1. **What was the response when you sent a GET request to /pet/1?**

200 a successful operation, I also had a response body and response headers sent to me.

2. **What was the response code, and what does it mean?**

Response code 200 and it means I was able to get a successful response so what I was looking for exists, I can read it because the server is working as it should and I have permission to see it.

Response body:

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


Response headers:

```
access-control-allow-headers: Content-Type,api_key,Authorization
 access-control-allow-methods: GET,POST,DELETE,PUT
 access-control-allow-origin: *
 content-type: application/json
 date: Wed,13 Sep 2023 07:47:38 GMT
 server: Jetty(9.2.9.v20150224)
```


3. What fields were in the response body, and what do they mean?

The fields are JSON files with data typically taken from a database.
"id" is typically a unique id often used as a primary key or foreign key.

"name" is self-explanatory, the dog's name is the string "doggie"

"photosUrls" would normally have a url to an image of the dog but that's no been added so they've just got an empty string array.

The "status" means the pet is available for sale, the other options would have been "sold" or "pending" were this not the case.

"tags" and "category" don't contain much info in themselves, instead you're mean to use the ids provided in them to find information in another JSON file (XML is less common these days) which comes from another database table to figure out what tags with the id of 1 mean and what categories with the id of 1 mean.

```
  "tags": [
    {
      "id": 1,
      "name": "string"
    }
 "category": {
    "id": 1,
    "name": "string"
  },
```

4. **When you sent a POST request to /pet, what did you put in the request body?**

You need to provide properly formatted JSON similar to the style below with the key value pair structure. I kept the original code with 1 change being done in the name changing from "doggie" to "DoggoChange".

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

Every pet needs ids which are numbers. They need a primary id which is the first one you see in the JSON file, a category id and a tag id.

They require a name in string format.

The "photoUrls" look to be optional so an empty string array can be provided or you could provide a url in the form of a string pointing to an image of the pet.

The pet needs a status of "available", "sold" or "pending"
Categories and tags have names as well as ids but in this context "string" is just used as a name.

5. **What was the response to the POST request?**

A 200, all went well and the input was valid

```
curl -X 'POST' \
  'https://petstore.swagger.io/v2/pet' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": 110,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "DoggoChange",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}'
```

6. **What was the response code of the POST request, and what does it mean?**

The response code was a 200 so the POST was successful, it would have been 405 if the input was invalid.

If it was creating a brand-new entry into the database with a unique id I would have possibly been given a 201 which means the entry I sent was created.

Read about 201 response from
MozDevNet (no date) 201 created - http: MDN, HTTP | MDN. Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/201 (Accessed: 13 September 2023).

7. **Can you explain the purpose of the /pet/{petId} and /pet endpoints?**

An endpoint is either a URL or URI that a client or a server talking to another server can use to perform an action or access a resource.

The {petId} is a placeholder for an id so you could have /pet/1 or pet/1234567.

You need these endpoints so you know where your GET, POST, DELETE, PUT and PATCH requests are targeting. Normally you'd want the {petId} part for deletion or editing-related requests so only the specific pets are impacted.

If you make an HTTP request to an API endpoint the server will respond with the relevant data or requested action (as long as the client has met both authorization and authentication criteria if such things are required).

8. **If the /pet/{petId} endpoint returned a 404 status code, what would that mean and what might cause it?**

A 404 means what you're looking for hasn't been found. This could mean your url has typos in it and you'd be for instance looking for an animal id that never existed. Or it could you were looking for a pet that at one point existed but then got deleted from the server and thus is no longer found.

Sometimes you want to flat-out lie and use a 404. If you're looking for a Private GitHub project that you don't have access to then they will respond with a 404 rather than a 401. The reason to be deceptive is for privacy reasons, if I looked in GitHub for say "/company-super-secret-project" then I'd know a company was working on a super secret project if I got a 401 even if I couldn't access it.

9. Choose one more endpoint from the Petstore API and explain its purpose and how you would use it.

I'm going with the GET request /pet/findByStatus

eg
https://petstore.swagger.io/v2/pet/findByStatus?status=sold

I can use this to filter results with a query, the "?status=" part. This allows me to see which pets are "sold", "pending" or "available" depending on which word I use at the end. It only recognizes those 3 words.

10. Optional: What are the advantages of using a tool like Swagger UI when working with APIs?

I find it more visually appealing than Postman or Insomnia although I'm more familiar with those two. All 3 do a fine job with the REST architectural style.

If I was to use the Swagger sales pitch from their site I'd go with the following as my 3 main reasons for liking it as a tool.

"The UI works in any development environment, be it locally or in the web"
"Cater to every possible scenario with Swagger UI working in all major browsers"
"Quickly find and work with resources and endpoints with neatly categorized documentation"

Taken from:
Swagger UI (no date) REST API Documentation Tool | Swagger UI. Available at: https://swagger.io/tools/swagger-ui/ (Accessed: 13 September 2023).