Postman

Prattya Datta

Tasks of the Pill

Task	Priority	Description	Difficulty	Time
Read the description of the project	5	This task involves complete understanding of requirements for the project	1	30 min
Create git repo	5	Creating of git repository for project execution and delivery	1	2 mins
Create files for the project	4	Creating the necessary files, folders and subfolders for this project. This is important especially for this project of PhP unit	1	5 mins
Download POSTMAN	3	Downloading the POSTMAN which is required for this project	2	10-20 mins
Install POSTMAN	4	Installing the POSTMAN	3	30 mins
Create all the requests	2	Create all the requests as required by the project	4	15 mins
Run the tests	2	Run the tests	5	30 mins
Documentation	1	Write the documentation and push it to github	2	1 hr

Knowledge Learned

- Request: It is used for sending requests like GET, POST, DELETE, PUT etc. When you send a request, Postman will display the response received from the API server in a way that lets you examine, visualize, and if necessary troubleshoot it.

- Collection:It lets us create collections of requests in postman. All the requests could be organized into the folder. The collections could be created and shared to others. To create collection: In the CREATE A NEW COLLECTION modal:
  - Enter a name and optional description.
  - Select an authorization type.
  - Enter a pre-request script to execute before the collection runs.
  - Add a test to execute after the collection runs.
  - Add variables to the collection and its requests.

Knowledge Learned

- Environment:Environments enables you to create robust requests that you can reuse. For more information about using variables and environments. You also can use environments in the Collection Runner.

- API documentation:You can automatically generate documentation for your Postman APIs. You can share your documentation privately or publish it on the web. Postman generates and hosts documentation based on collections, synced in realtime and accessible via the browser. You can use documentation to collaborate with team members and partners, or to support developer adoption for your public APIs.

Knowledge Learned

- Mock server:Delays on the front-end or back-end make it difficult for dependent teams to complete their work efficiently. Postman's mock servers can alleviate those delays in the development process.

  Before sending an actual request, front-end developers can create a mock server to simulate each endpoint and its corresponding response in a Postman Collection. Developers can view potential responses, without spinning up a back end. Two types of mock servers:

  - Private
  - Public

Knowledge Learned

- Monitor: Monitoring API's and websites. The types of monitoring are:
  - Monitoring specific end-point
  - Monitoring an entire API
  - Running an API suite
  - Monitoring HTTP response codes
- API: It is what is requested from the postman. It could be designed by the user itself or third-party APIs listed in various domains

