

API Testing using Postman

Table of Contents

What is an API?	3
What is Postman?	3
Why Choose Postman?	3
Postman Features:.....	3
How to Use Postman.....	4
Building Blocks of Postman:	5
Testing GET Requests:.....	8
Testing POST Requests:	11
How to Parameterize Requests:.....	13
Important HTTP response status codes:	15
Types of APIs:	16
API Testing Scenario Categories	16

What is an API?

API stands for Application Programming Interface. It is a collection of routines, protocols, and tools designed for building software applications. Essentially, APIs define how different software programs interact with one another.

In simpler terms, an API serves as a bridge between two software applications, enabling them to communicate. It consists of a set of functions that can be called and executed by other software programs.

What is Postman?

Postman is a collaborative platform for API development that serves as a widely-used API client. It allows users to design, build, share, test, and document APIs effectively.

With Postman, you can send HTTP and HTTPS requests to a service and receive responses, helping to ensure that the service is operational. Originally a Chrome browser extension, Postman now offers native applications for both Mac and Windows.

Why Choose Postman?

Postman has become the preferred tool for over 8 million users due to several key features:

- **Free:** It's free to download and use, accommodating teams of any size.
- **User-Friendly:** You can quickly download it and send your first request within minutes.
- **API Support:** It supports all types of API calls (REST, SOAP, or plain HTTP) and allows you to easily inspect even the largest responses.
- **Customizable:** You can tailor it to fit your specific needs using the Postman API.
- **Integration:** Easily incorporate test suites into your preferred CI/CD service using Newman, the command line collection runner.
- **Community & Support:** Postman boasts a large community forum for support and resources.

Postman Features:

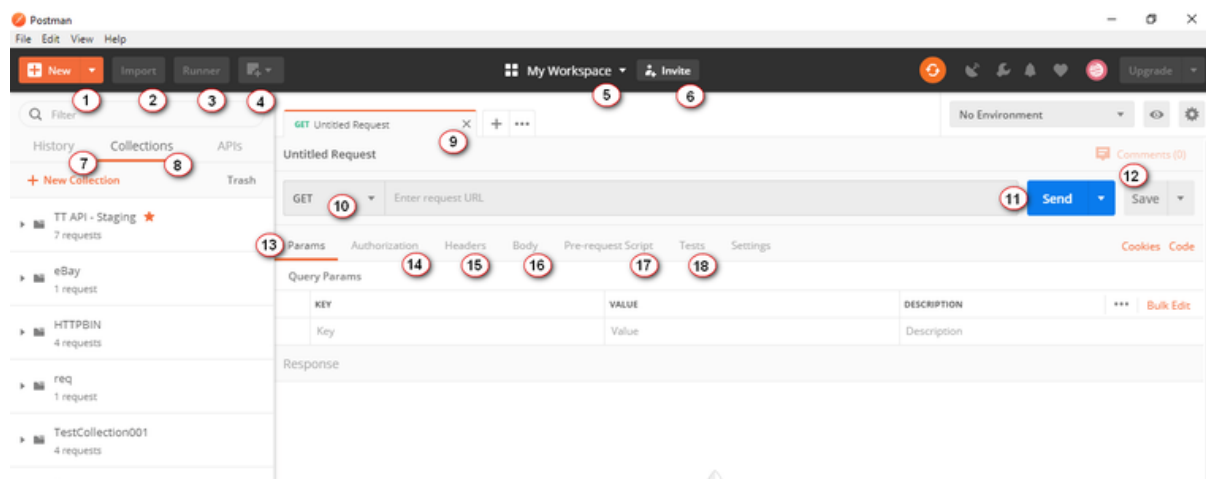
Postman offers a variety of features, including:

- **User-Friendly REST Client:** Easy to navigate and use.
- **Intuitive Interface:** Designed for a smooth user experience.

- **Versatile Testing:** Suitable for both manual and automated API testing.
- **Cross-Platform Compatibility:** Runs on Mac, Windows, Linux, and as a Chrome app.
- **Integration Capabilities:** Supports various formats like Swagger and RAML.
- **Comprehensive Features:** Includes running, testing, documenting, and monitoring functionalities.
- **No New Language Required:** Users can get started without needing to learn a new programming language.
- **Team Collaboration:** Easily share knowledge by packaging requests and expected responses to send to colleagues.
- **CI/CD Integration:** Works seamlessly with tools like Jenkins and TeamCity.
- **Detailed API Documentation:** Comes with extensive documentation to assist users.
- **API Development and Automation:** Facilitates the development of APIs and automates the execution of API tests.

