Beaconfire Inc, Home Work, Week6 Day26.
By (Ping) Nalongsone Danddank.
ndanddank@gmail.com
wechatID: ndanddank
Short Answer:

1. What is a Marker interface?

-> A marker interface is an interface that has no methods or constants inside it.
• It provides run-time type information about objects, so the compiler and JVM have additional information about the object
-> • Repository is The central interface in the Spring Data repository abstraction.
• It takes the domain class to manage as well as the ID type of the domain class as type arguments.
• This interface acts primarily as a marker interface to capture the types to work with and to help you to discover interfaces that extend this one.

2. How is caching achieved in Spring applications?

-> • To get started, we need following dependency,
• Then we have to enable caching by @EnableCaching annotation
• The @EnableCaching annotation triggers a post-processor that inspects every Spring bean for the presence of caching annotations on public methods
• If such an annotation is found, a proxy is automatically created to intercept the method call and handle the caching behavior accordingly
• After we enable caching – for the minimal setup – we must register a cacheManager
• Spring Boot automatically configures a suitable CacheManager to serve as a provider for the relevant cache (By default, it uses a ConCurrentHashMap)
• @Cacheable — Defines methods that are cacheable so that on subsequent invocations (with the same arguments), the value in the cache is returned without having to actually invoke the method
• @CachePut — When the cache needs to be updated without interfering with the method execution, you can use the @CachePut annotation.
• It supports the same options as @Cacheable
• Using @CachePut and @Cacheable annotations on the same method is generally strongly discouraged because they have different behaviors
• @Cacheable causes the method execution to be skipped by using the cache, but @CachePut forces the execution in order to execute a cache update, resulting in conflicting behaviours
• @CacheEvict — This process is useful for removing stale or unused data from the cache
• Using the allEntries attribute to evict all entries from the cache.
• @Caching — Sometimes, multiple annotations of the same type (such as @CacheEvict or @CachePut) need to be specified.

3. How can RESTful APIs be documented?

-> the API documentation should be informative, readable, and easy to follow. reference documentation should simultaneously describe every change in the API. So we should use SWAGGER

4. What is Docker, container, and images?

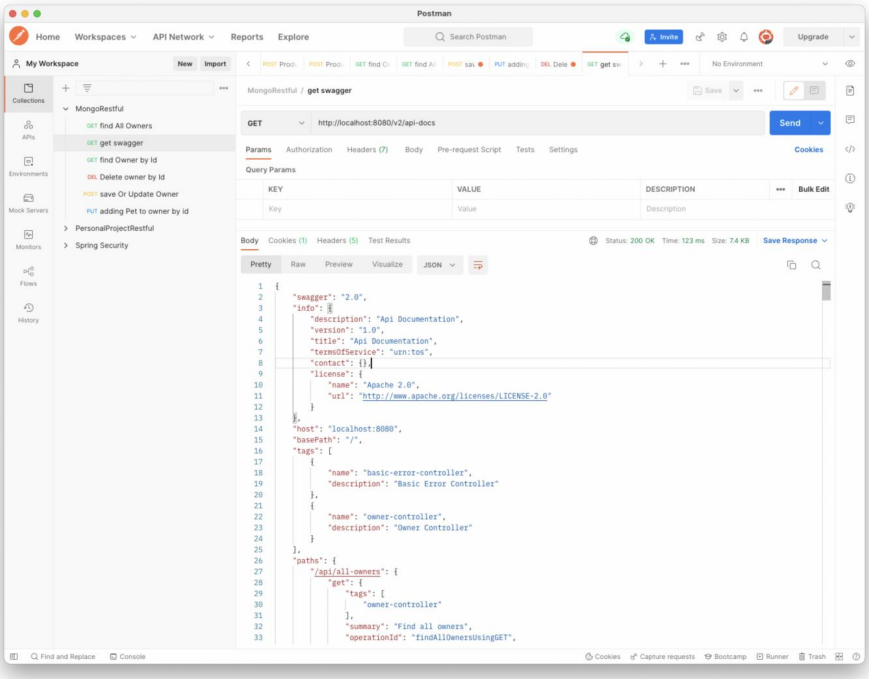
-> Docker is a standalone software that can be installed on any computer to run containerized applications. Containerization is an approach of running applications on an OS such that the application is isolated from the

