FATIMA JINNAH WOMEN UNIVERSITY
Department of Software Engineering

Software Construction & Development Lab Project

Spring Boot Project

Submitted To: **Engr. Muhammad Shahzad**

Submitted By: **Hira Shakeel(2021-BSE-049)**

Laiba Imran(2021-BSE-072)

Laiba Shahid(2021-BSE-051)

Date: **15th January-2024**

Lab Rubrics

Lab Title:	Experiment # :
Student Name:	Regd. #:
Experiment Name:	

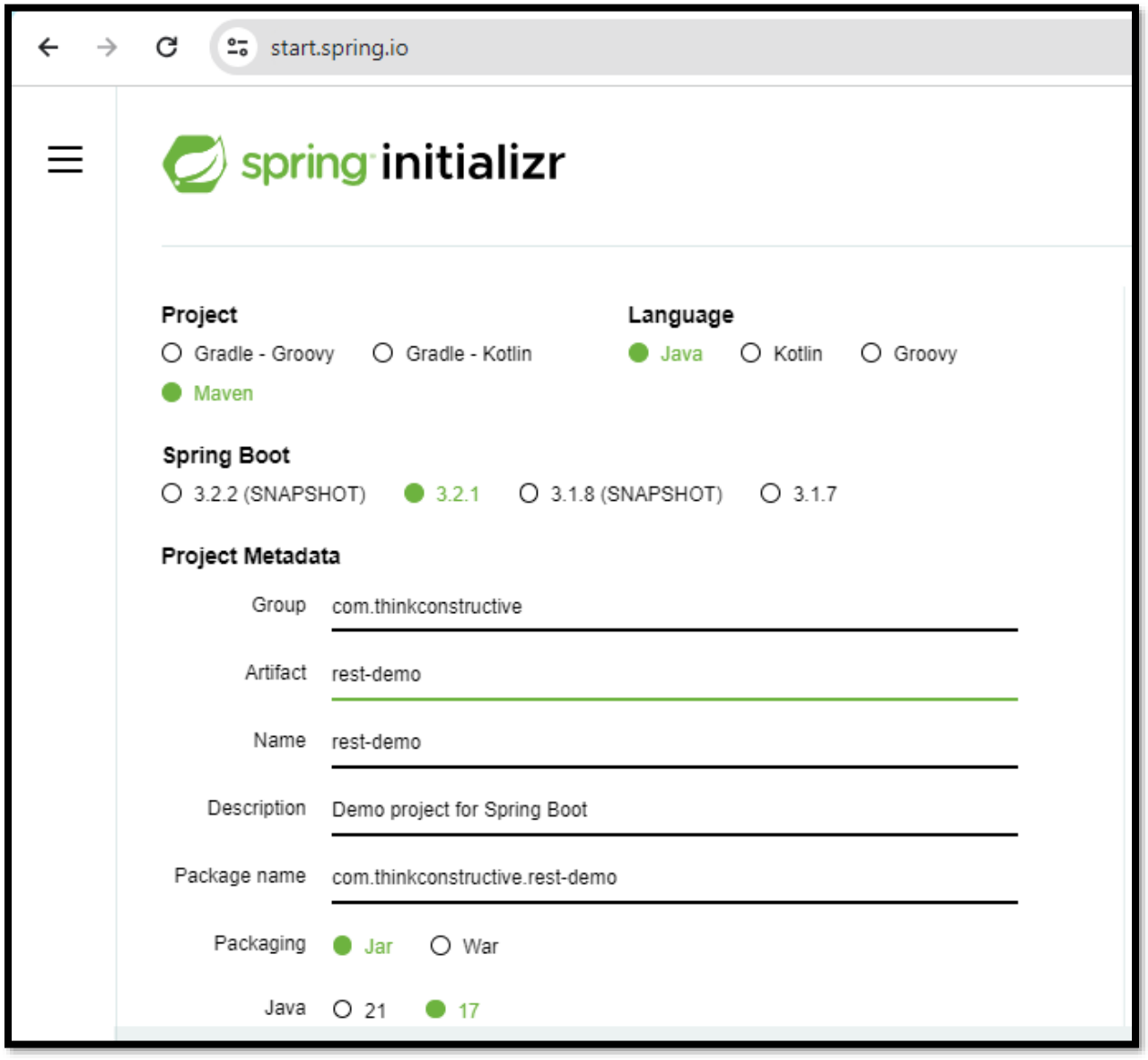
Performance Indicator	Level of Achievements				
	Excellent (5)	Good (4)	Average (3)	Below Average (2)	Poor (1)
Understanding & implementation Problem					
Report and presentation					
Maximum Marks		10	Obtained Marks		
Lab Instructor:					

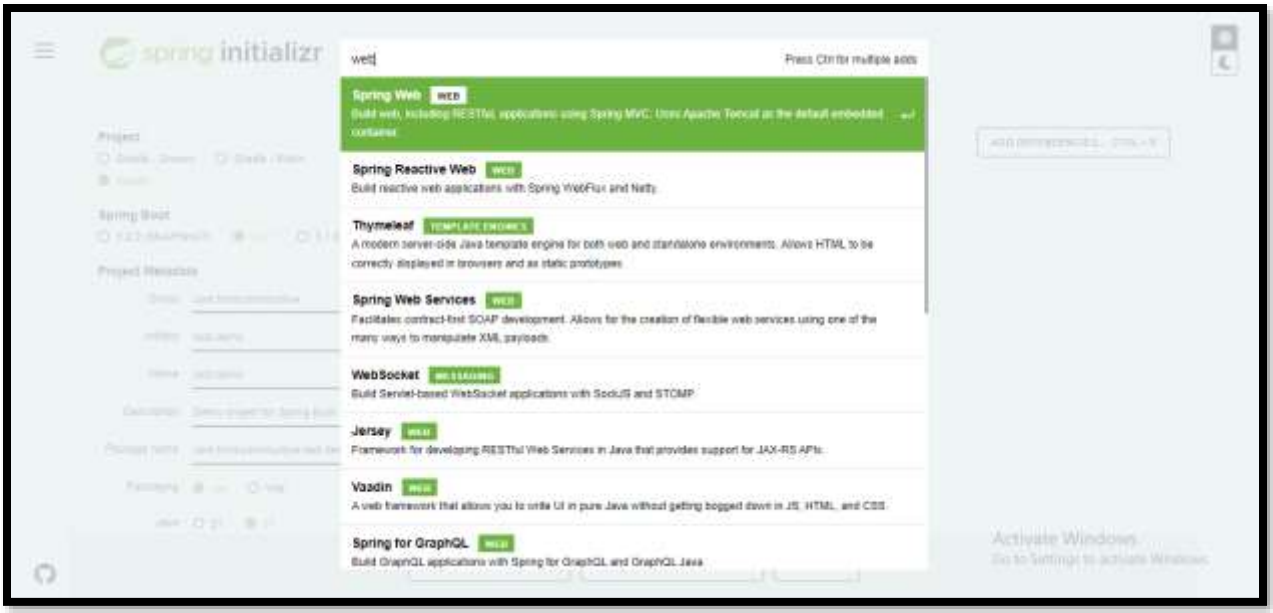
Contents

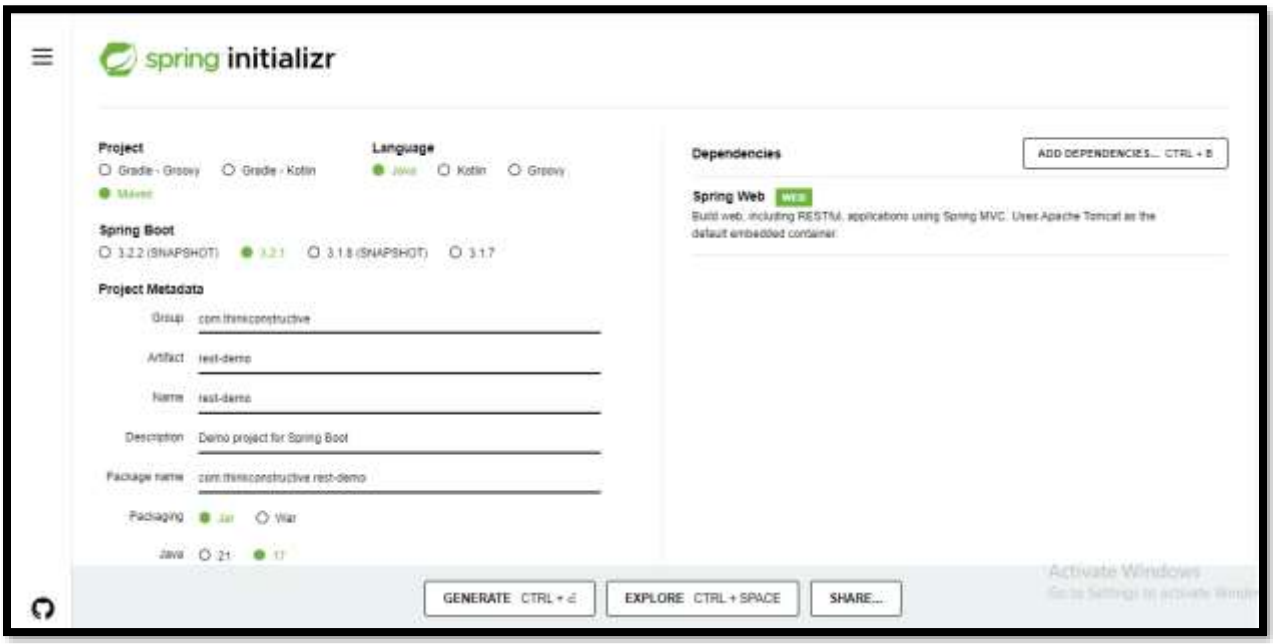
Session# 01

"Java REST API with Spring Boot Tutorial | REST API CRUD Implementation"

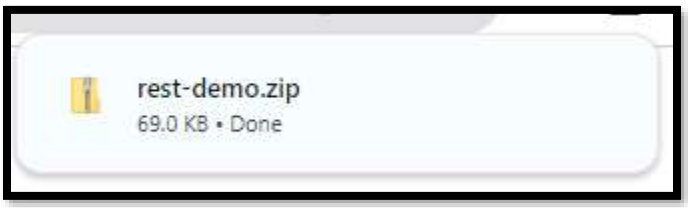
- Creating basic spring boot application.



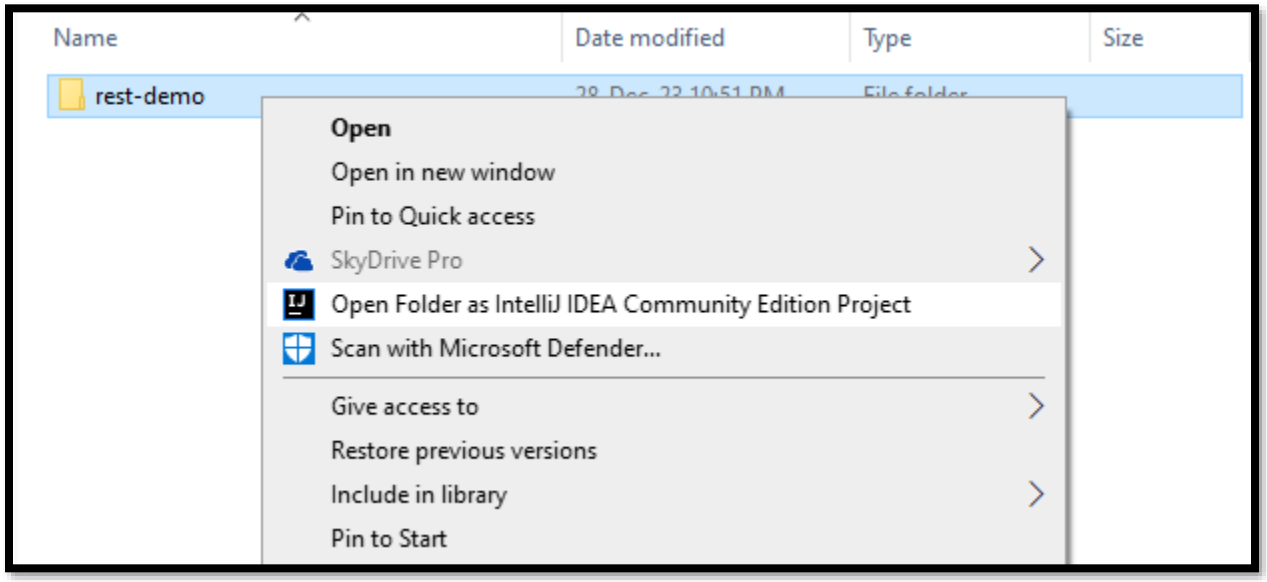




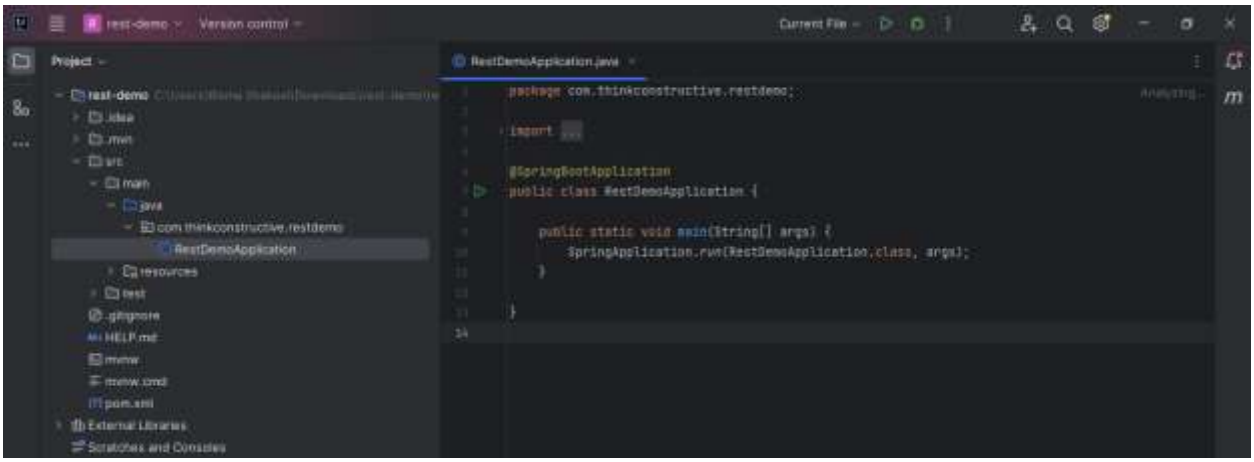
- Now Click on generate



- Save the zip and extract in any java editor, I will do in IntelliJ IDEA editor.



- Rest Demo Application

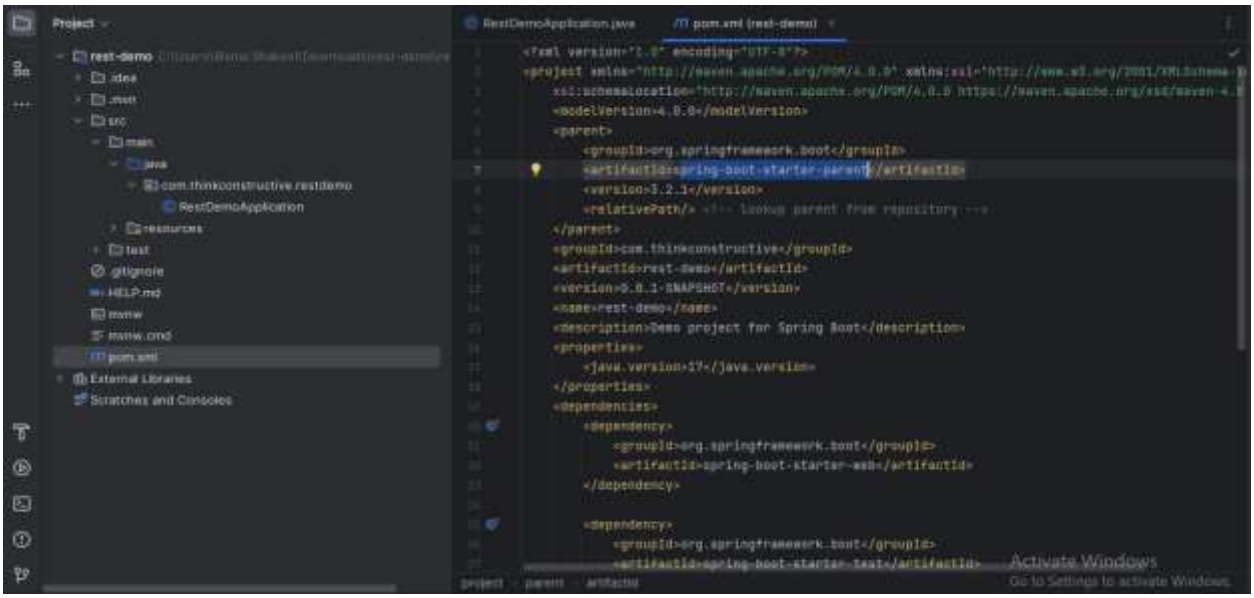


Output

- My Application is successfully started.

```
main] c.t.restdemo.RestDemoApplication        : Starting RestDemoApplication using Java 21 with PID 23904 (C:\Users\Bilal Shakeel\Downloads\rest-demo\r
main] c.t.restdemo.RestDemoApplication        : No active profile set, falling back to 1 default profile: "default"
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
main] o.apache.catalina.core.StandardService  : Starting service [Tomcat]
main] o.apache.catalina.core.StandardEngine   : Starting Servlet engine: [Apache Tomcat/10.1.17]
main] o.a.c.c.C.[Tomcat].[localhost].[/]      : Initializing Spring embedded WebApplicationContext
main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 8013 ms
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
main] c.t.restdemo.RestDemoApplication        : Started RestDemoApplication in 15.095 seconds (process running for 17.864)
```

- Dependencies we are getting.



- Check whether my application is actually working on port 8080 or not?

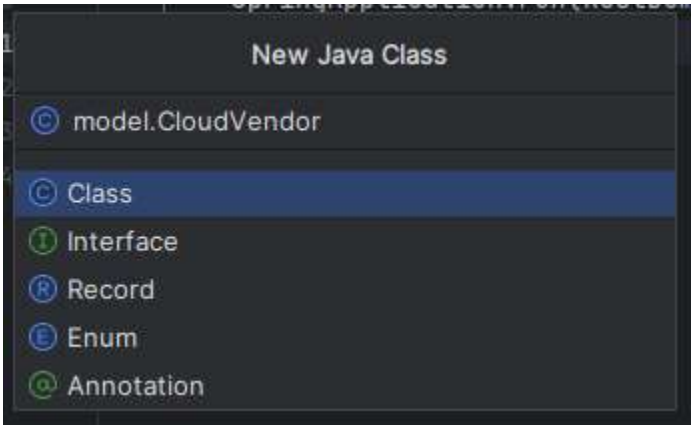
Whitelabel Error Page

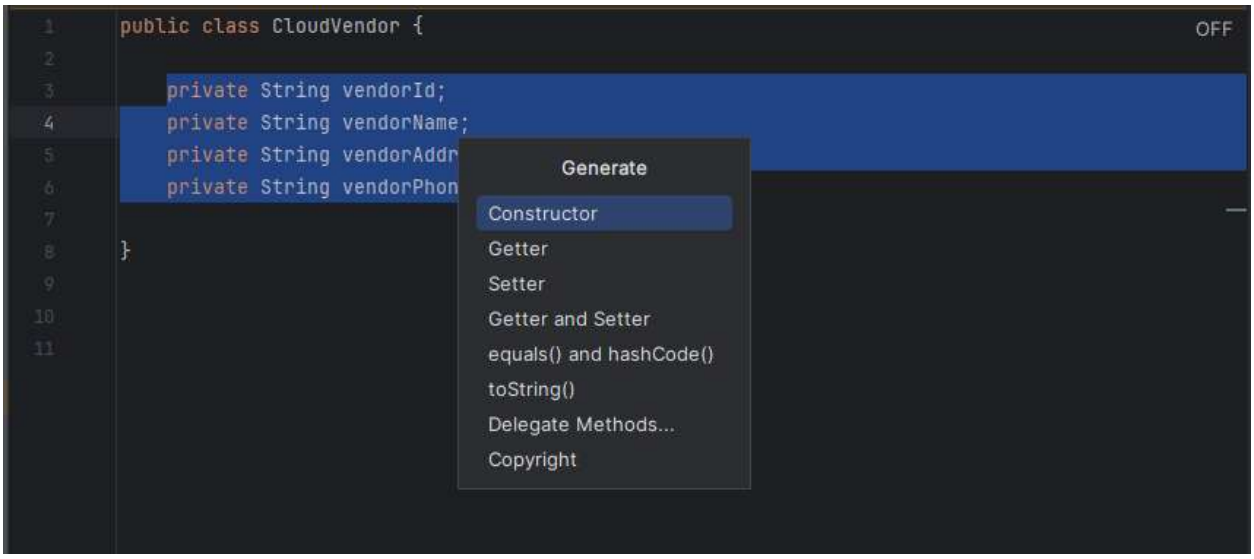
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Thu Dec 28 23:40:10 PKT 2023
There was an unexpected error (type=Not Found, status=404).