How to Use Postman

Every component of the Postman tool plays a vital role. Let's explore the various options available in the Postman workspace.



1. **New:** Use this option to create a new request, collection, or environment.
2. **Import:** Import a collection or environment from various sources, including files, folders, links, or by pasting raw text.
3. **Runner:** Execute automated tests with the Collection Runner.
4. **Open New:** Open either a new Postman window or the Runner window.

5. **My Workspace:** This is your personal workspace where you can create new ones. A workspace provides a shared context for building and consuming APIs, facilitating real-time collaboration among teams.
6. **Invite:** Invite team members to collaborate within a workspace.
7. **History:** Automatically saves requests and responses, allowing you to easily track your past actions.
8. **Collections:** Organize and manage related requests efficiently.
9. **Request Tab:** Displays the title of the current request, defaulting to 'Untitled Request'.
10. **HTTP Request:** View various request types such as GET, POST, COPY, DELETE, etc.
11. **Request URL:** Specify the link for the API communication, also known as the endpoint.
12. **Save:** Save a new request or update an existing one.
13. **Params:** Enter the key-value parameters required for your request.
14. **Authorization:** Provide authorization details, such as username, password, or bearer token, to secure client requests.
15. **Headers:** Set specific headers, like JSON or JavaScript, that some APIs require for additional metadata about your request.
16. **Body:** Specify the data to be sent with a request, supporting various formats to match your API's requirements.
17. **Pre-request Script:** Write JavaScript code that runs before the request is sent, ideal for tasks like adding a timestamp or generating a random string for URL parameters.
18. **Tests:** Execute scripts during the request to verify that your API functions correctly, ensuring reliable integrations and confirming that new changes haven't disrupted existing functionality.

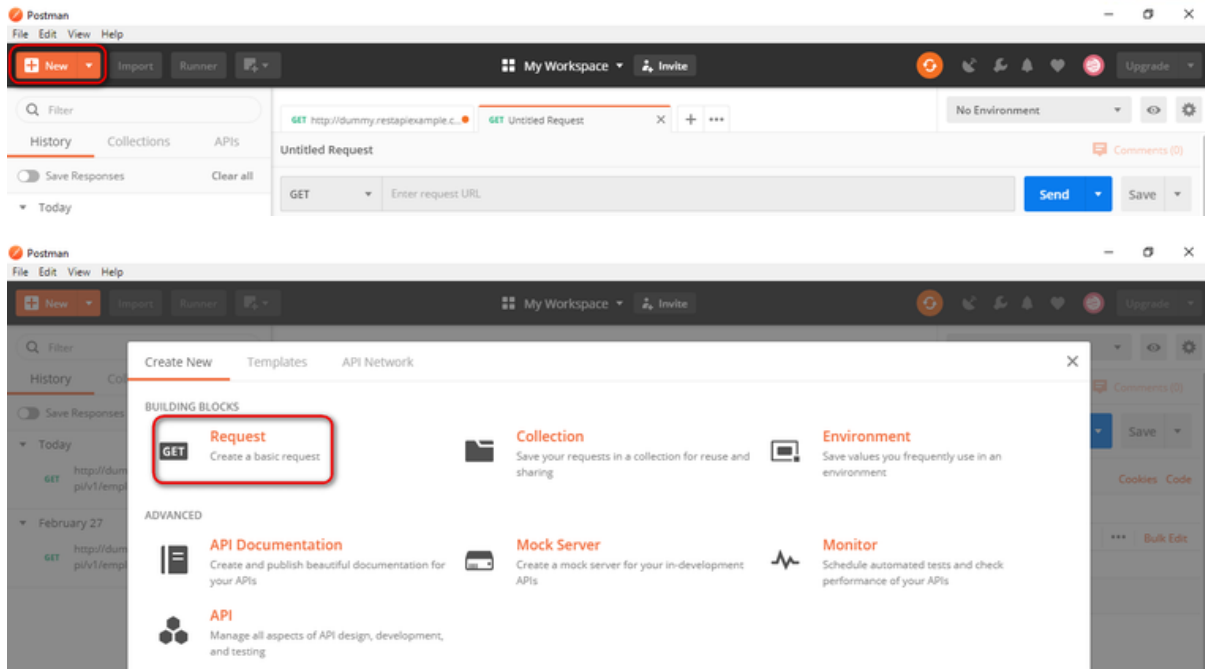
Building Blocks of Postman:

Before testing an API, it's important to understand some fundamental components of the Postman tool that are essential for all operations:

1. **Requests**
2. **Collections**
3. **Environments**

1. Requests:

A request consists of the URL, HTTP headers, and the body or payload. In Postman, you can save your requests for future use, allowing you to access them as needed. To create a new request, click on **New** and then select **Request**.



SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
[Learn more about creating collections](#)

Request name

Request description (Optional)

Make things easier for your teammates with a complete request description.

Descriptions support **Markdown**

Select a collection or folder to save to:

Cancel

Save

You can make API requests in Postman, which allow you to retrieve or send data to a data source. APIs operate on web servers and expose endpoints to facilitate the operations that client applications need to function.

Each API request utilizes an HTTP method.

What is HTTP?

HTTP stands for Hypertext Transfer Protocol. It enables communication between clients and servers. Typically, clients are web browsers, while servers are computers hosted in the cloud.

When a client submits an HTTP request to the server, the server responds with a reply that includes status information about the request and the requested content.

The most commonly used HTTP methods are:

1. **GET:** Used to retrieve data from an API.
2. **POST:** Used to send new data to an API.
3. **PUT:** Used to update existing data.
4. **PATCH:** Used to make partial updates to existing data.
5. **DELETE:** Used to remove existing data.

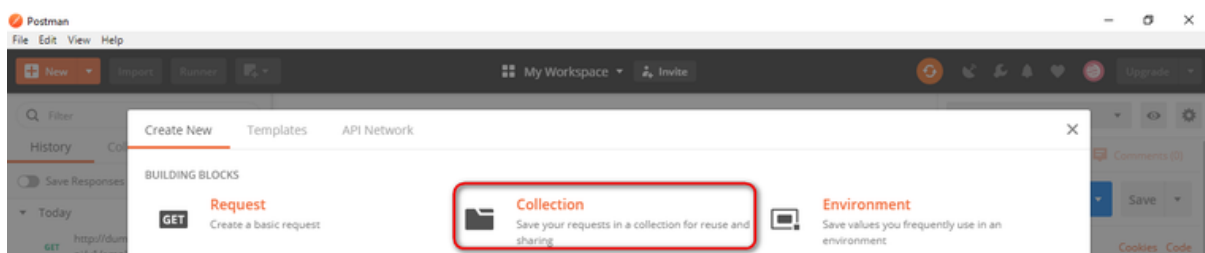
Now, let's explore how to create a simple request using the Postman application and examine the various components of the request and its response.

2. Collections:

Collections are groups of saved requests that you can organize into folders, effectively serving as a repository for your requests.

How to Create Collections in Postman:

To create a collection, click on **New** and then select **Collection**.



Input the Collection Name and description then click Create to create a new collection. You can add any number of requests in a Collection.

CREATE A NEW COLLECTION

Name

Collection Name

Description

Authorization

Pre-request Scripts

Tests

Variables

This description will show in your collection's documentation, along with the descriptions of its folders and requests.

Make things easier for your teammates with a complete request description.

Descriptions support **Markdown**

Cancel

Create

3. Environments:

Environments in Postman enable you to run requests and collections against various data sets. You can set up different environments for Development, Quality Assurance (QA), and Production, each with unique configurations such as URLs, token IDs, passwords, and API keys.

Environments consist of key-value pairs of variables, where each variable name acts as its key. By referencing a variable name, you can access its corresponding value.

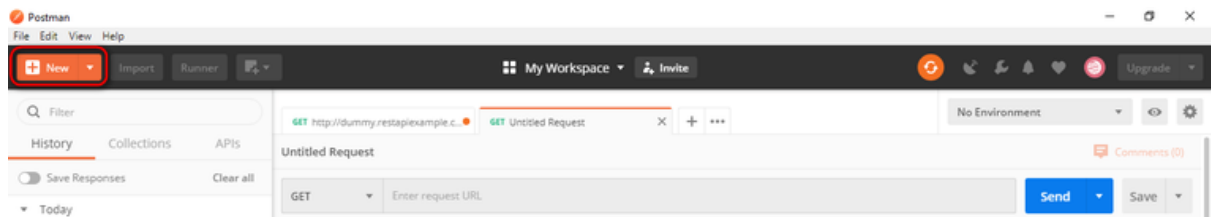
Testing GET Requests:

To retrieve information, use a GET request with the specified URL. GET requests do not affect the endpoint.

1. **Open the Postman Application:** It's recommended to sign in to save your requests, collections, and other actions for future use. If you're not logged in, use your credentials to log in.
2. **Postman UI:** You'll see the initial screen of Postman.

Step 1: Click on the **New** tab to create a new request.