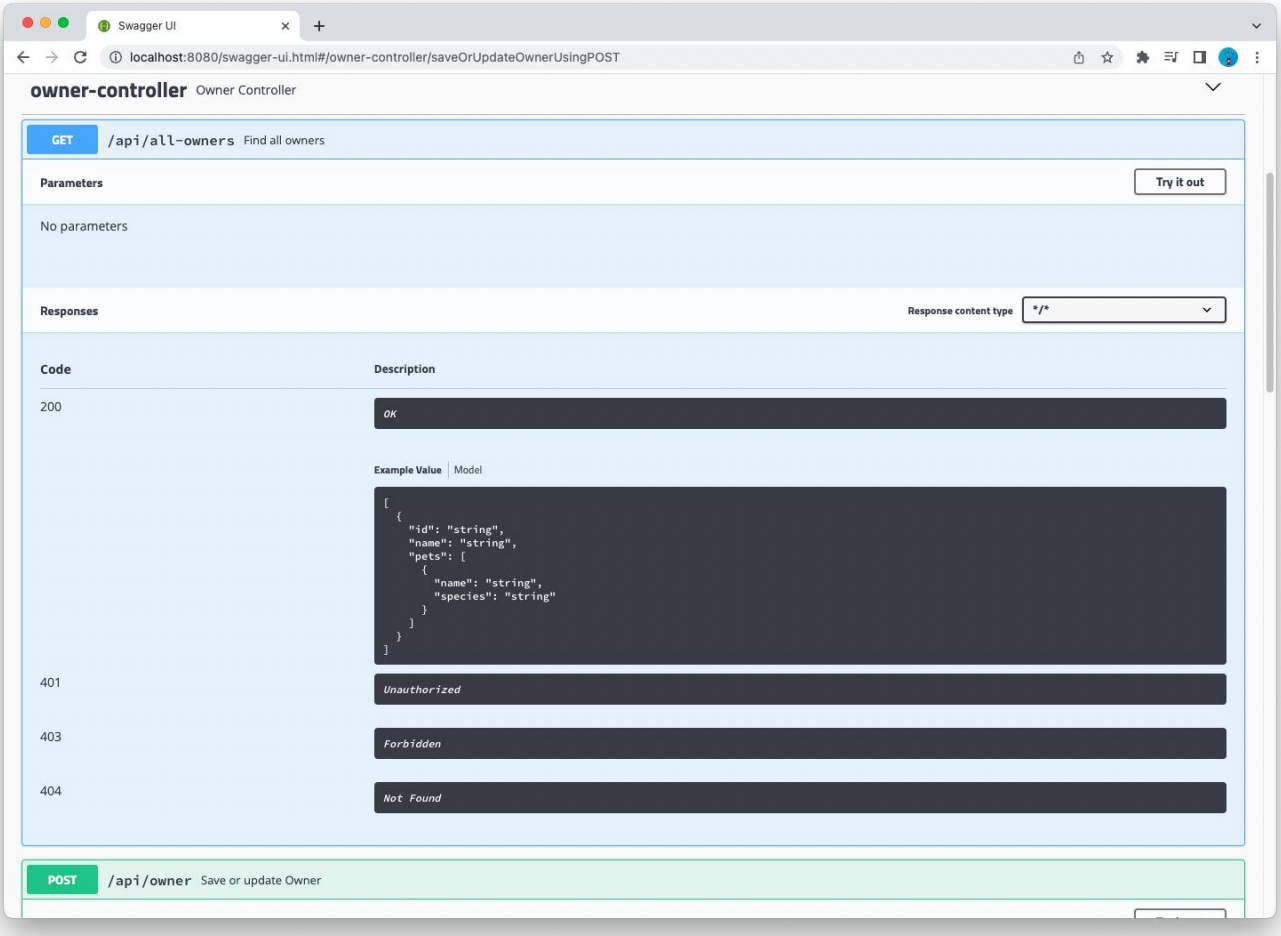
rest of the system.
Docker allows users to publish docker images and consume those published by others in repositories like Docker Hub.
• image: A static template - a set of bytes (class)
• container: Running version of your image, a image can have several containers (object)

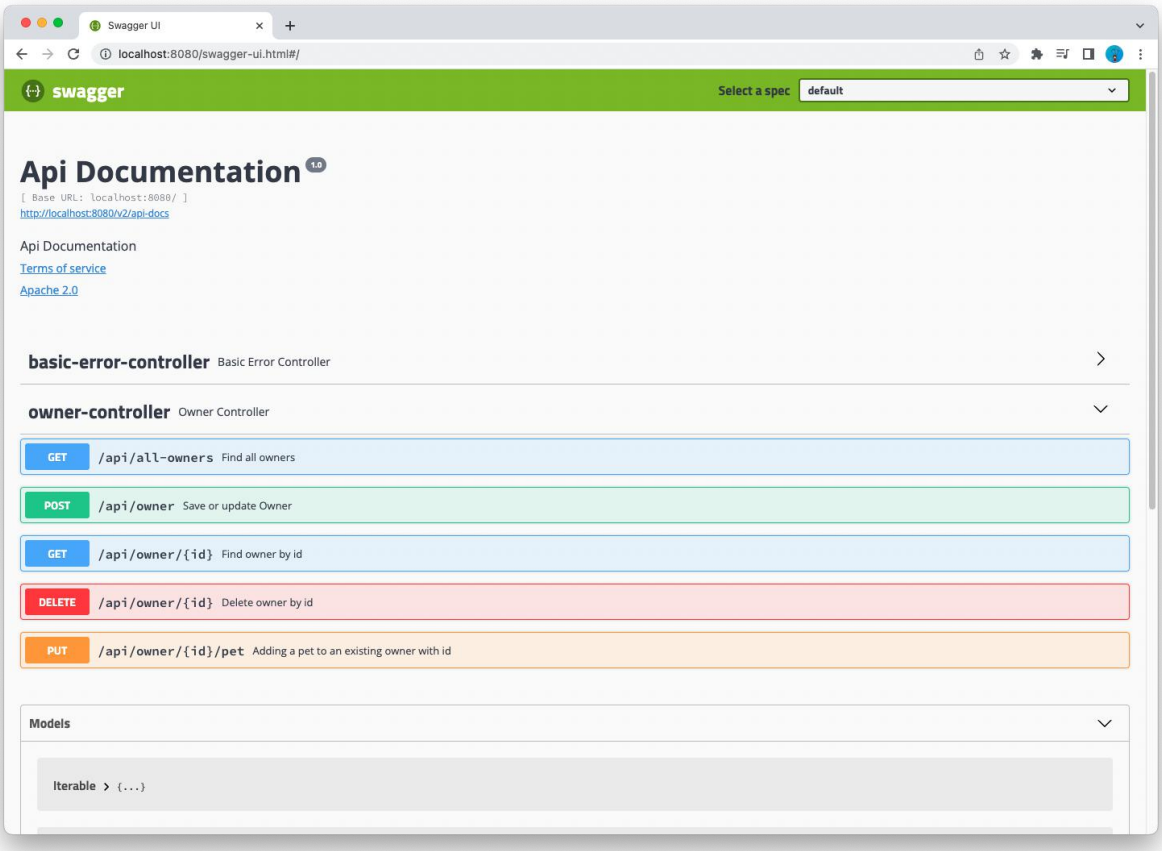
5. Describe the differences between SQL vs NoSQL databases?
-> • Schema: SQL databases have a rigid schema whereas NoSQL databases don't.
• Scalability: NoSQL databases are generally more horizontally scalable compared to SQL databases due to storing data as self-contained documents.
• Query: SQL syntax vs NoSQL database-dependent querying methods.
• Transaction: NoSQL databases generally do not support transaction across multiple documents whereas SQL databases do.

