API Testing Basics Notes:

End point: Address where API is hosted on the Server.

HTTP methods which are commonly used to communicate with Rest APIs are

**GET, POST, PUT, and DELETE**

GET- The GET method is used to extract information from the given server using a given URI. While using GET request, it should only extract data and should have no other effect on the data. No Payload/Body required

**How to send input data in GET?**  
Ans: Using Query Parameters

POST- A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

**How to send input data in POST?**  
Ans: Using Form Parameters /Body Payload

PUT- Replaces all current representations of the target resource with the uploaded content.

DELETE- Removes all current representations of the target resource given by a URI.

**Resources:  
Resources represent API/Collection which can be accessed from the Server**

Google.com/maps  
google.com/search  
google.com/images

**Path Parameters:**  
**Path parameters** are variable parts of a URL path. They are typically used to point to a specific resource within a collection, such as a user identified by ID

<https://www.google.com/Images/1123343>  
<https://www.google.com/docs/1123343>  
<https://amazon.com/orders/112>

<https://www.google.com/search?q=newyork&oq=newyork&aqs=chrome..69i57j0l7.2501j0j7&sourceid=chrome&ie=UTF-8>

**Query Parameters:**  
Query Parameter is used to sort/filter the resources.

Query Parameters are identified with?””

https://amazon.com/orders?sort\_by=2/20/2020

**Headers/Cookies**:

Headers represent the meta-data associated with the API request and response. In layman terms, we were sending Additional details to API to process our request.  
Example: Authorization details

**End Point Request URL can be constructed as below**  
Base URL/resource/(Query/Path) Parameters

**Library API:**

Base URI: <https://rahulshettyacademy.com>

1. **Method**: POST

**Add Book Complete URL** - <https://rahulshettyacademy.com/Library/Addbook.php>

**Input Payload: Json**:

{

"name":"Learn Appium Automation with Java",

"isbn":"bcd",

"aisle":"227",

"author":"John foe"

}

**Output Json**

{

"Msg": "successfully added",

"ID": "bcd227"

}

1. **Method:** GET**, Resource**: /Library/GetBook.php?AuthorName=somename

**Output Json**:

Output the array of Json object books with all below details

{

Name: “bookname”   ( String)

Isbn: “A2fdsf”  (String)

Aisle: 32 (Integer)

}

1. **Resource**: Library/GetBook.php?ID=3389      - **Method** : GET

**Output Json:**

{

"book\_name": "Selenium automation using Java",

"isbn": "a23hd738",

"aisle": "1223"

}

1. **Resource** :/Library/DeleteBook.php, **Method**: POST

**Input Payload: Json:**

{

"ID" : "a23h345122332"

}

**Output Response**:

{  
msg: book is successfully deleted”

}

Automation Scenarios to Test Library API -

Verify if API responses returns Success Codes with Proper Assertions

Verify if Response Json Schema is displayed as expected

Create Environment/Global/Collection variables to dynamically switch end points of APIs to test in different stages of QA Life cycle

Pass response ID of Add Book to Get Book and Delete ID request Parameters for full functional Automation Testing

Validate the book ID response calculation Logic

Validate if Get Book API retrieves the correct response with Book Details requested

Validate if Delete Book API successfully deleted Book

Implement Try Catch Error Handling Mechanism for every Automation test as Tear Down Script

If Book Already exists in DB, Implement Smart Strategy to delete the existing Book first before Adding the Book again as Prerequisite Automation Step

Parse the complete Json response and verify if the values are displayed as expected

Generate Unique ISBN/ aisle value for every run to make Book Unique using Automation Script

Import the Book Details from CSV/Json without hard coding for Validating API’s

Run the APIs with multiple data sets by iterating the data from the CSV as a Data driven testing

What are Environments and variables in Postman?

An environment is a set of [variables](https://learning.postman.com/docs/sending-requests/variables/) you can use in your Postman requests. You can use environments to group related sets of values together and manage access to shared Postman data if you are working as part of a team.