Page - 8

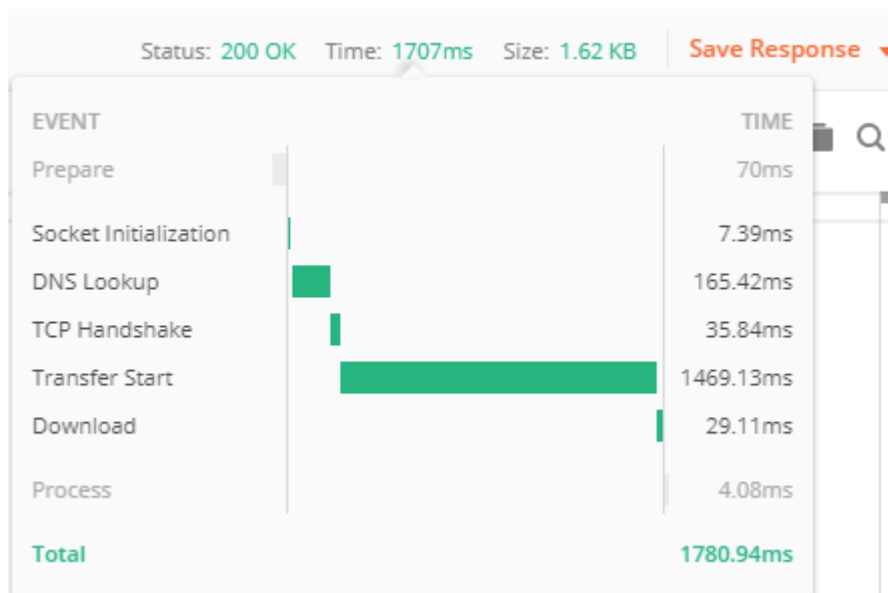


Step 2: Creating a GET Request for a REST API Endpoint

1. **Set the HTTP Request to GET.**
2. **Enter the Request URL:** Input the link: <https://jsonplaceholder.typicode.com/users>.
3. **Click SEND:** This will execute the request to the server hosting the endpoint.
4. **Check for a 200 OK Message:** This indicates that your request was successful. If it fails, it may be due to an invalid URL or incorrect authentication.

After the server responds, you'll see various data in the Body section.

- In the request headers, you can find the response status code, the time taken to complete the request, and the size of the payload.
- Hover over these details to see more information about response time and size, including components like Connect time, Socket time, and DNS lookup.



- **Response size:** We can see individual components like actual response size, how much size the headers are constituted etc.,

Status: 200 OK Time: 1707ms Size: 1.62 KB Save Response ▼

Response Size 1.62 KB

Body 595 B

Headers 1.04 KB

Request Size 252 B

Body 0 B

Headers 252 B

All size calculations are approximate

- **Cookies:** We can find session related information in the cookies that were returned from the server.

GET https://jsonplaceholder.typicode.com/users Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

▼ Headers (1)

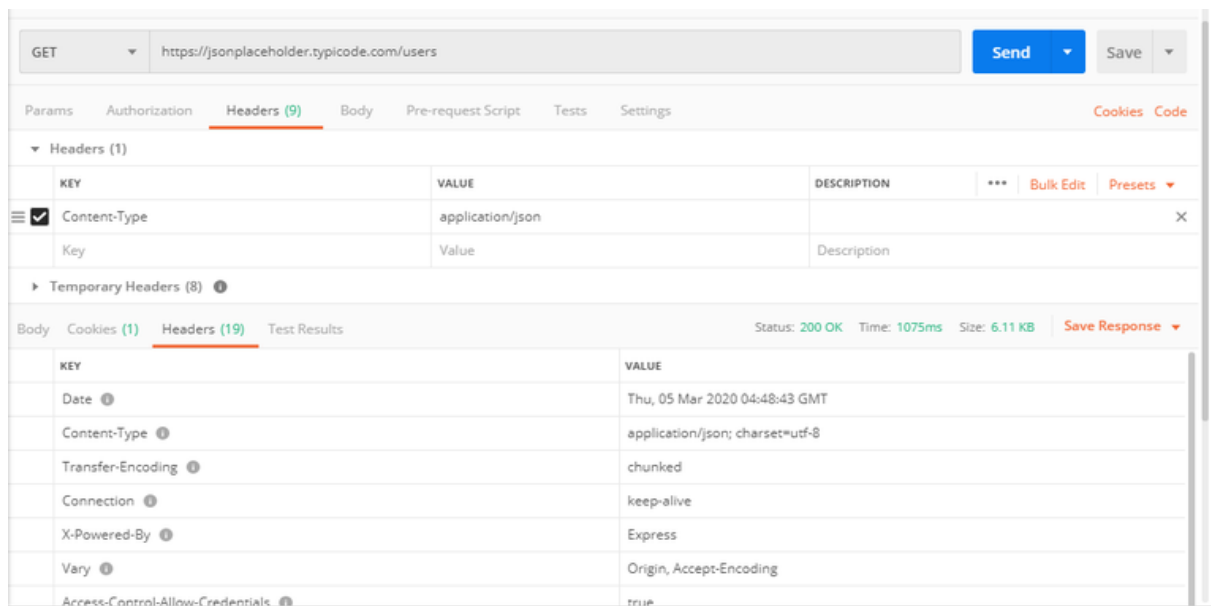
KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets ▼
<input checked="" type="checkbox"/> Content-Type	application/json				
Key	Value	Description			