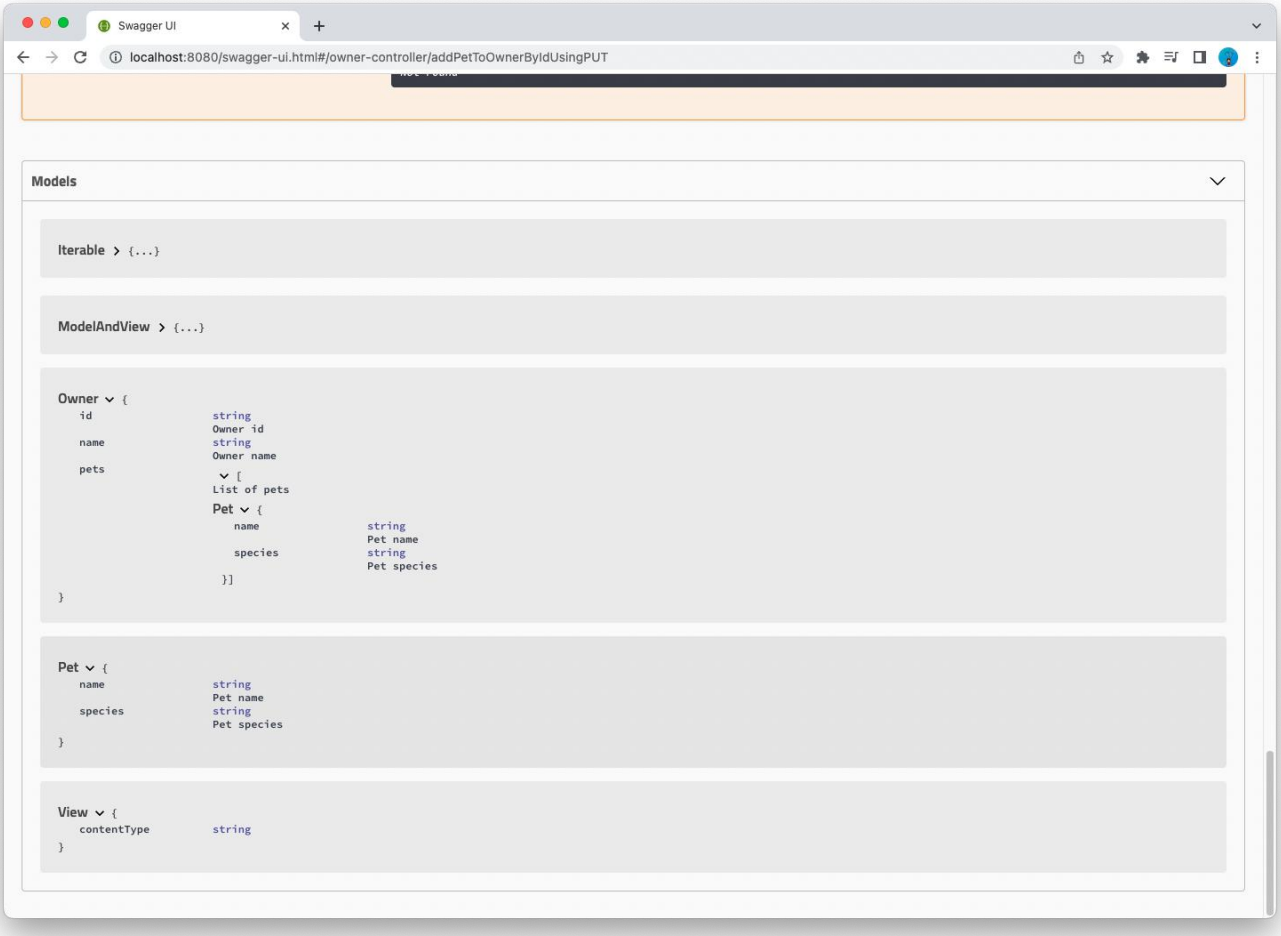
6. What are the two types of scaling?
-> PARTITIONING
• Vertical Partitioning
• id | name | age | hobbies
• id is the partition key
• id | name -> server 1
• id | age | hobbies -> server 2
• Horizontal Partitioning
• Let's say we have one billion rows
• id is the partition key
• 500 million rows -> server 1
• 500 million rows -> server 2

