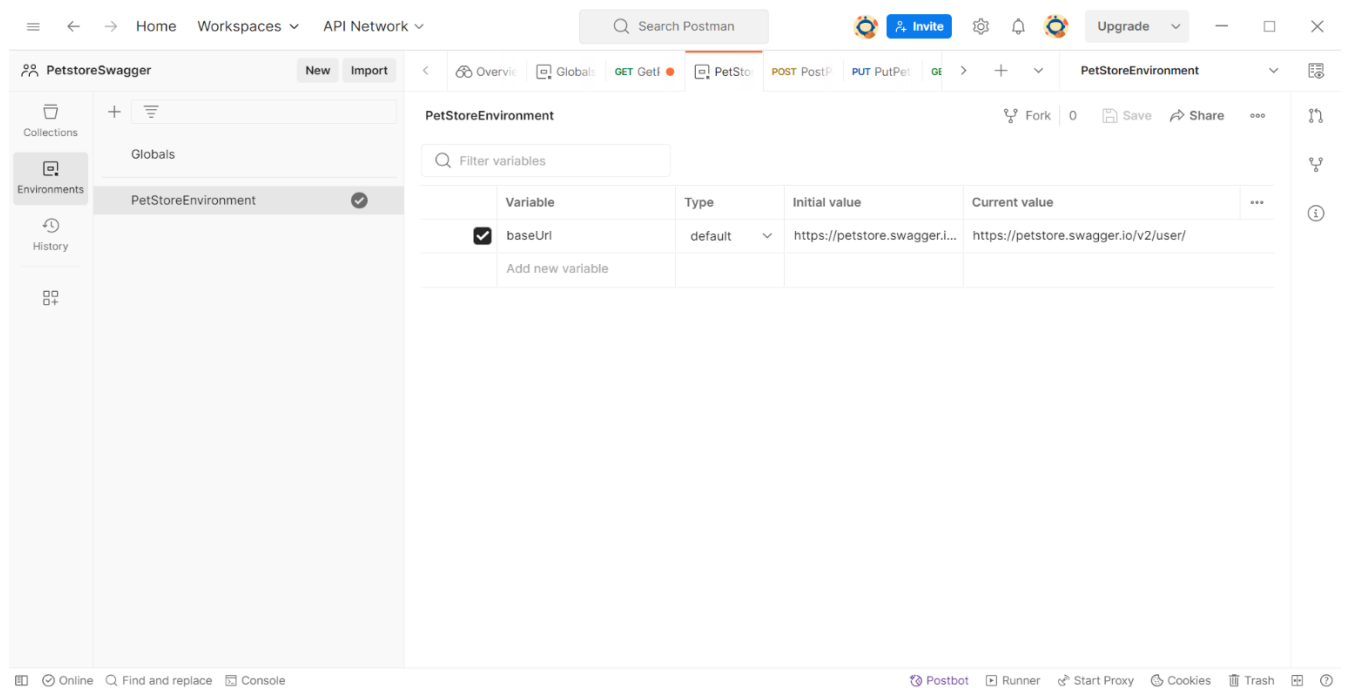


# PetStore

## Postman Screenshots

API: <https://petstore.swagger.io/#/user>

### 1. Creating a Environment



## 2. Creating a User – POST

The screenshot shows the Postman interface with a POST request configured for the endpoint `PostPetStoreUser`. The request body is a JSON object representing a user:

```
1 {
2   "id": "1001",
3   "username": "rosemary",
4   "firstName": "Rose",
5   "lastName": "Mary",
6   "email": "rose@gm.in",
7   "password": "12345",
8   "phone": "9876543210",
9   "userStatus": 1
10 }
```