Making Rest API's

Cloud vendor API






Choose Fields to Initialize by Constructor
CloudVendor
vendorId:String
vendorName:String
vendorAddress:String
vendorPhoneNumber:String
OK
Select None
Cancel


Generate
Constructor
Getter
Setter
Getter and Setter
equals() and hashCode()
toString()
Delegate Methods...
Copyright

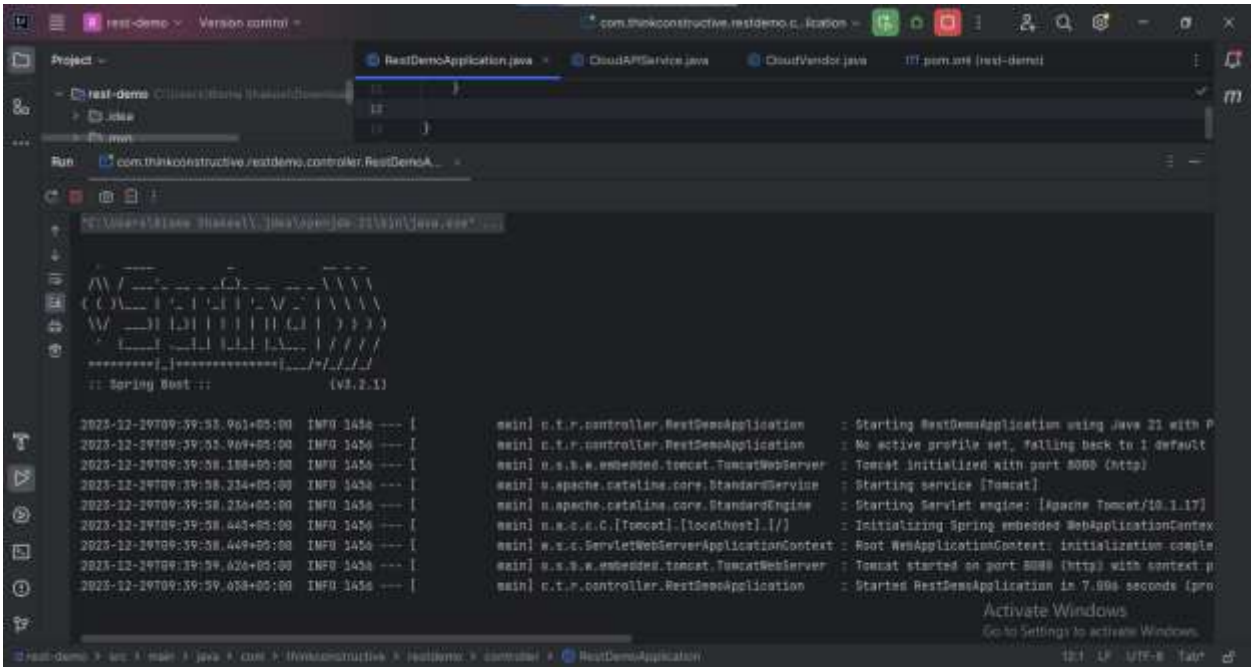
Code:

```
package com.thinkconstructive.restdemo.controller;

public class CloudVendor {

    private String vendorId;
    private String vendorName;
    private String vendorAddress;
    private String vendorPhoneNumber;

    public String getVendorId() {
        return vendorId;
    }
```

```
    public void setVendorId(String vendorId) {
        this.vendorId = vendorId;
    }

    public String getVendorName() {
        return vendorName;
    }

    public void setVendorName(String vendorName) {
        this.vendorName = vendorName;
    }

    public String getVendorAddress() {
        return vendorAddress;
    }

    public void setVendorAddress(String vendorAddress) {
        this.vendorAddress = vendorAddress;
    }

    public String getVendorPhoneNumber() {
        return vendorPhoneNumber;
    }

    public void setVendorPhoneNumber(String vendorPhoneNumber) {
        this.vendorPhoneNumber = vendorPhoneNumber;
    }



    public CloudVendor() {
    }

    public CloudVendor(String vendorId, String vendorName, String
vendorAddress, String vendorPhoneNumber) {
        this.vendorId = vendorId;
        this.vendorName = vendorName;
        this.vendorAddress = vendorAddress;
        this.vendorPhoneNumber = vendorPhoneNumber;
    }

}
```

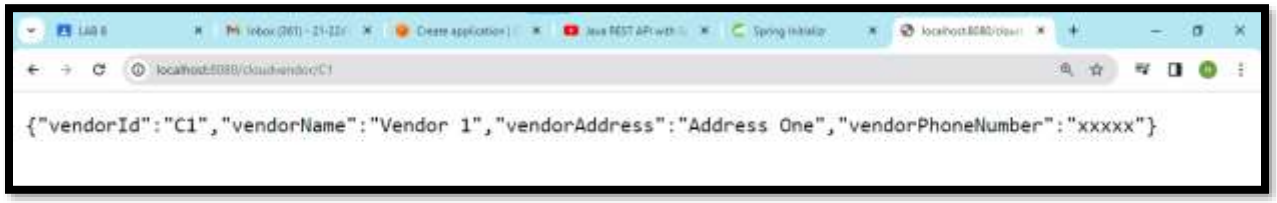
RestDemoApplication.java | CloudAPIService.java | CloudVendor.java | pom.xml (rest-demo)

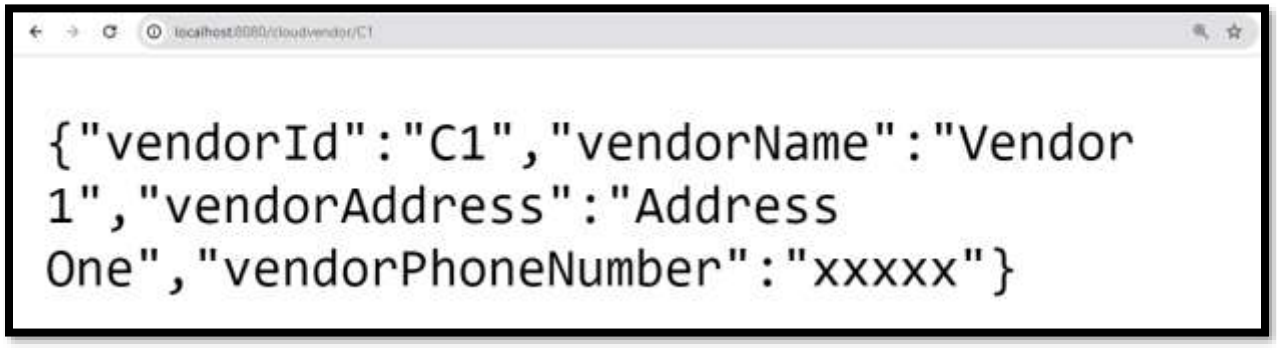
```java
package com.thinkconstructive.restdemo.controller.controller;


import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

no usages
@RestController
@RequestMapping("/cloudvendor")
public class CloudAPIService {
no usages
@GetMapping("{vendorId}")
    public CloudVendor getCloudVendorDetails(String vendorId)
    {
        return new CloudVendor( vendorId: "C1", vendorName: "Vendor 1",
                vendorAddress: "Address One", vendorPhoneNumber: "xxxxx");
    }
}

```



Cloud Vendor Properties:

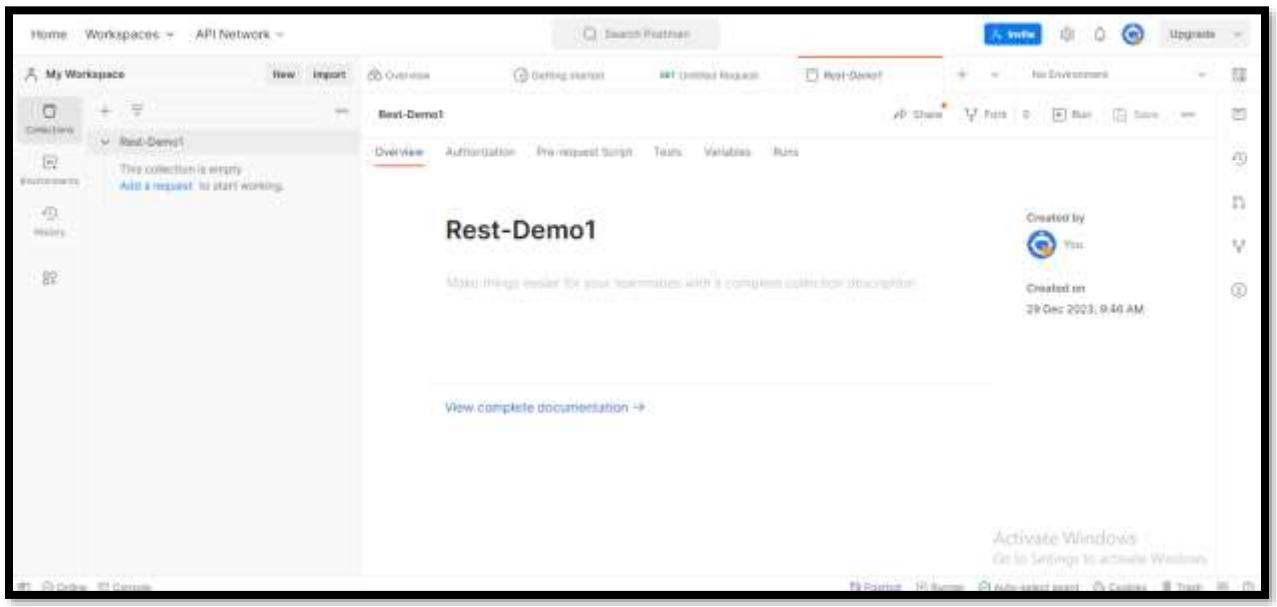
- Vendor ID
- Vendor Name
- Vendor Address
- Vendor Phone Number

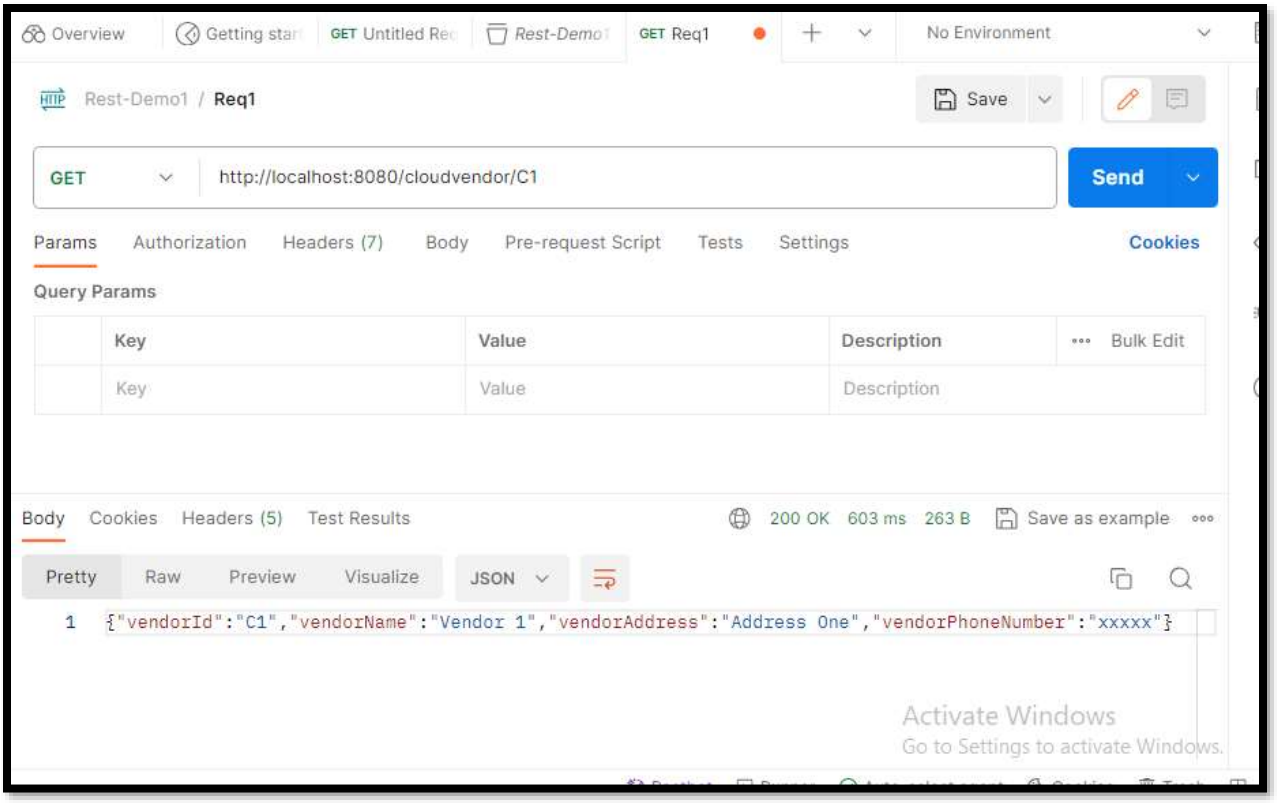




Postman

- Go to Postman

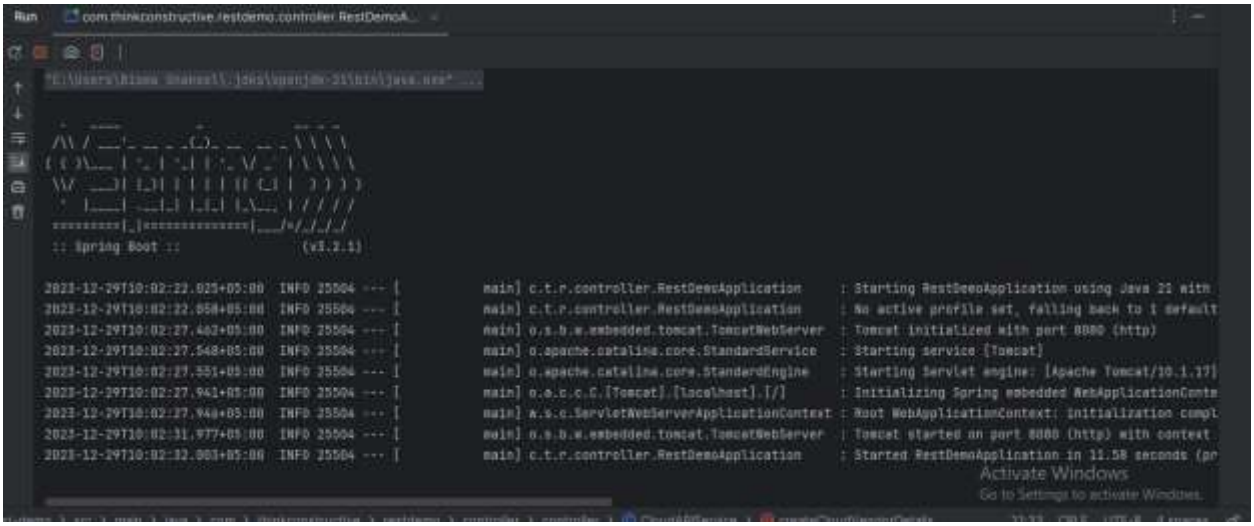


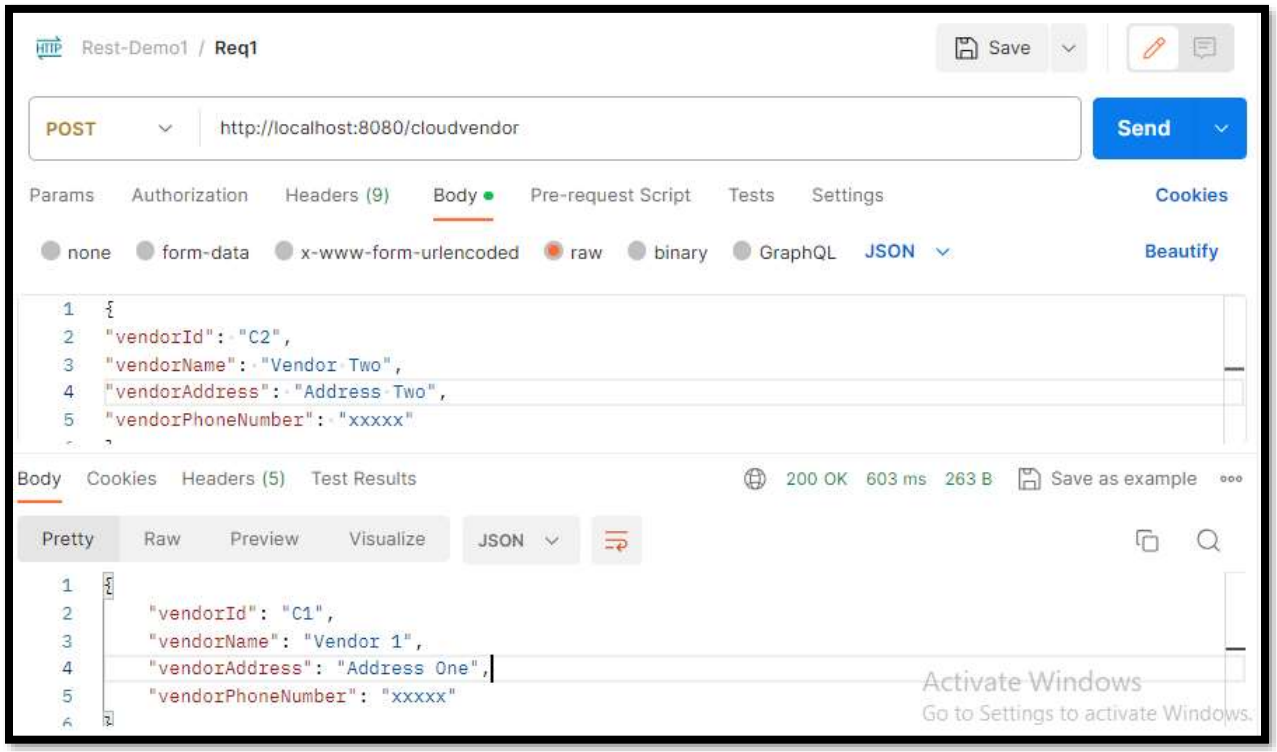


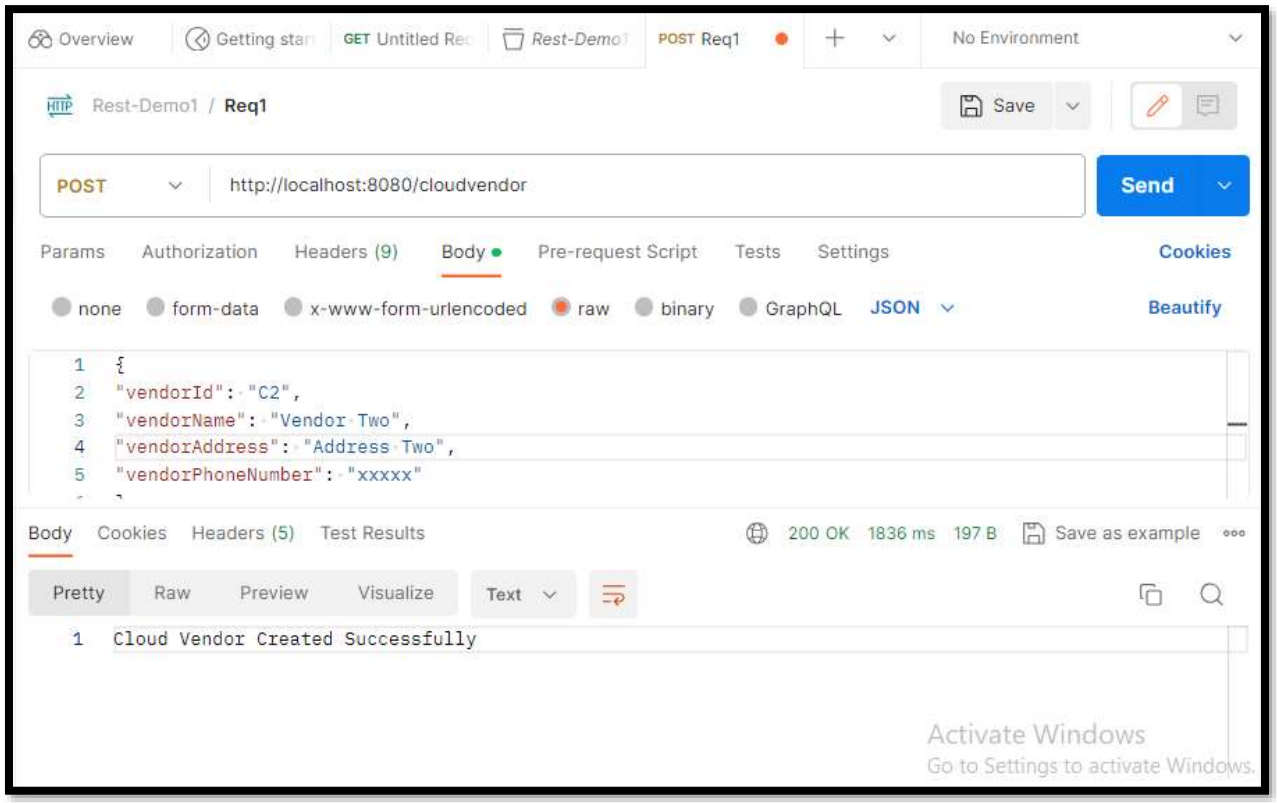
Code:

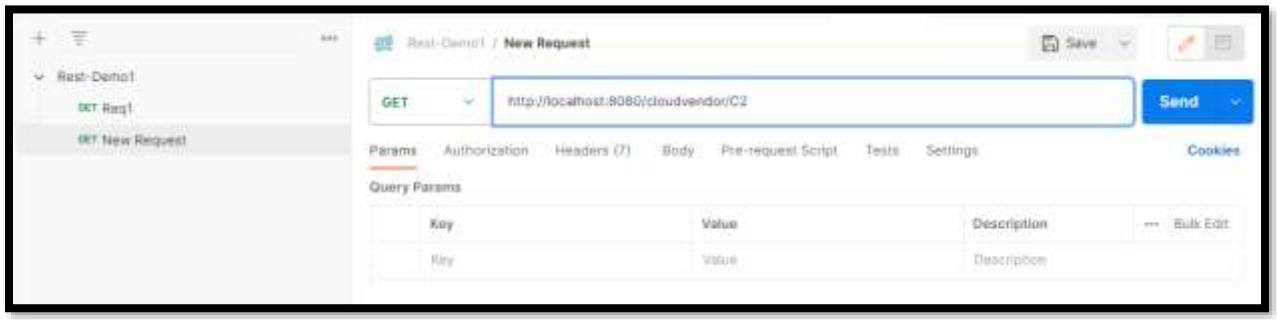
CloudAPIService

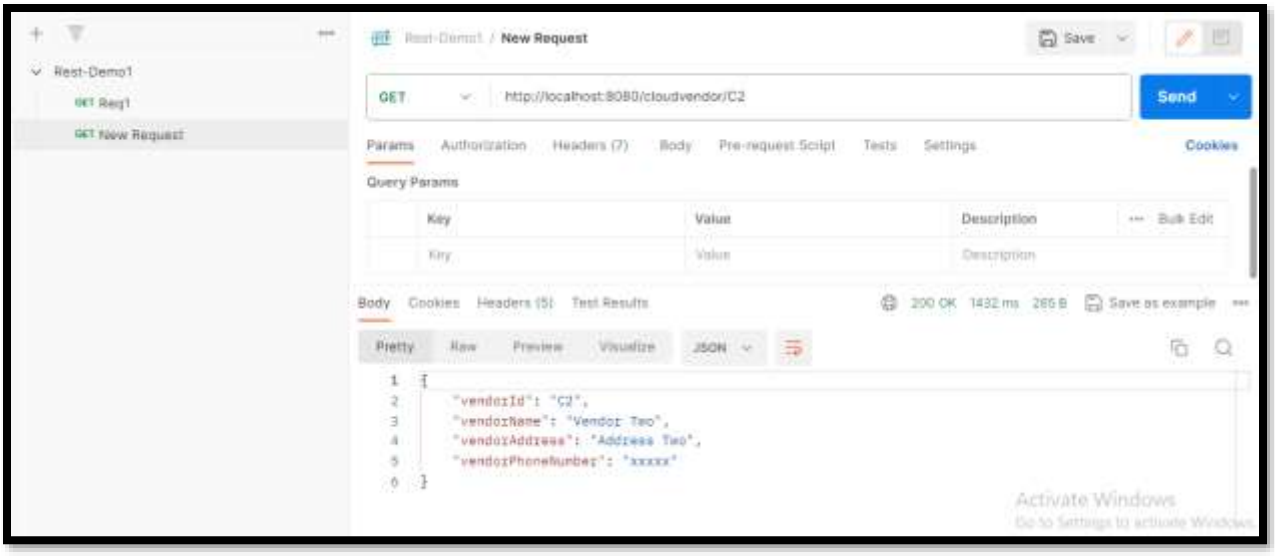
```
package com.thinkconstructive.restdemo.controller.controller;


import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/cloudvendor")
public class CloudAPIService {
    CloudVendor cloudVendor;
@GetMapping("{vendorId}")
    public CloudVendor getCloudVendorDetails(String vendorId)
    {
        return cloudVendor;
                //new CloudVendor("C1","Vendor 1",
                //"Address One","xxxxx");
    }
    @PostMapping
    public String createCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        this.cloudVendor = cloudVendor;
        return "Cloud Vendor Created Successfully";
    }
}
```


com.thinkconstructive.restdemo.controller.RestDemoA...
:: Spring Boot ::
(v3.2.1)
2023-12-29T10:02:22.025+05:00 INFO 25504 --- [main] c.t.r.controller.RestDemoApplication : Starting RestDemoApplication using Java 21 with
2023-12-29T10:02:22.050+05:00 INFO 25504 --- [main] c.t.r.controller.RestDemoApplication : No active profile set, falling back to 1 default
2023-12-29T10:02:27.462+05:00 INFO 25504 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2023-12-29T10:02:27.548+05:00 INFO 25504 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-12-29T10:02:27.551+05:00 INFO 25504 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.17]
2023-12-29T10:02:27.941+05:00 INFO 25504 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-12-29T10:02:27.944+05:00 INFO 25504 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization compl
2023-12-29T10:02:31.977+05:00 INFO 25504 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context
2023-12-29T10:02:32.003+05:00 INFO 25504 --- [main] c.t.r.controller.RestDemoApplication : Started RestDemoApplication in 11.58 seconds (pr
Activate Windows
Go to Settings to activate Windows.


Rest-Demo1 / Req1
Save
POST
http://localhost:8080/cloudvendor
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"vendorId": "C2",
"vendorName": "Vendor Two",
"vendorAddress": "Address Two",
"vendorPhoneNumber": "xxxxx"
}
Body
Cookies
Headers (5)
Test Results
200 OK
603 ms
263 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"vendorId": "C1",
"vendorName": "Vendor 1",
"vendorAddress": "Address One",
"vendorPhoneNumber": "xxxxx"
}
Activate Windows
Go to Settings to activate Windows.


Overview
Getting star
GET Untitled Re
Rest-Demo1
POST Req1
No Environment
Rest-Demo1 / Req1
Save
POST
http://localhost:8080/cloudvendor
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {
2 "vendorId": "C2",
3 "vendorName": "Vendor Two",
4 "vendorAddress": "Address Two",
5 "vendorPhoneNumber": "xxxxx"
Body
Cookies
Headers (5)
Test Results
200 OK
1836 ms
197 B
Save as example
Pretty
Raw
Preview
Visualize
Text
1 Cloud Vendor Created Successfully
Activate Windows
Go to Settings to activate Windows.

