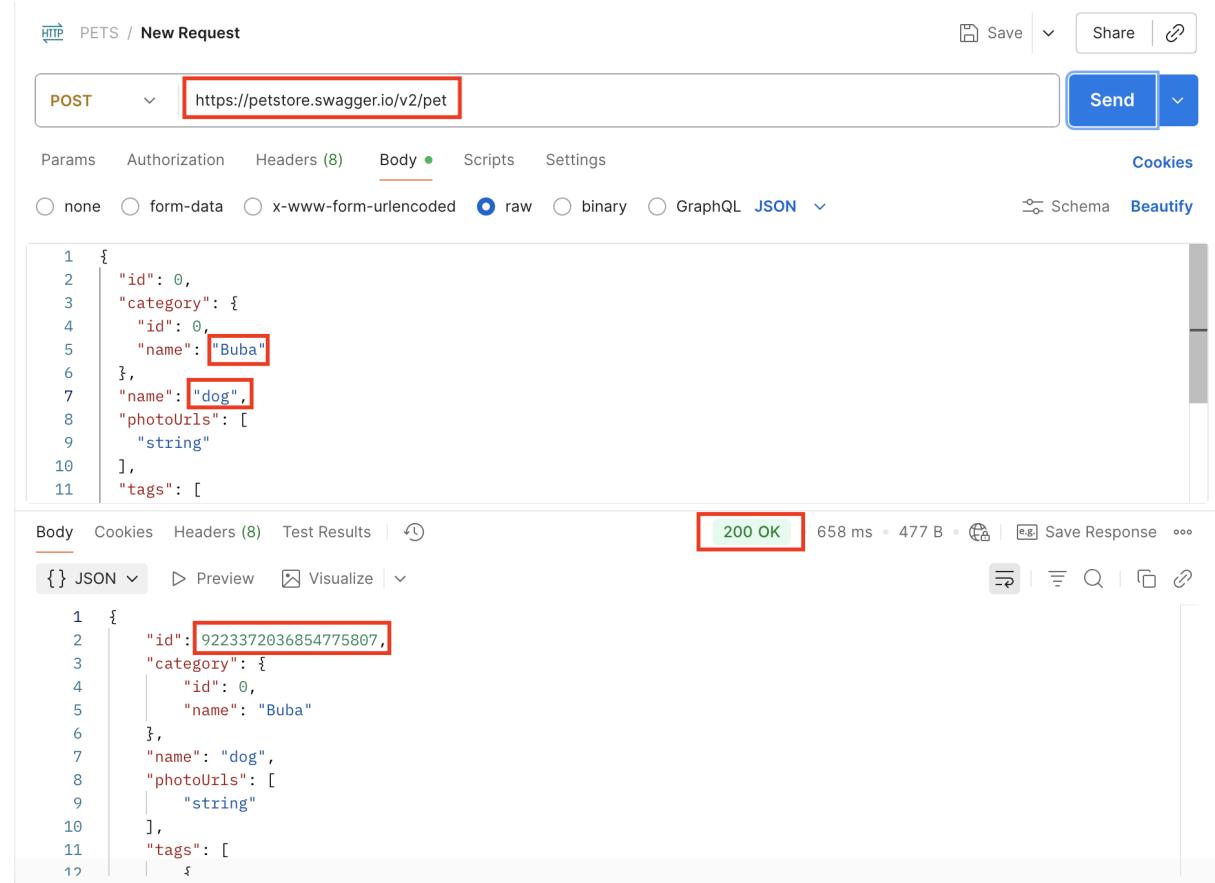


## 1.PETS

### Petstore Swagger API Testing

Tool: Postman

#### POST



HTTP PETS / New Request

POST  Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Body raw binary GraphQL JSON Schema Beautify

```
1 {  
2   "id": 0,  
3   "category": {  
4     "id": 0,  
5     "name": "Buba"  
6   },  
7   "name": "dog",  
8   "photoUrls": [  
9     "string"  
10    ],  
11   "tags": [  
12     {"name": "dog"}  
13   ]  
14 }
```