The response is displayed in the bottom panel, showing a 200 OK status with the following JSON body:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "1001"
5 }
```

Tests:-

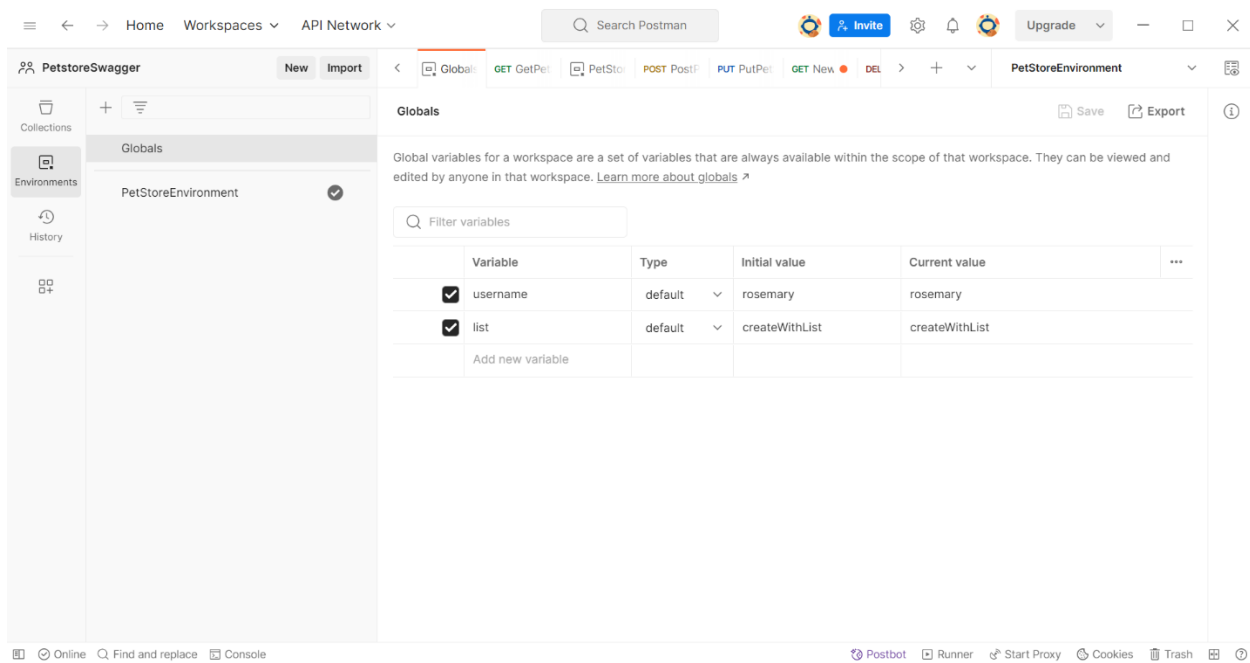
The screenshot shows the Postman interface with the same POST request, but the **Tests** tab is selected. The test scripts are as follows:

```
12 pm.response.to.have.header('content-type', 'application/json');
13 });
14 pm.test("Successful POST request", function () {
15   pm.expect(pm.response.code).to.be.oneOf([200, 202]);
16 });
17 pm.test("Status code name has string OK", function () {
18   pm.response.to.have.status("OK");
19 });
20
```

The Test Results panel shows the following results:

- PASS Status code is 200
- PASS Code Test
- PASS Content-Type is present
- PASS Content-Type is application/json

### 3. Creating Global Variable

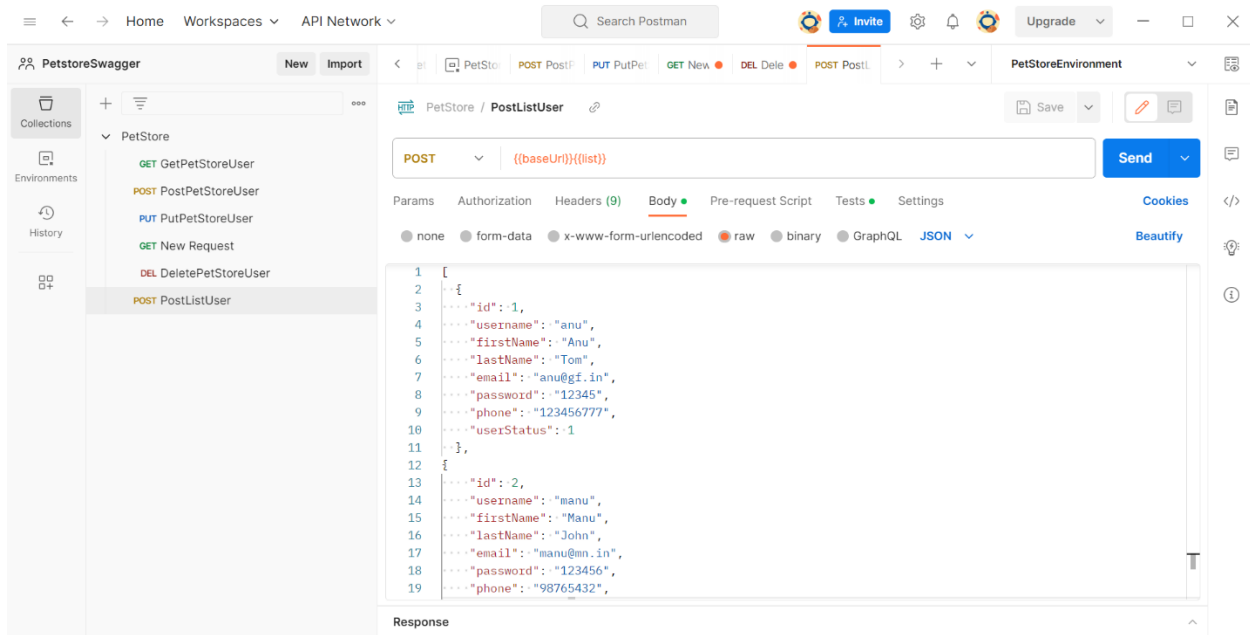


The screenshot displays the Postman application interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows 'Collections', 'Environments', 'History', and 'Bookmarks'. The main workspace is titled 'PetStoreEnvironment' and is currently in the 'Globals' tab. A text description explains that global variables are available within the workspace scope. Below this, a search bar 'Filter variables' is present. A table lists the existing global variables:

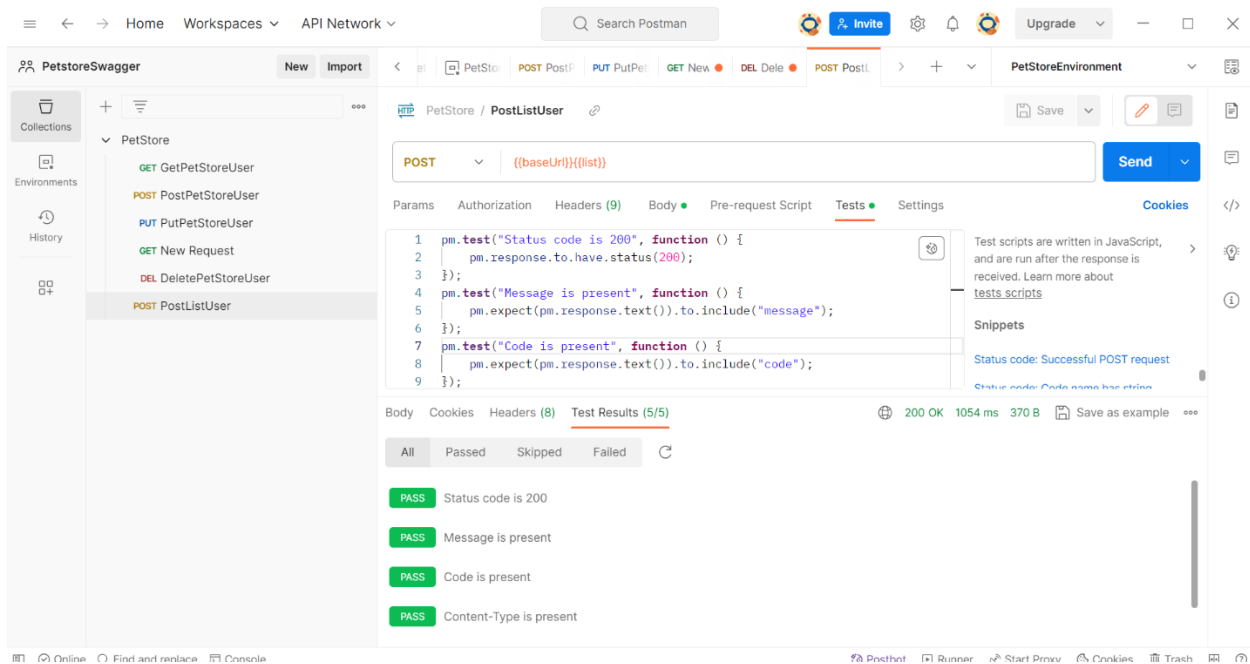
	Variable	Type	Initial value	Current value	...
<input checked="" type="checkbox"/>	username	default	rosemary	rosemary	
<input checked="" type="checkbox"/>	list	default	createWithList	createWithList	
	Add new variable				

The bottom status bar shows 'Online', 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Trash', and a help icon.

## 4. Creating a List of Users - POST



### Tests:-



## 5. Get a User - GET

The screenshot shows the Postman interface for a GET request to the endpoint `/petstore/users/{username}`. The request is configured with the method `GET` and the URL `{{baseUrl}}/{{username}}`. The `Tests` tab is active, displaying two test scripts:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4
5 pm.test("Body matches string", function () {
6   pm.expect(pm.response.text()).to.include("firstName");
7 });
8
```

