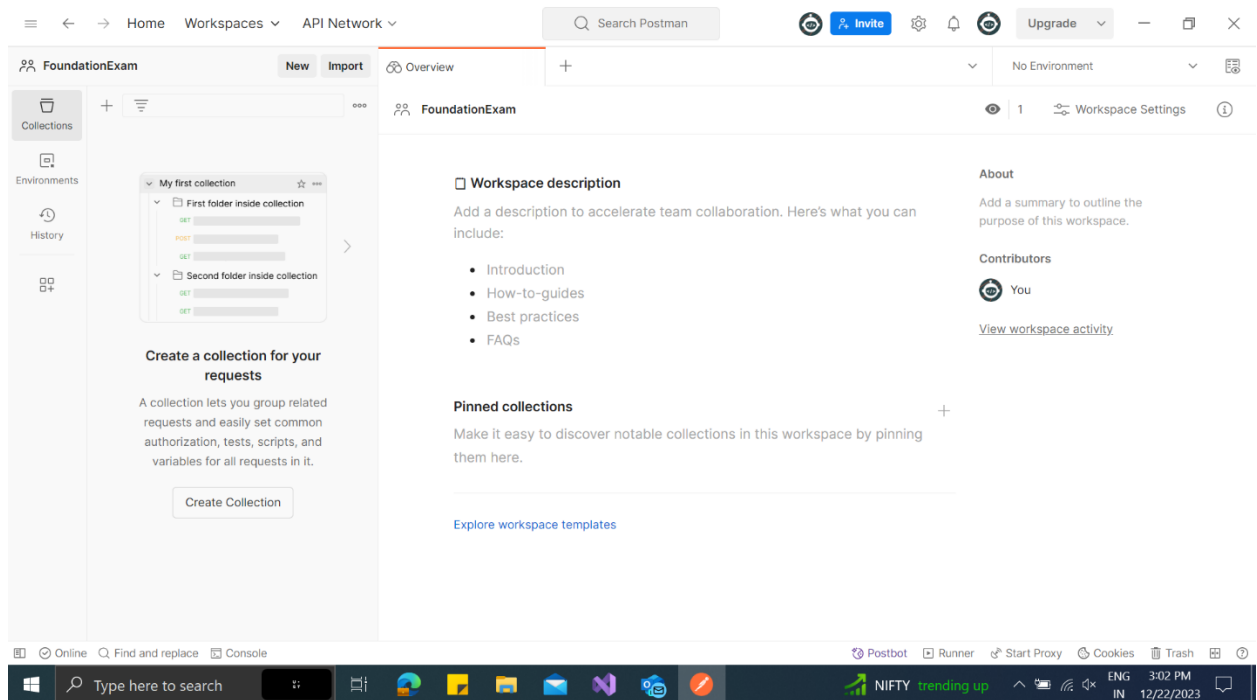


POSTMAN

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

Done HTTP method (GET, POST, PUT, DELETE) for a given scenario.

1. Creating a new workspace “FoundationExam”



2. Creating a new Environment called “PetStoreEnv” with variable “baseUrl”

The screenshot shows the Postman interface with the 'Environments' tab selected. A new environment named 'PetStoreEnv' has been created. The 'Variables' tab is active, showing a table with one variable: 'baseUrl' of type 'default' with an initial value of 'https://petstore.swagger.io/v2/user' and a current value of 'https://petstore.swagger.io/v2/user'. A notification at the bottom states: 'Use variables to reuse values and protect sensitive data. Store sensitive data in variable type secret to keep its values masked on the screen. Learn more about variable type. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values.'

Variable	Type	Initial value	Current value
<input checked="" type="checkbox"/> baseUrl	default	https://petstore.swagger.io/v2/user	https://petstore.swagger.io/v2/user
Add new variable			

3. Creating a new Collection “PetStoreAPI”

The screenshot shows the Postman interface with the 'Collections' tab selected. A new collection named 'PetStoreAPI' has been created. The 'Overview' tab is active, showing the collection name 'PetStoreAPI' and a description: 'Make things easier for your teammates with a complete collection description.' The collection was created by 'You' on '22 Dec 2023, 3:10 PM'. A link to 'View complete documentation' is provided.

4. GET – GetUserByUsername Response Body

The screenshot shows the Postman interface with the 'GET /user/2' endpoint selected. The response body is displayed in the 'Body' tab, showing a JSON object with user details. The status is 200 OK, and the response time is 431 ms.

GET /user/2

Params: Authorization, Headers (7), Body, Pre-request Script, Tests, Settings

Query Params:

Key	Value	Description
Key	Value	Description

Body: Cookies, Headers (8), Test Results

200 OK 431 ms 477 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 2,
3   "username": "user2",
4   "firstName": "Jane",
5   "lastName": "Smith",
6   "email": "user2@example.com",
7   "password": "password",
8   "phone": "0987654321",
9   "userStatus": 1
10 }
```

5. GET – GetUserByUsername Tests

The screenshot shows the Postman interface with the 'GET /user/2' endpoint selected. The 'Tests' tab is active, showing three test scripts. The status is 200 OK, and the response time is 1028 ms.

GET /user/2

Params: Authorization, Headers (7), Body, Pre-request Script, Tests, Settings

Test scripts are written in JavaScript, and are run after the response is received. Learn more about [tests scripts](#)

Snippets

- Response body: Contains string
- Response body: JSON value check
- Response body: Is equal to a string

Body: Cookies, Headers (8), Test Results (3/3)

200 OK 1028 ms 477 B Save as example

All Passed Skipped Failed

PASS Status code is 200

PASS Username check

PASS First Name check

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Username check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.username).to.eql("user2");
7 });
8 pm.test("First Name check", function () {
9   pm.expect(pm.response.text()).to.include("Jane");
10 });
```