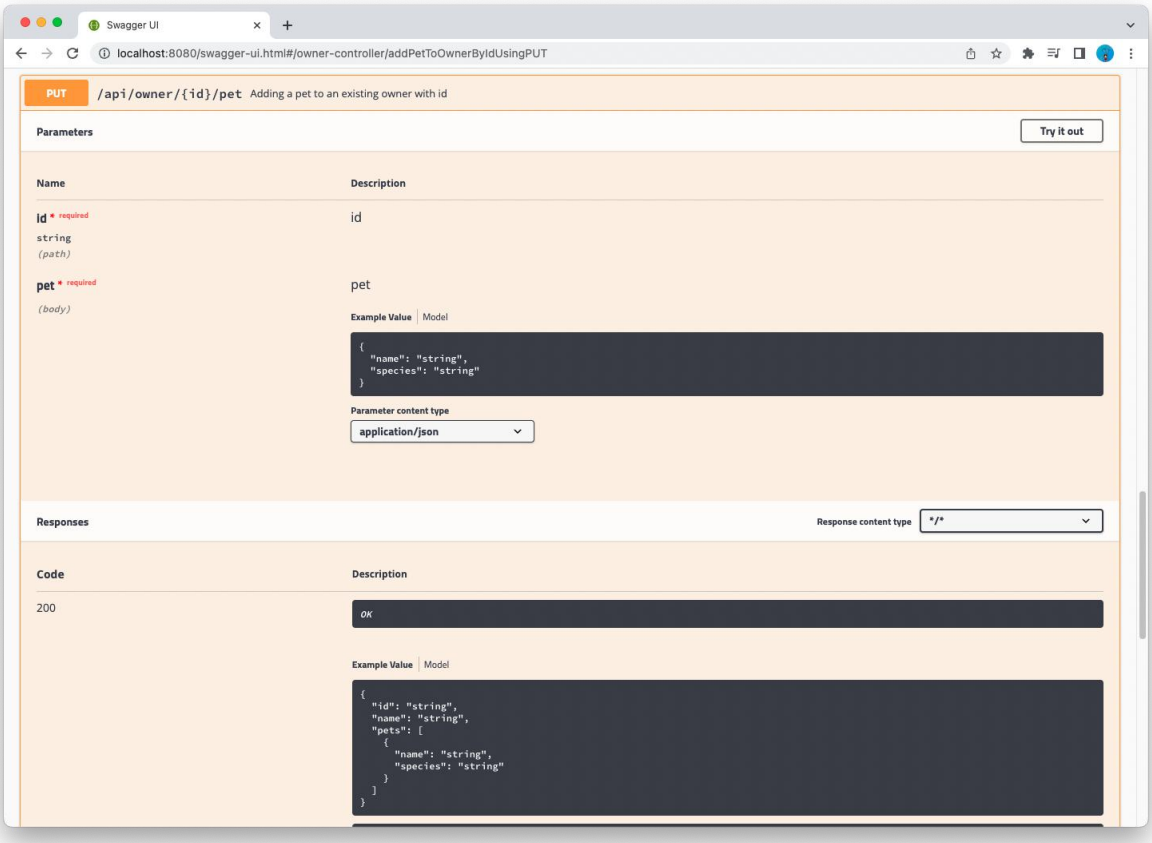
Swagger Screen short cut:



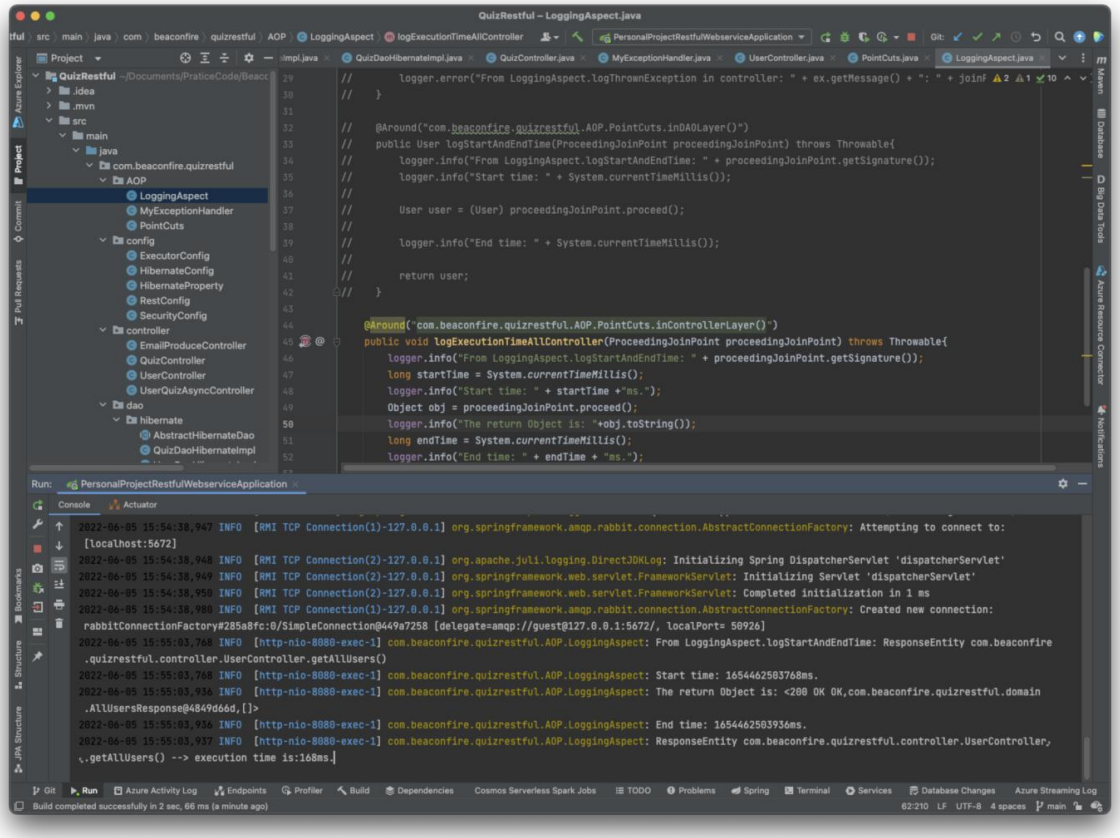

Swagger UI
localhost:8080/swagger-ui.html#/owner-controller/saveOrUpdateOwnerUsingPOST
owner-controller Owner Controller
GET /api/all-owners Find all owners
Parameters
Try it out
No parameters
Responses
Response content type */*
Code
Description
200
OK
Example Value | Model
[{ "id": "string", "name": "string", "pets": [{ "name": "string", "species": "string" }] }]
401
Unauthorized
403
Forbidden
404
Not Found
POST /api/owner Save or update Owner

