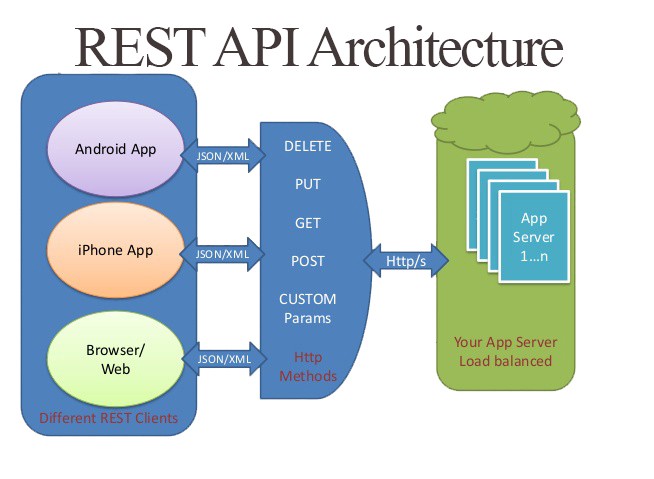
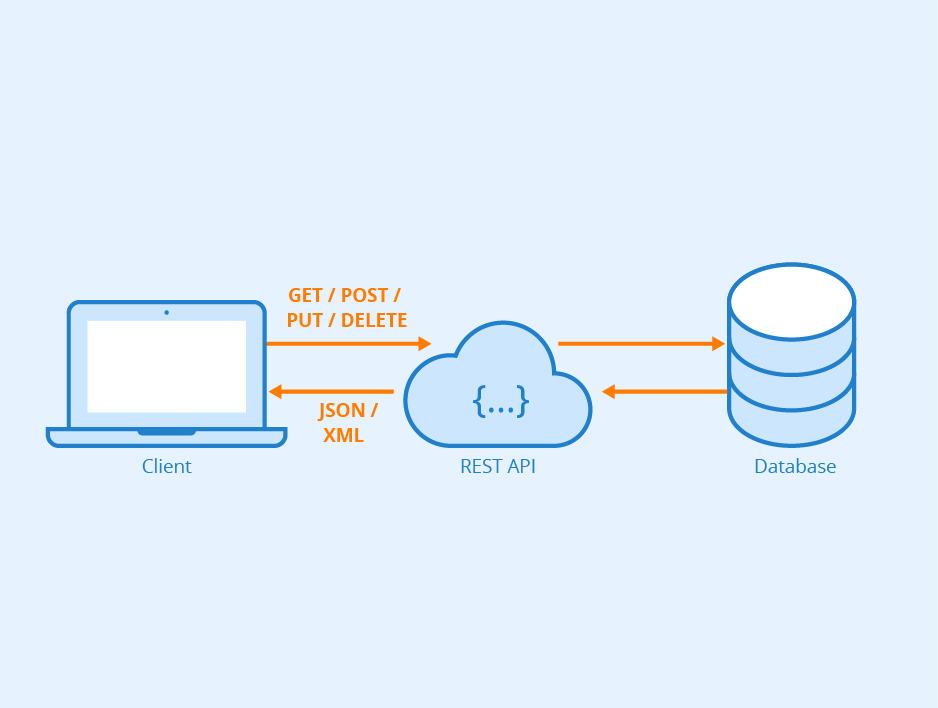
POSTMAN FOR RESTful API TESTING

# Why need for REST API Testing?

Now a days, REST API testing skill is the most wanted expertise. You can see many SQA jobs post with requirement of REST API testing experience. **Mobile and web applications** mostly used **RESTful API** for **reading, creating, updating and deleting data** from database server (e.g. SQL server, Oracle, AWS etc.) via **http request/response methods**.



