Hands on Workshop

On PostMan

At first, I want to Congratulate You Guys

You are selected 😍

https://www.pexels.com/photo/man-facing-road-1248418

ToTo Company

WHAT IS THIS?

ToTo Company (API Provider)

Public

Private

Partnership

API Types

Public

Accessible by anyone

API Key

Private / Internal

Used within the same org.

Access is restricted (OAuth, OpenID Connect)

Communication between different services

Partner

Access by different organizations.

Access is restricted

Agreements between creator and consumer

BUT!

It's Not Working!

Developer
think I fixed it guys
production
bug fix

Focus on The Following Problems

- **Documentation Deficiency:** Their APIs lack proper documentation. For instance, details on how their partners should call API endpoints from different clients, the necessary payload, headers required for sending requests, etc., are missing.
- **Validation and Testing Gaps:** There are no solutions in place to validate or test their API responses.
- **Collaboration Challenges:** There is a problem with collaboration and easily sharing APIs across the development team.
- **Environment Mix-up:** They cannot easily manage their APIs in different environments (e.g., development, testing, production).

- **Performance Analysis Dilemma:** They lack an easy solution to analyze and identify performance issues, errors, or any unexpected behavior in their APIs, hampering improvements.
- **Frontend-Backend Coordination Issues:** They regularly observe that their frontend teams sometimes wait for the backend team to complete APIs, causing delays in completing projects on time.
- **Unclear authorization process:** Toto Company uses OAuth2 and OpenID Connect to secure their APIs. However, their backend team often receives questions from the frontend team, such as "Frontend team is unable to authorize and obtain a token from the authorization server." This is common for the frontend team because the backend team cannot demonstrate simple steps to work with their Auth server.

❑ **Data-Driven Testing Hurdles:** They want to perform Data-Driven testing.

❑ **Automated Testing Goals:** They also aim to automate their API testing process, generating reports automatically.

❑ **API Data Flows:** Their backend team desires a system where they can directly manipulate APIs and observe the data flowing between them without creating a new application. This helps them create a mind map of their API functionality.

HOW DO WE FIX THIS?
SNL

HERO

Rasel Hossain

Software Developer at Stack Learner

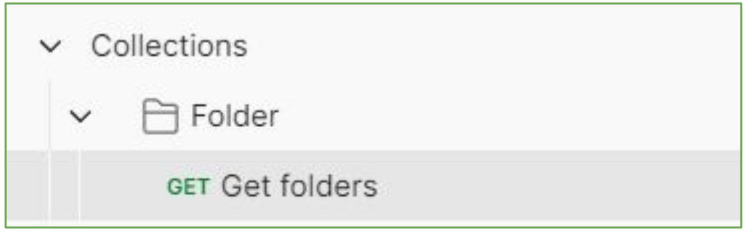
What is this PostMan?

Postman is a popular API client that allows you to *make requests* to APIs, test API endpoints, and *automate API testing*. It provides a user-friendly interface for constructing and sending HTTP requests, as well as for inspecting the responses. Postman also offers features for *collaboration*, *documenting* APIs, and *monitoring API, etc*

Let's explore the Postman.

Collection

In Postman, a collection is a grouping or container for a set of related API requests. It allows you to organize, structure, and manage your API requests in a *logical manner*. Collections can include multiple requests, scripts, and even folders to help you streamline and categorize your API development and testing workflows