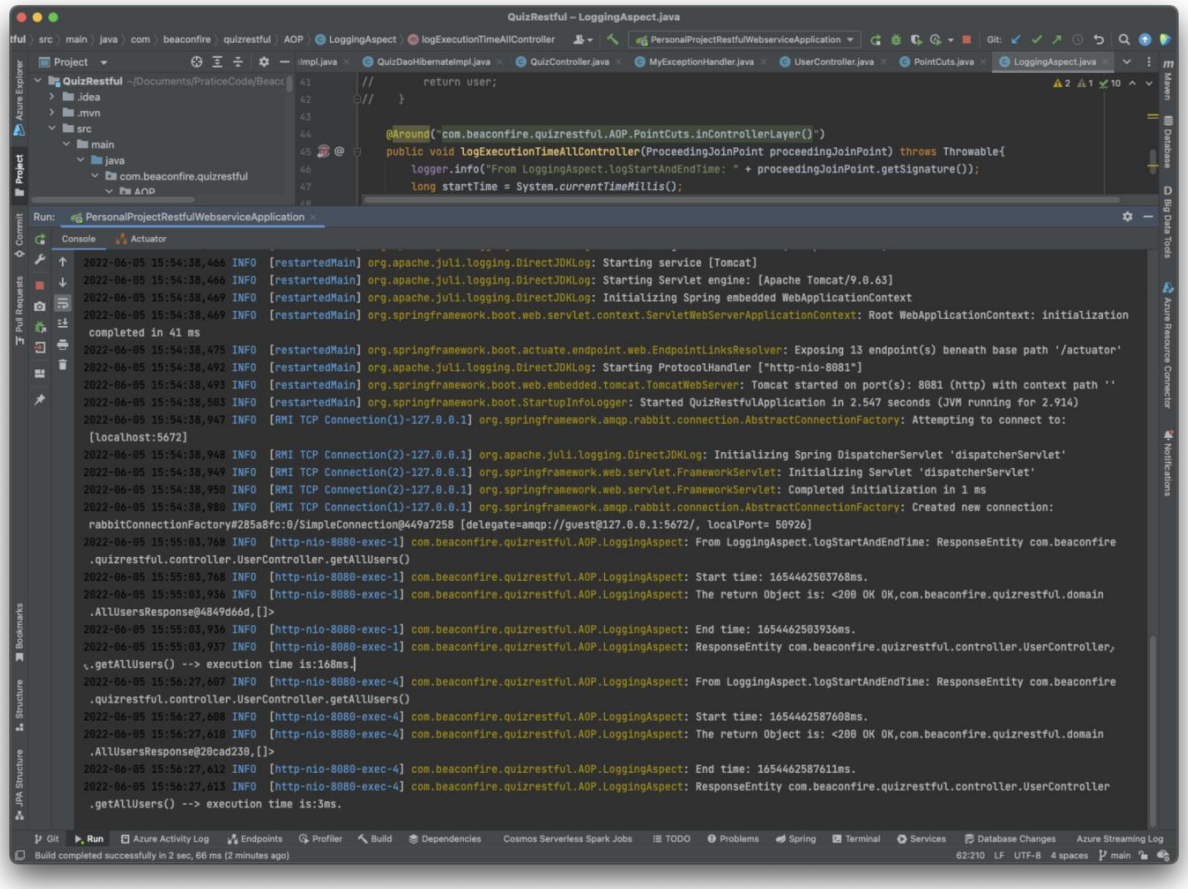

Swagger UI
localhost:8080/swagger-ui.html#/
swagger
Select a spec default
Api Documentation 1.0
[Base URL: localhost:8080/]
http://localhost:8080/v2/api-docs
Api Documentation
Terms of service
Apache 2.0
basic-error-controller Basic Error Controller
owner-controller Owner Controller
GET /api/all-owners Find all owners
POST /api/owner Save or update Owner
GET /api/owner/{id} Find owner by id
DELETE /api/owner/{id} Delete owner by id
PUT /api/owner/{id}/pet Adding a pet to an existing owner with id
Models
Iterable > {...}


Swagger UI
localhost:8080/swagger-ui.html#/owner-controller/addPetToOwnerByIdUsingPUT
Models
Iterable > {...}
ModelAndView > {...}
Owner { id string Owner id name string Owner name pets [List of pets Pet { name string Pet name species string Pet species }] }
Pet { name string Pet name species string Pet species }
View { contentType string }

