**API\_Testing**

What is API – Application Programming Interface. It is an intermediator software that communicates two software.

Webservices example 🡪 MakeMyTrip flights communicating with diff flight service provider.

Response of any request contains many things like Response Body, Headers, Status Code, Response Time etc. Response body contains key value pair mostly in Json format. Each key value pair is known as Node. In REST web services, response is generally shared in JSON format.

|  |  |
| --- | --- |
| **Web Services** | **API** |
| All the web services are APIs. | All the APIs are not web services. |
| Web services need internet connection to work. | API not always need internet connection, like create a function and deliver it as JAR file e.g Apache POI |
| Web services might not contain all the specifications and can not perform all the tasks that API can perform. E.g. If user login using credentials then webservices will be called which will send the data and receive response but API will manage everything like extracting the data from response, authenticating it and showing validation result on screen like login successful or login failed. |  |
| Web services uses three types of styles : SOAP, REST and XML-RPC. | API may be exposed to multiple ways. |

There are two types of Web Services - SOAP and Rest.

For manual testing of these web services, we use POSTMAN and SOAPUI.

POSTMAN –It is free automation tool / plugin used to automate Webservice API. It is used to test both SOAP and Rest services.

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| Simple Object Access Protocol | Representational State Transfer |
| SOAP is a protocol | REST is architecture and it uses HTTP protocol |
| SOAP exposes behavior which represents logic | REST exposes resources which represents data |
| It is somewhat complex | It is much simpler |
| SOAP uses XML for parsing which is complex for conversion so it is slow. | REST uses JSON / XML so it is fast |
| SOAP provide good security options compared with REST |  |

Commonly used HTTP methods 🡪

* GET Method – It simply retrieves data from server. No change on server or resource.
* POST Method – This method performs changes in server. In simple words, post always creates resources in server.
* DELETE Method- It simply deletes data from server.
* PUT Method – This method updates the existing resource. In simple words, this method can update the existing file on server.
* After POST request, you always get 201 response means post successfully created.
* Get, Delete, Put will not create additional load on server while making multiple calls.
* But POST can create load on server if made multiple calls.

Architectural styles of different web APIs 🡪

* + HTTP for client server communication
  + XML/JSON for data exchange
  + Simple URI as address for service

What is API Testing 🡪

API testing is a kind of software testing which determines if the developed APIs meet expectations regarding functionality, reliability, performance and security of application. Generally any website has three layers GUI, business logic and dataset layer. API testing works with core business logic testing.

Advantages of API Testing 🡪

* Testing of core functionality as it certifies back end.
* Time effective as compared to UI testing / automation (selenium) testing.
* Language independent, data is exchanged using XML or JSON.

Common protocols used in APIs 🡪

* JMS (Java Messaging Services)
* HTTP
* SOAP

API Testing types 🡪

* Functional Testing
* Load Testing
* Validation Testing
* Security Testing
* Interoperability Testing

Tools used for API Testing 🡪

* Postman
* JMeter
* RestAssured
* SoapUI

Challenges in API Testing 🡪

<https://www.tutorialspoint.com/rest-api-testing-tutorial-sample-manual-test-case>

* Required info like what data to be passed, parameters to be passed, what headers to be passed
* Access to DB for asserting the API response. This is mainly when APIs are from third party company where database is not shared.
* Authenticate / Authorization overhead

Difference bet POST and PUT 🡪

* In POST request we create new object on server.
* In PUT request we update the existing object.

Difference bet PUT and PATCH 🡪

* PUT means replace the entire resource with given data, while PATCH means replace only specified fields.
* Example – If you want to update the first name then in PUT you need to send all other payloads also along with first name (though other fields are to be kept unchanged) while in PATCH you send only first name to update it.

Authentication techniques in API 🡪

* Basic Authentication (it requires username and password)
* OAuth 2.0 (we need to generate access token and then need to provide it in the request)
* Session / Cookie based authentication
* Digest Authentication

<https://www.youtube.com/watch?v=8cyMZ87J7zo&t=422s>

What is REST API 🡪

REST stands for Representational State Transfer. It is a set of API helping developers to send the request and receive responses. Interaction is made through HTTP protocol in REST API.