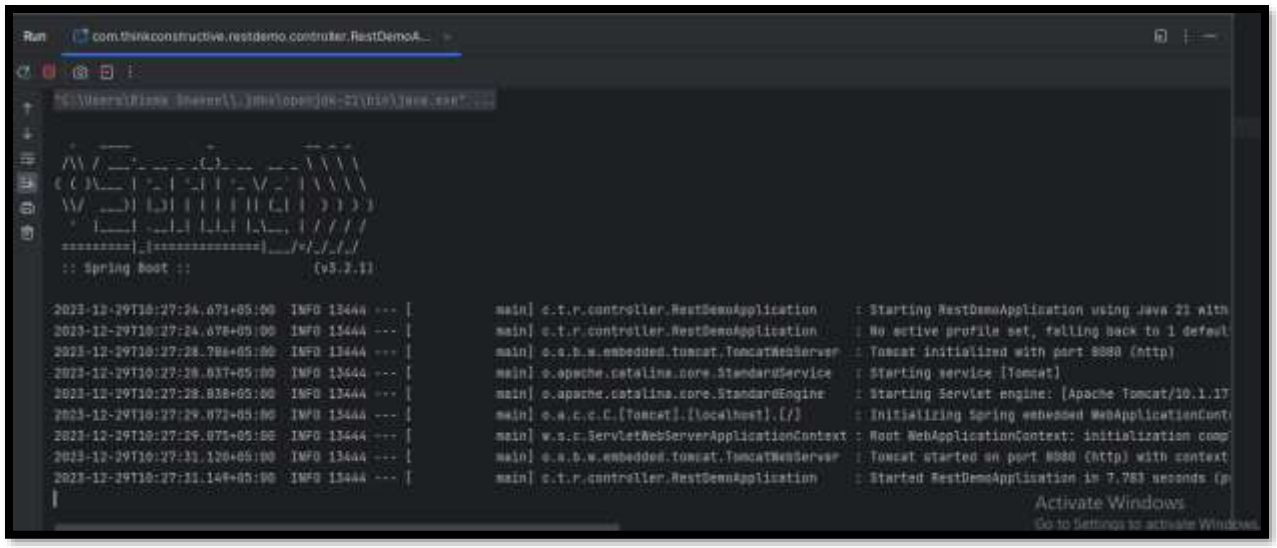

Rest-Demo1
GET Req1
GET New Request
Rest-Demo1 / New Request
Save
GET
http://localhost:8080/cloudvendor/C2
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit

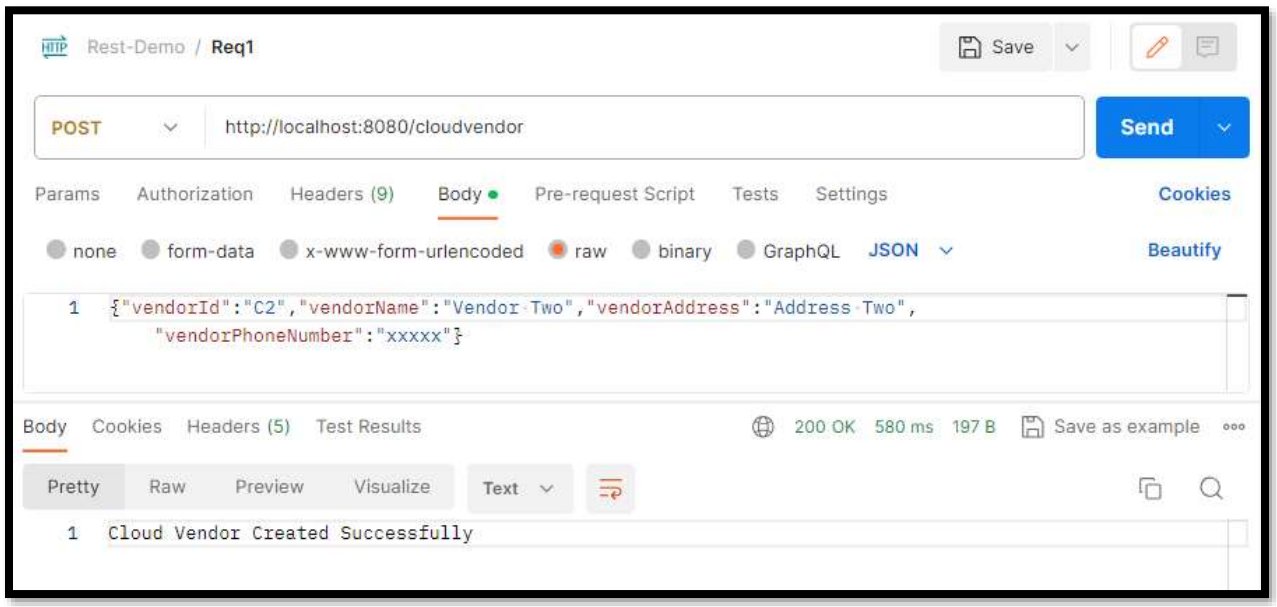


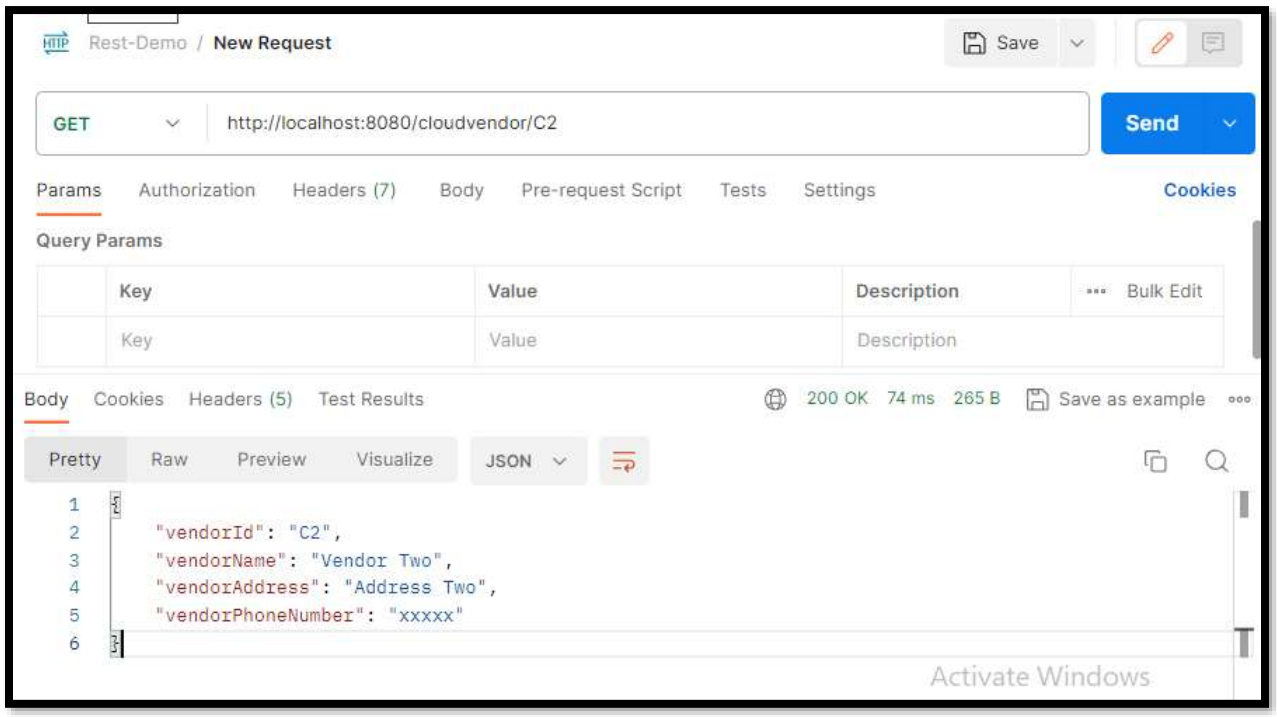
Code:

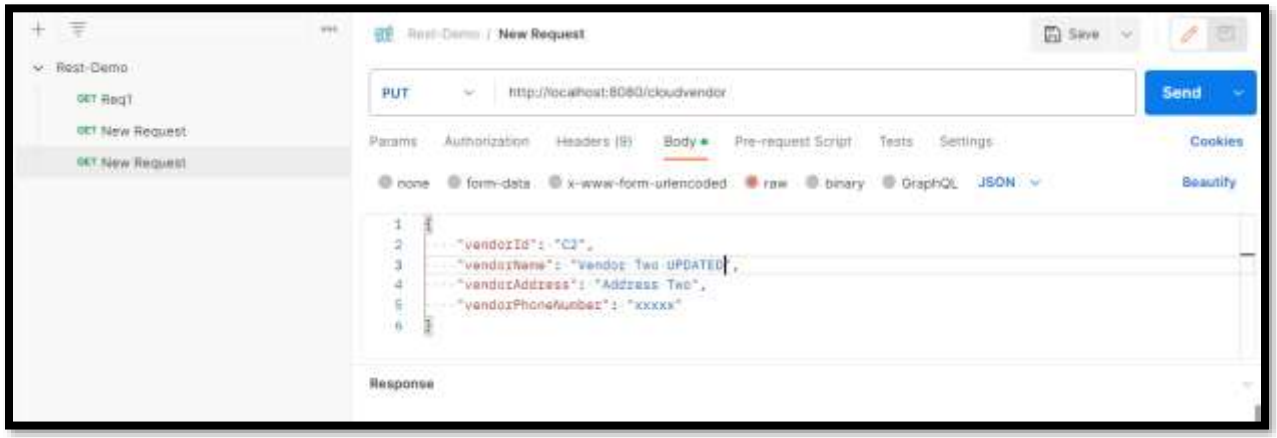
```
package com.thinkconstructive.restdemo.controller.controller;


import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/cloudvendor")
public class CloudAPIService {
    CloudVendor cloudVendor;
@GetMapping("{vendorId}")
    public CloudVendor getCloudVendorDetails(String vendorId)
    {
        return cloudVendor;
                //new CloudVendor("C1","Vendor 1",
                //"Address One","xxxxx");
    }
    @PostMapping
    public String createCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        this.cloudVendor = cloudVendor;
        return "Cloud Vendor Created Successfully";
    }
    @PutMapping
    public String updateCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        this.cloudVendor = cloudVendor;
        return "Cloud Vendor Updated Successfully";
    }
}
```

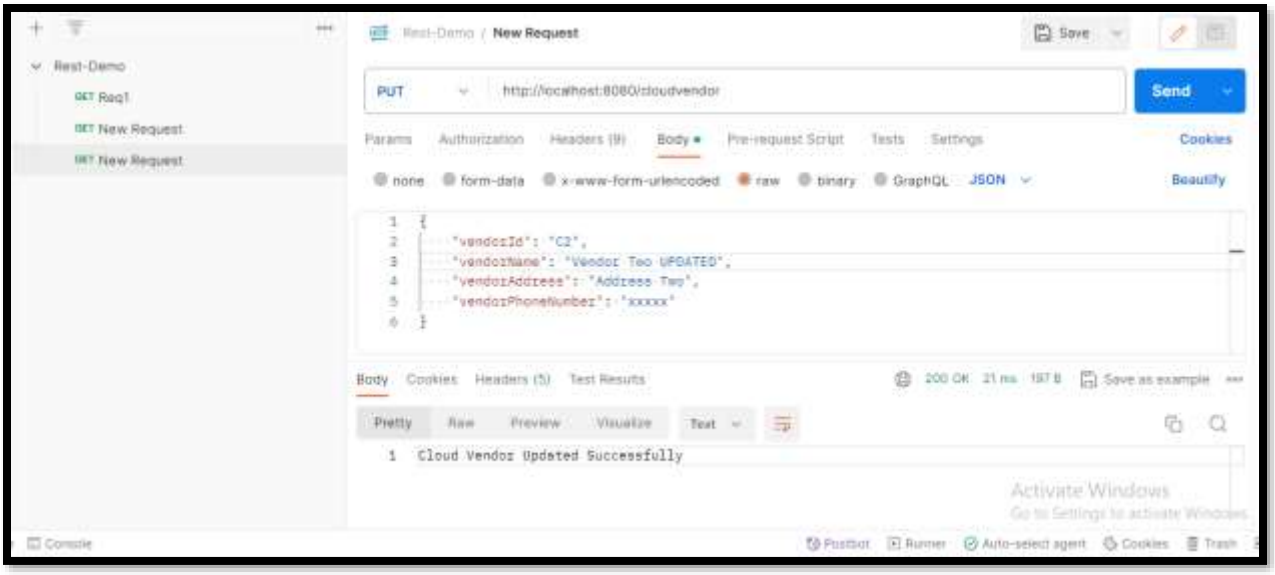
Output:


Run
com.thinkconstructive.restdemo.controller.RestDemoA...
:: Spring Boot ::
Activate Windows

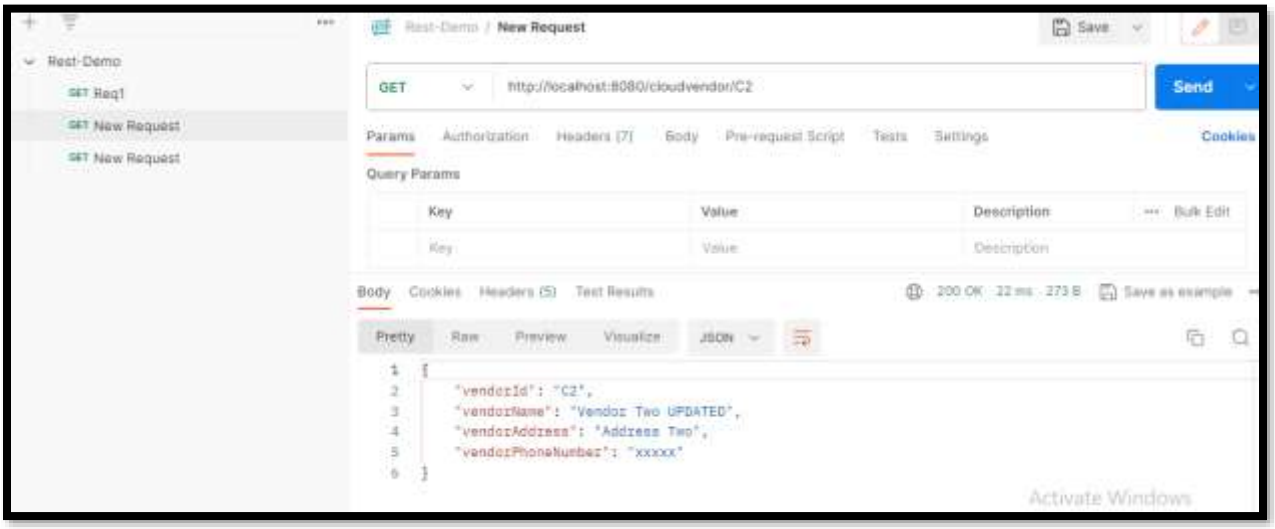

Rest-Demo / Req1
Save
POST
http://localhost:8080/cloudvendor
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{"vendorId":"C2","vendorName":"Vendor Two","vendorAddress":"Address Two",
"vendorPhoneNumber":"xxxxx"}
Body
Cookies
Headers (5)
Test Results
200 OK
580 ms
197 B
Save as example
Pretty
Raw
Preview
Visualize
Text
Cloud Vendor Created Successfully


Rest-Demo / New Request
Save
GET
http://localhost:8080/cloudvendor/C2
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
200 OK
74 ms
265 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"vendorId": "C2",
"vendorName": "Vendor Two",
"vendorAddress": "Address Two",
"vendorPhoneNumber": "xxxxx"
}
Activate Windows


Rest-Demo
GET Req1
GET New Request
GET New Request
New Request
Save
PUT
http://localhost:8080/cloudvendor
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"vendorId": "C2",
"vendorName": "Vendor Two UPDATED",
"vendorAddress": "Address Two",
"vendorPhoneNumber": "xxxxx"
}
Response



- Now We will get the updated one.

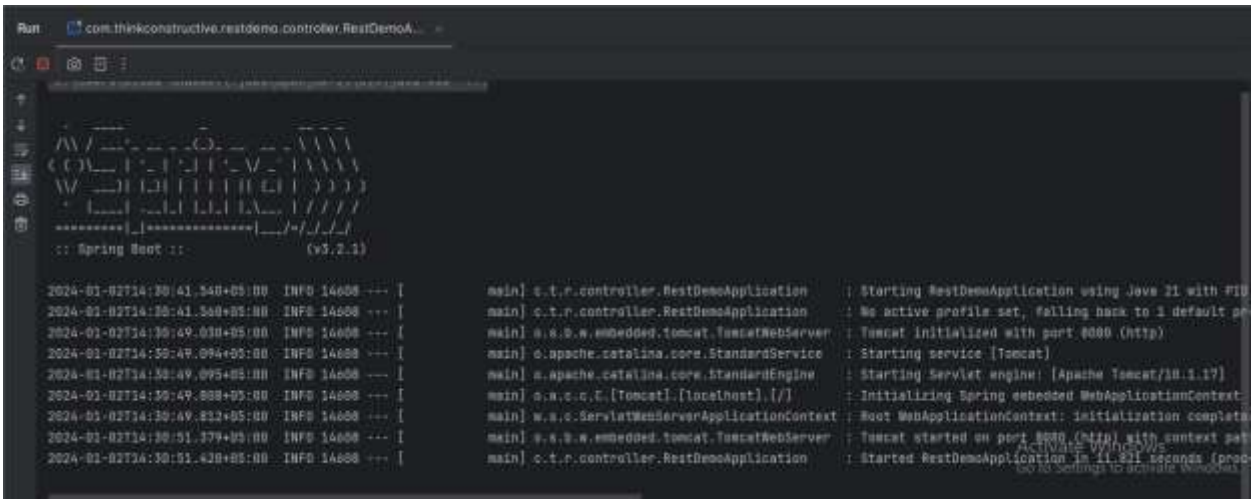


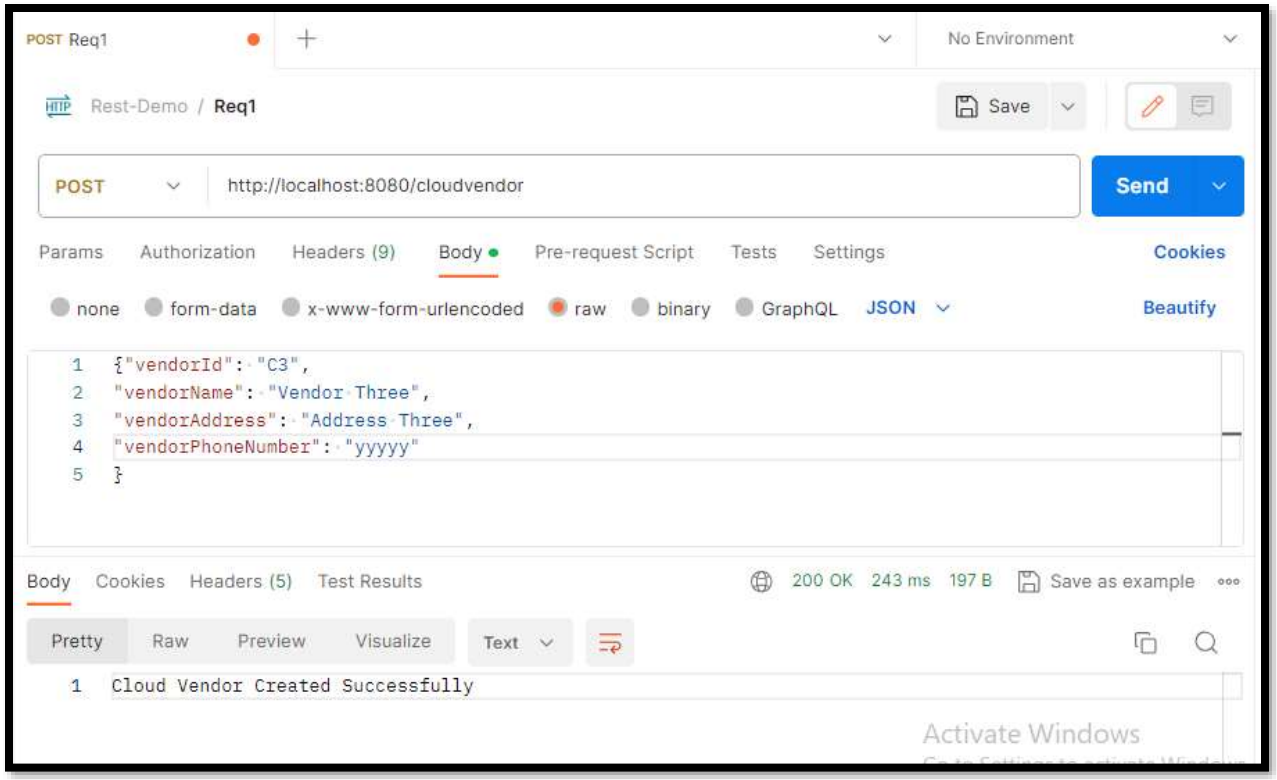
Creating Delete API

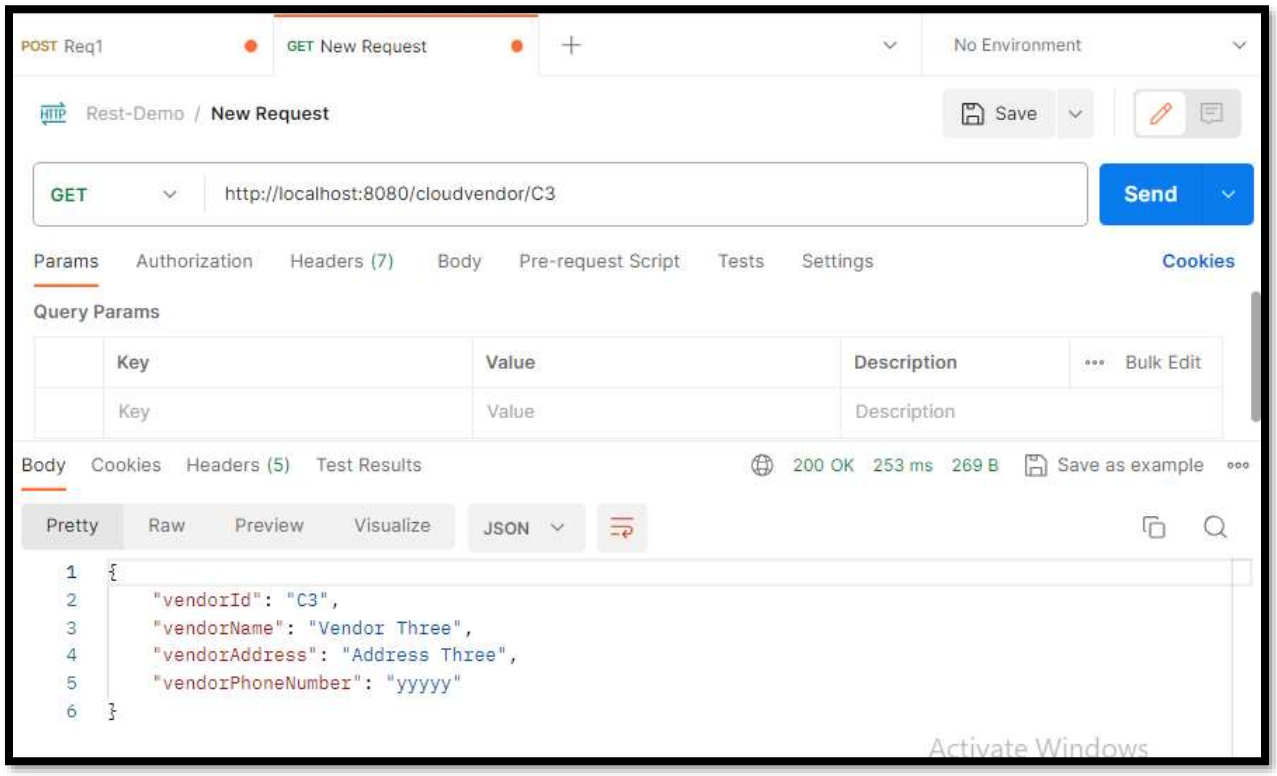
Code

```
package com.thinkconstructive.restdemo.controller.controller;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/cloudvendor")

public class CloudAPIService
{  CloudVendor cloudVendor;
    @GetMapping("{vendorId}")
```