200 OK 658 ms 477 B Save Response

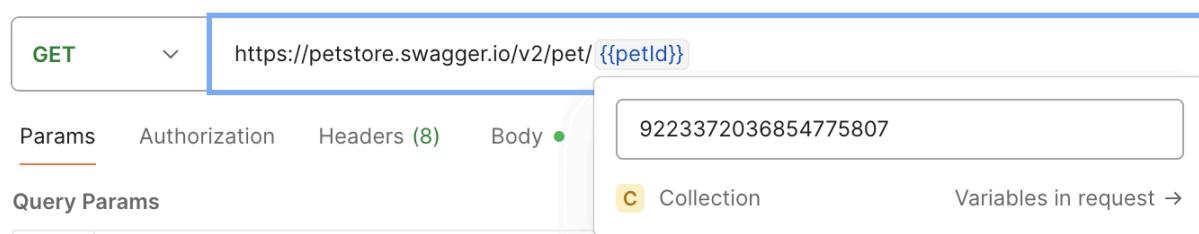
Body Cookies Headers (8) Test Results

{ } JSON Preview Visualize

```
1 {  
2   "id": 9223372036854775807,  
3   "category": {  
4     "id": 0,  
5     "name": "Buba"  
6   },  
7   "name": "dog",  
8   "photoUrls": [  
9     "string"  
10    ],  
11   "tags": [  
12     {"name": "dog"}  
13   ]  
14 }
```

#### GET

1)Get by ID ( we have to save the ID into the parameter `{{petId}}` so that we can use this ID later)



GET https://petstore.swagger.io/v2/pet/ {{petId}}

Params Authorization Headers (8) Body

Query Params

9223372036854775807

C Collection Variables in request →

HTTP PETS / Get pet by id

Save | Share | [Edit](#)

GET <https://petstore.swagger.io/v2/pet/{{petId}}> Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	<a href="#">Description</a>

Body Cookies Headers (8) Test Results | [Copy](#)

200 OK 556 ms 472 B [Save Response](#)

{ JSON ▾ ▶ Preview [Visualize](#) ▾

```

1 {
2   "id": 9223372036854775807,
3   "category": {
4     "id": 0,
5     "name": "Buba"
6   },
7   "name": "dog",
8   "photoUrls": [
9     "string"
10 ],
11   "tags": [
12   ]

```

## PUT

HTTP PETS / Update pet

Save | Share | [Edit](#)

PUT <https://petstore.swagger.io/v2/pet/> Send

Params Authorization Headers (8) Body [Raw](#) Scripts Settings Cookies

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL [JSON](#) [Schema](#) [Beautify](#)

1 {
2 "id": {{petId}},
3 "category": {
4 "id": 0,
5 "name": "Luna"
6 },
7 "name": "dog",
8 "photoUrls": [
9 "string"
10 ],
11 "tags": [
12 ]

Body Cookies Headers (8) Test Results | [Copy](#)

200 OK 1.29 s 477 B [Save Response](#)

{ JSON ▾ ▶ Preview [Visualize](#) ▾

```

1 {
2   "id": 9223372036854775807,
3   "category": {
4     "id": 0,
5     "name": "Luna"
6   },
7   "name": "dog",
8   "photoUrls": [
9     "string"
10 ],
11   "tags": [
12   ]

```

## **DELETE**

The screenshot shows the Postman interface for a DELETE request. The URL is `https://petstore.swagger.io/v2/pet/{{petId}}`. The response status is **200 OK**, and the response body is:

```
1 {  
2   "code": 200,  
3   "type": "unknown",  
4   "message": "9223372036854775807"  
5 }
```

**To make sure that we have actually deleted the pet we can use “get”:  
GET**

The screenshot shows the Postman interface for a GET request. The URL is `https://petstore.swagger.io/v2/pet/{{petId}}`. The response status is **404 Not Found**, and the response body is:

```
1 {  
2   "code": 1,  
3   "type": "error",  
4   "message": "Pet not found"  
5 }
```

## GET

Find pet by status using params:

The screenshot shows the Postman interface with the following details:

- HTTP Method:** GET
- URL:** https://petstore.swagger.io/v2/pet/findByStatus?status=sold
- Params:** status = sold
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 [  
2 {  
3     "id": 8004,  
4     "category": {  
5         "id": 8,  
6         "name": "Tasty Wooden Keyboard"  
7     },  
8     "name": "UpdatedPetName",  
9     "photoUrls": [  
10        "https://picsum.photos/seed/vRMQoTWIX/1477/1578",  
11        "https://picsum.photos/seed/oV5aZFN/481/1377"  
12    ],  
13    "tags": [  
14        {  
15            "id": 2,  
16            "name": "Unbranded Steel Soap"  
17        }  
18    ],  
19    "status": "sold"  
20},
```

Adding a pet one more time:

## POST:

The screenshot shows the Postman application interface. At the top, it says "PETS / Add new pet Copy". Below that is a search bar with "POST https://petstore.swagger.io/v2/pet" and a "Send" button. Underneath are tabs for "Params", "Authorization", "Headers (8)", "Body", "Scripts", and "Settings". The "Body" tab is selected and has a dropdown menu with "none", "form-data", "x-www-form-urlencoded", "raw", "binary", "GraphQL", and "JSON". The "JSON" option is selected and highlighted in blue. The "Body" field contains the following JSON code:

```
1 {
2     "id": 0,
3     "category": {
4         "id": 0,
5         "name": "Buba"
6     },
7     "name": "dog",
8     "photoUrls": [
9         "string"
10    ],
11    "tags": [
12        {
13            "id": 0,
14            "name": "blue dog"
15        }
16    ]
}
```

At the bottom, there are tabs for "Body", "Cookies", "Headers (8)", "Test Results", and "Visualize". The "Test Results" tab is selected and shows a green "200 OK" status with response details: 850 ms, 477 B, and a link to "Save Response". The "Visualize" tab is also visible.

## POST:

Changing an existing pet using POST and params:

The screenshot shows the Postman application interface. At the top, it says "PETS / Update pet using POST". Below that is a search bar with "POST https://petstore.swagger.io/v2/pet/{petId}?name=Nona&status=available" and a "Send" button. Underneath are tabs for "Params", "Authorization", "Headers (8)", "Body", "Scripts", and "Settings". The "Params" tab is selected and has a dropdown menu with "none", "query", "path", and "header". The "query" option is selected and highlighted in blue. The "Query Params" table shows two entries with checked checkboxes:

Key	Value	Description	Bulk Edit
name	Nona		Type
status	available		

Below the table, there are tabs for "Body", "Cookies", "Headers (8)", "Test Results", and "Visualize". The "Test Results" tab is selected and shows a green "200 OK" status with response details: 1.15 s, 387 B, and a link to "Save Response". The "Visualize" tab is also visible.

Check if data has updated:

**GET**

The screenshot shows the Postman interface with a successful API call. The URL is `https://petstore.swagger.io/v2/pet/{petId}`. The response status is **200 OK** with a response time of 781 ms and a body size of 477 B. The response body is a JSON object representing a pet, with the name field highlighted in red.

```
1 {
2     "id": 9223372036854775807,
3     "category": {
4         "id": 0,
5         "name": "Nona"
6     },
7     "name": "dog",
8     "photoUrls": [
9         "string"
10    ],
11    "tags": [
12        {
13            "id": 0,
14            "name": "string"
15        }
16    ],
17    "status": "available"
18 }
```

## 2.USERS

**POST**

Create a new user:

The screenshot shows the Postman interface with a successful API call. The URL is `https://petstore.swagger.io/v2/user`. The response status is **200 OK** with a response time of 1.05 s and a body size of 372 B. The response body is a JSON object with a code of 200, type of "unknown", and message of "8899".