► Temporary Headers (8) ⓘ

Body Cookies (1) Headers (19) Test Results Status: 200 OK Time: 1075ms Size: 6.11 KB Save Response ▼

Name	Value	Domain	Path	Expires	HttpOnly	Secure
__cfduid	d27e2f460f649a7000038d0196eaf1e161583369076	typicode.com	/	Sat, 04 Apr 2020 00:44:36 GMT	true	false

- **Response header:** Here we can find information about the request that got processed.



GET <https://jsonplaceholder.typicode.com/users> Send Save

Params Authorization **Headers (9)** Body Pre-request Script Tests Settings Cookies Code

▼ Headers (1)

KEY	VALUE	DESCRIPTION
Content-Type	application/json	

▶ Temporary Headers (8)

Body Cookies (1) **Headers (19)** Test Results Status: 200 OK Time: 1075ms Size: 6.11 KB Save Response

KEY	VALUE
Date	Thu, 05 Mar 2020 04:48:43 GMT
Content-Type	application/json; charset=utf-8
Transfer-Encoding	chunked
Connection	keep-alive
X-Powered-By	Express
Vary	Origin, Accept-Encoding
Access-Control-Allow-Credentials	true

Testing POST Requests:

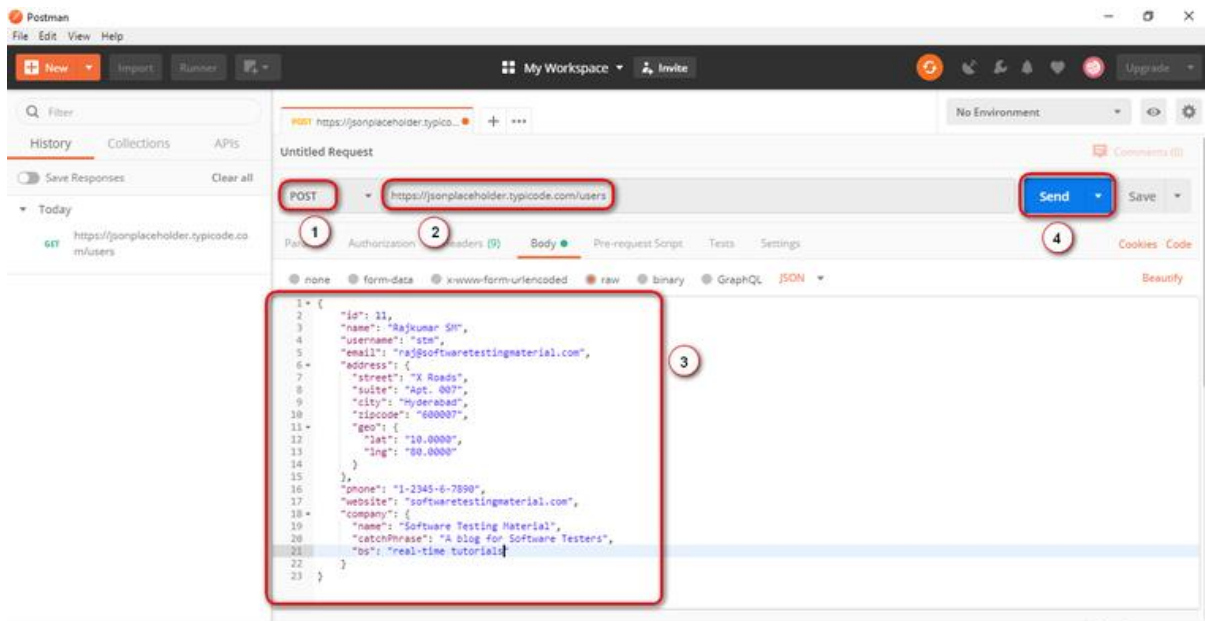
POST requests are used to manipulate data by adding new information to the endpoint. Let's add a user to the application. To do this, we need to send data using a POST request, which includes the data in the body of the request. The API will return a response confirming that the user has been created.