# What is RESTful API?

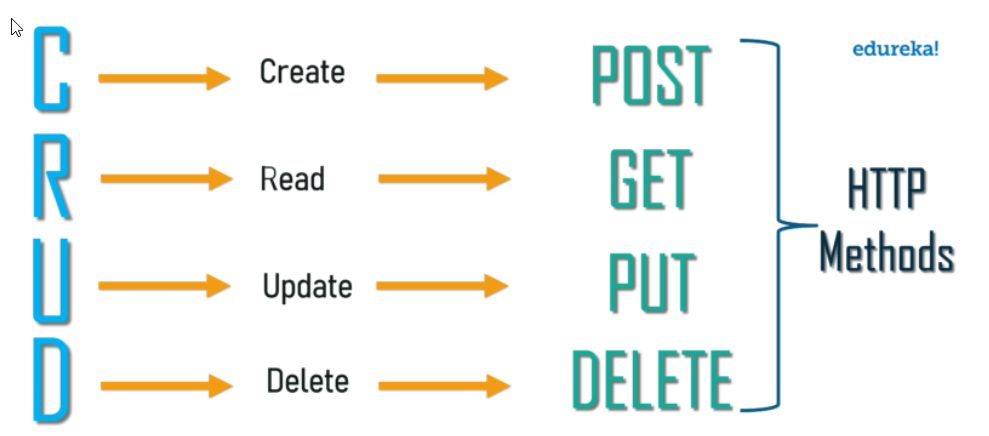


A RESTful API (Application program interface) -- also referred to as a RESTful web service or REST API -- is based on representational state transfer REST technology, an architectural style and approach to communications often used in web service development. REST is based on **Request/Response architecture**, it expose methods(resources) via HTTP URL, **no contract defined between Client & Server**, it is based on e **Stateless Connection**. Means, if server down or restart, it has no impact at client application. For example, If WhatsApp application server down for a while, it has no impact on my WhatsApp application in mobile. I will get my all messages when server up because HTTP **read can be cache** and it is based on **asynchronous call** (email is an example of asynchronous call; phone call is an example of synchronous call).

# What are the HTTP Methods?

There are different types of HTTP request method.

* **GET -** To get resources/data from server.
* **PUT -** To update resources/data at server. E.g. update order details.
* **POST-** To insert into table. e.g. Add new order
* **DELETE -** From removing request e.g. Delete order



# How to install postman at your computer?

Download & install postman via link <https://www.getpostman.com/downloads/>

# How to start testing of your assigned website?

For example, a task assigned you to test the REST APIs of dummy employee information application. You get a list of APIs from development team to test all the implement APIs. For this, first, you must open your browser. Enter URL<http://dummy.restapiexample.com/> . See available details different methods.

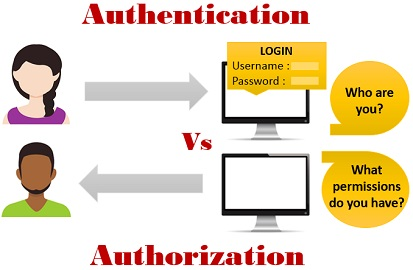
# How to create GET, PUT, POST & DELETE Request in postman for testing?

1. Click New Collection. Save your collection with named ‘DemoAPITesting’.
2. Select authorization. Add basic authorization. Enter username & password.
3. Add variable baseURL. Enter base URL value ‘[http://dummy.restapiexample.com](http://dummy.restapiexample.com/) ‘
4. Add new http request.
5. Enter variable base URL {{baseURL}}.
6. Enter relevant URL /api/v1/employee. Click Send button. User get response in HTTP form with HTTP response 200Ok. 200-OK , shows success

# What are the different API Testing Tools?



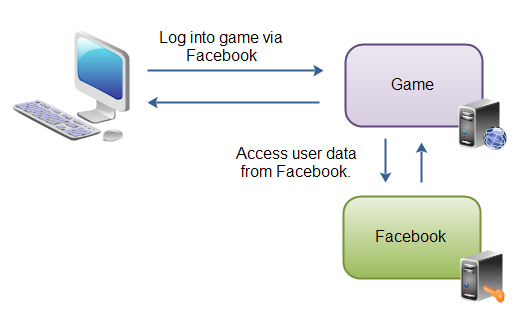
**What is the difference between authentication and authorization?**



Authorization process verifies whether you have permission to access the data you want from the server. When you send a request, you often must include parameters to ensure the request has permission to access and return the data you want. Postman provides authorization types that make it easy for you to handle authentication protocols in Postman native apps. When you select “Authorization” in the request builder, you see the TYPE drop down menu.