```
1 {
2     "code": 200,
3     "type": "unknown",
4     "message": "8899"
5 }
```

## Adding new variables:

Variable	Value	Actions
{{petId}}	9223372036854775807	Share
{{userId}}	8899	
username	Olha	
Add variable		

## **GET:**

Try to log in using variables and params:

The screenshot shows the Postman interface for a GET request to `https://petstore.swagger.io/v2/user/login`. The 'Params' tab is selected, showing two parameters: 'username' and 'password', both set to `{{{username}}}`. The response status is 200 OK, and the JSON body contains a message about logging in.

HTTP USERS / log in

Save Share ⌂

Send

Params • Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
username	{{{username}}}		
password	{{{username}}}		
Key	Value	Description	

Body Cookies Headers (10) Test Results ⌂

200 OK 901 ms 471 B Save Response ⌂

{ } JSON ▾ Preview Visualize ▾

```
1 {  
2   "code": 200,  
3   "type": "unknown",  
4   "message": "logged in user session:1762797736799"  
5 }
```

## **GET:**

Try to log out using variables and params:

The screenshot shows the Postman interface for a GET request to the endpoint `https://petstore.swagger.io/v2/user/logout`. The response status is **200 OK**, and the response body is a JSON object with the following content:

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

## **DELETE:**

The screenshot shows the Postman interface for a DELETE request to the endpoint `https://petstore.swagger.io/v2/user/{{username}}`. The response status is **200 OK**, and the response body is a JSON object with the following content:

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

## Run a Postman collection using the Runner to test multiple API requests and check responses:

USERS - Run results

Ran today at 08:36:55 PM · [View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	petstore	1	2s 828ms	0	362 ms

All Tests Passed (0) Failed (0) Skipped (0) [View Summary](#)

Iteration 1 1

**POST Add a new user**  
https://petstore.swagger.io/v2/user

200 • 865 ms • 372 B •  
No tests found

**GET log in**  
https://petstore.swagger.io/v2/user/login?username=Olha&password=Olha

200 • 171 ms • 471 B •  
No tests found

**GET logout**  
https://petstore.swagger.io/v2/user/logout

200 • 175 ms • 370 B •  
No tests found

**DELETE Delete user**  
https://petstore.swagger.io/v2/user/Olha

200 • 236 ms • 372 B •  
No tests found

### 3.TESTS - Status code+response time + negative test.

1)"Status code is 200"

2)"Response time is less than 200ms"

The screenshot shows a Postman collection named "USERS / Add a new user". A POST request is made to <https://petstore.swagger.io/v2/user>. The "Scripts" tab contains the following pm.test blocks:

```
1 pm.test("Status code is 200", function () {  
2 | pm.response.to.have.status(200);  
3 };  
4 pm.test("Response time is less than 200ms", function () {  
5 | pm.expect(pm.response.responseTime).to.be.below(200);  
6 });
```

The "Test Results" section shows two entries:

- PASSED Status code is 200
- FAILED Response time is less than 200ms | Assertion Error: expected 1415 to be below 200

The status bar at the bottom indicates a response time of 1.42 s.

3)"Response time is less than 1000ms"

The screenshot shows a Postman collection named "USERS / Add a new user". A POST request is made to <https://petstore.swagger.io/v2/user>. The "Scripts" tab contains the following pm.test blocks:

```
1 pm.test("Status code is 200", function () {  
2 | pm.response.to.have.status(200);  
3 };  
4 pm.test("Response time is less than 1000ms", function () {  
5 | pm.expect(pm.response.responseTime).to.be.below(200);  
6 });
```

The "Test Results" section shows two entries:

- PASSED Status code is 200
- PASSED Response time is less than 1000ms

The status bar at the bottom indicates a response time of 174 ms.

## 4) Negative test:

The screenshot shows the Postman interface for a 'Negative test' scenario. The URL is `https://petstore.swagger.io/v2/user`. The 'Scripts' tab is selected, containing the following code:

```
1 pm.test("Status code is 200", function () {  
2     pm.response.to.have.status(404); ←  
3 });  
4 pm.test("Response time is less than 1000ms", function () {  
5     pm.expect(pm.response.responseTime).to.be.below(200);  
6 });
```

The 'Post-response' section is highlighted. Below the scripts, the results are displayed:

Test Results (1/2) | ⏪

- FAILED** Status code is 200 | Assertion: expected response to have status code 404 but got 200
- PASSED** Response time is less than 1000ms

Details: 200 OK | 174 ms | 372 B | Save Response | ⋮