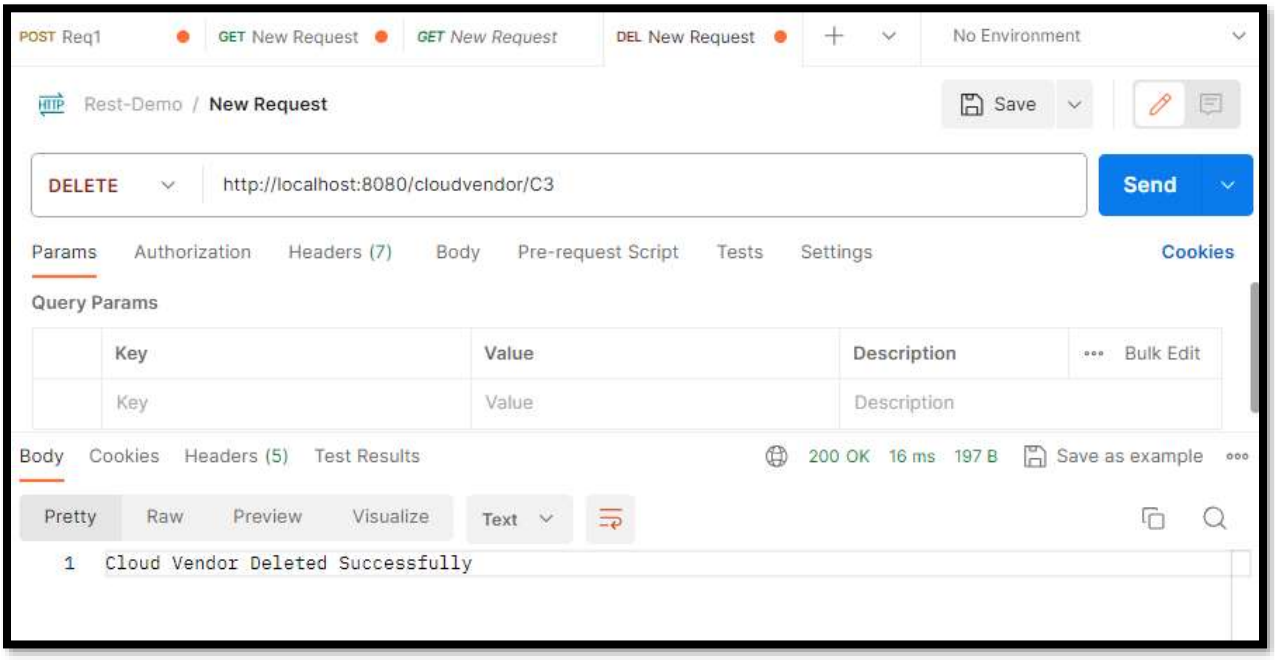
```
    public CloudVendor getCloudVendorDetails(String vendorId)
    {
        return cloudVendor;
        //new CloudVendor("C1","Vendor 1",
        //"Address One","xxxxx");
    }
    @PostMapping
    public String createCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        this.cloudVendor = cloudVendor;
        return "Cloud Vendor Created Successfully";
    }
    @PutMapping
    public String updateCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        this.cloudVendor = cloudVendor;
        return "Cloud Vendor Updated Successfully";
    }
    @DeleteMapping("{vendorId}")
    public String deleteCloudVendorDetails(String vendorId)
    {
        this.cloudVendor = null;
        return "Cloud Vendor Deleted Successfully";
    }
}
```

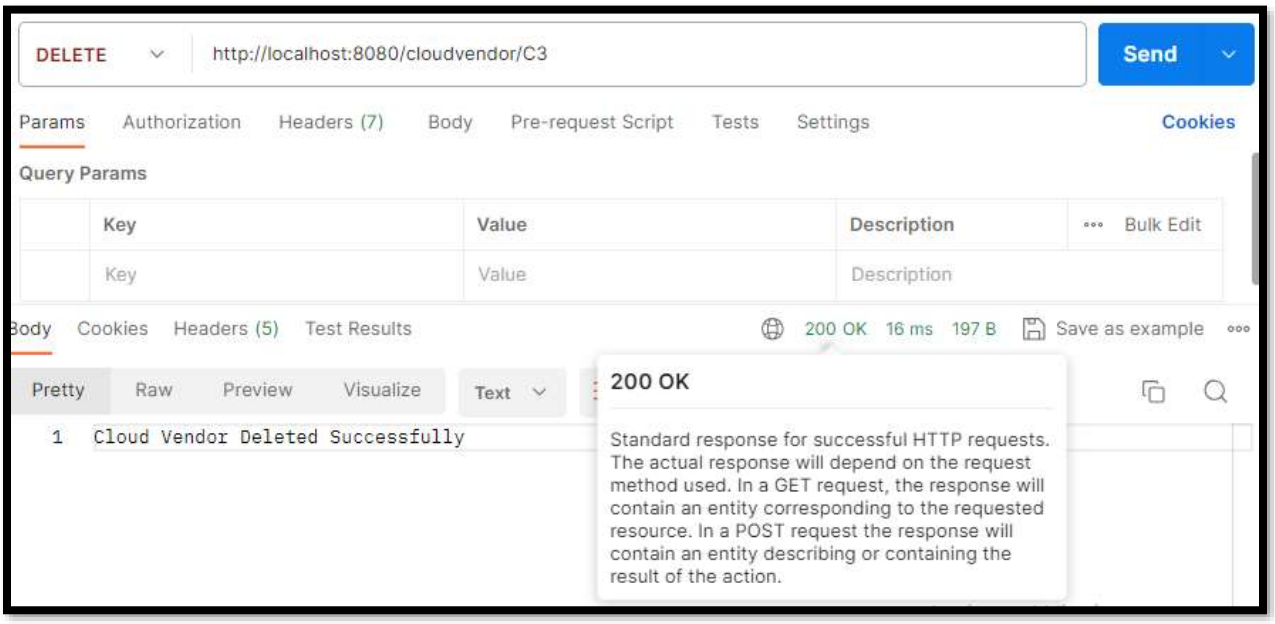



POST Req1
No Environment
Rest-Demo / Req1
Save
POST
http://localhost:8080/cloudvendor
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{"vendorId": "C3",
"vendorName": "Vendor Three",
"vendorAddress": "Address Three",
"vendorPhoneNumber": "yyyyy"
}
Body Cookies Headers (5) Test Results
200 OK 243 ms 197 B
Save as example
Pretty Raw Preview Visualize Text
Cloud Vendor Created Successfully
Activate Windows

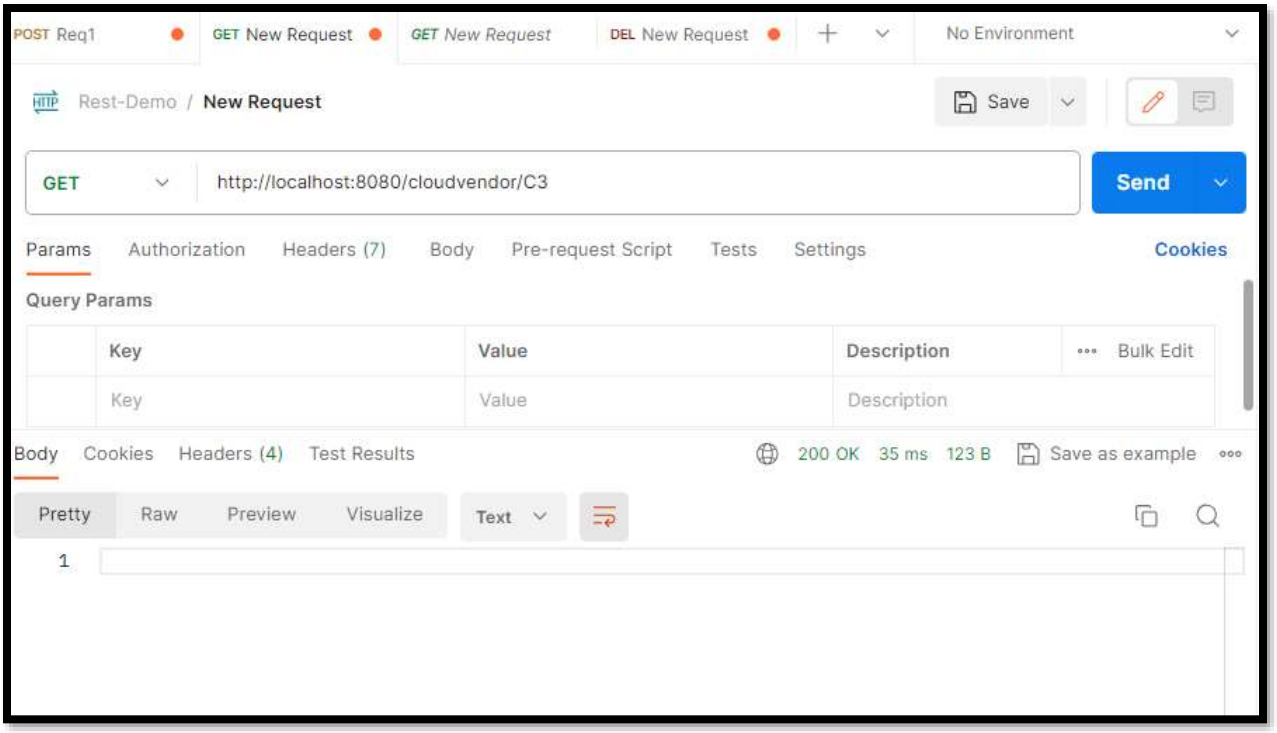

POST Req1
GET New Request
No Environment
Rest-Demo / New Request
Save
GET
http://localhost:8080/cloudvendor/C3
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings
Cookies
Query Params
Key Value Description Bulk Edit
Key Value Description
Body Cookies Headers (5) Test Results
200 OK 253 ms 269 B
Save as example
Pretty Raw Preview Visualize JSON
{
"vendorId": "C3",
"vendorName": "Vendor Three",
"vendorAddress": "Address Three",
"vendorPhoneNumber": "yyyyy"
}
Activate Windows

- Now wanted to delete for that I will add another request, and used the method as delete.





- Now, Let's check whether we will Get C3 Information or not.
- I didn't get any value, request is successful but we didn't have any data so it has given me a blank response body. So, our vendor details are deleted.



Cloud Vendor Information Service:

All Four Methods Code

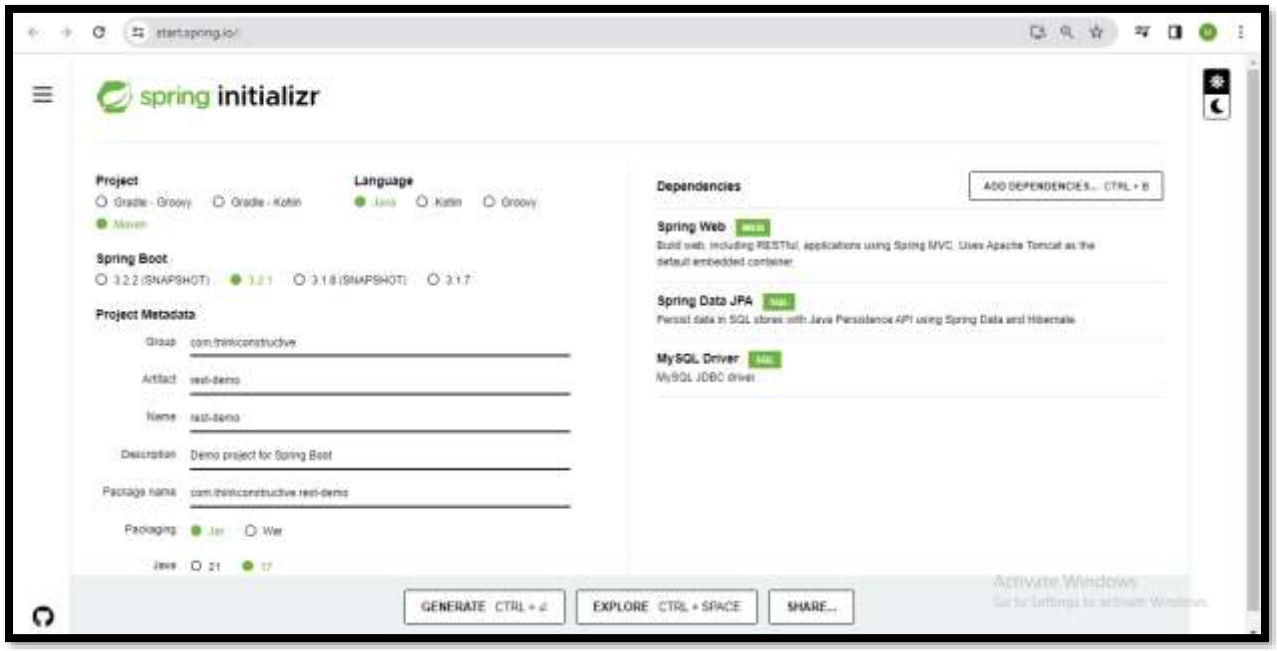
- GET
- POST
- PUT
- DELETE

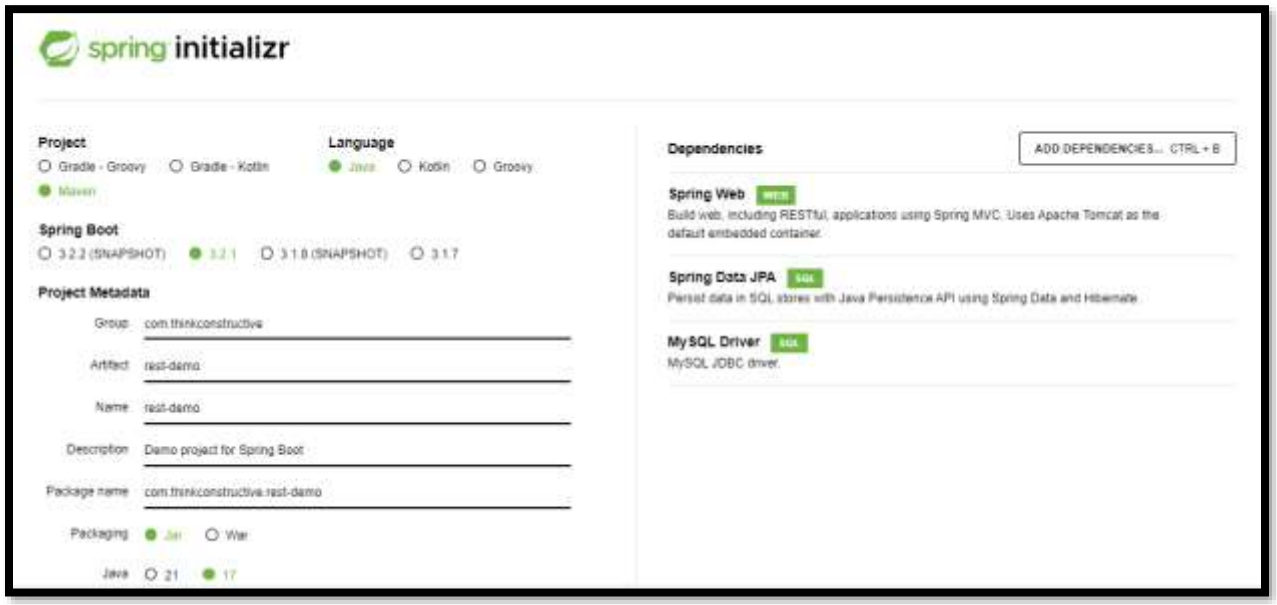
```
package com.thinkconstructive.restdemo.controller.controller;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/cloudvendor")

public class CloudAPIService
{  CloudVendor cloudVendor;
    @GetMapping("{vendorId}")
    public CloudVendor getCloudVendorDetails(String vendorId)
    {
        return cloudVendor;
        //new CloudVendor("C1","Vendor 1",
        //"Address One","xxxxx");
    }
    @PostMapping
    public String createCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
```

```
        this.cloudVendor = cloudVendor;
        return "Cloud Vendor Created Successfully";
    }
    @PutMapping
    public String updateCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        this.cloudVendor = cloudVendor;
        return "Cloud Vendor Updated Successfully";
    }
    @DeleteMapping("{vendorId}")
    public String deleteCloudVendorDetails(String vendorId)
    {
        this.cloudVendor = null;
        return "Cloud Vendor Deleted Successfully";
    }
}
```

Session# 02

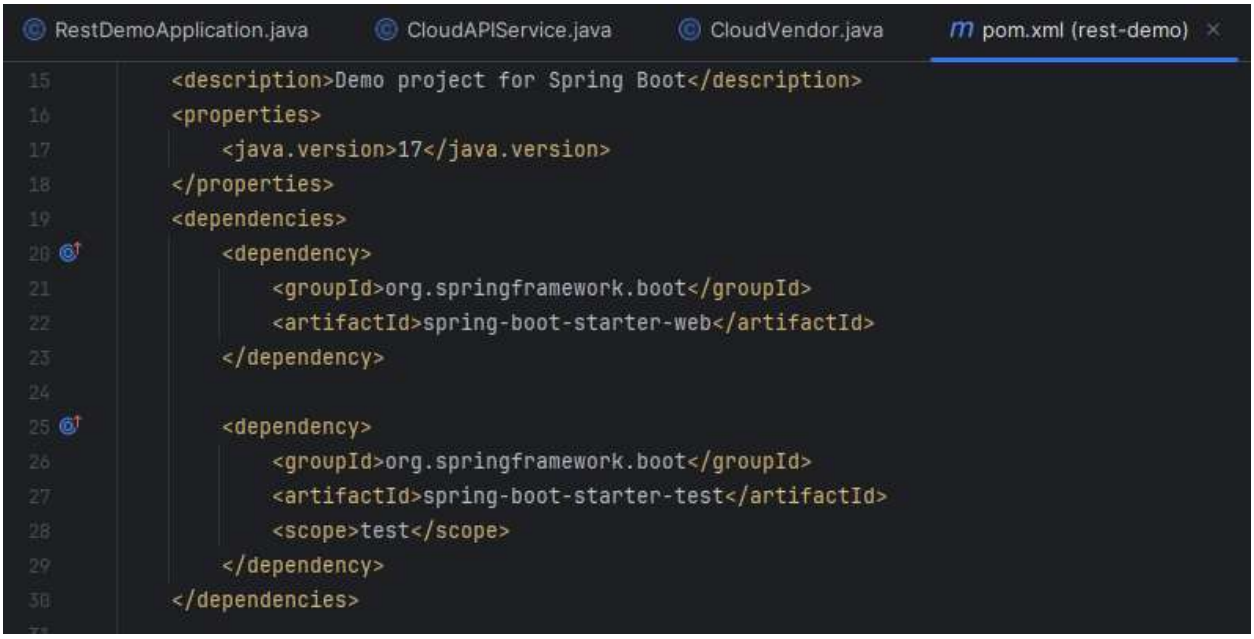
Creating Java REST API with Spring Boot, Spring Data JPA and MySQL | REST API CRUD Operations.



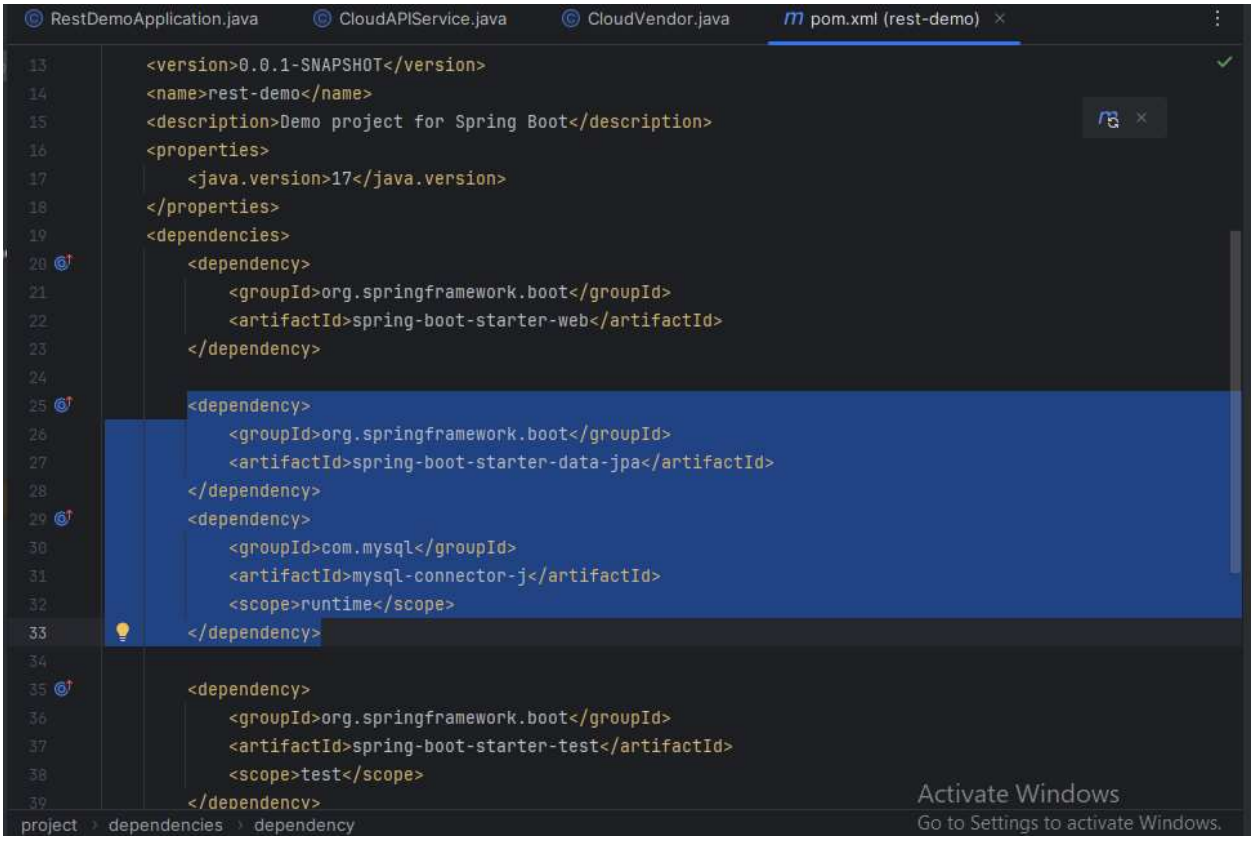


- Extract the zip file.

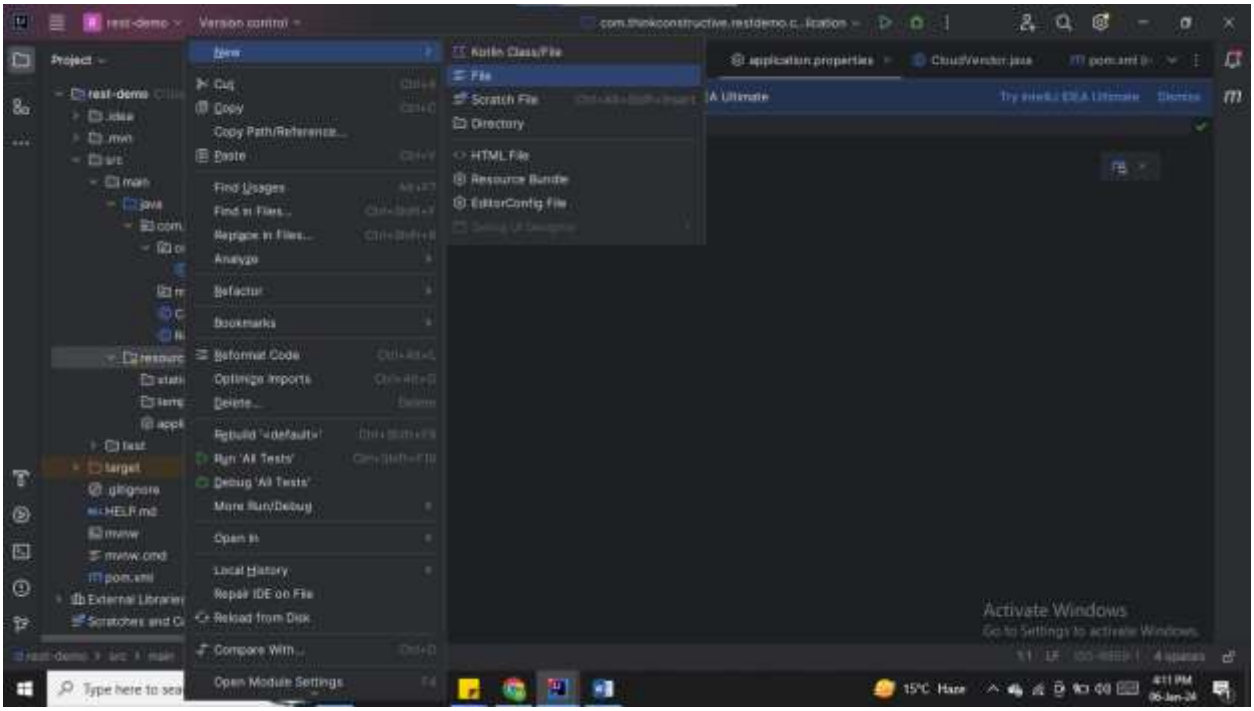
- Now, Open the previous session Rest-Demo project in intelliJ IDEA, in which we have created the controller layer. And add dependencies in Pom.xml
- Currently we have these dependencies.