Difficulties during the PILL

None

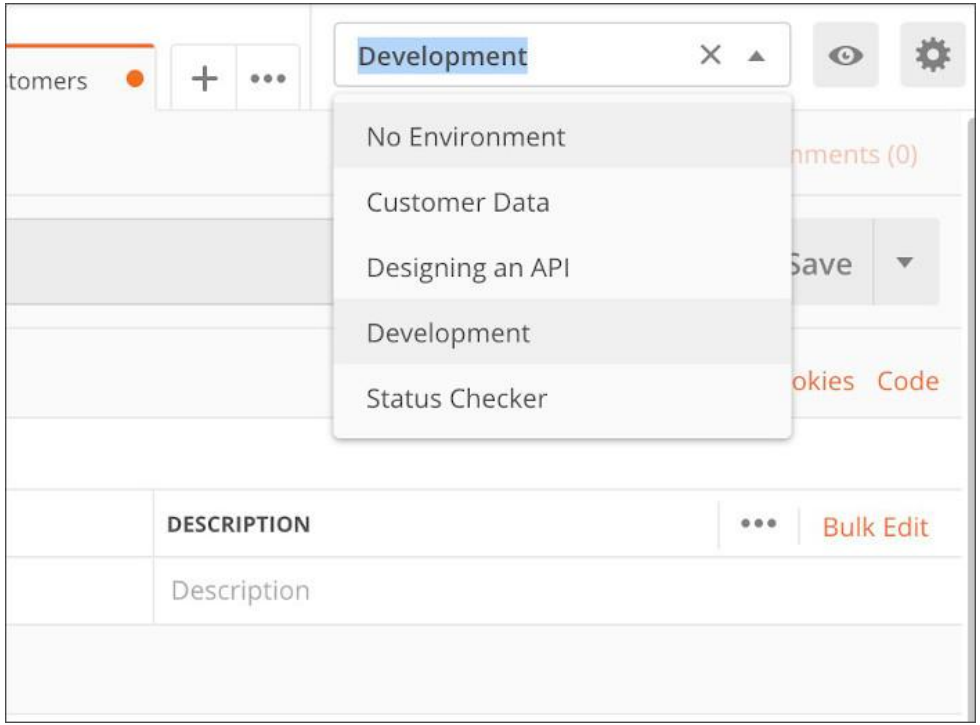
Environment variables and Scopes

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 have to change it in one place. Using variables increases your ability to work efficiently and minimizes the likelihood of error.

Environments allow you to run requests and collections against different data sets. For example, you could have an environment for development, one for testing, and another for production. You can use variables to pass data between requests and tests, for example if you are chaining requests using a collection

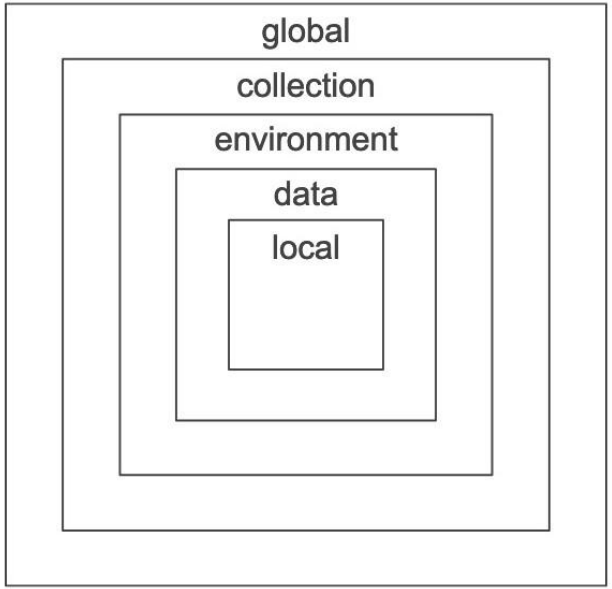
Environment variables and Scopes

To create environments



Environment variables and Scopes

Postman supports the following variable scopes:



- **Global**: allow to access data between requests and collections
- **Collection**: are available throughout the requests in a collection and are independent of environments, so do not change based on the selected environment.
- **Environment:** allow you to tailor your processing to different environments, for example local development vs testing or production. Only one environment can be active at a time.
- **Data:**come from external CSV and JSON files to define data sets you can use when running collections via Newman or the Collection Runner.
- **Local:**are temporary, and only accessible in your request scripts.

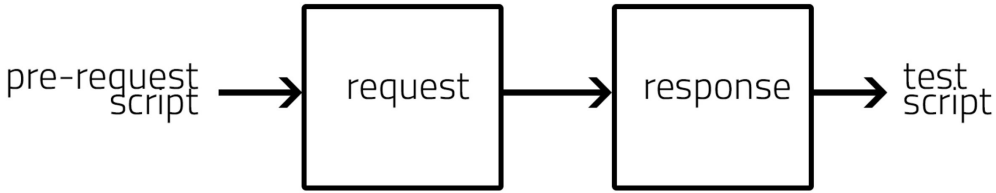
Tests

Test scripts are run after a request is sent and a response has been received from the server.

Setting a nested object as an environment variable:

```
var array = [1, 2, 3, 4];

pm.environment.set("array", JSON.stringify(array, null, 2));



var obj = { a: [1, 2, 3, 4], b: { c: 'val' } };

pm.environment.set("obj", JSON.stringify(obj));
```

Tests



Tests are scripts written in JavaScript that are executed after a response is received. Tests can be run as part of a single request or run with a collection of requests.

In the Postman app, the request builder at the top contains the **Tests** tab where you write your tests. The response viewer at the bottom contains a corresponding **Test Results** tab where you can view the results of your tests.