1. **Set the HTTP Request to POST.**
2. **Enter the Request URL:** Use this link: <https://jsonplaceholder.typicode.com/users>.
3. **Click on the Body Tab:** Select the "Raw" radio button and then choose **JSON**.
4. **Add User Data:** Copy and paste one user result from your previous GET request into the body as shown in the screenshot.

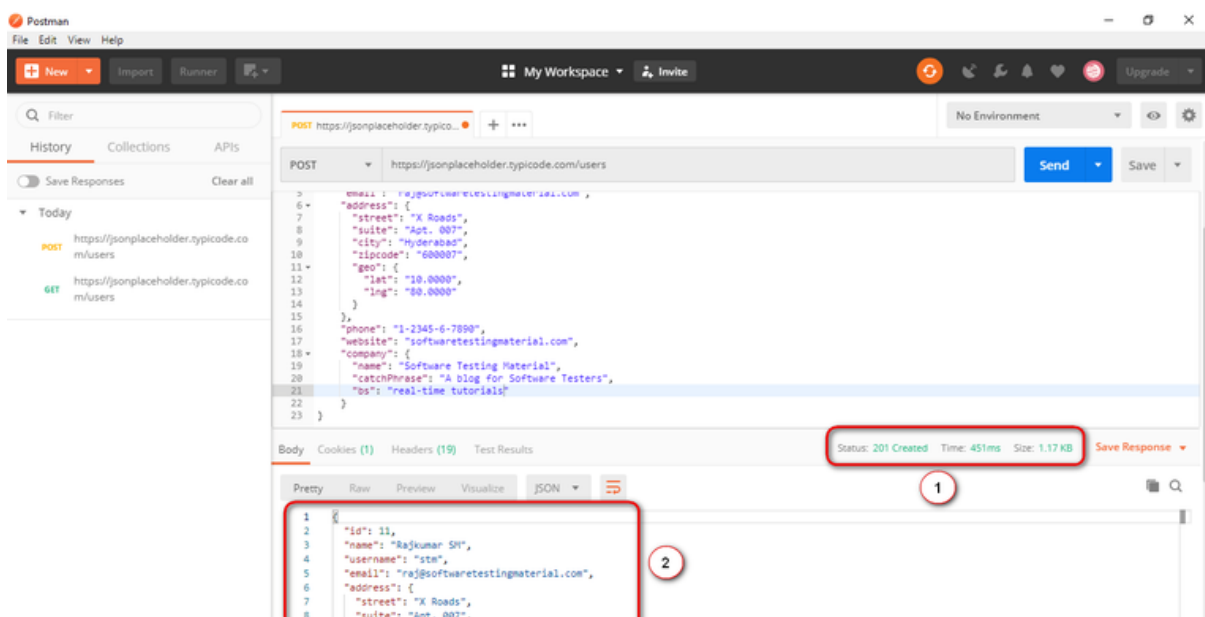
```
{
  "id": 11,
  "name": "Rajkumar SM",
  "username": "stm",
  "email": "raj@softwaretestingmaterial.com",
  "address": {
    "street": "X Roads",
    "suite": "Apt. 007",
    "city": "Hyderabad",
    "zipcode": "600007",
    "geo": {
      "lat": "10.0000",
      "lng": "80.0000"
    }
  },
  "phone": "1-2345-6-7890",
}
```

```
"website": "softwaretestingmaterial.com",
"company": {
  "name": "Software Testing Material",
  "catchPhrase": "A blog for Software Testers",
  "bs": "real-time tutorials"
}
```

1. Click on **SEND** to execute the request to the server hosting the endpoint



1. You can see **201 Created message** in the screenshot below because our request is successful.
2. You can see the posted data in the body.



Likewise, we will test other requests PUT, PATCH & DELETE

Note: For every request, you need to check expected result, status code, response time. Also don't forgot to do negative tests to verify whether the API is responding properly or not

How to Parameterize Requests:

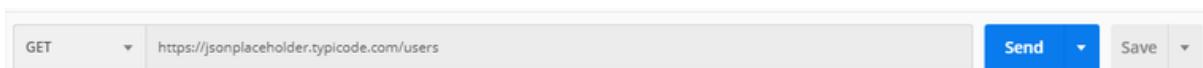
Parameterization in Postman is a valuable feature that allows you to reuse the same request with different data. You can store this data in environment variables or data files.

To create parameters in Postman, use double curly brackets, like `{{test}}`.