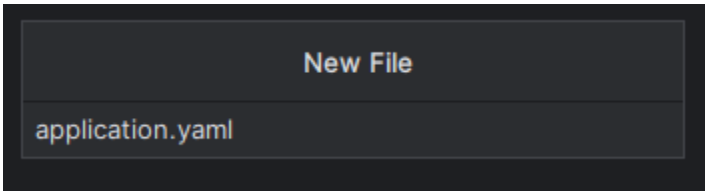


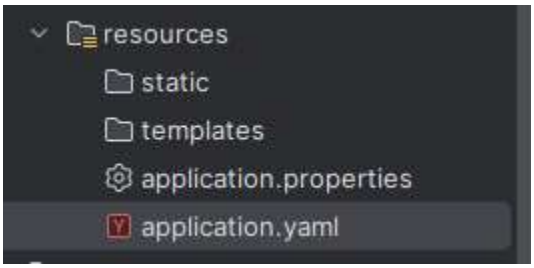
- Add Jpa and sql dependency in it.



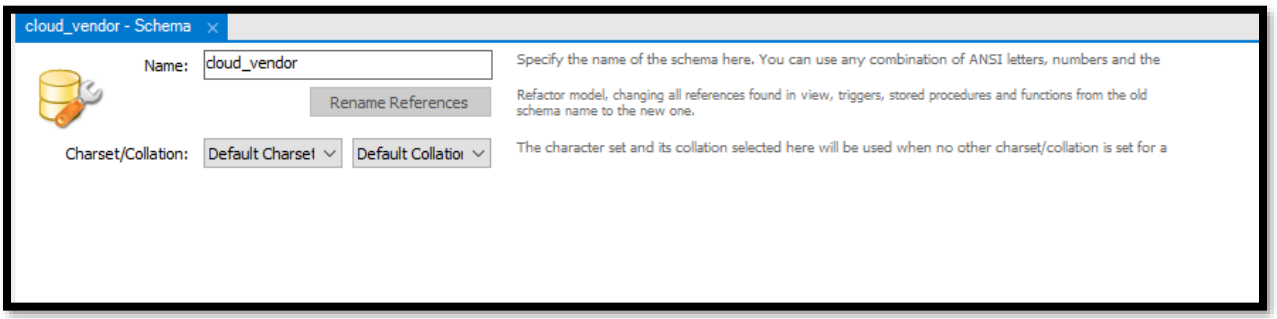
- Now, we will evolve this application connect it to the database and query the database.
- Configure MySQL database connector, so that this application can be connected to the database.

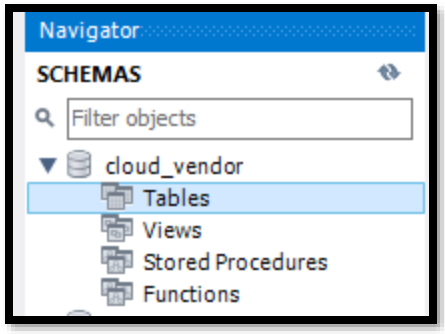




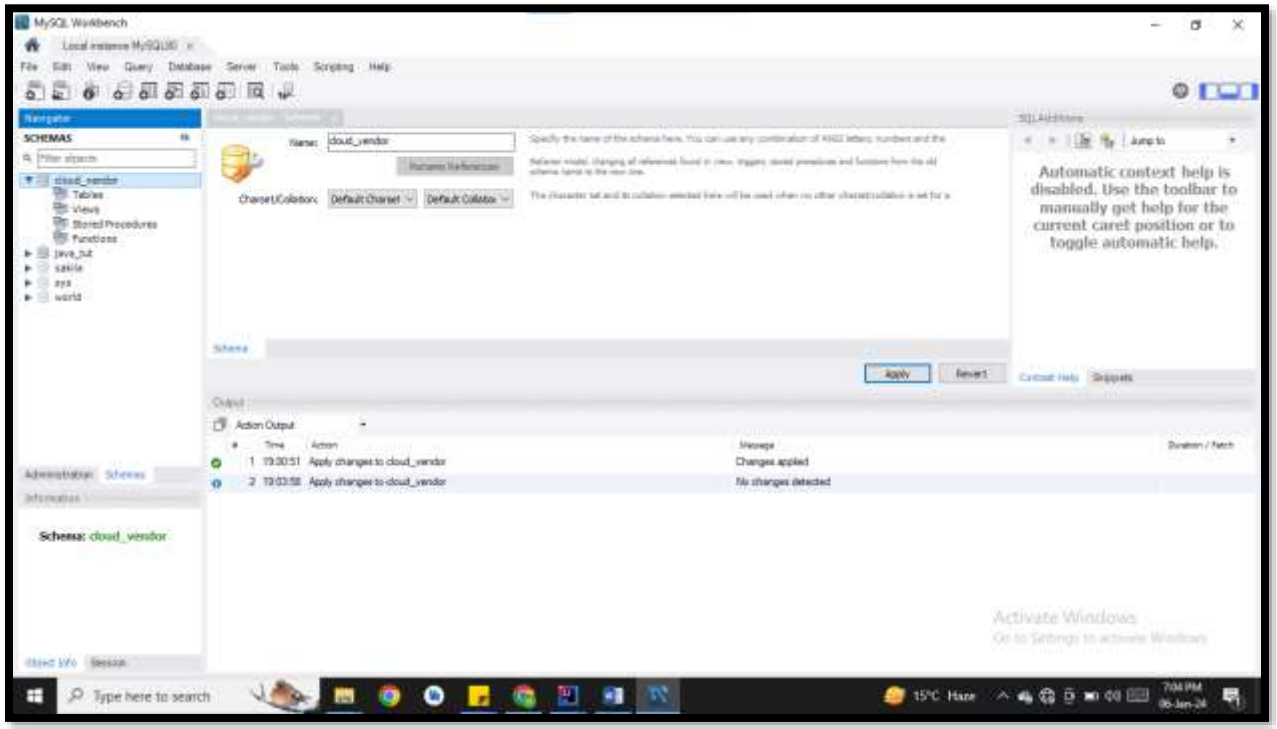


- Making new schema in MySQL Workbench



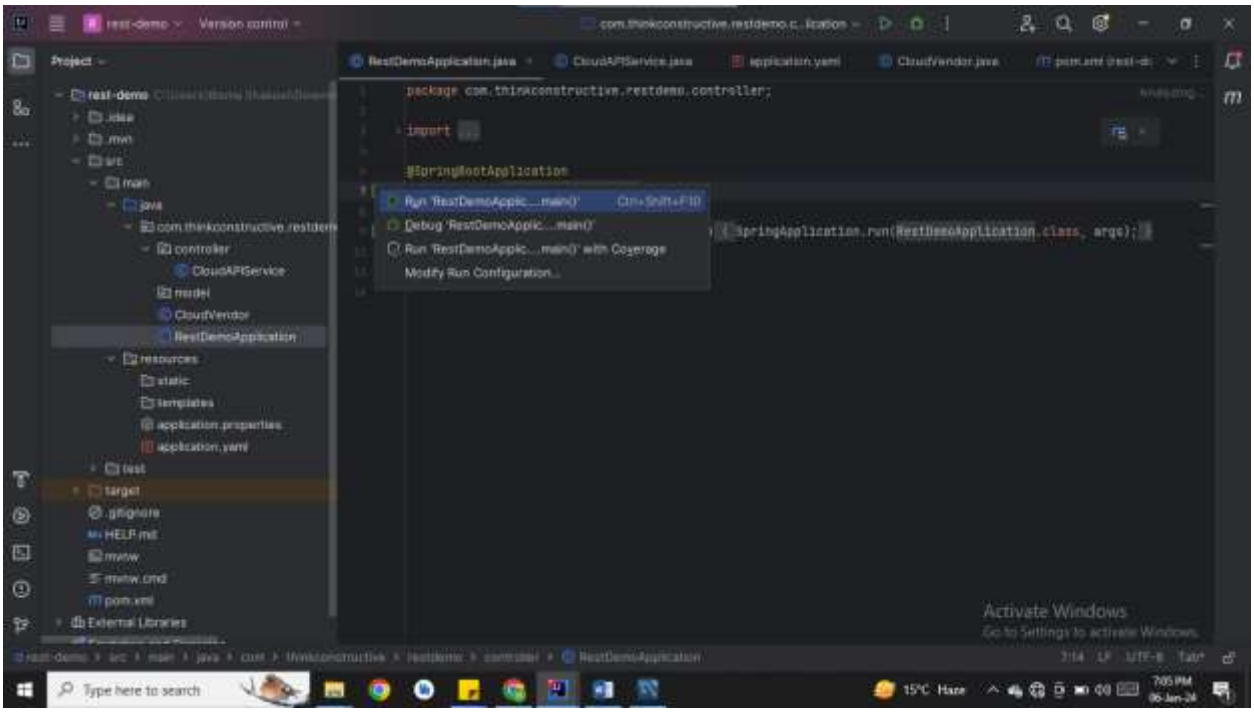


- Currently my tables are empty, and I am expecting my tables should get created with the help of my spring boot application.

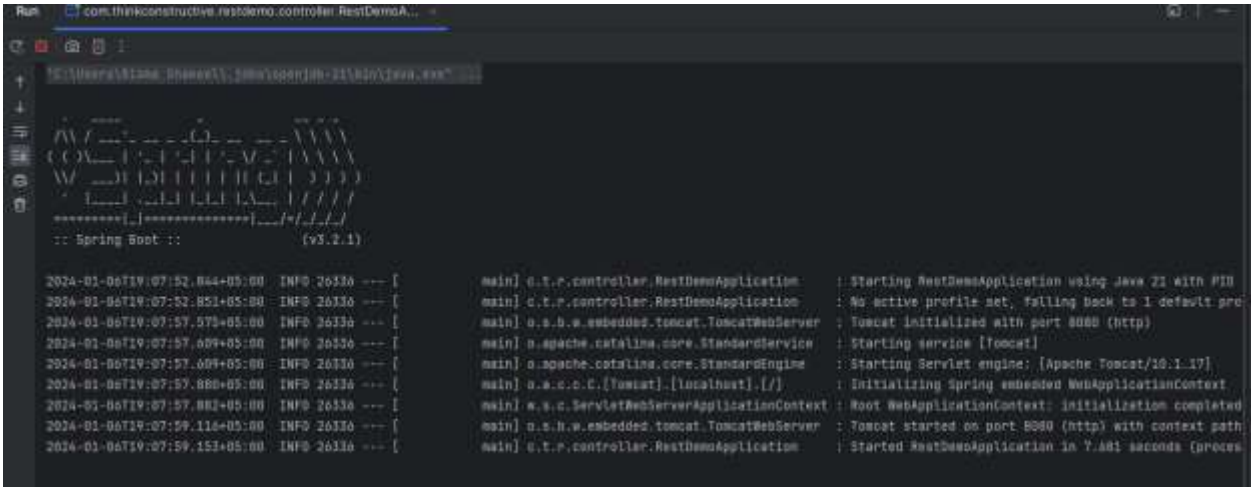


- Here we have done all the configuration.

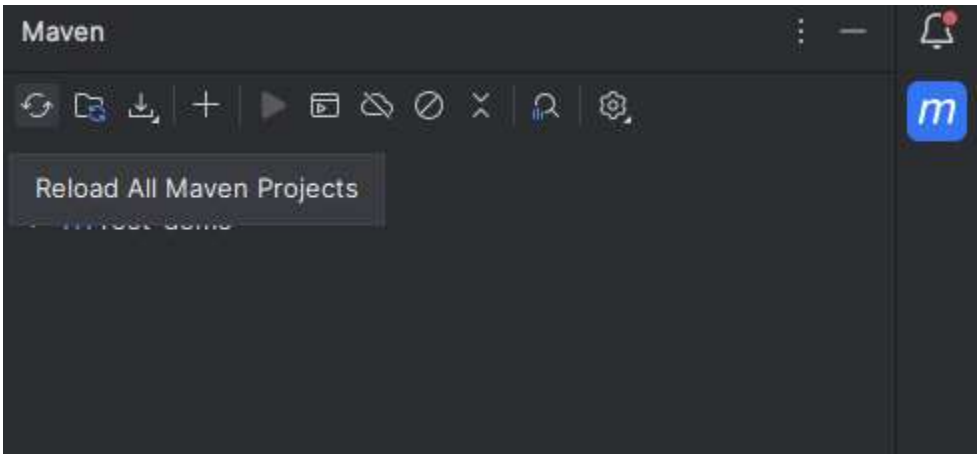
- Now, go to RestDemoApplication and run it to check that our database is connected successfully.



- My application is successfully connected to database.



- Whenever you add any dependency in pom.xml you should do maven rebuild.



- Do these changes in CloudVendor.java Class

```
emoApplication.java   CloudAPIService.java   application.yaml   CloudVendor.java   pom.xml (rest-demo)
```

```java
package com.thinkconstructive.restdemo.controller;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

5 usages
@Entity
@Table(name="cloud_vendor_info")

public class CloudVendor {

    3 usages
    @Id
    private String vendorId;
    3 usages
    private String vendorName;
    3 usages
    private String vendorAddress;
    3 usages
    private String vendorPhoneNumber;

    no usages
    public String getVendorId() {
        return vendorId;
    }

    no usages
```

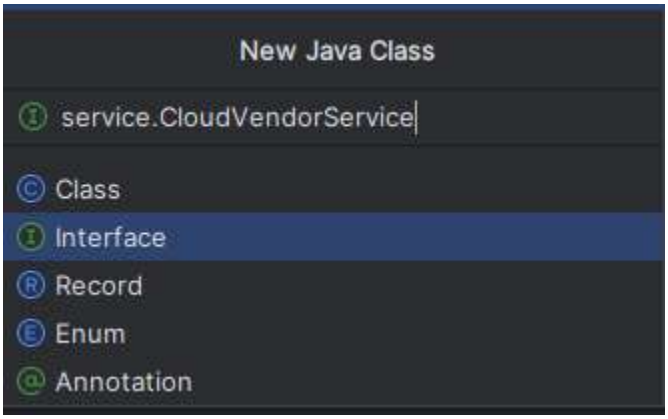

- Create new repository


```
emoApplication.java   CloudAPIService.java   CloudVendorRepository.java   application.yaml   CloudVendor.jav
```

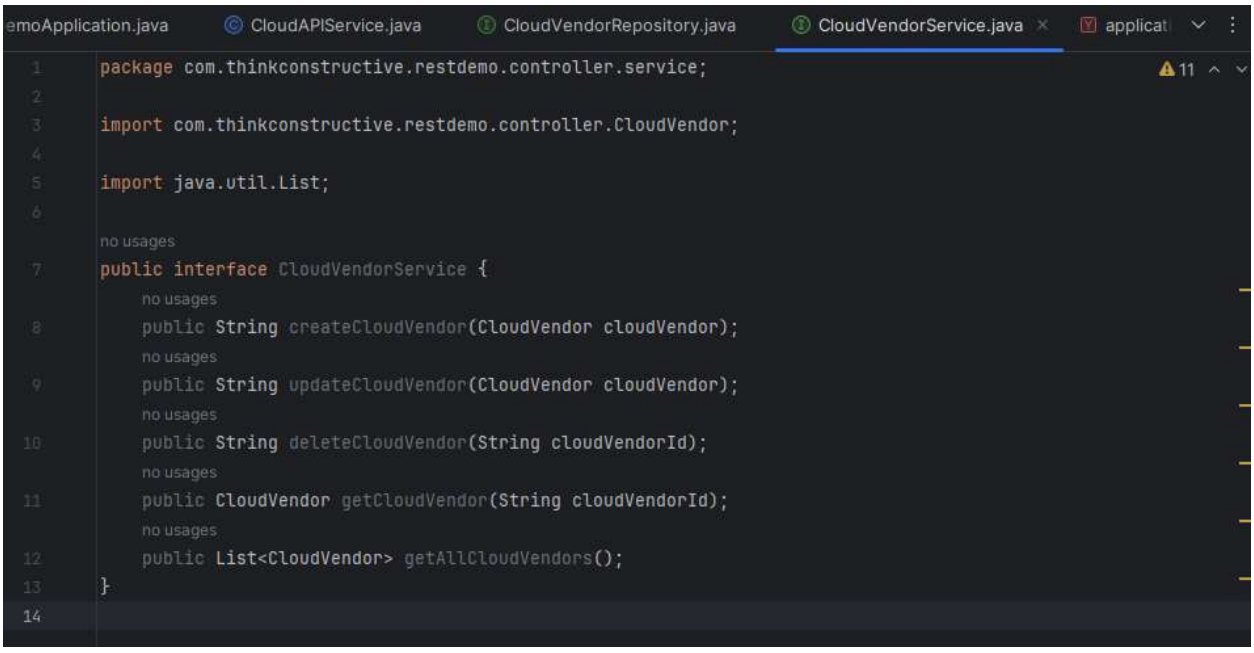
```java
package com.thinkconstructive.restdemo.controller.repository;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.data.jpa.repository.JpaRepository;

no usages
public interface CloudVendorRepository extends JpaRepository<CloudVendor, String> {

}
```

```
package com.thinkconstructive.restdemo.controller.repository;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CloudVendorRepository extends JpaRepository<CloudVendor,
String> {

}
```

- Create New Repository

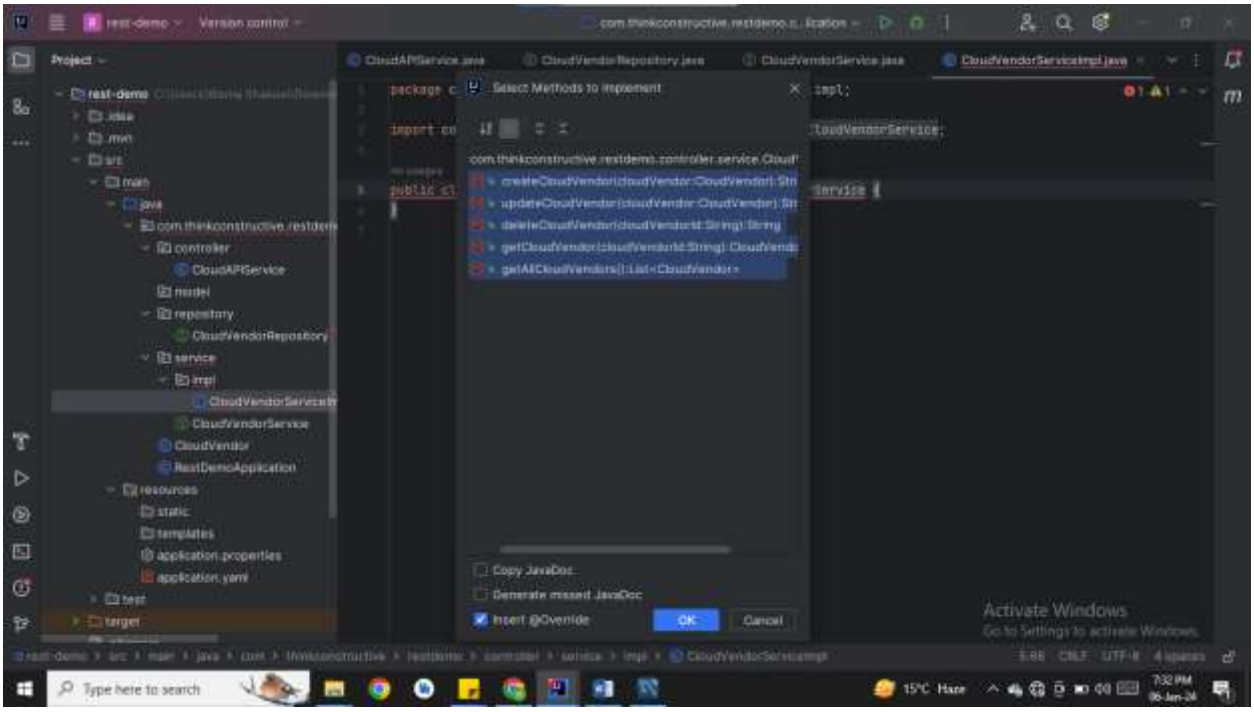


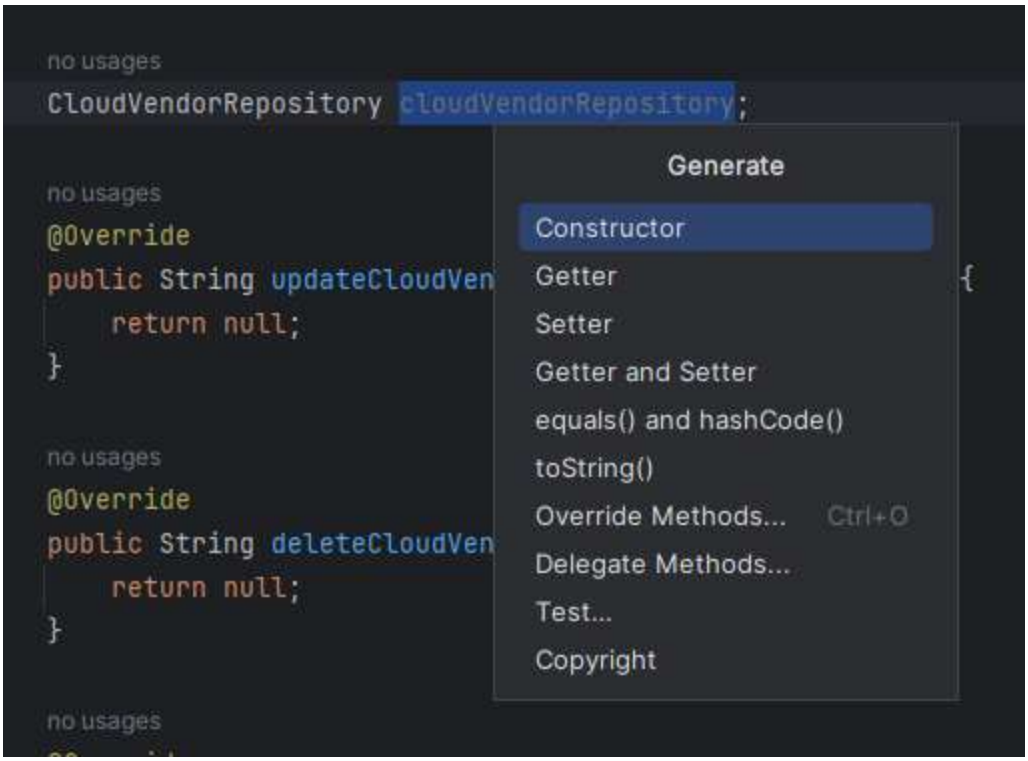
- Cloud vendor service interface is defined.




```
package com.thinkconstructive.restdemo.controller.service;

import com.thinkconstructive.restdemo.controller.CloudVendor;

import java.util.List;
```

```java
public interface CloudVendorService {
    public String createCloudVendor(CloudVendor cloudVendor);
    public String updateCloudVendor(CloudVendor cloudVendor);
    public String deleteCloudVendor(String cloudVendorId);
    public CloudVendor getCloudVendor(String cloudVendorId);
    public List<CloudVendor> getAllCloudVendors();
}
```