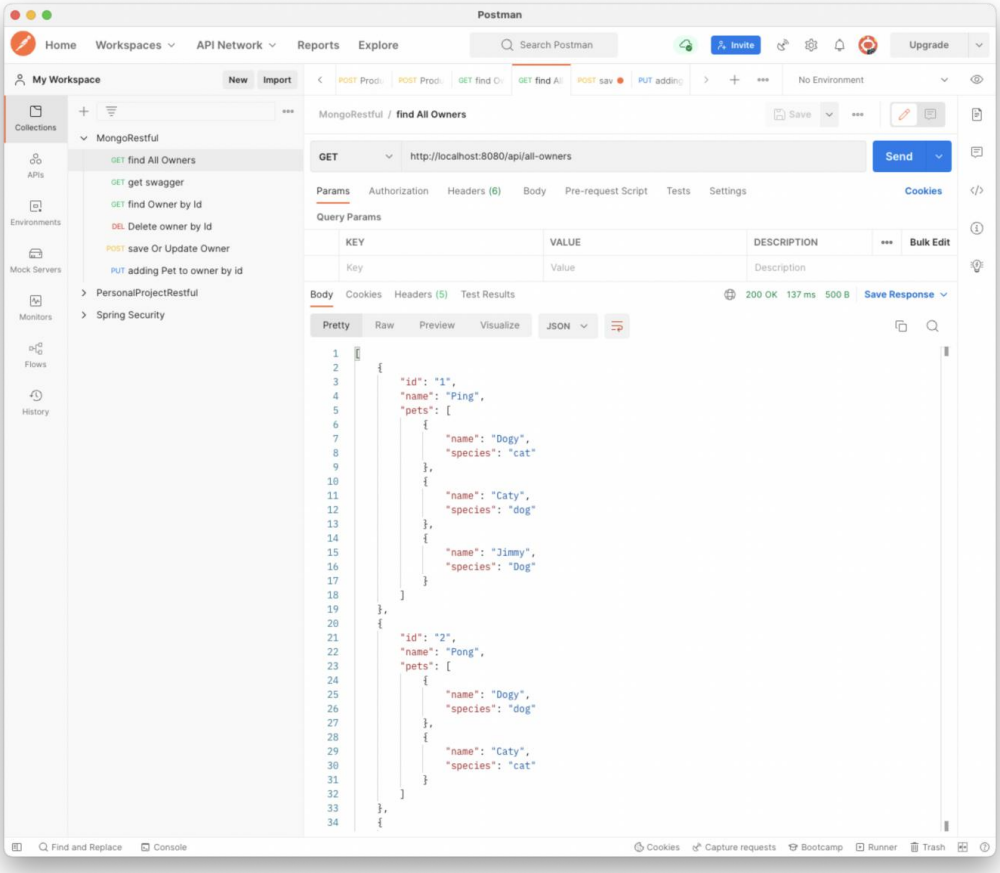

Swagger UI
localhost:8080/swagger-ui.html#/owner-controller/addPetToOwnerByIdUsingPUT
PUT /api/owner/{id}/pet Adding a pet to an existing owner with id
Parameters
Try it out
Name
Description
id * required
string
(path)
id
pet * required
(body)
pet
Example Value | Model
{ "name": "string", "species": "string" }
Parameter content type
application/json
Responses
Response content type
/
Code
Description
200
OK
Example Value | Model
{ "id": "string", "name": "string", "pets": [{ "name": "string", "species": "string" }] }

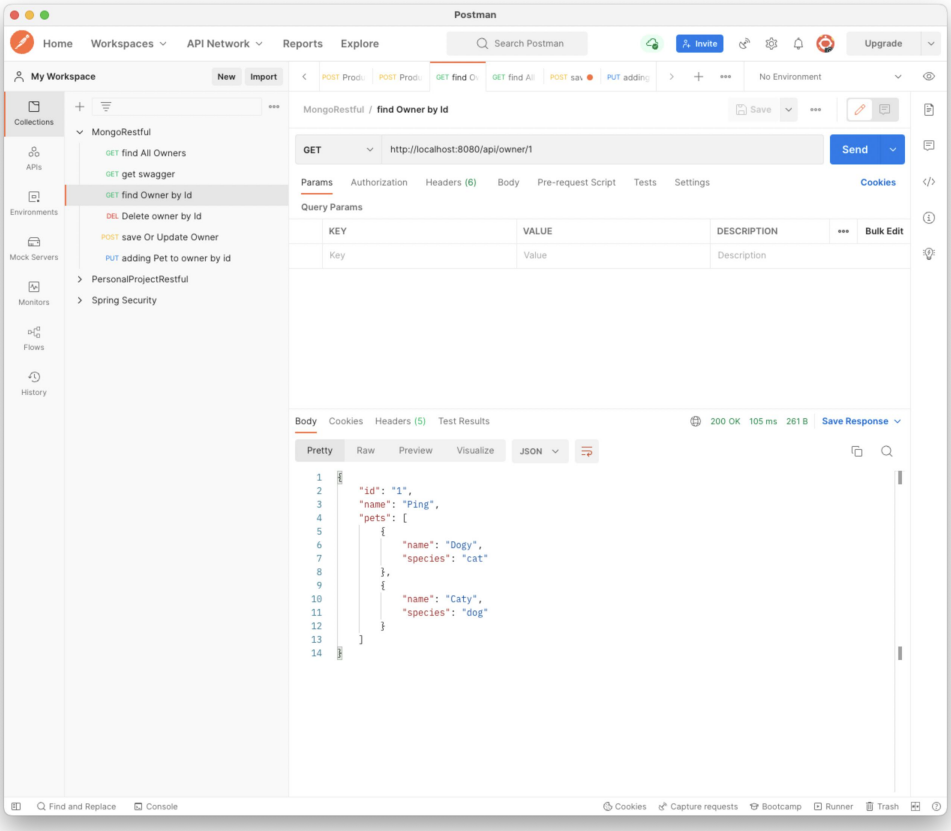
Using Cache before and after comparing Screen short cut:

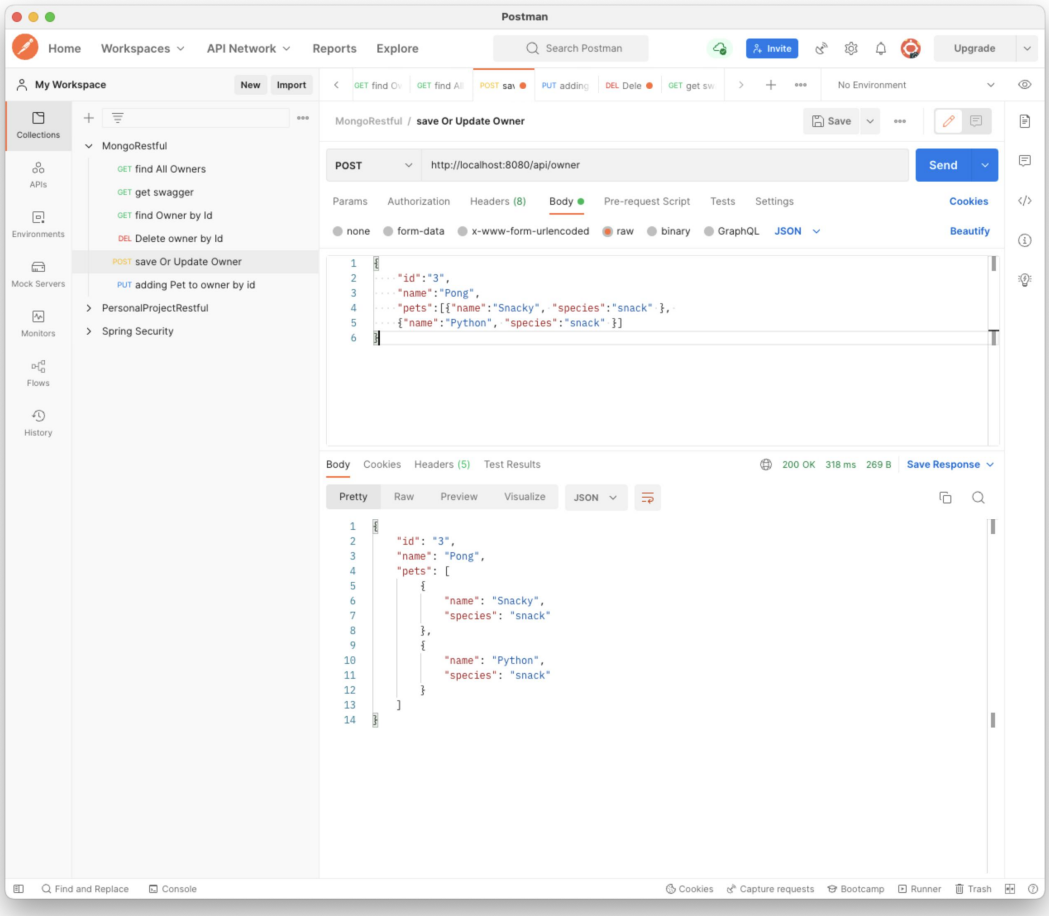


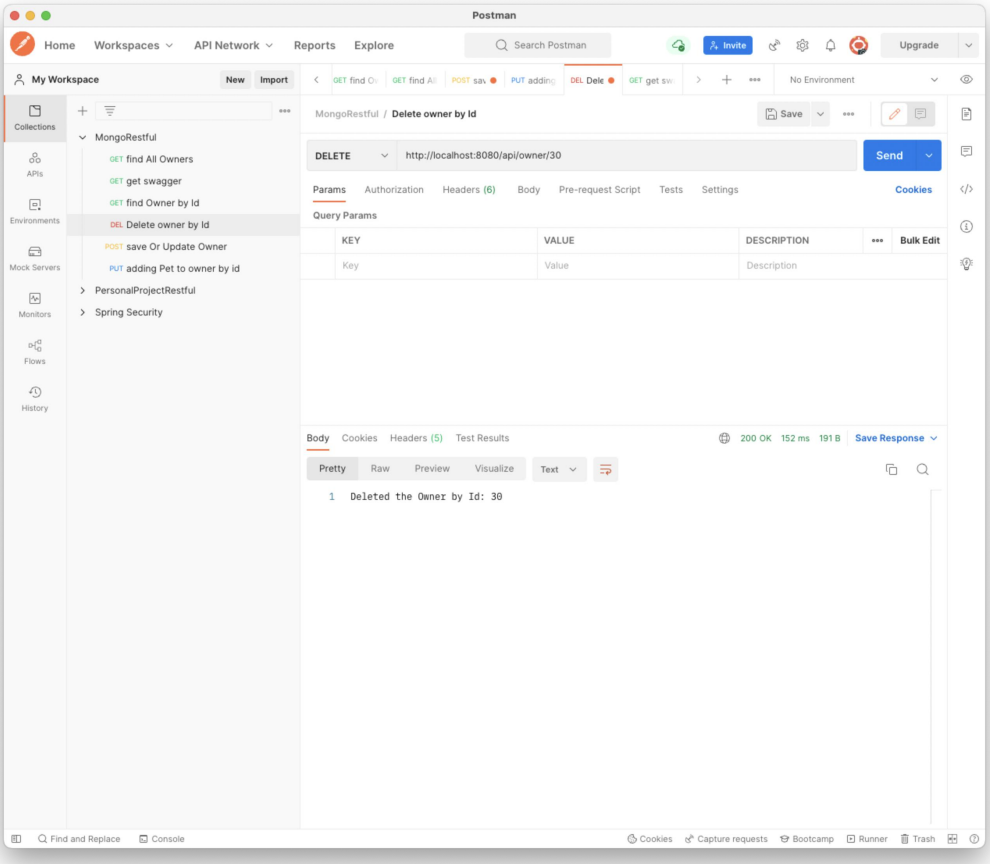


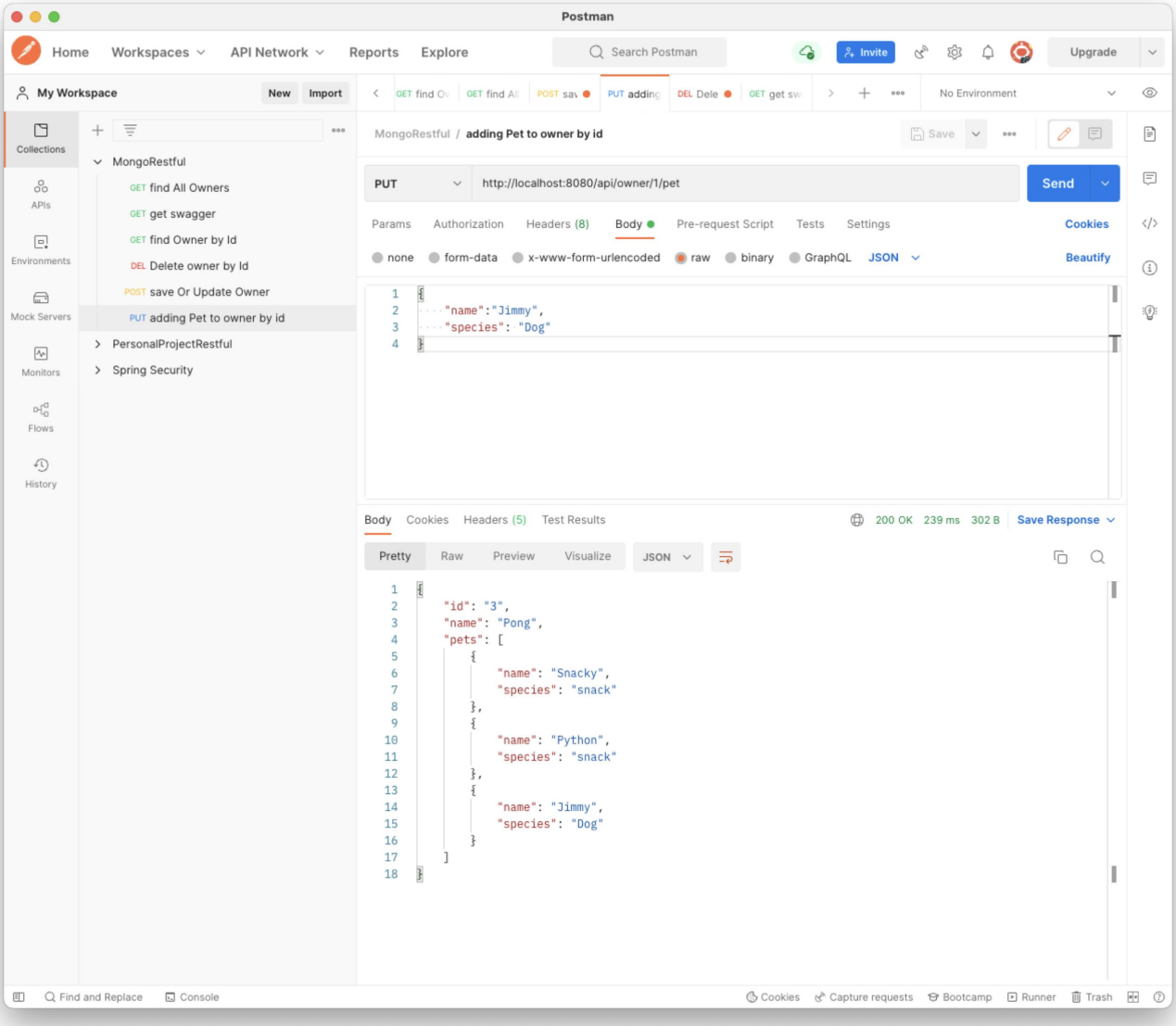
Show MongoDB Restful api Screen short cut:






Postman
Home
Workspaces
API Network
Reports
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
MongoRestful
GET find All Owners
GET get swagger
GET find Owner by Id
DEL Delete owner by Id
POST save Or Update Owner
PUT adding Pet to owner by id
PersonalProjectRestful
Spring Security
No Environment
MongoRestful / save Or Update Owner
Save
POST
http://localhost:8080/api/owner
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id":"3",
"name":"Pong",
"pets":[{"name":"Snacky", "species":"snack" },
{"name":"Python", "species":"snack" }]
}
Body
Cookies
Headers (5)
Test Results
200 OK
318 ms
269 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": "3",
"name": "Pong",
"pets": [
{
"name": "Snacky",
"species": "snack"
},
{
"name": "Python",
"species": "snack"
}
]
}
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash


Postman
Home
Workspaces
API Network
Reports
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
MongoRestful
GET find All Owners
GET get swagger
GET find Owner by Id
DEL Delete owner by Id
POST save Or Update Owner