Variables allow you to store and reuse values in your requests and scripts. By storing a value in a variable, you can reference it throughout your collections, environments, and requests—and if you need to update the value, you only must change it in one place. Using variables increases your ability to work efficiently and minimizes the likelihood of error.

How to use Environments and Variables in Postman?

Variable scopes

Postman supports the following variable scopes:

* Global
* Collection
* Environment
* Data
* Local

Scripting in Postman

Postman contains a powerful runtime based on Node.js that allows you to add dynamic behavior to requests and collections. This allows you to write test suites, build requests that can contain dynamic parameters, pass data between requests,

Diagram

Description automatically generated

## The pm object

You will carry out most of the Postman JavaScript API functionality using “pm” which provides access to request and response data, and variables.

Verify if there is an entry called Cypress.

Verify if Cypress entry has course Title and Price properties/keys

Verify if sum of API Courses equals to 90

Verify if Web Automation Course titles are shown as expected titles

**Postman can also test Soap Webservices in addition to Rest HTTP requests**

How Soap Webservices are different from Rest Services?

Rest API’s use HTTP protocol to send the request and receive the response

Whereas Soap Webservices/API’s use Soap Protocol to send the request ad receive the response

In Soap Services, requests/responses are sent in XML Format.

What will we learn from this Course?

How to Call Soap Services from Postman- Understand the rules of setting up Soap Project in Postman

How to write automation tests on the XML response of Soap Services.

Soap End Point example  
<https://www.dataaccess.com/webservicesserver/NumberConversion.wso>

What is Newman?

* Newman is a command line Collection Runner for Postman.
* It allows you to run and test a Postman Collection directly from the command line.
* Using Newman, you can easily integrate it with your continuous integration servers and build systems.
* Newman is built on Node.js. To run Newman, Install Node.js as prerequisite.
* After Node.js install, Install Newman from npm globally on your system, which allows you to run it from anywhere.  
  npm install -g newman
* How to run collection from Newman?

$ newman run <CollectionFile>

* Learn about Newman Configuration options
* Generate HTML reports for Test execution results with newman - htmlextra plugin  
  <https://www.npmjs.com/package/newman-reporter-htmlextra>
* Integrate Postman Automation Scripts to Jenkins CI/CD with the help of Newman commands

Trigger the Postman Automation Scripts from Terminal with the help of Newman CLI.

Integrate the Automation Tests with Jenkins for CI/CD Implementation.

Prepare neat HTML reports for the Postman API Test Automation results.

Understand how to collaborate as a Team by forking the existing Project – Creating branches- Creating Pull requests – Merging.

**OAuth 2.0 Contract Details:**

|  |  |
| --- | --- |
| GrantType | Authorization code |
| redirect URL/Callback URL | https://rahulshettyacademy.com/getCourse.php |
| Authorization server url | https://accounts.google.com/o/oauth2/v2/auth |
| Access token url | https://www.googleapis.com/oauth2/v4/token |
| Client ID | 692183103107-p0m7ent2hk7suguv4vq22hjcfhcr43pj.apps.googleusercontent.com |
| Client Secret | erZOWM9g3UtwNRj340YYaK\_W |
| Scope | https://www.googleapis.com/auth/userinfo.email |
| State | Any random string |
| How to pass oauth in request | Headers |

Mandatory fields for GetAuthorization Code Request,

**End Point**: Authorization server url  
**Query Params**:Scope, Auth\_url, client\_id, response\_type, redirect\_uri  
Output : Code

Mandatory fields for GetAccessToken Request:  
**End point**: Access token url

**Query Params**: Code, client\_id, client\_secret, redirect\_uri, grant\_type

Output: Access token

Google Maps

Add API (POST): This API Will add new place into Server

**Complete URL:** [https://rahulshettyacademy.com](https://rahulshettyacademy.com/)**/maps/api/place/add/json?key=** **qaclick123**

