

Hands-on Lab - Modernize JPetStore with Microservices



Estimated Time: 30 minutes

In this lab, you will become familiar with using the Swagger UI. The Swagger UI is an open source project to visually render documentation for an API defined with the OpenAPI (Swagger) specification. REST APIs endpoint is one end of a communication channel. When an API interacts with another system, the touchpoints of this communication are considered endpoints. For APIs, an endpoint can include a URL of a server or service.

Learning Objectives:

After completing this exercise, you should be able to perform the following tasks:

- Use the Swagger UI and understand the various components.
- Access endpoints through the Swagger UI.

Pre-requisites

- You must be familiar with Docker applications and commands.
- You must have a good understanding of REST API.

Task 1 - Getting the Swagger UI

1. Open a terminal window by using the top menu in the IDE: **Terminal > New Terminal**.

2. In the terminal, run the following command to pull the docker image for Swagger.

```
docker pull swaggerapi/swagger-ui
```

3. To run the swagger application and access the UI, run the following command.

```
docker run -dp 8080:8080 swaggerapi/swagger-ui
```

Note - You will get a hex code in return. This means the server has successfully started.

4. Click on the button below or click the **Skills Network** icon and choose **Launch Application** from the menu and enter the port number *8080*.

Launch Application

5. It opens the browser and connects to port 8080 which is where the Swagger application is running.

Swagger UI comes up connecting to the default, preconfigured sample [PetStore JSON](#). You can check this JSON file by opening in a new page

The screenshot shows the Swagger Petstore interface. At the top left is the Swagger logo with the text "Supported by SMARTTEAM". The URL "https://petstore.swagger.io/v2/swagger.json" is displayed in the address bar. On the right, there is a green "Explore" button. Below the header, the title "Swagger Petstore 1.0.6" is shown, along with a note about it being a sample server. A red arrow points from the text "JSON being used" to the URL in the address bar. Another red arrow points from the text "Protocol Scheme" to the "Schemes" dropdown menu, which is currently set to "HTTPS".

pet Everything about your Pets

- POST** /pet/{petId}/uploadImage uploads an image
- POST** /pet Add a new pet to the store
- PUT** /pet Update an existing pet
- GET** /pet/findByStatus Finds Pets by status
- GET** /pet/findByTags Finds Pets by tags
- GET** /pet/{petId} Find pet by ID
- POST** /pet/{petId} Updates a pet in the store with form data
- DELETE** /pet/{petId} Deletes a pet

store Access to Petstore orders

- POST** /store/order Place an order for a pet
- GET** /store/order/{orderId} Find purchase order by ID
- DELETE** /store/order/{orderId} Delete purchase order by ID
- GET** /store/inventory Returns pet inventories by store

user Operations about user

- POST** /user/createWithArray Creates list of users with given input array
- POST** /user/createWithList Creates list of users with given input array
- GET** /user/{username} Get user by user name
- PUT** /user/{username} Updated user
- DELETE** /user/{username} Delete user
- GET** /user/login Logs user into the system
- GET** /user/logout Logs out current logged in user session
- POST** /user Create user

The page will look as shown below. It lists the following:

1. JSON being used
2. The protocol scheme
3. The end points along with the type of REST requests - GET, POST, PUT, UPDATE, DELETE
4. It also lists the Models that are used in the application

Task 2 - Accessing endpoints through the Swagger UI

As a part of this task you will -

1. Add a pet
2. Get the details of the pet by id
3. Update the pet status to **sold**
4. Get the details of the pet by id to see if the status has been updated
5. Delete the pet
6. Get the details of the pet by id to see that it doesn't exist

You will try POST, GET and DELETE endpoints.