Let's Create Auth And Inventory Collection

Variables

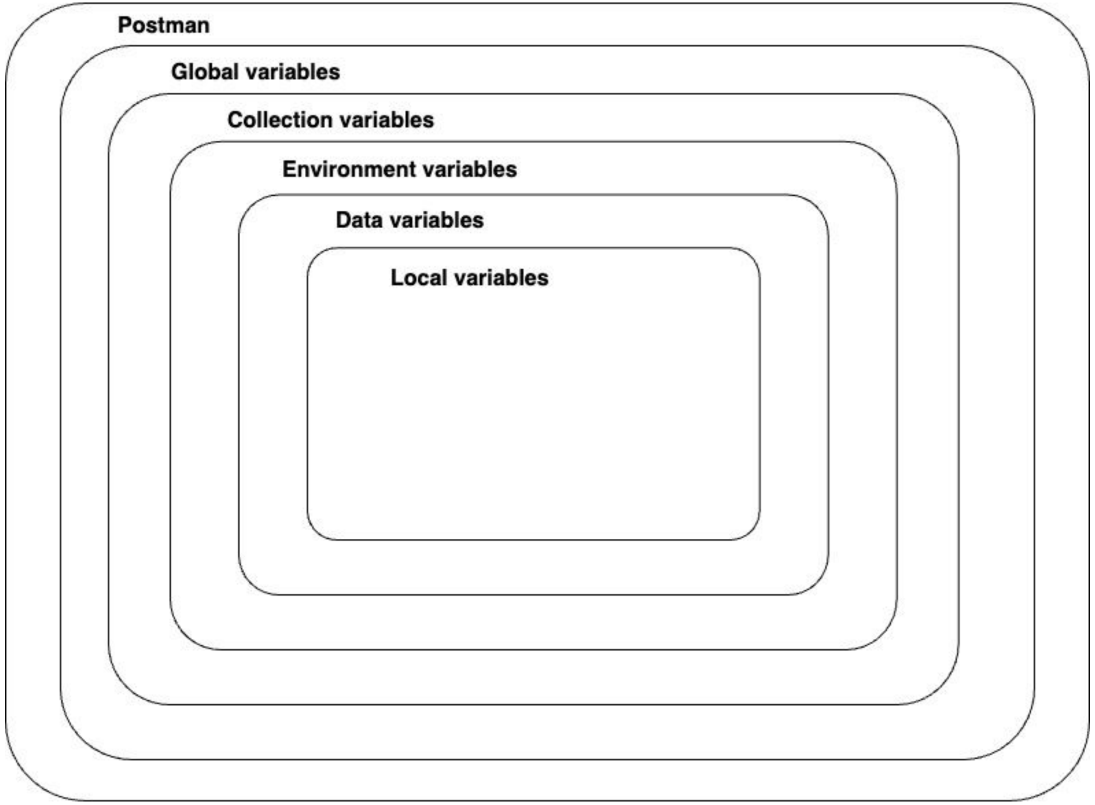
In Postman, variables are placeholders or containers for values that can be used throughout your API testing workflows. They are used to make your requests and scripts dynamic, allowing you to reuse values, handle dynamic data, and create more flexible and maintainable workflows.

In postman there are 5 types of variables available

1. Global variables
2. Collection Variables
3. Environment Variables
4. Data Variables
5. Local Variables

Variables

1. **Global variables** enable you to access data between collections, requests, and test scripts. Global variables are available throughout a workspace. Global variables are a helpful tool for sharing variables across the workspace.
2. **Collection variables** are available throughout the requests in a collection. Collection variables are useful for keeping variables private within a collection of APIs and don't want to share variables outside the collection.
3. **Environment variables** enable you to scope your work to different environments, for example, Development, Testing, Production. One environment can be active at a time.
4. **Data Variables** in your data come from external CSV and JSON files. You use these sets of data when running collections with Newman or the Collection Runner. These variables have current values that only last during the request or collection runs.
5. **Local variables** are temporary variables that are accessed in your request scripts. Local variable values are scoped to a single request or collection run, and are no longer available when the run is complete. Local variables are suitable if you need a value to override all other variable scopes but don't want the value to persist once execution has ended.