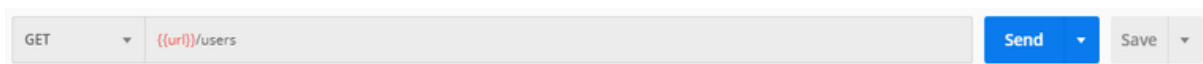
For example, if you have a base URL such as `https://stm.org` stored in a variable named `base_url`, you can reference it in your requests as `{{base_url}}`.

To send a request to this base URL to get a list of new customers, you would format the request URL like this: `{{base_url}}/get?customers=new`. Postman will then send the request to `https://stm.org/get?customers=new`.

1. **Set the HTTP Request to GET** and enter the URL.

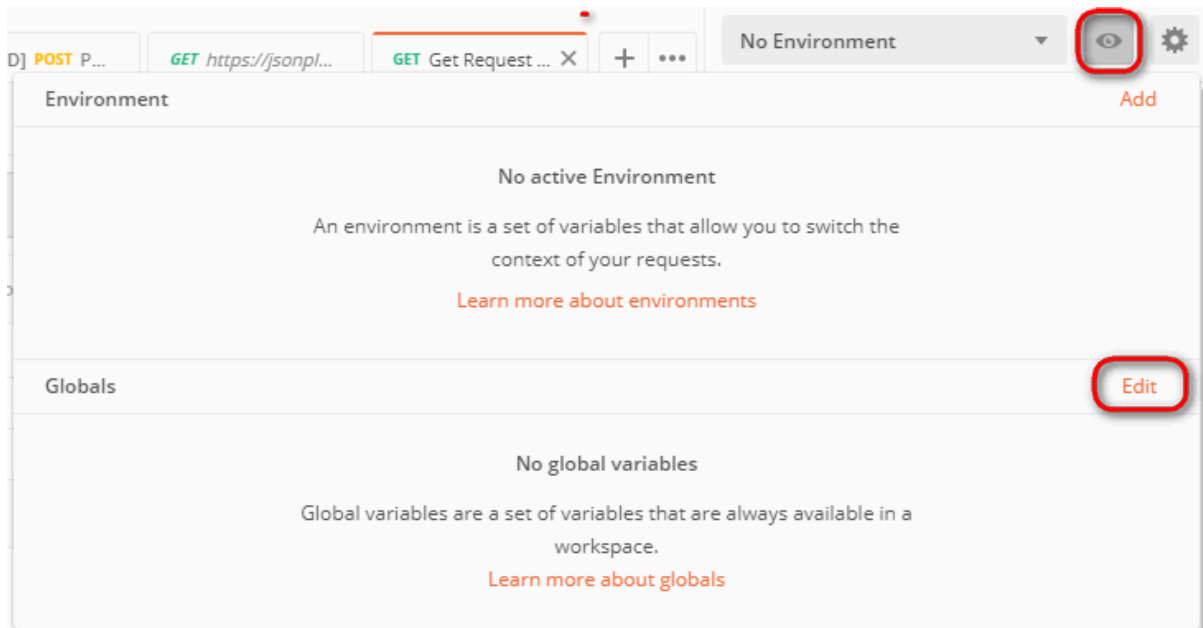


2. Replace the URL with a parameter such as `{{url}}`. Request URL should be `{{url}}/users`.



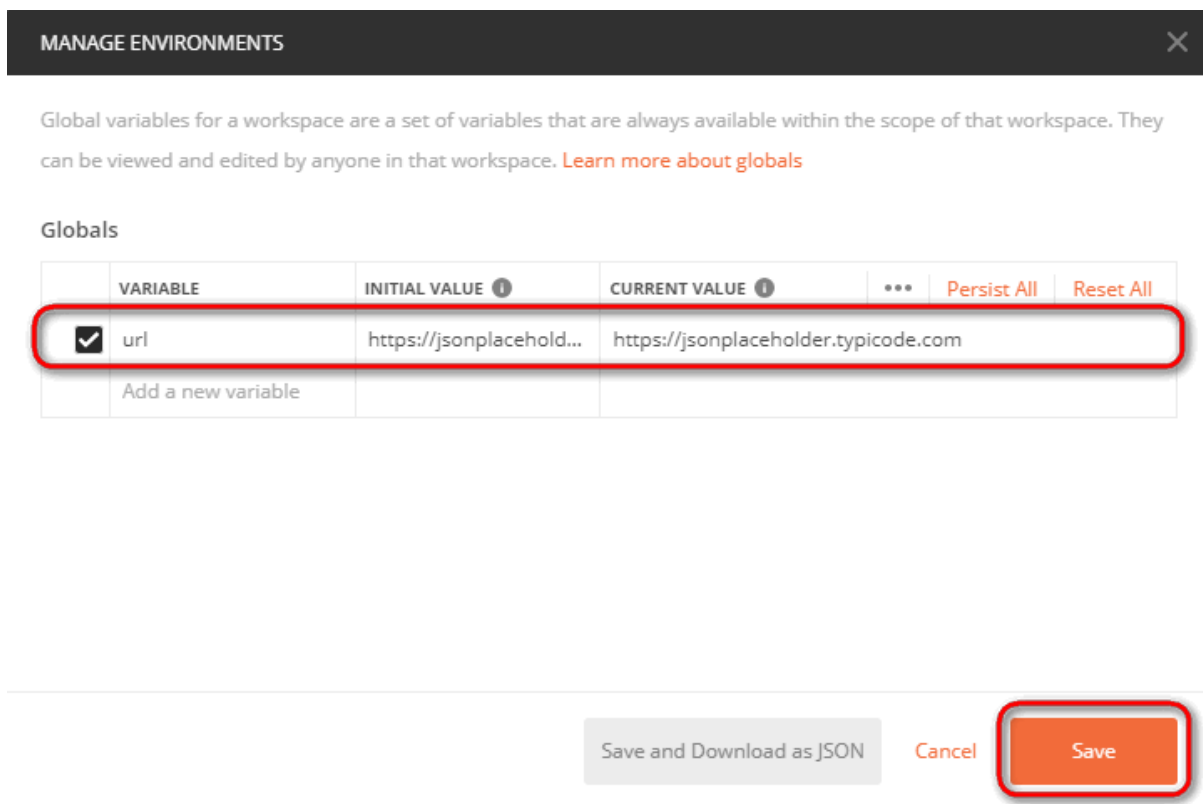
3. **Set Environment Variables for Parameters:**

To do this, click on the eye icon in the top right corner and then select **Edit**. From there, you can create a global environment variable that will be accessible across all collections.



4. Set the Variable:

Name the variable url and set its value to `https://jsonplaceholder.typicode.com`. Then click **Save**.



5. Go back to GET request and click send.

Get Request Parameterize

GET `{{url}}/users` Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (19) Test Results Status: 200 OK Time: 1285ms Size: 6.11 KB Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 1,
3   "name": "Leanne Graham",
4   "username": "Bret",
5   "email": "Sincere@april.biz",
6   "address": {
7     "street": "Kulas Light",
8     "suite": "Apt. 556",
9     "city": "Gwenborough",
10    "zipcode": "92998-3874",
11    "geo": {
12      "lat": "-37.3159",
13      "lng": "81.1496"
14    }
15  }
16 }
```

Important HTTP response status codes:

Status Code	Description
200	OK – The request was successful.
201	Created – The resource was created.
204	No Content – The request was successful but there is no content to return.
400	Bad Request – The server could not understand the request due to invalid syntax.