PUT adding Pet to owner by id
PersonalProjectRestful
Spring Security
No Environment
MongoRestful / Delete owner by Id
Save
DELETE
http://localhost:8080/api/owner/30
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
200 OK
152 ms
191 B
Save Response
Pretty
Raw
Preview
Visualize
Text
Deleted the Owner by Id: 30
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash


Postman
Home
Workspaces
API Network
Reports
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
MongoRestful
GET find All Owners
GET get swagger
GET find Owner by Id
DEL Delete owner by Id
POST save Or Update Owner
PUT adding Pet to owner by id
PersonalProjectRestful
Spring Security
MongoRestful / adding Pet to owner by id
Save
PUT
http://localhost:8080/api/owner/1/pet
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {
2 "name":"Jimmy",
3 "species": "Dog"
4 }
Body
Cookies
Headers (5)
Test Results
200 OK
239 ms
302 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "id": "3",
3 "name": "Pong",
4 "pets": [
5 {
6 "name": "Snacky",
7 "species": "snack"
8 },
9 {
10 "name": "Python",
11 "species": "snack"
12 },
13 {
14 "name": "Jimmy",
15 "species": "Dog"
16 }
17]
18 }
Find and Replace
Console
Cookies
Capture requests
Bootcamp
Runner
Trash