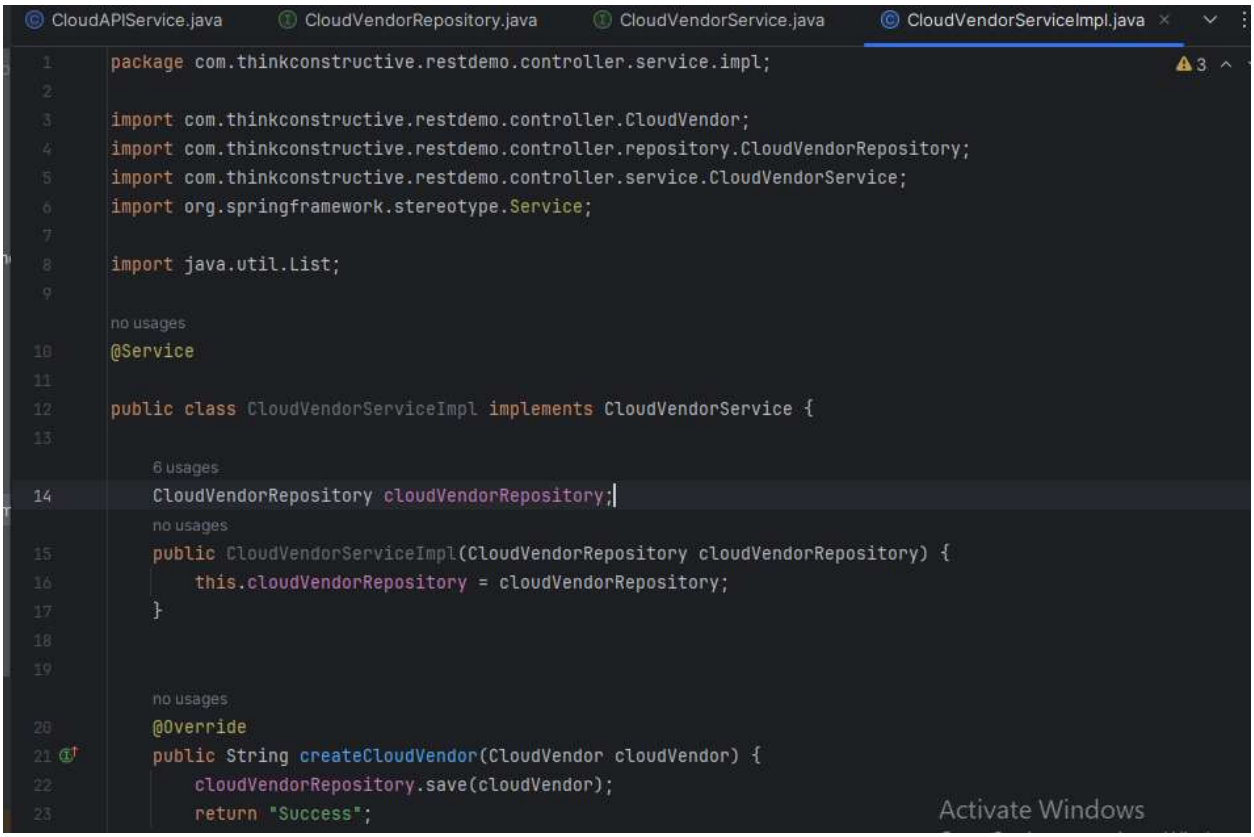


- Implemented service layer completely

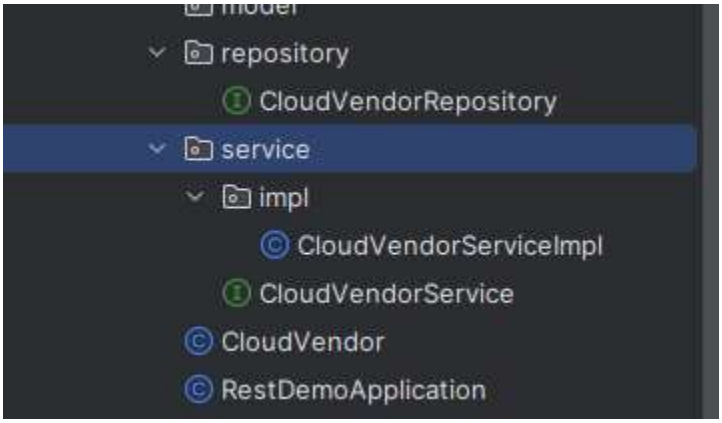
CODE:

```
package com.thinkconstructive.restdemo.controller.service.impl;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import
com.thinkconstructive.restdemo.controller.repository.CloudVendorRepository;
import com.thinkconstructive.restdemo.controller.service.CloudVendorService;
import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CloudVendorServiceImpl implements CloudVendorService {

    CloudVendorRepository cloudVendorRepository;
    public CloudVendorServiceImpl(CloudVendorRepository
cloudVendorRepository) {
        this.cloudVendorRepository = cloudVendorRepository;
    }


    @Override
    public String createCloudVendor(CloudVendor cloudVendor) {
        cloudVendorRepository.save(cloudVendor);
        return "Success";
    }
    @Override
```

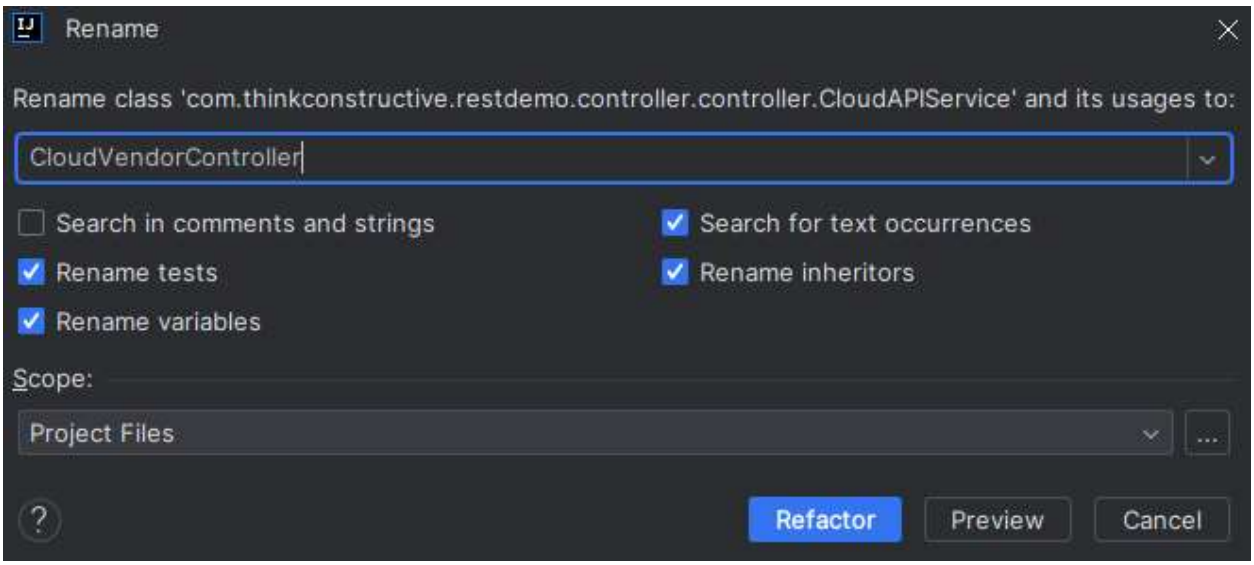
```
    public String updateCloudVendor(CloudVendor cloudVendor) {
        cloudVendorRepository.save(cloudVendor);
        return "Success";
    }

    @Override
    public String deleteCloudVendor(String cloudVendorId) {
        cloudVendorRepository.deleteById(cloudVendorId);
        return "Success";
    }

    @Override
    public CloudVendor getCloudVendor(String cloudVendorId) {

        return cloudVendorRepository.findById(cloudVendorId).get();
    }

    @Override
    public List<CloudVendor> getAllCloudVendors() {
        return cloudVendorRepository.findAll();
    }
}
```



- We have implemented out service layer and repository layer.



- **Controller Layer**



Code:

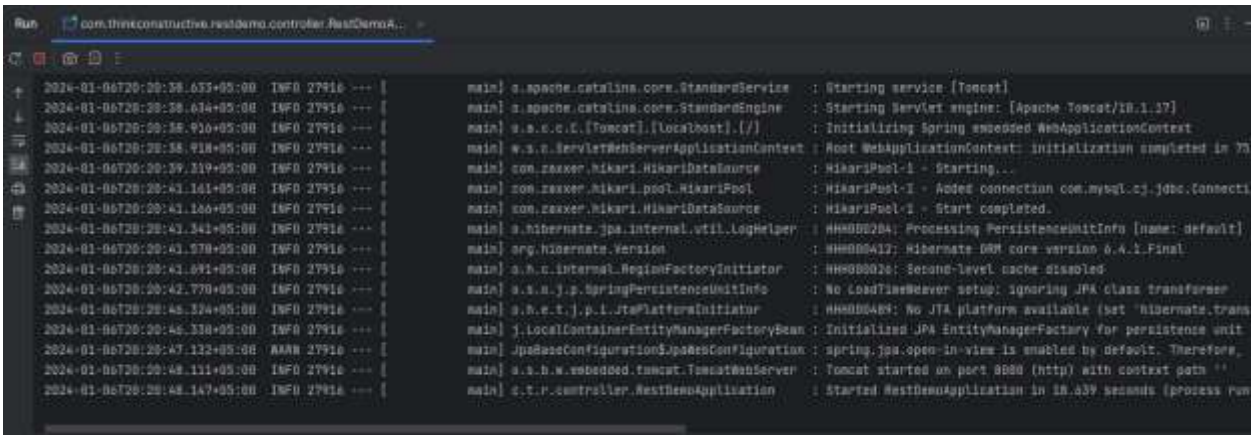
```
package com.thinkconstructive.restdemo.controller.controller;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import com.thinkconstructive.restdemo.controller.service.CloudVendorService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/cloudvendor")

public class CloudVendorController
{
    CloudVendorService cloudVendorService;
    public CloudVendorController(CloudVendorService cloudVendorService) {
        this.cloudVendorService = cloudVendorService;
    }
```

```
    // Read Specific Cloud Vendor Details from DB
    @GetMapping("{vendorId}")
    public CloudVendor getCloudVendorDetails(@PathVariable("vendorId") String
vendorId)
    {
        return cloudVendorService.getCloudVendor(vendorId);

    }
    // Read All Cloud Vendor Details from DB
    @GetMapping()
    public List<CloudVendor> getAllCloudVendorDetails()
    {
        return cloudVendorService.getAllCloudVendors();

    }

    @PostMapping
    public String createCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        cloudVendorService.createCloudVendor(cloudVendor);
        return "Cloud Vendor Created Successfully";
    }
    @PutMapping
    public String updateCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        cloudVendorService.updateCloudVendor(cloudVendor);
        return "Cloud Vendor Updated Successfully";
    }
    @DeleteMapping("{vendorId}")
    public String deleteCloudVendorDetails(@PathVariable("vendorId") String
vendorId)
    {
        cloudVendorService.deleteCloudVendor(vendorId);
        return "Cloud Vendor Deleted Successfully";
    }
}
```

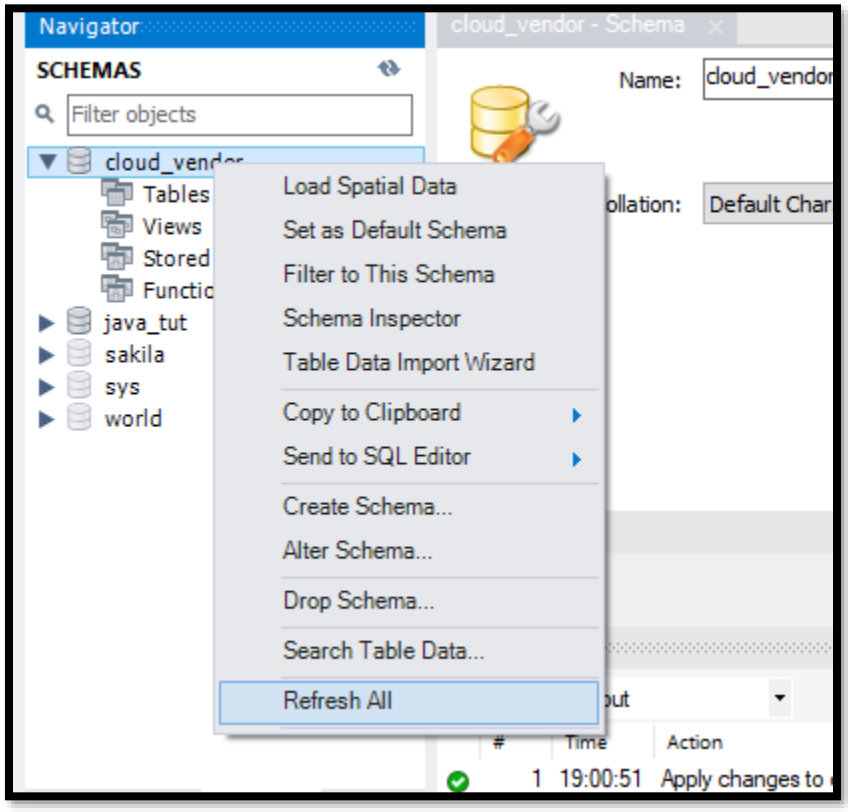
```java
package com.thinkconstructive.restdemo.controller.controller;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import com.thinkconstructive.restdemo.controller.service.CloudVendorService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/cloudvendor")

public class CloudVendorController
{
    CloudVendorService cloudVendorService;
    public CloudVendorController(CloudVendorService cloudVendorService) {
        this.cloudVendorService = cloudVendorService;
    }

    // Read Specific Cloud Vendor Details from DB
    @GetMapping("{vendorId}")
    public CloudVendor getCloudVendorDetails(@PathVariable("vendorId") String vendorId)
    {
```

```
spring.jpa.hibernate.ddl-auto=create
```

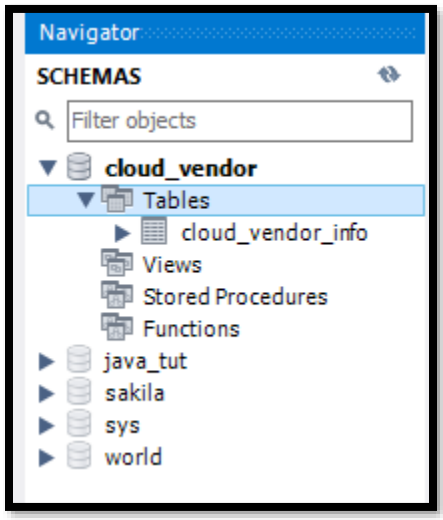
- Now, run the RestDemoApplication again to check that it is running successfully.
- My application is up and running successfully.

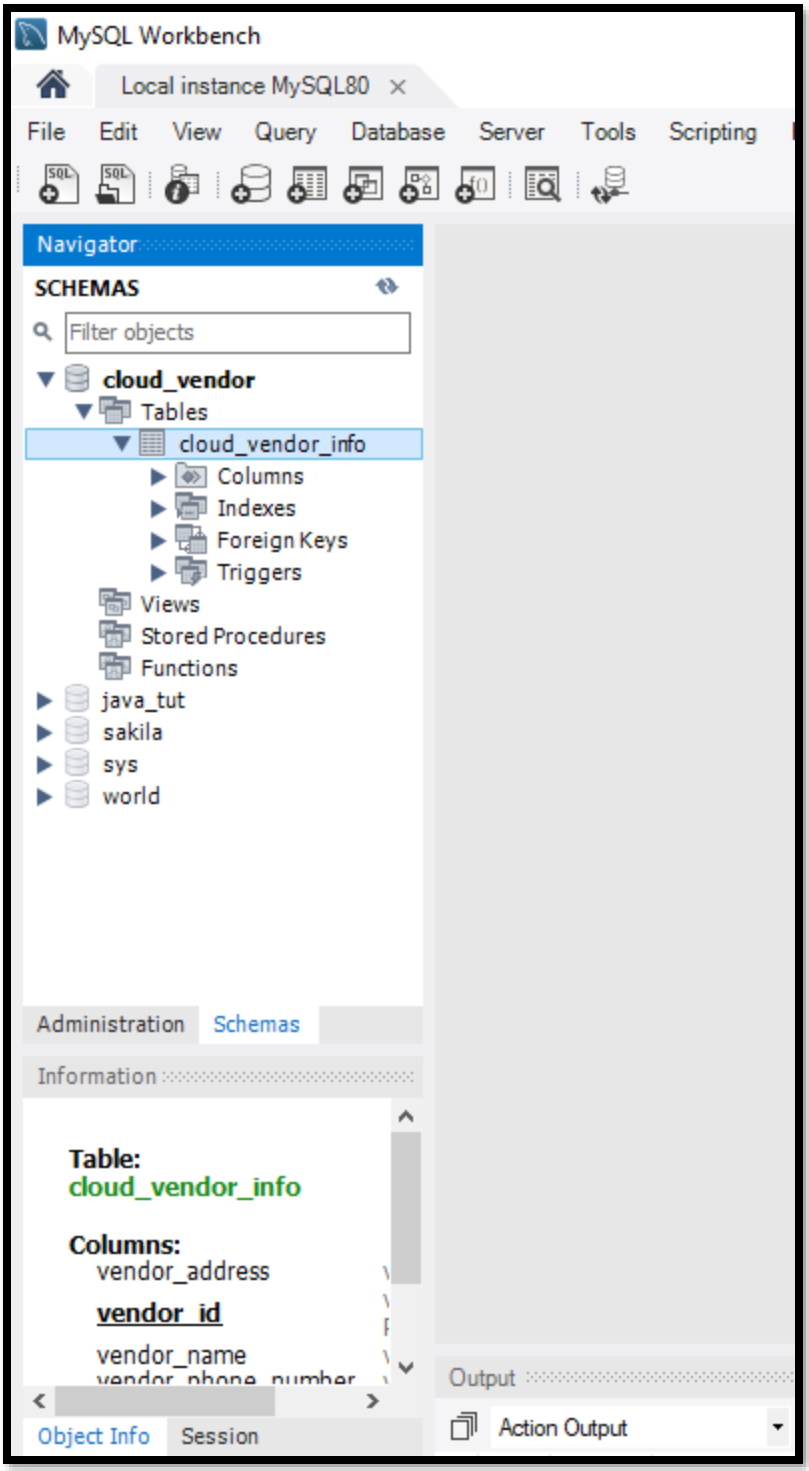


- Refresh and check whether any table is created.

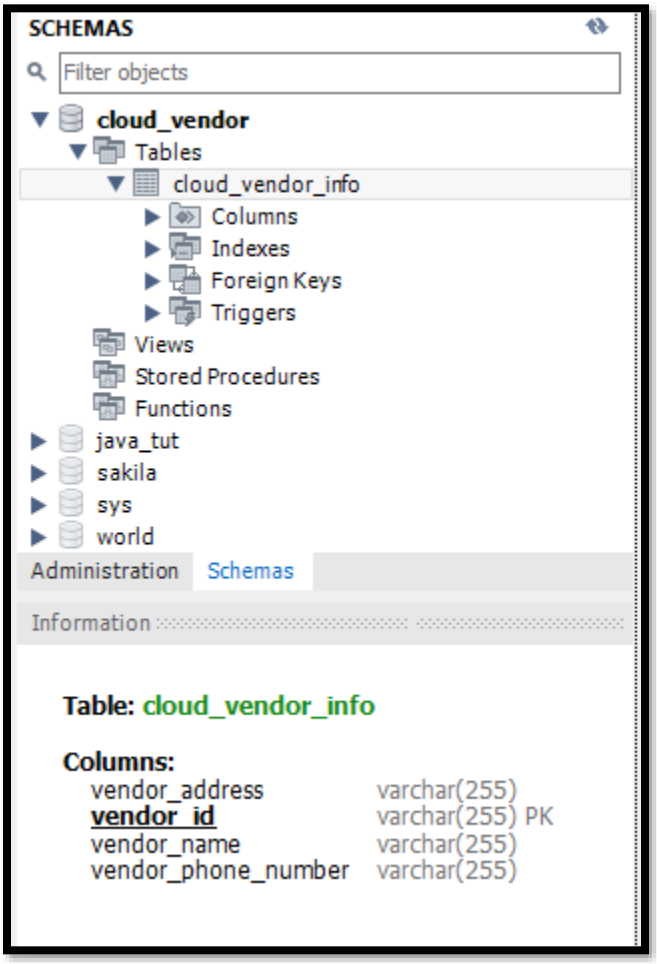


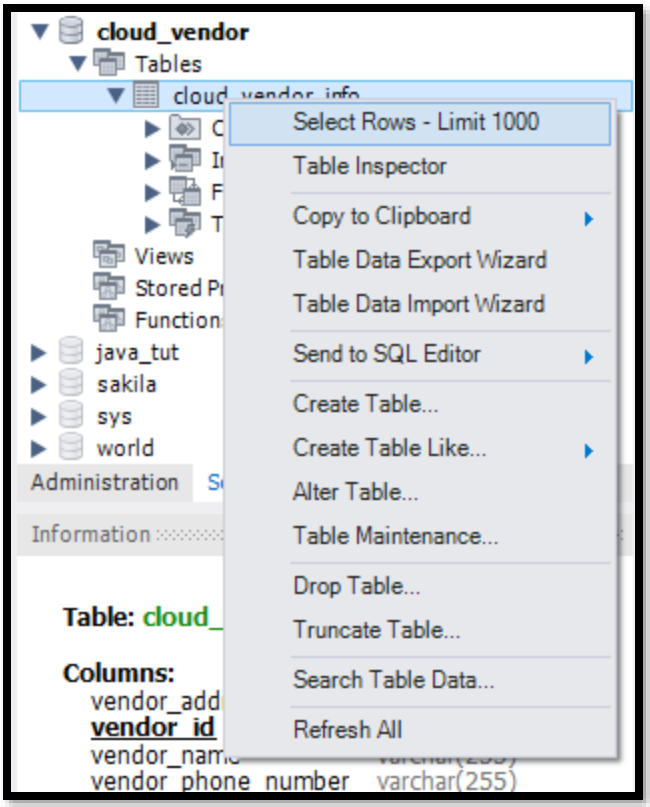
- Successfully table is created in the database.



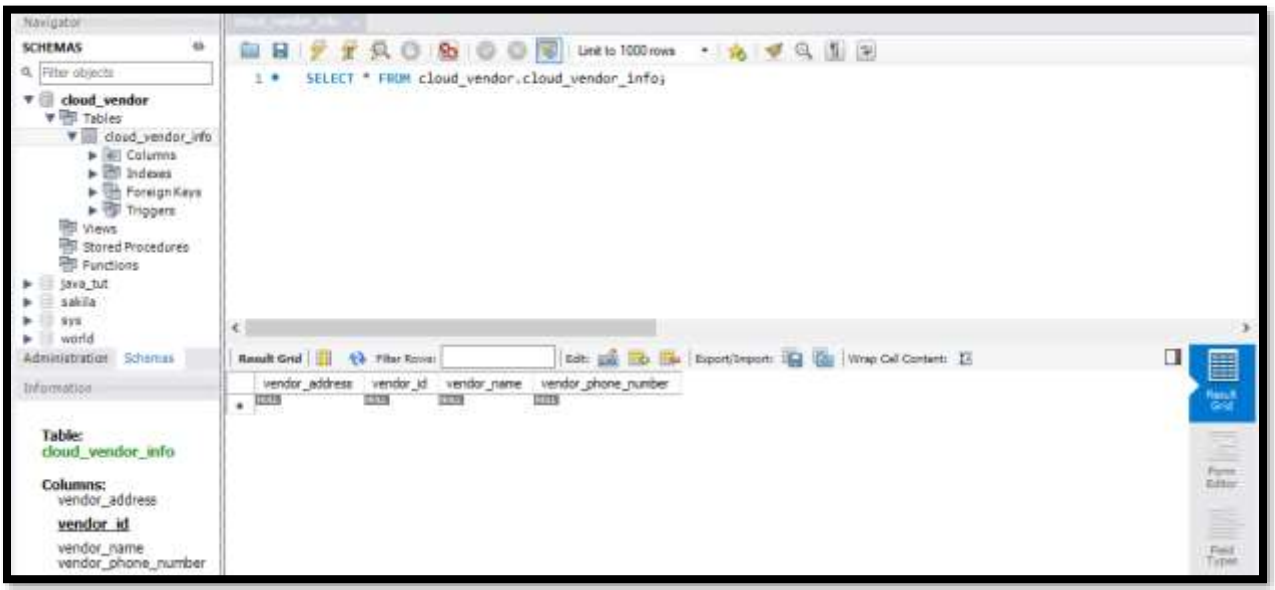

MySQL Workbench
Local instance MySQL80
File
Edit
View
Query
Database
Server
Tools
Scripting
Navigator
SCHEMAS
Filter objects
cloud_vendor
Tables
cloud_vendor_info
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
java_tut
sakila
sys
world
Administration
Schemas
Information
Table:
cloud_vendor_info
Columns:
vendor_address
vendor_id
vendor_name
Object Info
Session
Output
Action Output

- All four fields are visible here.

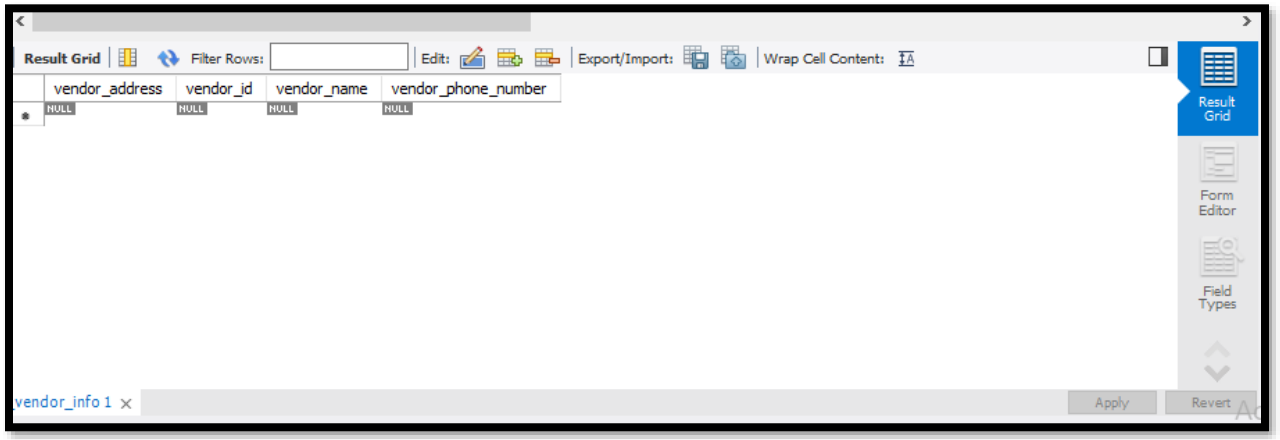




- Table should be empty since we have not added any data.