Authorization types

* Inherit auth from parent
* **No Auth**
* **Bearer Token**
* **Basic auth**
* Digest Auth
* Hawk Authentication
* AWS Signature
* NTLM Authentication [Beta]
* **OAuth 1.0**
* **OAuth is an open-standard authorization protocol or framework that describes how unrelated servers and services can safely allow authenticated access to their assets without actually sharing the initial, related, single logon credentials. In authentication parlance, this is known as secure, third-party, user-agent, delegated authorization.**

******

* OAuth 2.0

# What are the different types of http response methods?

1. 200-ok , shows success
2. 201-Created , When a resource is successful created using POST or PUT request. Return link to newly created resource using location header.
3. 204-No content, When response body is empty, for example, a DELETE request.
4. 304-Not Modified, If the client has performed a conditional GET request and access is allowed, but the document has not been modified, the server SHOULD respond with this status code.
5. 400-Bad Request , Request you sent to the website server, often something simple like a request to load a web page, was somehow incorrect or corrupted and the server couldn't understand it.
6. 401-FORBIDDEN/UNAUTHORIZED , States that user is not having access to method being used for example, delete access without admin rights.
7. 404-Not Found , States that method is not available.
8. 409-Conflict , States conflict situation while executing the method for example, adding duplicate entry.
9. 500-Internal server error, States that server has thrown some exception while executing the method.

# What is JSON?

1. JSON (JavaScript Object Notation) is a lightweight data-interchange format.
2. It is easy for humans to read and write.
3. It is easy for machines to parse and generate.

# How to add assertion for response status code 200 Ok/404 Not found?

A Postman test is essentially JavaScript code executed after the request is sent, allowing access to the pm.response object.

// example using pm.response.to.have

pm.test("response is 200ok", function () {

    pm.response.to.have.status(200);

});

pm.test("response is Not found", function () {

    pm.response.to.have.status(404);

});

# How to add assertion for response message(string)?

pm.test("Verify Load not Found message", function () {

     pm.expect(pm.response.text()).to.include("Load ML028779 not found.");

})

# How to add assertion for JSON response(single value)?

pm.test("Verify Loadkey value", function()

{

    json\_response = JSON.parse(responseBody);

    x = json\_response[0].LoadKey;

    pm.expect(x).to.equal("ML287791");

});

# How to add assertion for verifying JSON schema?

Go to <https://jsonschema.net/>

Generate schema for your JSON response

var jsonData = JSON.parse(responseBody);

var schema = {

  "definitions": {},

  "$schema": "http://json-schema.org/draft-07/schema#",

  "$id": "http://example.com/root.json",

  "type": "array",

  "title": "The Root Schema",

  "items": {

    "$id": "#/items",

    "type": "object",

    "title": "The Items Schema",

    "required": [

      "LoadKey",

      "Site",

      "LoadStatus",

      "LoadStatusDesc",

        "BayKey",

      "BayKeyFormatted",

        ],

    "properties": {

      "LoadKey": {

        "$id": "#/items/properties/LoadKey",

        "type": "string",

        "title": "The Loadkey Schema",

        "default": "",

        "examples": [

          "ML287791"

        ],

        "pattern": "^(.\*)$"

      },

      "Site": {

        "$id": "#/items/properties/Site",

        "type": "string",

        "title": "The Site Schema",

        "default": "",

        "examples": [

          "ML"

        ]

  }

};

pm.test('Schema is valid', function() {

  pm.expect(tv4.validate(jsonData, schema)).to.be.true;

});

# How to add assertion for validating all data of JSON response?

const responseKey = [pm.info.requestName, 'response'].join('/');

let res = '';

try {

    res = JSON.stringify(pm.response.json());

} catch(e) {

    res = pm.response.text();

}

if (!pm.globals.has(responseKey)) {

    pm.globals.set(responseKey, res);

} else {

    // this tests the actual response with one saved in the global for the same request

    pm.test(responseKey, function () {

        const response = pm.globals.get(responseKey);

        pm.globals.unset(responseKey);

        try {

            const data = pm.response.json();

            pm.expect(JSON.stringify(data)).to.eql(response);

        } catch(e) {

            const data = pm.response.text();

            pm.expect(data).to.eql(response);

        }

    });

}

pm.globals.get(responseKey);

# What are the environmental variables?

Environments are a portable scope for variables. **Use for variables mostly used**.

# What is the Monitor?

User can create a monitor to run a collection periodically to check its **performance and response**. User can set a monitor to run as frequently as every 5 minutes.

# What is API Documentation?

Use for creating API documentation for rest API.