1. Click the dropdown next to the end point **POST /pet**.

POST**/pet** Add a new pet to the store

2. Click on Try it out to add a pet.

Parameters

Name	Description
body * required object (body)	Pet object that needs to be added to the store

```
{  
    "id": 0,  
    "category": {  
        "id": 0,  
        "name": "string"  
    },  
    "name": "doggie",  
    "photoUrls": [  
        "string"  
    ],  
    "tags": [  
        {  
            "id": 0,  
            "name": "string"  
        }  
    ],  
    "status": "available"  
}
```

3. This will allow you to edit the values. Replace the prepopulated JSON, with the following:

```
{  
    "id": 10,  
    "category": {  
        "id": 1,  
        "name": "string"  
    },  
    "name": "doggie",  
    "photoUrls": [  
        "string"  
    ],  
    "tags": [  
        {  
            "id": 1,  
            "name": "string"  
        }  
    ],  
    "status": "available"  
}
```

```
        "name": "dogs"
    },
    "name": "Hershey",
    "photoUrls": [
        "https://i.natgeofe.com/n/4f5aaece-3300-41a4-b2a8-ed2708a0a27c/domestic-dog_thumb_3x2.jpg"
    ],
    "tags": [
        {
            "id": 1,
            "name": "Friendly"
        }
    ],
    "status": "available"
}
```

POST**/pet** Add a new pet to the store

Parameters

Name	Description
------	-------------

body * requiredobject
(body)

Pet object that needs to be added to the store

[Edit Value](#) | [Model](#)

```
{
  "id": 10,
  "category": {
    "id": 1,
    "name": "dogs"
  },
  "name": "Hershey",
  "photoUrls": [
    "https://i.natgeofe.com/n/4f5aaece-3300-41a4-b2a8-ed2708a0"
  ],
  "tags": [
    {
      "id": 1,
      "name": "Friendly"
    }
  ],
  "status": "available"
}
```

[Cancel](#)**Parameter content type**

application/json

Execute

4. Click **Execute** to add the pet. You should see a **Server Response** with Code 200, which means the POST request was successful.

Server response

Code	Details
------	---------

200

Undocumented Response body

```
{  
    "id": 10,  
    "category": {  
        "id": 1,  
        "name": "dogs"  
    },  
    "name": "Hershey",  
    "photoUrls": [  
        "https://i.natgeofe.com/n/4f5aaece-3300-41a4-b2a8-ed2708a0a27c/domest  
    ],  
    "tags": [  
        {  
            "id": 1,  
            "name": "Friendly"  
        }  
    ],  
    "status": "available"  
}
```

Response headers

content-type: application/json

5. Close the dropdown for **POST /pet**.
6. Now you will get the details of the pet you just added. Click the dropdown next to **GET /pet/{petid}**.
7. Click **Try it out** and enter the id of the pet whose details you want to retrieve. Our pet's id is **10**. Click **Execute**.

GET**/pet/{petId}** Find pet by ID

Returns a single pet

Parameters

Name	Description
------	-------------

petId * required	
-------------------------	--

<code>integer(\$int64)</code>	ID of pet to return (path)
-------------------------------	-------------------------------

10

Execute

Responses

8. In the Server Response obtained, you will see **200** indicating the request was successful and the details of the pet.

Responses

Curl

```
curl -X 'GET' \
'https://petstore.swagger.io/v2/pet/10' \
-H 'accept: application/json' \
-H 'api_key: special-key'
```

Request URL

<https://petstore.swagger.io/v2/pet/10>

Server response

Code Details

200

Response body

```
{
  "id": 10,
  "category": {
    "id": 1,
    "name": "dogs"
  },
  "name": "Hershey",
  "photoUrls": [
    "https://i.natgeofe.com/n/4f5aaece-3300-41a4-b2a8-ed2708a0a27c/domesti"
  ],
  "tags": [
    {
      "id": 1,
      "name": "Friendly"
    }
  ],
  "status": "available"
}
```

9. Close the **GET /pet/{petid}** dropdown.
10. You will now update the status of the pet to **sold**. To do this, click the dropdown next to **POST /pet/{petid}**.
11. Now you will change the status of the pet with id **10** to **sold**. Click **Try it out** and make the changes as shown below, and then click **Execute**.