**Base URL**: [https://rahulshettyacademy.com](https://rahulshettyacademy.com/)

**Resource**: /maps/api/place/add/json

**Query Parameters**: key =qaclick123

**Http Method**: POST

**Sample Body**:

{

"location": {

"lat": -38.383494,

"lng": 33.427362

},

"accuracy": 50,

"name": "Frontline house",

"phone\_number": "(+91) 983 893 3937",

"address": "29, side layout, cohen 09",

"types": [

"shoe park",

"shop"

],

"website": "http://google.com",

"language": "French-IN"

}

**Sample Response**

{

"status": "OK",

"place\_id": "928b51f64aed18713b0d164d9be8d67f",

"scope": "APP",

"reference": "736f3c9bec384af62a184a1936d42bb0736f3c9bec384af62a184a1936d42bb0",

"id": "736f3c9bec384af62a184a1936d42bb0"

}

Google Maps Delete API (POST): This API Will delete existing place from Server

**Complete URL**: https://rahulshettyacademy.com/maps/api/place/delete/json?key=qaclick123

**Base URL**: https://rahulshettyacademy.com

**Resource**: /maps/api/place/delete/json

**Query Parameters:** key

**Http request**: DELETE

**Sample Body**:

{

"place id":"928b51f64aed18713b0d164d9be8d67f"

}

**Sample Response**

{

"status": "OK"

}

Google Maps get Place API (GET): This API Will get existing place details from Server

**Complete URL:** http://rahulshettyacademy.com/maps/api/place/get/json?place\_id=xxxx&key=qaclick123

**Base URL**: [https://rahulshettyacademy.com](https://rahulshettyacademy.com/)

**Resource**: /maps/api/place/get/json

**Query Parameters**: key, place\_id //(place\_id value comes from Add place response)

**Http request**: GET {Note: Key value is hardcoded, and it is always qaclick123}

**Sample Response for the Provided Place\_Id**

{

"location": {

"lat": -38.383494,

"lng": 33.427362

},

"accuracy": 50,

"name": "Frontline house",

"phone\_number": "(+91) 983 893 3937",

"address": "29, side layout, cohen 09",

"types": ["shoe park","shop"],

"website": "http://google.com",

"language": "French-IN"

}

Google Maps Put Place API (PUT): This API Will update existing place in Server

**Complete URL:** http://rahulshettyacademy.com/maps/api/place/get/json?place\_id=xxxx&key=qaclick123

**Base URL**: [https://rahulshettyacademy.com](https://rahulshettyacademy.com/)

**Resource**: /maps/api/place/update/json

**Query Parameters**: key

**Http Method**: PUT {Note: Key value is hardcoded, and it is always qaclick123}

**Sample Request:**

{

"place\_id":"8d2573bdf6ceec0e474c5f388fa917fb",

"address":"70 Summer walk, USA",

"key":"qaclick123"

}

**Sample Response for the Provided Place\_Id**

{

"location": {

"lat": -38.383494,

"lng": 33.427362

},

"accuracy": 50,

"name": "Frontline house",

"phone\_number":"(+91) 983 893 3937",

"address": "29, side layout, cohen 09",

"types": ["shoe park","shop"],

"website": "http://google.com",

"language": "French-IN"

}

**E-Commerce API End to End Testing**

* Injecting JWT Session token into API Calls using Postman
* How to work with POST Call form-data
* How to Send Attachments through request using Postman
* Unauthorized 403 API Example with Postman

**End to End Flow:**

Login API -> Create Product -> Purchase Order on created Product -> Delete order -> Delete Product

**Create Product Contract:**

End point – <https://rahulshettyacademy.com/api/ecom/product/add-product>

Http Method - POST

Form Data -

productName: qwerty

productAddedBy: {{userId}}

productCategory: fashion

productSubCategory: shirts

productPrice: 11500

productDescription: Addias Originals

productFor: women

**Delete Product:**

[https://rahulshettyacademy.com/api/ecom/product/delete-product/{{productId}}](https://rahulshettyacademy.com/api/ecom/product/delete-product/%7b%7bproductId%7d%7d)

Http Method: DELETE