6. POST – CreateUser Response Body

The screenshot shows the Postman interface for a POST request to `PetStoreAPI / CreateUser`. The request is configured with the method `POST` and the URL `{{baseUrl}}`. The response body is displayed in the `Body` tab, showing a JSON object with the following structure:

```
1 {
2   "id": 1234,
3   "username": "user1234",
4   "firstName": "Ashwin",
5   "lastName": "Gopal",
6   "email": "ashwin@gmail.com",
7   "password": "12345",
8   "phone": "9878768767",
9   "userStatus": 1
10 }
```

The response status is `200 OK`, with a response time of `1062 ms` and a body size of `372 B`. The response is also displayed in the `Body` tab, showing a JSON object with the following structure:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "1234"
5 }
```

7. POST – CreateUser Tests

The screenshot shows the Postman interface for a POST request to `PetStoreAPI / CreateUser`. The request is configured with the method `POST` and the URL `{{baseUrl}}`. The `Tests` tab is selected, showing the following test script:

```
1 pm.test("Successful POST request", function () {
2   pm.expect(pm.response.code).to.be.oneOf([200, 201]);
3 });
4 pm.test("Message check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.message).to.eql("1234");
7 });
8 pm.test("Status code name has string OK", function () {
9   pm.response.to.have.status("OK");
10 });
```

The response status is `200 OK`, with a response time of `1062 ms` and a body size of `372 B`. The `Test Results` tab shows the following results:

All	Passed	Skipped	Failed
	PASS		
	PASS		
	PASS		

The test results are:

- Successful POST request
- Message check
- Status code name has string OK

8. PUT – UpdateUser Response Body

The screenshot shows the Postman interface for the **PUT /UpdateUser** endpoint. The request is a PUT method to the URL `{{baseUrl}}/user1`. The response body is displayed in the **Body** tab, showing a JSON object with the following structure:

```
1 {
2   "id": "909901",
3   "username": "user1",
4   "firstName": "user2F",
5   "lastName": "user1L",
6   "email": "test@gmail.com",
7   "password": "test",
8   "phone": "9810222889",
9   "userStatus": 0
10 }
```

Below the request body, the **Body** tab shows the response body in the **Pretty** view:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "909901"
5 }
```

The status bar at the bottom indicates a 200 OK response with a response time of 1106 ms and a response size of 374 B.

9. PUT – UpdateUser Tests

The screenshot shows the Postman interface for the **PUT /UpdateUser** endpoint. The request is a PUT method to the URL `{{baseUrl}}/user1`. The response body is displayed in the **Body** tab, showing a JSON object with the following structure:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "909901"
5 }
```

Below the request body, the **Tests** tab shows the test scripts:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Type check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.type).to.eql("unknown");
7 });
8 pm.test("Message check", function () {
9   pm.expect(pm.response.text()).to.include("message");
10 });
```

The test results are displayed in the **Test Results (3/3)** tab, showing three passed tests:

- PASS** Status code is 200
- PASS** Type check
- PASS** Message check

The status bar at the bottom indicates a 200 OK response with a response time of 1106 ms and a response size of 374 B.

10. DELETE – DeleteUserByUsername Response Body

The screenshot shows the Postman interface for the `DELETE /DeleteUserByUsername` endpoint. The URL is `{{baseUrl}}/user1`. The response body is displayed in the `Body` tab, showing a JSON object with the following structure:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "user1"
5 }
```

The status bar at the bottom indicates a 200 OK response with 1040 ms latency and 373 B of data.

11. DELETE – DeleteUserByUsername Tests

The screenshot shows the Postman interface for the `DELETE /DeleteUserByUsername` endpoint, with the `Tests` tab selected. The tests are defined as follows:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Message value check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.message).to.eql("user1");
7 });
8 pm.test("Status code name has string OK", function () {
9   pm.response.to.have.status("OK");
10 });
```

The test results are displayed in the `Test Results (3/3)` tab, showing that all three tests passed:

- PASS Status code is 200
- PASS Message value check
- PASS Status code name has string OK

The status bar at the bottom indicates a 200 OK response with 1040 ms latency and 373 B of data.

12. GET – LogsUser Response Body

The screenshot shows the Postman interface for the **GET LogsUser** endpoint. The URL is `{{baseUrl}}/login?username=test&password=abc123`. The response status is **200 OK** with a response time of **1055 ms** and a body size of **471 B**. The response body is displayed in the **Body** tab, showing a JSON object:

```
1 {
2   "code": 200,
3   "type": "unknown",
4   "message": "logged in user session:1703240741979"
5 }
```

The left sidebar shows the **FoundationExam** workspace with the **PetStoreAPI** collection. The **LogsUser** endpoint is selected. The bottom status bar shows the system clock as 3:55 PM on 12/22/2023.

13. GET – LogsUser Tests

The screenshot shows the Postman interface for the **GET LogsUser** endpoint with the **Tests** tab selected. The URL is `{{baseUrl}}/login?username=test&password=abc123`. The response status is **200 OK** with a response time of **476 ms** and a body size of **471 B**. The tests are written in JavaScript and are all passing:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 pm.test("Code value check", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.code).to.eql(200);
7 });
8 pm.test("Message value check", function () {
9   pm.expect(pm.response.text()).to.include("logged in user session");
10 });
```

The **Test Results (3/3)** section shows the following results:

- PASS** Status code is 200
- PASS** Code value check
- PASS** Message value check

The left sidebar shows the **FoundationExam** workspace with the **PetStoreAPI** collection. The **LogsUser** endpoint is selected. The bottom status bar shows the system clock as 4:01 PM on 12/22/2023.