POST**/pet/{petId}** Updates a pet in the store with form data

Parameters

Name	Description
------	-------------

petId * required

integer(\$int64) ID of pet that needs to be updated
(path)

10

name

string

(*formData*)

Hershey

status

string

(*formData*)

sold

Execute

12. If the update ran successfully, you get a server response with code **200**.

Responses

Curl

```
curl -X 'POST' \
  'https://petstore.swagger.io/v2/pet/10' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'name=Hershey&status=sold'
```

Request URL

```
https://petstore.swagger.io/v2/pet/10
```

Server response

Code Details

200

Undocumented

Response body

```
{  
  "code": 200,  
  "type": "unknown",  
  "message": "10"  
}
```

Response headers

```
content-type: application/json
```

13. Close the **POST /pet/{petid}** dropdown.
14. You can check if the pet you just updated reflects as **sold**. Click the dropdown next to **GET /pet/{petid}**.
15. Enter the id of the pet whose details you want to retrieve. The pet you updated has the id **10**. Click **Execute**.

GET**/pet/{petId}** Find pet by ID

Returns a single pet

Parameters

Name	Description
------	-------------

petId * required	
-------------------------	--

<code>integer(\$int64)</code>	ID of pet to return (path)
-------------------------------	-------------------------------

10

Execute

Responses

16. In the Server Response obtained, you will see **200** indicating the request was successful and the details of the pet.

Responses

Curl

```
curl -X 'GET' \
  'https://petstore.swagger.io/v2/pet/10' \
  -H 'accept: application/json' \
  -H 'api_key: special-key'
```

Request URL

```
https://petstore.swagger.io/v2/pet/10
```

Server response

Code	Details
------	---------

200	Response body
-----	---------------

```
{
  "id": 10,
  "category": {
    "id": 1,
    "name": "dogs"
  },
  "name": "Hershey",
  "photoUrls": [
    "https://i.natgeofe.com/n/4f5aaece-3300-41a4-b2a8-ed2708a0a27c/domest"
  ],
  "tags": [
    {
      "id": 1,
      "name": "Friendly"
    }
  ],
  "status": "sold"
}
```

17. Close the **GET /pet/{petid}** dropdown.

18. Now that the pet is sold, you can delete it from your system. Click the dropdown next to **DELETE /pet/{petid}**. Click **Try it out** and enter the **petid** as 10.

DELETE /pet/{petId} Deletes a pet

Parameters

Name	Description
------	-------------

api_key <small>string (header)</small>	api_key
---	---------

petId * required <small>integer(\$int64)</small> (path)	Pet id to delete
--	------------------

10

Execute

19. Click **Execute**. If the delete ran successfully, you get a server response with code **200**.

Responses

Curl

```
curl -X 'DELETE' \
  'https://petstore.swagger.io/v2/pet/10' \
  -H 'accept: application/json'
```

Request URL

```
https://petstore.swagger.io/v2/pet/10
```

Server response

Code	Details
------	---------

200

Undocumented

Response body

```
{  
  "code": 200,  
  "type": "unknown",  
  "message": "10"  
}
```

20. You can check if the pet you just deleted, has been removed from the system. Click the dropdown next to **GET /pet/{petid}**.

21. Enter the id of the pet whose details you want to retrieve. The pet you deleted has the id **10**. Click **Execute**.

GET**/pet/{petId}** Find pet by ID

Returns a single pet

Parameters

Name	Description
------	-------------

petId * required	
-------------------------	--

<code>integer(\$int64)</code>	ID of pet to return (path)
-------------------------------	-------------------------------

10

Execute

Responses

22. In the Server Response obtained you will see **404** indicating the request was erroneous and there is no such pet.

Responses

Curl

```
curl -X 'GET' \
  'https://petstore.swagger.io/v2/pet/10' \
  -H 'accept: application/json' \
  -H 'api_key: special-key'
```

Request URL

```
https://petstore.swagger.io/v2/pet/10
```

Server response

Code	Details
------	---------

404

Error: response status is 404

Response body

```
{  
  "code": 1,  
  "type": "error",  
  "message": "Pet not found"  
}
```

Congratulations! You have successfully completed the task.

Tutorial details

Author: Lavanaya T S

Contributors: Pallavi Rai

© IBM Corporation. All rights reserved.