401	Unauthorized – Authentication is required and has failed or not been provided.
403	Forbidden – The server understood the request, but refuses to authorize it.
404	Not Found – The requested resource could not be found.
500	Internal Server Error – The server encountered an unexpected condition.
503	Service Unavailable – The server is not ready to handle the request.

Types of APIs:

1. **Private APIs:** These APIs are designed for internal use within an organization. They serve as in-house applications that help employees automate business processes and improve delivery.
2. **Public/Partner APIs:** These APIs are openly promoted and accessible to known developers or business partners. They typically facilitate software integrations between different organizations.
3. **External APIs:** As the name suggests, these APIs are available to any third-party developer. They are primarily designed for end-users or customers.

API Testing Scenario Categories

1. **Basic Positive Tests (Happy Paths):**
 - Test scenarios that verify the API works as expected with valid inputs and parameters.
2. **Extended Positive**
3. **Testing with Optional Parameters:**
 - Validate how the API handles additional optional parameters while still returning the correct response.
4. **Destructive Testing:**
 - Assess the API's robustness by testing how it behaves under extreme conditions, such as high load or invalid data formats.
5. **Security, Authorization, and Permission Tests:**
 - (Out of scope for this post) Test for security vulnerabilities, ensuring proper authorization and permission settings.
6. **Negative Testing with Valid Input:**
 - Evaluate how the API responds to valid inputs that should produce errors under certain conditions (e.g., missing required fields).
7. **Negative Testing with Invalid Input:**
 - Test the API's response to completely invalid inputs, ensuring it handles errors gracefully and provides appropriate error messages.