\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**REST API**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

1. **Client/server Error Codes**

4XX Series

These are specific to the client-side error.

400 (Bad Request) : A bad request means that the syntax of the request was incorrect. It can happen if you have sent wrong parameters along with the request url or in the body of the request.

401 (Unauthorized): Indicates that the request requires user authentication information. The client MAY repeat the request with a suitable Authorization header field.

402 Payment Required (Experimental).

403(forbidden): Unauthorized request. The client does not have access rights to the content. Unlike 401, the client’s identity is known to the server.

404 (Not Found): A response code 404 means that the server was connected but it could not find what was requested. You can normally see this status code when you request a web page which is not available.

405 (Method Not Allowed): The request HTTP method is known by the server but has been disabled and cannot be used for that resource.

407 (proxy authentication required):Indicates that the client must first authenticate itself with the proxy.

408 (Request Timeout): Indicates that the server did not receive a complete request from the client within the server’s allotted timeout period.

409: Conflict : The request could not be completed due to a conflict with the current state of the resource**.**

429: To Many requestes

**5XX Series**

These are specific to the server-side error.

500 – Internal Server Error: The server encountered an unexpected condition which prevented it from fulfilling the request.

501 – Not Implemented : The HTTP method is not supported by the server and cannot be handled.

502 – Bad Gateway : The server got an invalid response while working as a gateway to get a response needed to handle the request.

503 – Service Unavailable

504 – Gateway Timeout

505 – HTTP Version Not Supported : The HTTP version used in the request is not supported by the server.

506 – Variant Also Negotiates

507 – Insufficient Storage

508 – Loop Detected

510 – Not Extended

511 – Network Authentication Required: Indicates that the client needs to authenticate to gain network access.

**2. Success codes**

2XX Series

200 (OK): Defines that the request was correct.

201 (Created): The value wrapped with the request has been created in the database. It is needless to say that the request was correct.

202(Accepted): Indicates that the request has been received but not completed yet. It is typically used in log running requests and batch processing.

204(No Content): This status code means that the request was correct and received but there is no response to send to the client by the server.

205 (Reset Content): Indicates the client to reset the document which sent this request.

**3. Use of GET, PUT, POST, PATCH, DELETE**

**1.** ***GET -****As the name suggests, the Get method fetches the information from the server. Moreover, it is the most commonly used method which does not have a request body. Every time you open a website, the Get request fires to retrieve the website contents. Additionally, it is equivalent to the read operation. Some of the main features of the GET method are-*

* *We can easily bookmark the data using the GET method.*
* *The limit of the length of values is generally 255 characters for the GET method.*
* *GET supports only string data types.*
* *GET requests are cacheable.*
* *The parameters passed in GET methods store in the browser history.*

*An example GET request would look like this -*

***GET https://bookstore.toolsqa.com/BookStore/v1/Books HTTP/1.1***

**2.** ***HEAD:****The Head method is similar to the Get method, but it retrieves only the header data and not the entire response body. Moreover, we use it when you need to check the document's file size without downloading the document.*

**3.** ***POST:****The Post method works to send data to the server. You may add or update data using the Post request. We send the information that needs to update in the request body. In the real world, the form data on website updates using the Post request. Some of the critical features of a POST method are-*

* *Data passed through the POST method is not visible in the browser URL.*
* *Additionally, values passed through POST are not stored in browser history.*
* *Moreover, there is no restriction on the length of data sent through the POST method.*
* *Also, POST method request supports different data types like String, binary, integers, etc.*

**4.** ***PUT:****The Put method is similar to the Post method since it updates the data. The only difference is that we use it when we have to replace an existing entity completely. Also, PUT methods are idempotent, i.e., they return the same result on executing repeatedly.*

**5.** ***PATCH:****This method is again similar to Post and Put methods, but we use it when we have to update some data partially. Moreover, unlike the Post and Put methods, you may send only the entity that needs updation in the request body with the Patch method.*

**6.** ***DELETE:****Like its name, the Delete method deletes the server's representations of resources through the specific URL. Additionally, just like the Get method, they do not have a request body.*

**7.** ***OPTIONS:****This is not a widely used method when compared to other ones. It returns data specifying the different methods and the operations supported by the server at the given URL. Moreover, it responds with an Allow header giving a list of the HTTP methods allowed for the resource.*

**4. Difference between Put and Patch**

**5. Difference between Put and Post**

## PUT v/s POST Request

* PUT  is idempotent while POST  is not.

As Wikipedia puts it,

Idempotent is the property of specific procedures in mathematics and computer science, that can be implemented multiple times ***without altering the result*** beyond the initial application. To clarify:

* Therefore, the ***PUT method is idempotent*** because no matter how many times we send the same request, the results will always be the same.
* POST is not an idempotent method. In other words, Creating a POST multiple times may result in various resources getting created on the server.
* Moreover, there is no distinction between PUT and POST if the resource already exists.
* The following example will help in conceptually clarifying the difference between PUT and POST requests.

POST /vehicle-management/vehicle: Create a new device

PUT /vehicle-management/vehicle/{id} : Updates the vehicle information identified by “***id***”

### ***Appropriate status codes obtained for PUT and POST requests:***

***POST***

* ***201*** and a location header pointed to the new resource
* ***400*** if we are not able to create an item

***PUT***

* ***204*** for OK (but no content)
* ***200*** for OK with Body (Updated response)
* ***400*** if the supplied data was invalid

**6. How to handle authentication**

In rest assured by passing the username and password in request header.

**7. Difference between webserver and API**

**8. Json/xml structure format**

{ "name":"Chaya", "age":55, "city":"Boulder", "type":"Canine" }

**9. How to validate json response**

By using jsonPath

**10. Use of JsonPath**

**JSONPath** is a query language for JSON with features similar to XPath for XML. **JSONPath** is used for selecting and extracting a sub-section from the JSON document.