Verification Points in API 🡪

* Accuracy of data (say confirmation with DB)
* HTTP Status Code
* Response Time (mostly useful for performance testing)
* Authorization

**Diff bet URI & URL 🡪**

Please read at https://www.guru99.com/url-vs-uri-difference.html

**Reading an URL 🡪**

URL = <https://www.vjthoratacademy.com/courses/11256?locationIND>

End point = <https://www.vjthoratacademy.com>

Resource = courses

Path parameter = 11256 (It is followed by ‘/’)

Query parameter = locationIND (It is followed by ‘?’)

**Difference between Path parameter and Query parameter 🡪**

Path Parameter is basically used to identify a specific resource or resources whereas Query Parameter is used to filter/sort those resources

**Format of URL 🡪**

<protocol>://<service-name>/<ResourceType>/<ResourceID>

**Core Components of HTTP Request 🡪**

* What type of HTTP request method it is like GET, POST, PUT or DELETE
* Base URI (uniform Resource Identifier)
* Resource and parameters
* Request Headers (carries info necessary for API request like connection type, proxies)
* Request Body / Payload (like for POST request need to send info name, id, salary etc)

**Serialization and Deserialization in RESTAssured 🡪**

* Serialization in RESTAssured means converting a java object into file format (payload of JSON format) and deserialization means converting the file format (from response body) back to java objects.
* For serialization ‘FileOutputStream’ and ‘ObjectOutputStream’ classes are used. While for deserialization, ‘FileInputStream’ and ‘ObjectInputStream’ classes are used.
* Serialization is mainly used in RestAssured during POST request. All the body can be converted to file format and will be send (instead of using Map collection).

**Documentation of API should include 🡪**

* What kind of request it is
* What is the URL
* What is the resource
* What are the path parameters, query parameters
* What is the JSON payload need to provide
* What response code will be received

**API Documentation Templates 🡪**

* Swagger
* Slate
* API Blueprint

**Points to be considered during API documentation 🡪**

* Source of content (URL, URI, path parameter)
* Information needed for every call
* Delivery layout (flow diagram of API)

**What is ‘Resource’ in REST 🡪**

* Any content which is fetched through server API is treated as resource like text, image, video, HTML Page.

**Popular ways to represent a resource in REST 🡪**

* JSON
* XML
* Text

**In API why there is always Authorization section and not Authentication 🡪**

* Authentication means ‘who you are’
* Authorization means ‘what access you have’
* In API, we try to access the resource. So authorization is required, just authentication is not enough.