https://learning.postman.com

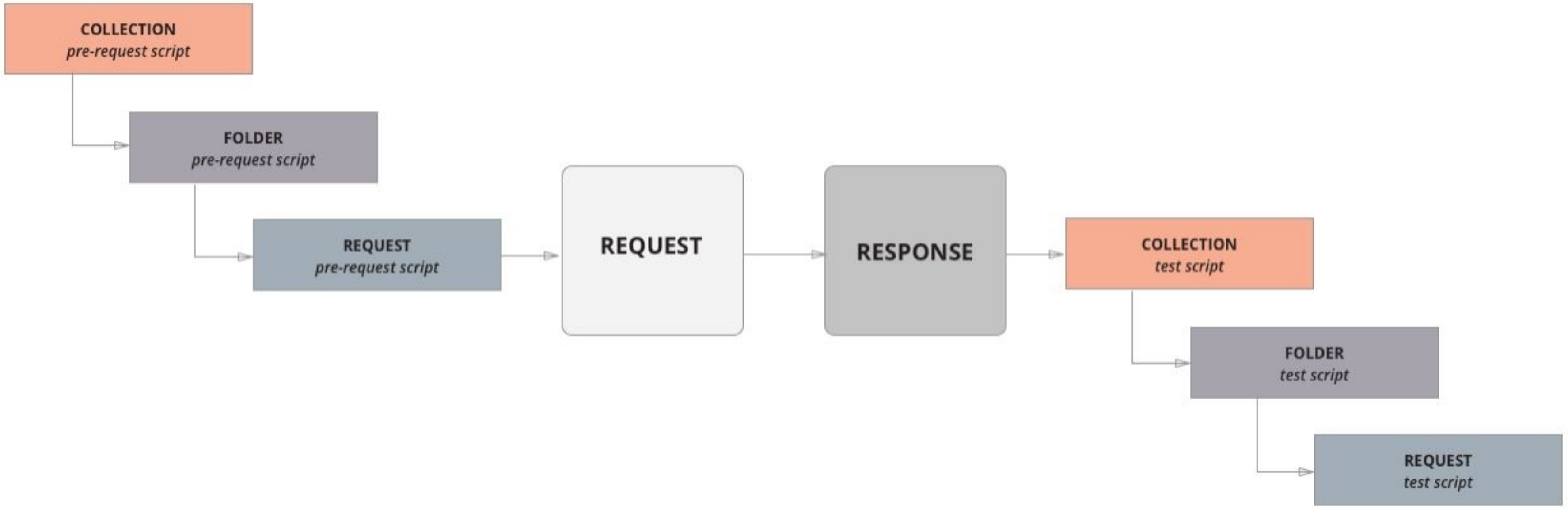
Let's Explore Postman Variables

Scripts

In Postman, scripts are used to automate and customize API requests and tests. They allow you to write code that will be executed before or after a request is sent, giving you the ability to manipulate data, set up test conditions, or perform other tasks.

In Postman there are two types of scripts available.

1. **Pre-scripts:** This script is run before sending the request
2. **Test scripts:** This script is run after sending the request

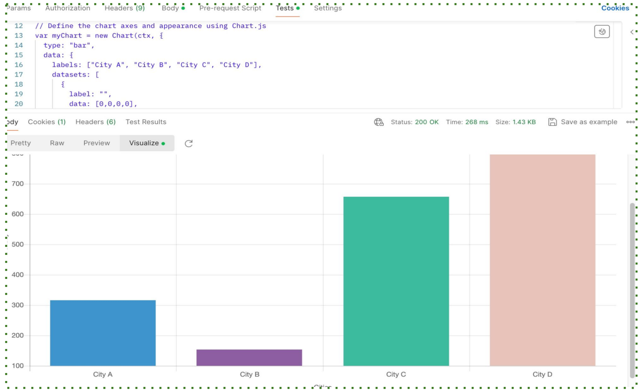


https://learning.postman.com

Let’s make our hand dirty

Visualizer

The Visualizer enables you to present your response data in ways that help to make sense of it. You can use this to model and highlight the information that's relevant to your project, instead of having to read through raw response data. When you share a Postman Collection, other people on your team can also understand your visualizations within the context of each request. **#Handlebars** template



```
// Define the chart axes and appearance using Chart.js
var myChart = new Chart(ctx, {
  type: "bar",
  data: {
    labels: ["City A", "City B", "City C", "City D"],
    datasets: [
      {
        label: "",
        data: [0,0,0,0],
```

Today we have solved

- ❏ **Validation and Testing Gaps:** There are no solutions in place to validate or test their API responses.
- ❏ **Environment Mix-up:** They cannot easily manage their APIs in different environments (e.g., development, testing, production).

Thank You