**jsonPath is used to validate json response in restassured**

**11. How to construct json body for Put, Post, Patch request.**

By using JsonObject class

**12. Use of JsonObject.**

**JSONObject**, **JSONObject** class is an unordered collection of key-value pairs. It provides methods to access values by key and to put values

jsonObject class is used to construct json body for post and put request.

**13. What's is URI**

URI stands for Uniform Resource Identifier. It is a string of characters designed for unambiguous identification of resources and extensibility via the URI scheme.

The purpose of a URI is to locate a resource(s) on the server hosting of the web service.

A URI’s format is <protocol>://<service-name>/<ResourceType>/<ResourceID>.

**14. What's a resource**

REST architecture treats any content as a resource, which can be either text files, HTML pages, images, videos or dynamic business information.  
REST Server gives access to resources and modifies them, where each resource is identified by URIs/ global IDs.

**15. What are the common headers used.**

Here are some of the most common API Headers you will encounter when testing any API.

* **Authorization:** Contains the authentication credentials for HTTP authentication.
* **WWW-Authenticate:** The server may send this as an initial response if it needs some form of authentication before responding with the actual resource being requested. Often following this header is the response code 401, which means “unauthorized”.
* **Accept-Charset:** This header is set with the request and tells the server which character sets (e.g., UTF-8, ISO-8859-1, Windows-1251, etc.) are acceptable by the client.
* **Content-Type:**  Tells the client what media type (e.g., application/json, application/javascript, etc.) a response is sent in. This is an important header field that helps the client know how to process the response body correctly.
* **Cache-Control:** The cache policy defined by the server for this response, a cached response can be stored by the client and re-used till the time defined by the Cache-Control header.
* Content-Encoding
* Transfer-Encoding
* Server

**16. Common content types**

**“text/plain”, “application/xml”, “text/html”, “application/json”, “image/gif”, and “image/jpeg”**

**17. Difference between SOAP and REST API.**

* SOAP stands for Simple Object Access Protocol whereas REST stands for Representational State Transfer.
* SOAP is a protocol whereas REST is an architectural pattern.
* SOAP uses service interfaces to expose its functionality to client applications while REST uses Uniform Service locators to access to the components on the hardware device.
* SOAP needs more bandwidth for its usage whereas REST doesn’t need much bandwidth.
* Comparing SOAP vs REST API, SOAP only works with XML formats whereas REST work with plain text, XML, HTML and JSON.
* SOAP cannot make use of REST whereas REST can make use of SOAP

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| * SOAP stands for Simple Object Access Protocol | * REST stands for Representational State Transfer |
| * SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service. | * REST is an Architectural style in which a web service can only be treated as a   RESTful service if it follows the constraints of being   * 1. Client Server   2. Stateless   3. Cacheable   4. Layered System   5. Uniform Interface |
| * SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern. | * REST can make use of SOAP as the underlying protocol for web services, because in   the end it is just an architectural pattern. |
| * SOAP uses service interfaces to expose its functionality to client applications. In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer. | * REST use Uniform Service locators to access to the components on the hardware device. For example, if there is an object which represents the data of an employee hosted on a URL as http://demo.guru99 , the below are some of URI that can exist to access them.   http://demo.guru99.com/Employee  http://demo.guru99.com/Employee/1 |
| * SOAP requires more bandwidth for its usage. Since SOAP Messages contain a lot of information inside of it, the amount of data transfer using SOAP is generally a lot.   <?xml version="1.0"?>  <SOAP-ENV:Envelope  xmlns:SOAP-ENV  ="http://www.w3.org/2001/12/soap-envelope"  SOAP-ENV:encodingStyle  =" http://www.w3.org/2001/12/soap-encoding">  <soap:Body>  <Demo.guru99WebService  xmlns="http://tempuri.org/">  <EmployeeID>int</EmployeeID>  </Demo.guru99WebService>  </soap:Body>  </SOAP-ENV:Envelope> | * REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages. Below is an example of a JSON message passed to a web server. You can see that the size of the message is comparatively smaller to SOAP.   {"city":"Mumbai","state":"Maharastra"} |
| * SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format. | * REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON. |

18. Difference between wsdl url and rest API.

**19.What is difference between API and WebService.**

**KEY DIFFERENCE**

* Web service is a collection of open source protocols and standards used for exchanging data between systems or applications whereas API is a software interface that allows two applications to interact with each other without any user involvement.
* Web service is used for REST, SOAP and XML-RPC for communication while API is used for any style of communication.
* Web service supports only HTTP protocol whereas API supports HTTP/HTTPS protocol.
* Web service supports XML while API supports XML and JSON.
* All Web services are APIs but all APIs are not web services.

| **Web Serviced** | **API** |
| --- | --- |
| All web services are APIs. | All APIs are not web services. |
|  |  |
| It supports XML. | Responses are formatted using Web API’s MediaTypeFormatter  into XML, JSON, or any other given format. |
| You need a SOAP protocol to send or receive and data over the network. Therefore it does not have light-weight architecture. | API has a light-weight architecture. |
| It can be used by any client who understands XML. | It can be used by a client who understands JSON or XML. |
| Web service uses three styles: REST, SOAP, and XML-RPC for communication. | API can be used for any style of communication. |
| It provides supports only for the HTTP protocol. | It provides support for the HTTP/s protocol: URL Request/  Response Headers, etc. |

What is Mocking of webservice/API?

What do you validate in API testing?

What is Authentication in API Testing?

Different Authorization used in API Testing?