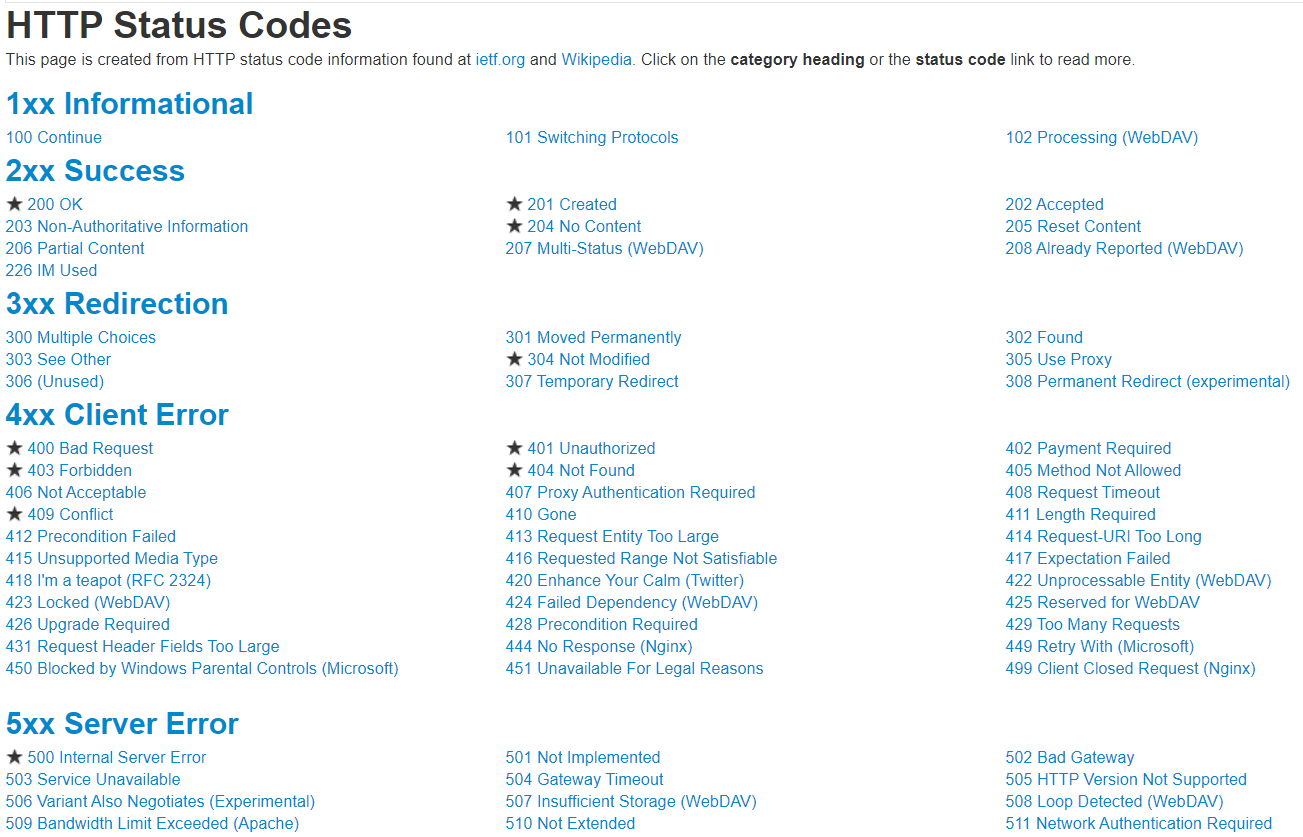
**What is idempotent http method 🡪**

An HTTP method is idempotent if an identical request can be made once or several times but server remains in the same state. In other words, it shouldn’t create any new load on the server.

PUT request is idempotent but POST request is not idempotent.

**Status Codes in API**

<https://www.restapitutorial.com/httpstatuscodes.html>



2xx 🡪 Success family

200 🡪 OK

201 🡪 Created (like create user, create new object)

202 🡪 Accepted & resource is marked for update / deletion but actual operation will take some time, It is generally received in case of DELETE request. Action will likely succeed but has not yet been enacted due to asynchronous tasks (other thread if using it) / dependencies.

204 🡪 Success but nothing to return (everything went well but nothing to return), received in case of POST / DELETE requests. In case of DELETE request, we get this status code if the action has been enacted and no further information is to be supplied.

3xx 🡪 Redirection

304 🡪 Not Modified (it is returned if file is unchanged since it was last accessed on the server. While making a request, browser includes ‘If-Modified-Since’ header and if file hasn’t been modified, request won’t be fulfilled and cached version of file will be served. It is used for optimization. If webpage you are requesting today is same as yesterday then cached version will be served instead of fetching data from server again.)

4xx 🡪 Client Error family (means something is wrong with webservice request)

400 🡪 Bad Request (like all the needed info is not sent)

401 🡪 Unauthorized (haven’t provided the needed authentication or wrongly provided), server says ‘I don’t know who you are’

403 🡪 Forbidden (provided authentication is not enough like with basic authentication, user is trying to access admin info), server says ‘I know who you are, but you do not have sufficient rights’

404 🡪 Not Found (trying to access a part of web which do not exists)

5xx 🡪 Server Error family

500 🡪 Internal server error (something broke on the server)

**JSON Extraction**

* JSON stands for Java Script Object Notation.
* We can use <http://jsonpath.com/> to check our json extraction syntax.
* We can align our JSON response properly using <https://jsonpathfinder.com/>.
* If key contains multiple set of values, then it becomes array & is always represented by [] square brackets.
* In JSON for any key has value set surrounded by {} curly brackets then all elements inside it must be in key value pair (can’t be normal array values).

{

"firstName": "John",

"lastName" : "doe",

"age" : 26,

"location" : ["Pune", "Mumbai", "Delhi"],

"address" : {

"streetAddress": "naist street",

"city" : "Nara",

"postalCode" : "630-0192"

},

"phoneNumbers": [

{

"type" : "iPhone",

"number": {

"own" : "0123-4567-8888",

"family" : "91-2046387469"

}

},

{

"type" : "home",

"number": "0123-4567-8910"

}

]

}

|  |  |
| --- | --- |
| **JSON Path** | **Extracted Value** |
| .address.city | "Nara" |
| .location[0] | "Pune" |
| .phoneNumbers[1].type | "home" |
| .phoneNumbers[1] | "type": "home",  "number": "0123-4567-8910" |
| .age | 26 |
| .phoneNumbers[0].number.family | "91-2046387469" |