publish",
    "author": 1
}
```

Output on wordpress (post created via API):



DELETE:

```
DELETE http://restful.dev/wp-json/wp/v2/posts/15
Authorization: Basic morten [redacted]
```

But in the WordPress API there is an option that when we fire delete it will go to trash so we checked in the OPTIONS how to permanently delete post using the arguments. The option force can be set as true to delete the post.

The below command deletes the posts permanently from an API