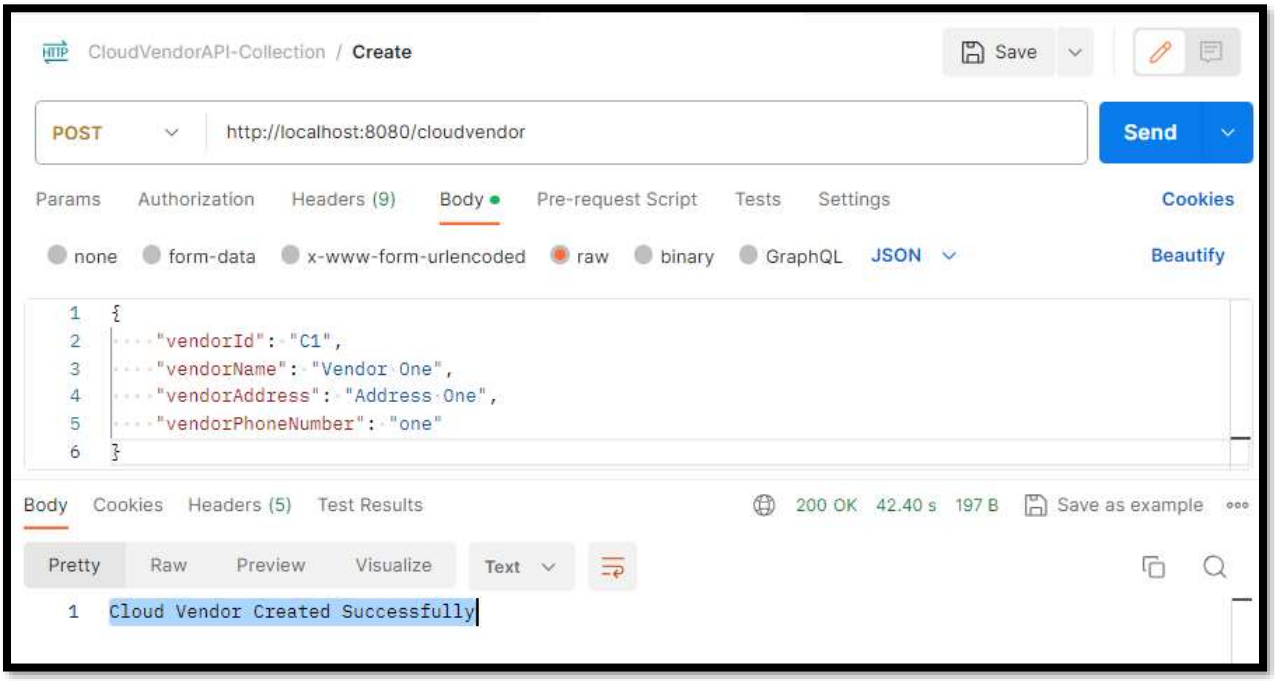


- All four fields are here without any values.

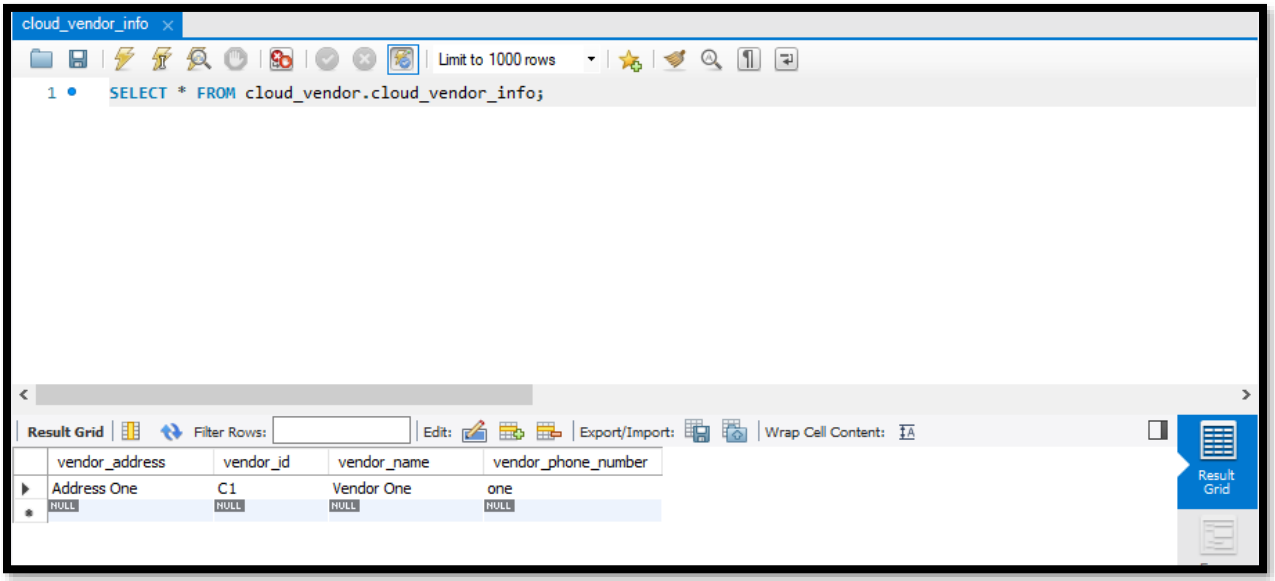


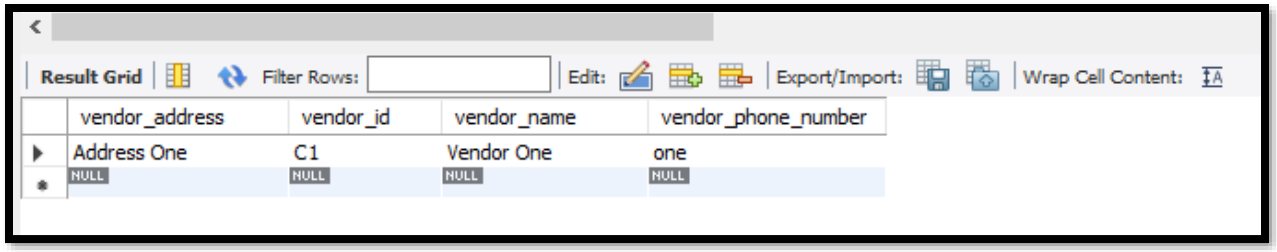
- Now, open postman and trigger some request.
- This is my message body.

- Create cloud vendor request is successfully sent.

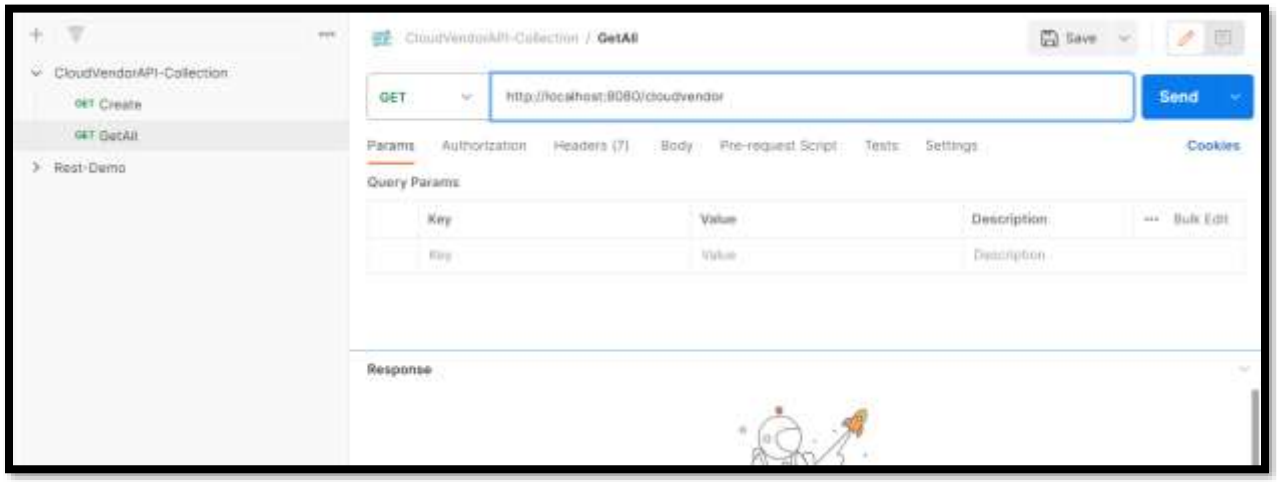


- Now, go to MySQL database and check the table.
- Entries are added.

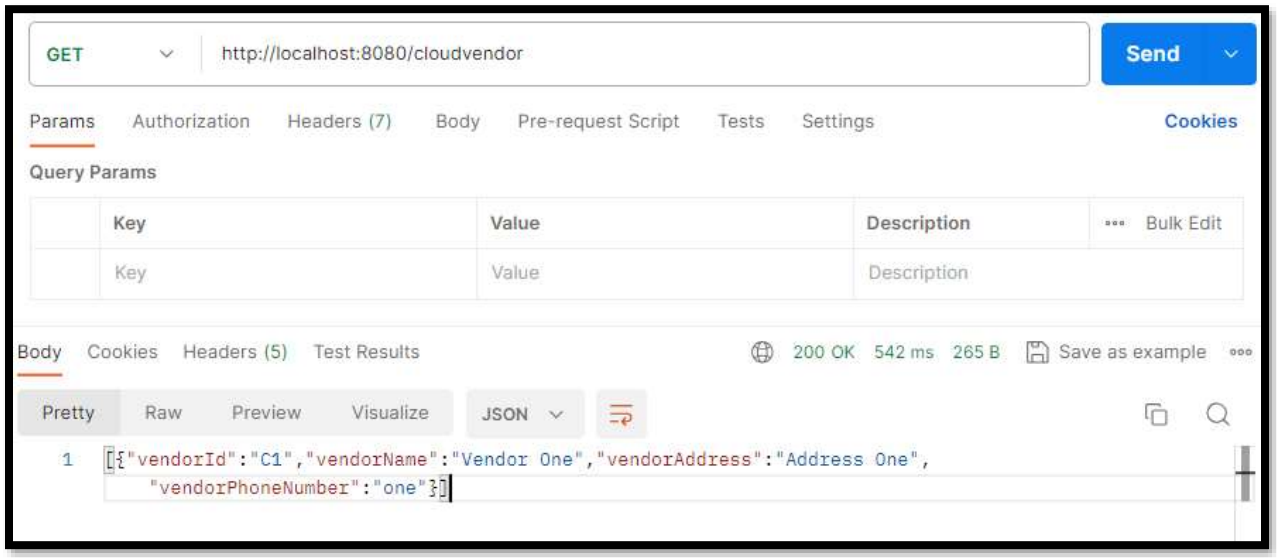




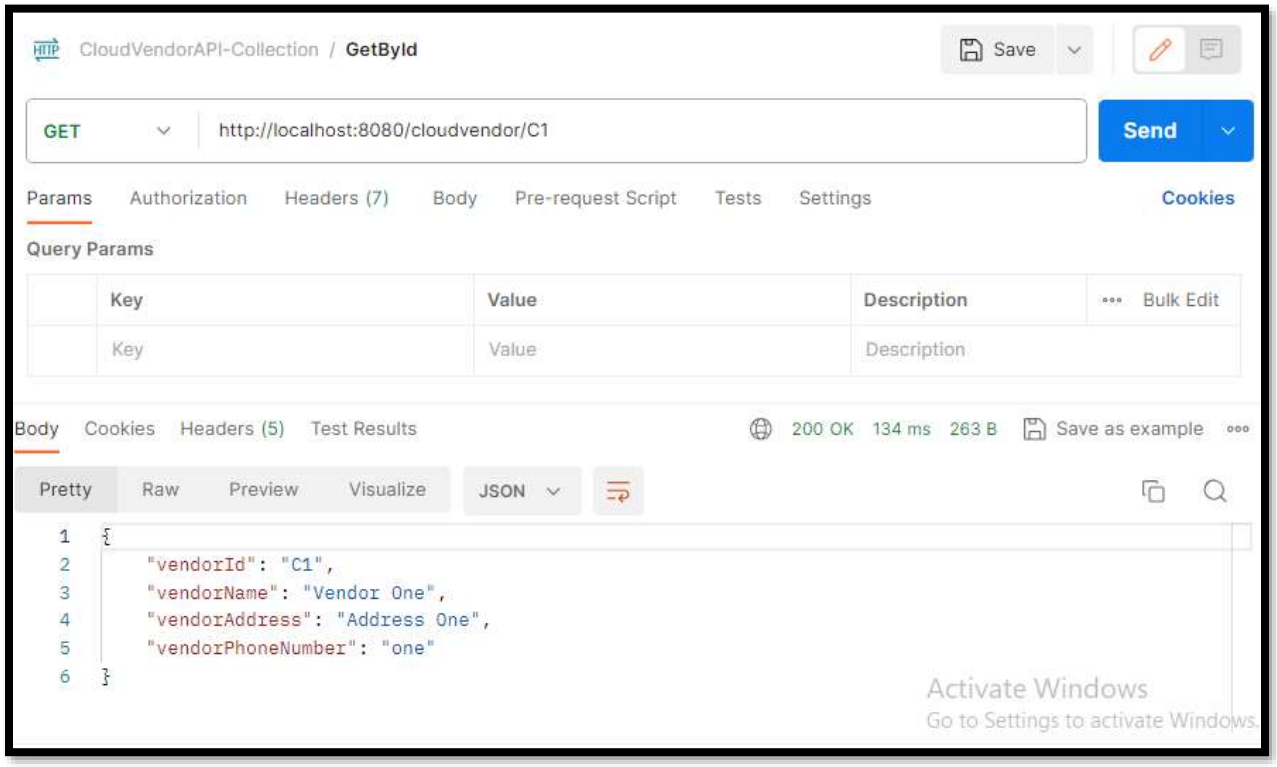
- Check what should we get.



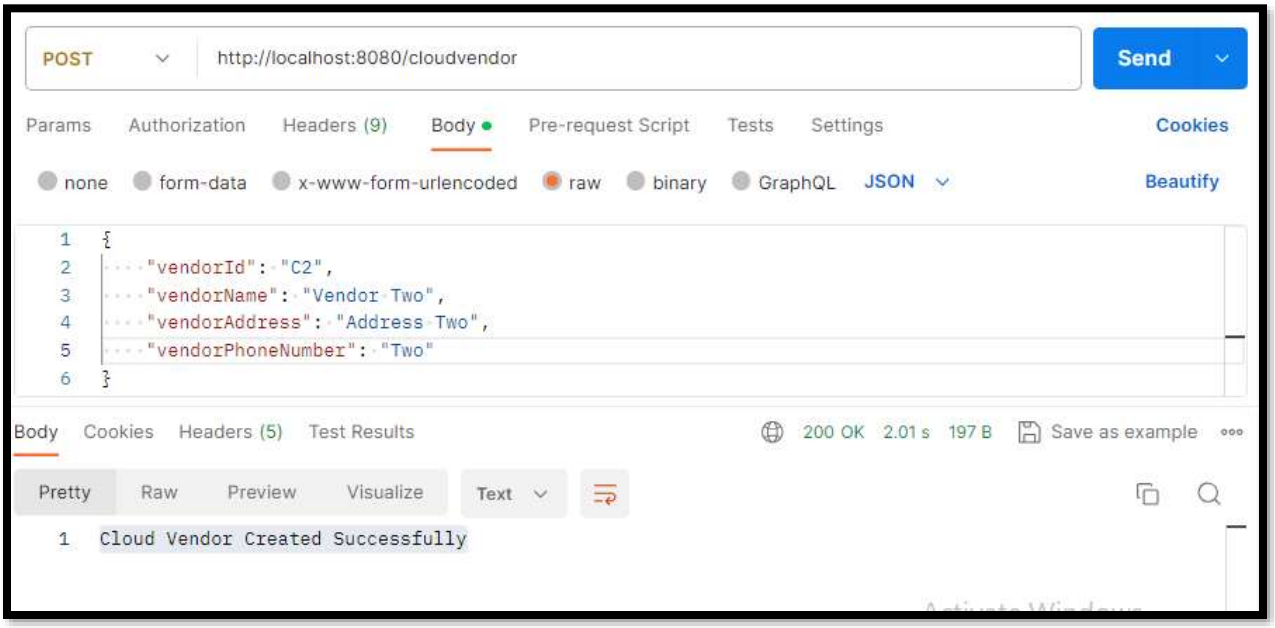
- We got all the four fields.



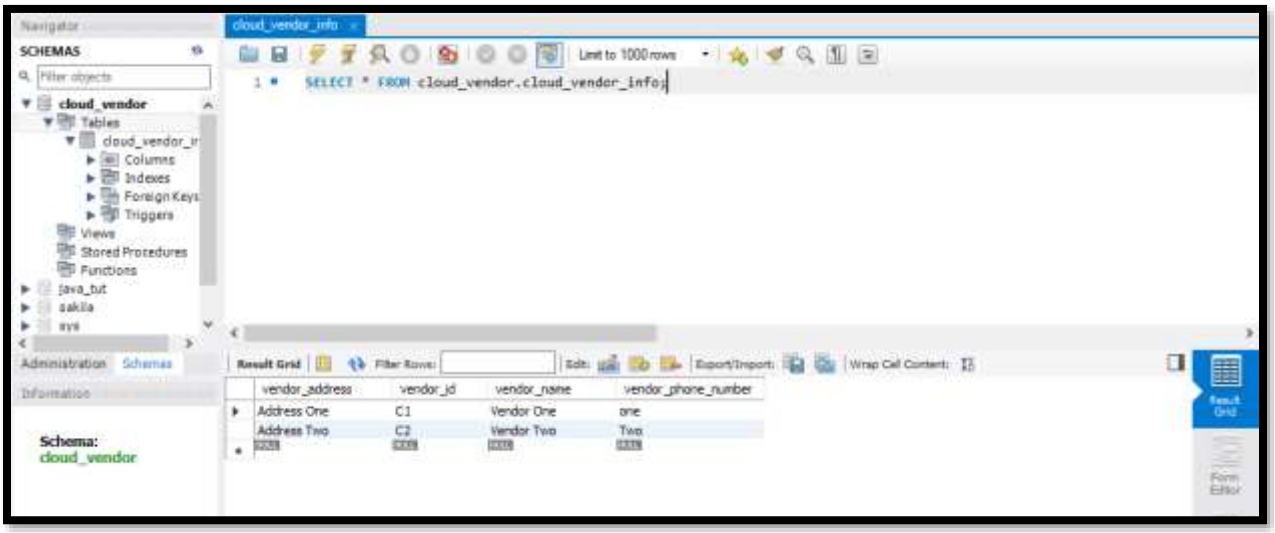
- Check Get By Id for C1.



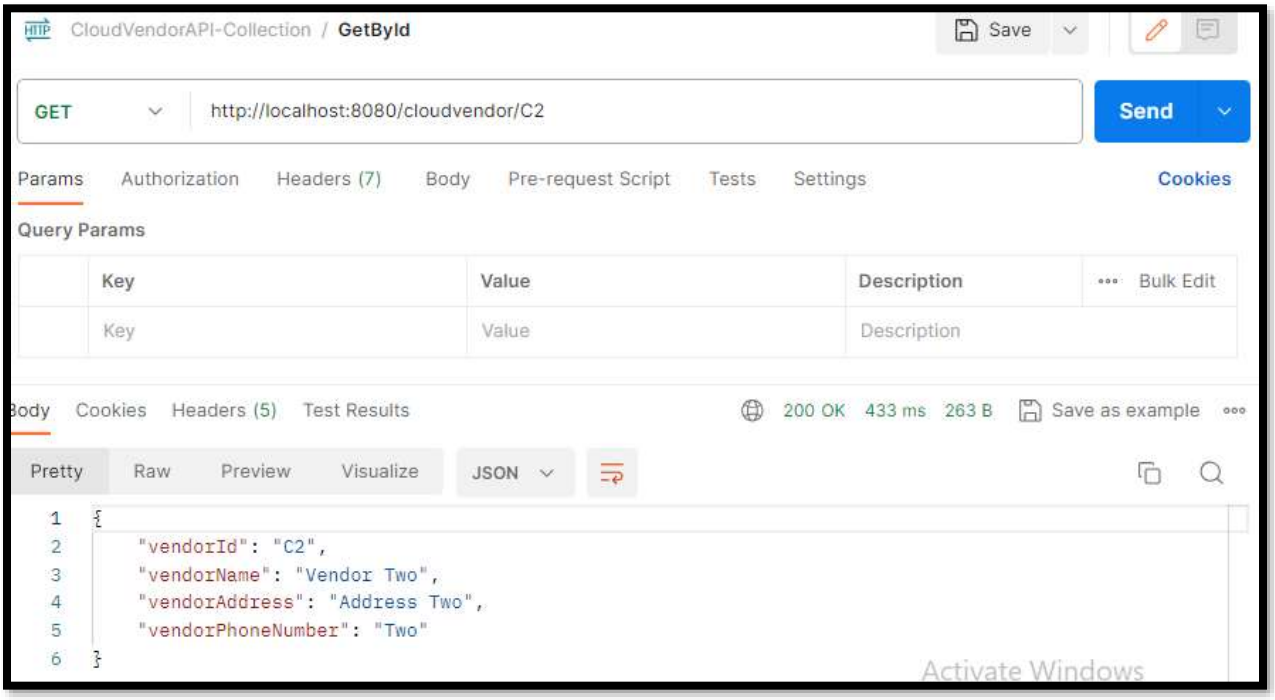
- Now, Let’s create two to three more records and check the database.



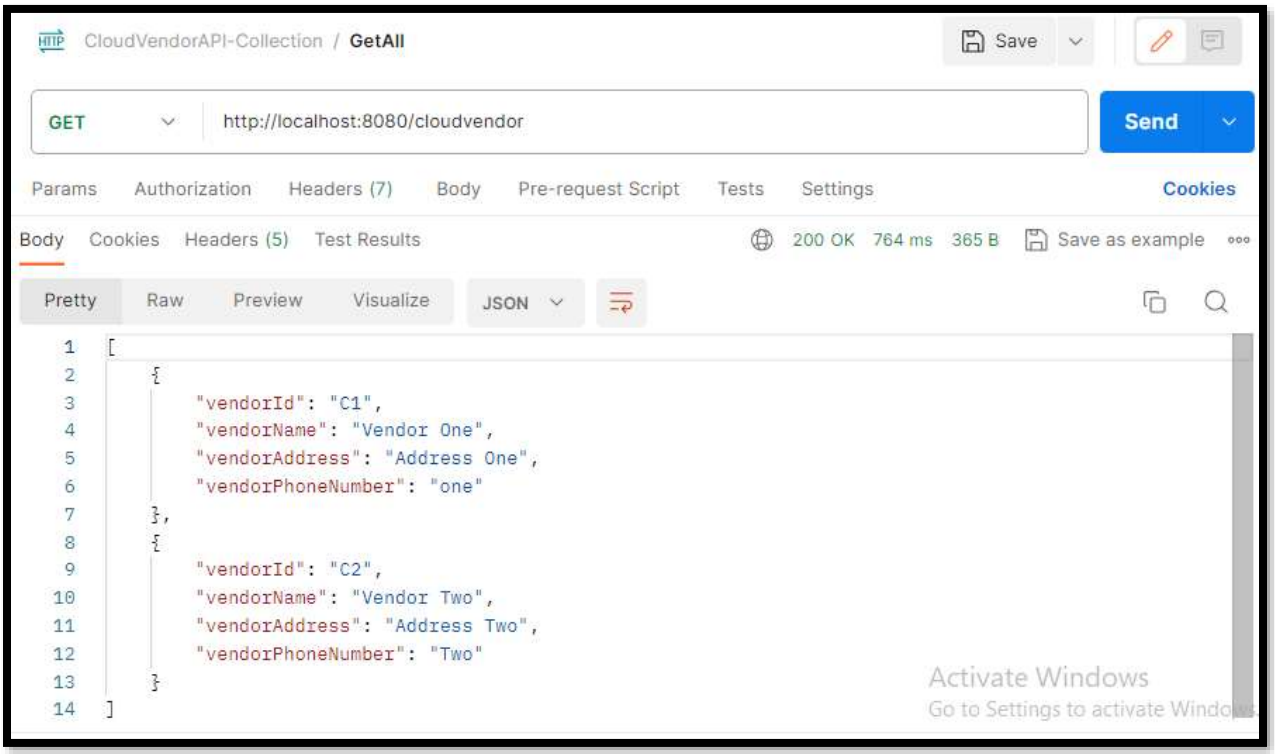
- Check in database.