The `Body` tab shows the response in JSON format:

```
1 {
2   "id": 1901,
3   "username": "rosemary",
4   "firstName": "Rose",
5   "lastName": "Mary",
6   "email": "rose@gm.in",
7   "password": "12345",
8   "phone": "9876543210",
9   "userStatus": 1
10 }
```

The status bar indicates a `200 OK` response with `1076 ms` and `472 B` of data.

Test:-

The screenshot shows the same GET request in Postman, but now the `Test Results` tab is active. It displays the results of the tests run against the response:

- PASS** Status code is 200
- PASS** Body matches string
- PASS** Schema is valid
- PASS** User Id test
- PASS** Status code name has string

The `Test Results` tab also shows the test scripts from the previous screenshot, confirming that all tests passed.

## 6. Updating a user – PUT

The screenshot shows the Postman interface with a PUT request configured for the endpoint `PUT {{baseUrl}}/{{username}}`. The request body is a JSON object representing a user update:

```
1 {
2   "id": 1001,
3   "username": "rosemary",
4   "firstName": "Rosey",
5   "lastName": "Marlet",
6   "email": "rose@gm.in",
7   "password": "12345",
8   "phone": "9876543210",
9   "userStatus": 1
10 }
```

The response is displayed in the bottom panel, showing a 200 OK status with the following JSON body:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "1001"
5 }
```

Test:-

The screenshot shows the same Postman interface, but with the 'Tests' tab selected. The test scripts are written in JavaScript and verify the response status, content type, and status code name:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Content-Type is application/json", function () {
5   pm.response.to.have.header("Content-Type", "application/json");
6 });
7 pm.test("Status code name has string", function () {
8   pm.response.to.have.status("OK");
9 });
```

The Test Results panel at the bottom shows that all four tests passed:

- PASS Status code is 200
- PASS Content-Type is application/json
- PASS Status code name has string
- PASS Response time is less than 700ms

## 7. Deleting a user - DELETE

The screenshot shows the Postman interface with a workspace named 'PetStoreSwagger'. On the left sidebar, under 'Collections', the 'PetStore' collection is expanded, showing several endpoints. The 'DELETE DeletePetStoreUser' endpoint is selected. The main panel displays the details of this endpoint:

- Method:** DELETE
- URL:** `{{baseUrl}}/{{username}}`
- Params:** None
- Authorization:** None
- Headers (7):** None
- Body:** None
- Pre-request Script:** None
- Tests:** A JavaScript test script is provided:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Body matches string", function () {
5   pm.expect(pm.response.text()).to.include("message");
6 });
7 pm.test("Status code name has string", function () {
8   pm.response.to.have.status("OK");
9 });
10 pm.test("Message Test" function () {
```
- Settings:** None
- Send:** A blue button to execute the request.

Below the request details, the response is shown in the 'Body' tab, formatted as JSON:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "rosemary"
5 }
```

The status bar at the bottom indicates a successful response: 200 OK, 715 ms, 376 B.

Test:-

The screenshot shows the same Postman interface as above, but with the 'Tests' tab selected. The test results are displayed as follows:

Test Name	Result
Status code is 200	PASS
Body matches string	PASS
Status code name has string	PASS
Message Test	PASS

The status bar at the bottom indicates a successful response: 200 OK, 715 ms, 376 B.