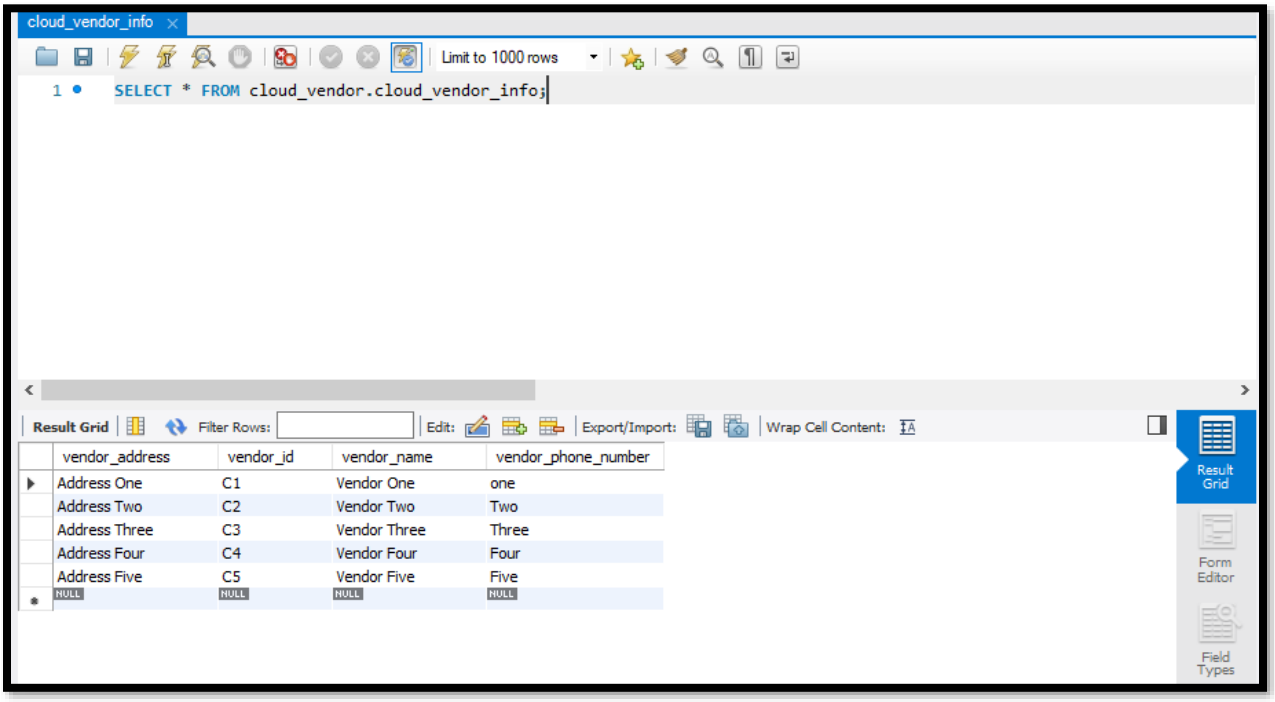
- Now Check some get by id for C2.



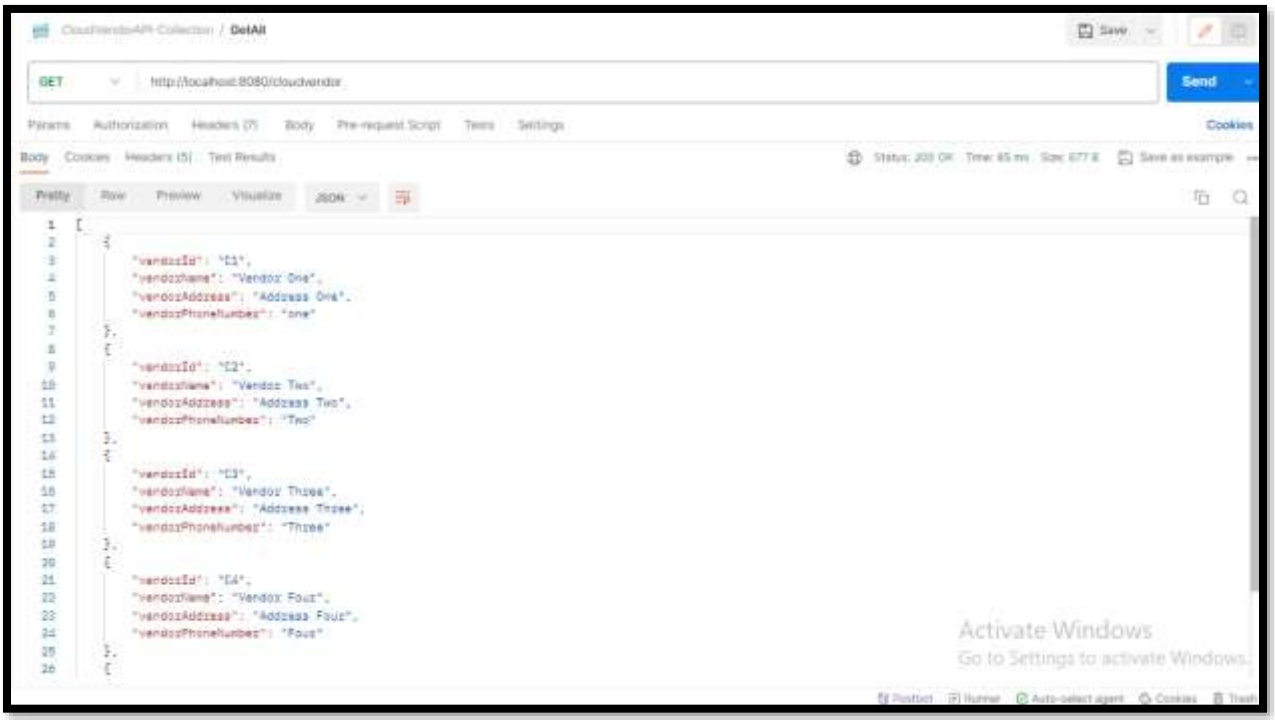
- In get all, I should get two records.



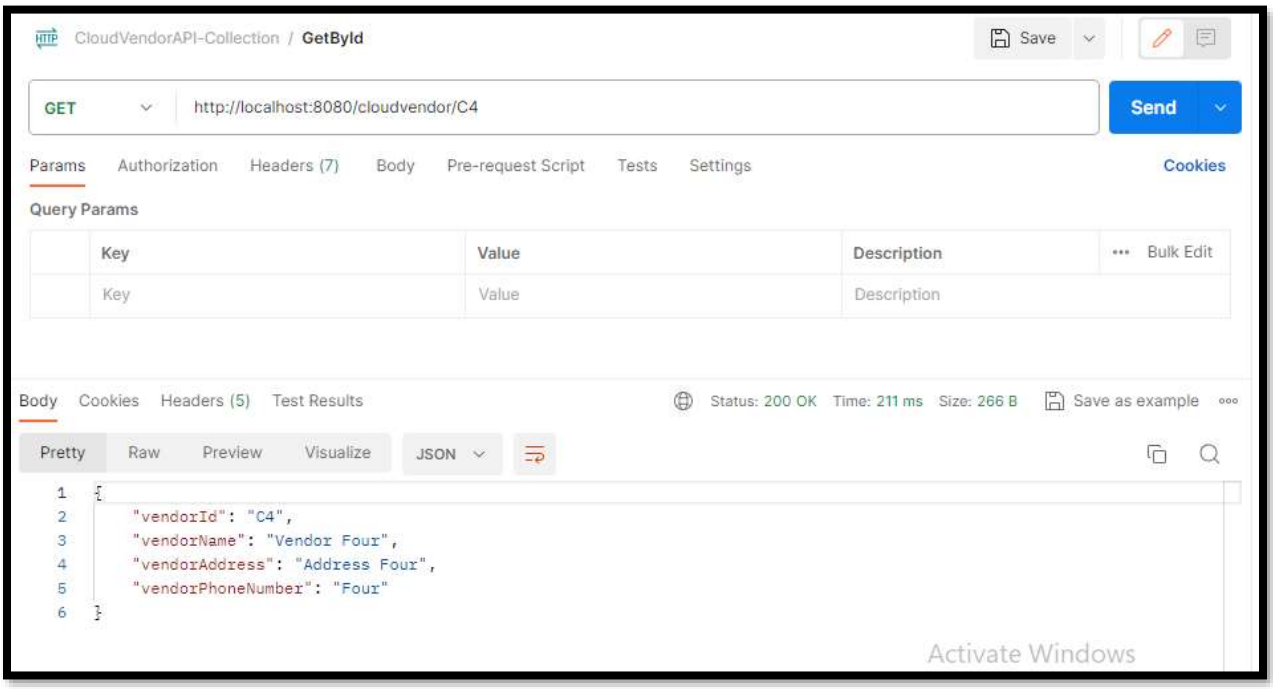
- Now, I have created 5 cloud vendors. Let us test them in MySQL Workbench.



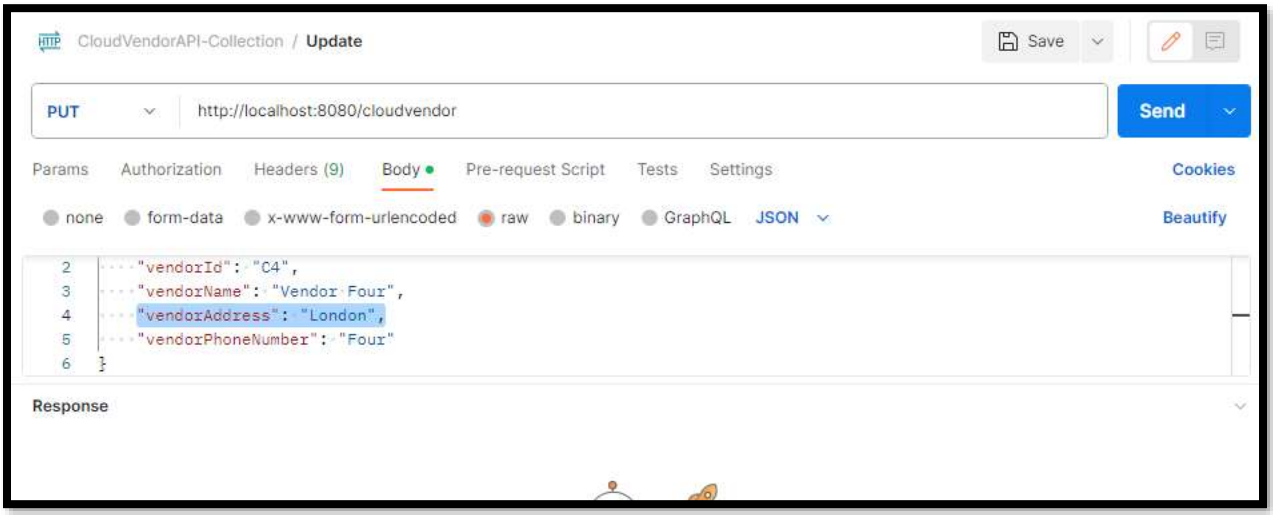
- All five records will be visible in get all in postman.

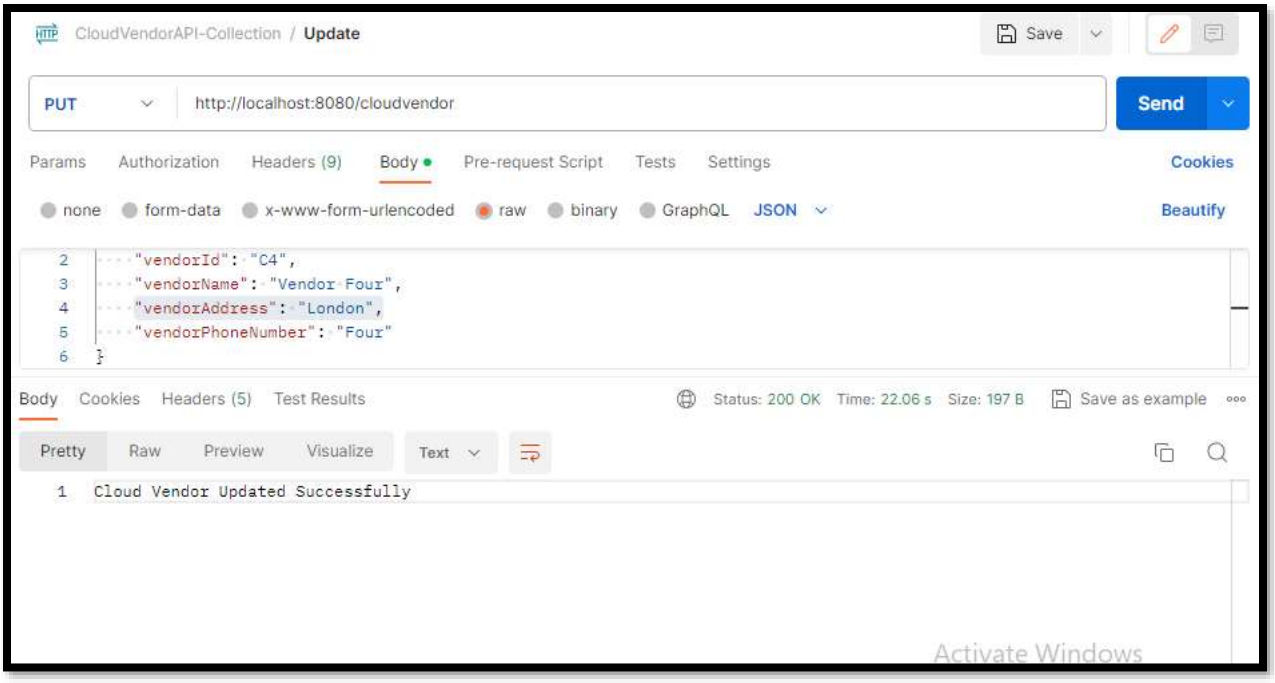


- Now, when ill check fot get by id for C4, I will get the following.

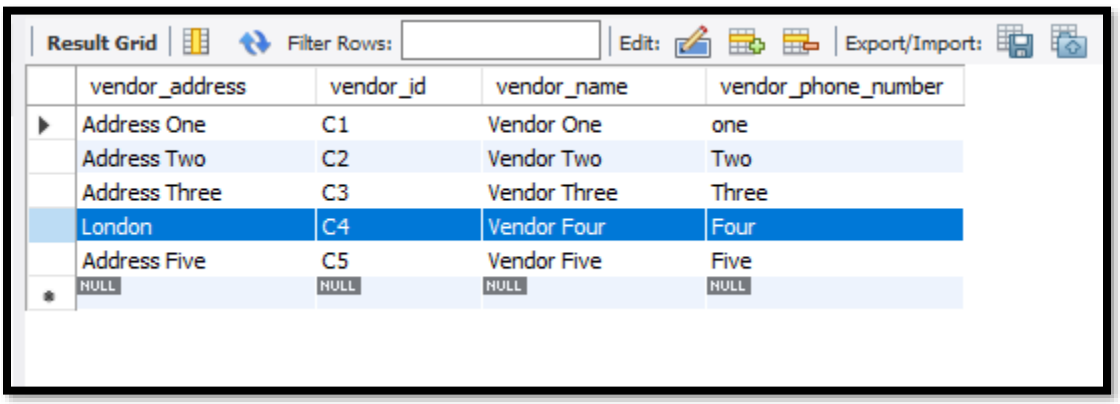


- Now, I want to update the “Address Four” to “London”.



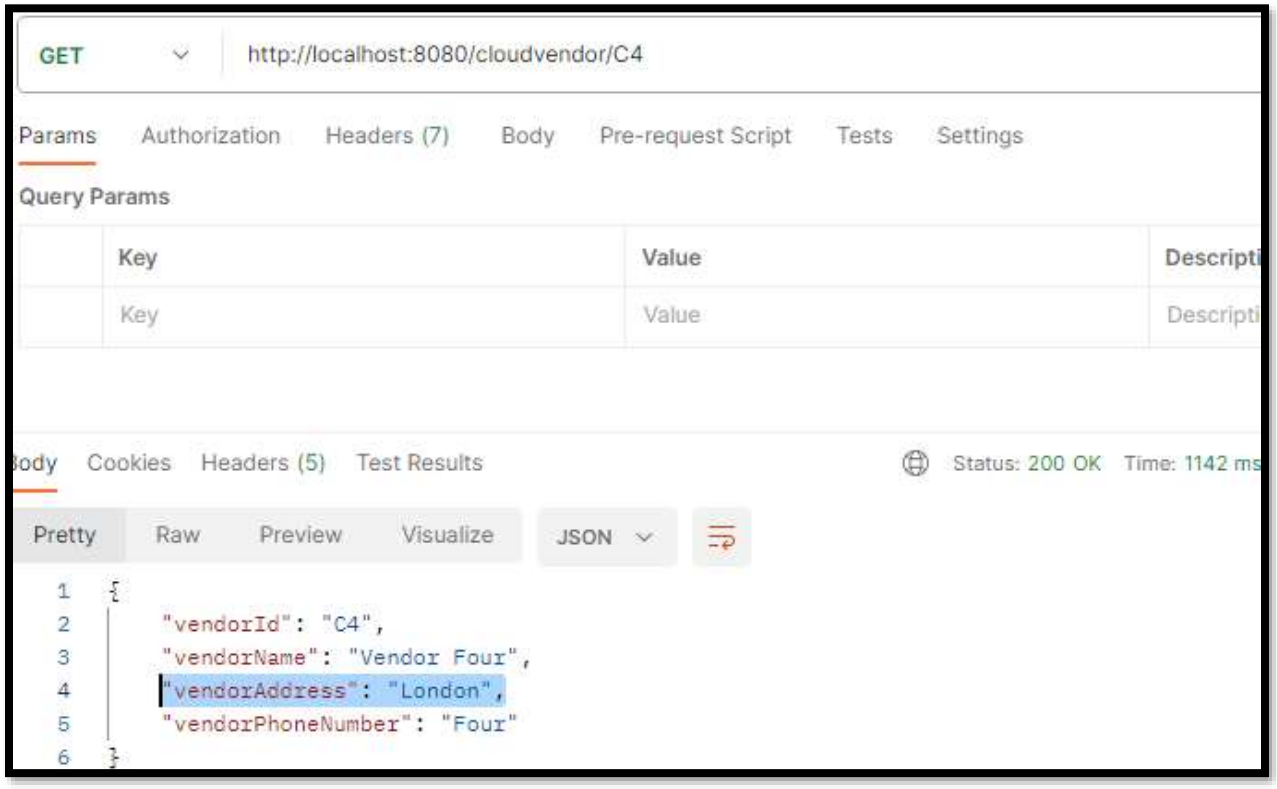


- Updated successfully in MySQL Workbench.

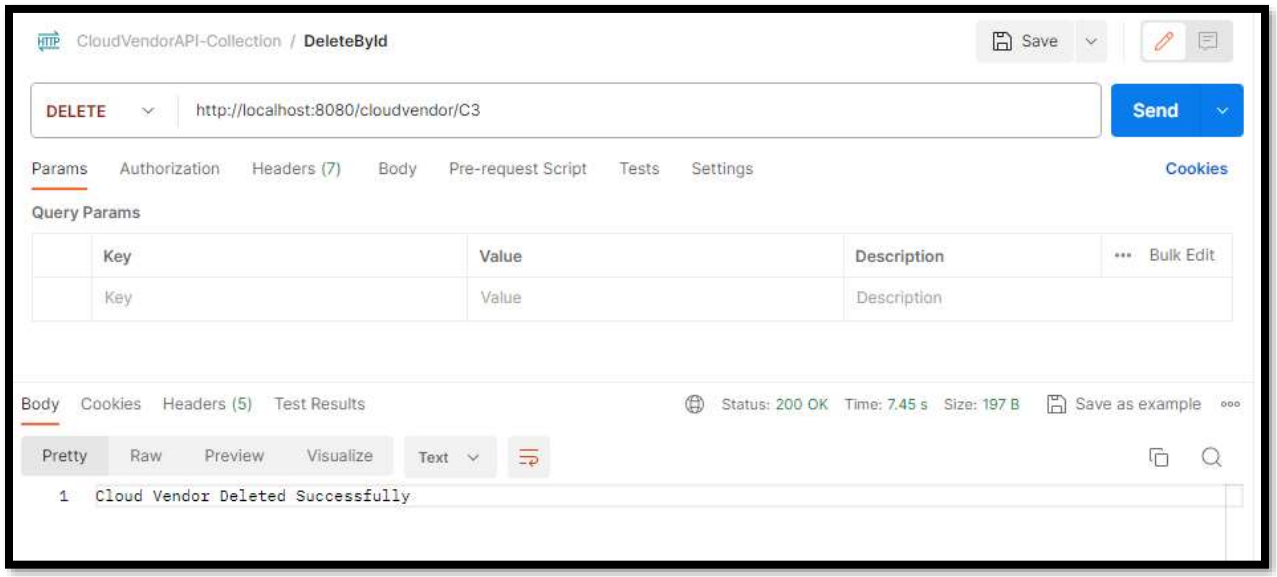


vendor_address	vendor_id	vendor_name	vendor_phone_number
Address One	C1	Vendor One	one
Address Two	C2	Vendor Two	Two
Address Three	C3	Vendor Three	Three
London	C4	Vendor Four	Four
Address Five	C5	Vendor Five	Five
NULL	NULL	NULL	NULL

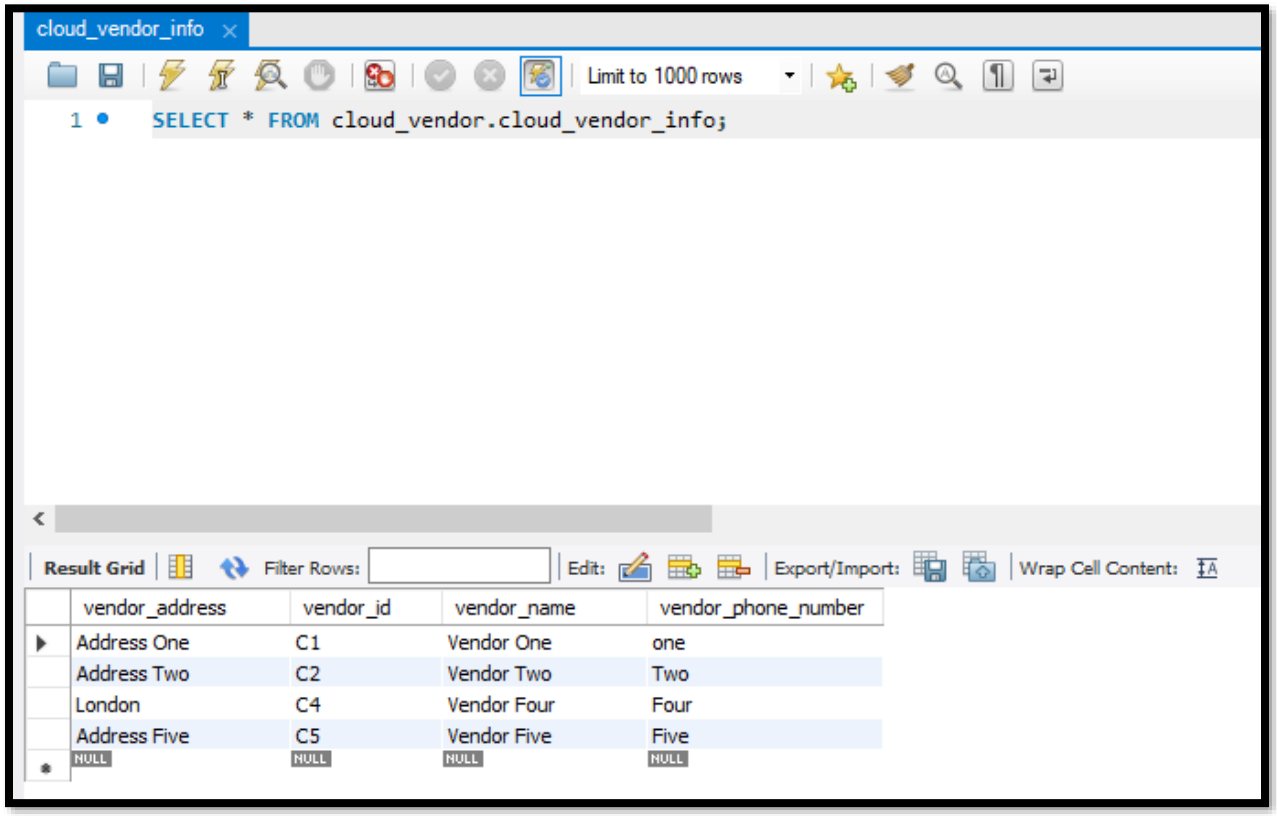
- Test by postman also, go to get by id for C4 and verify.



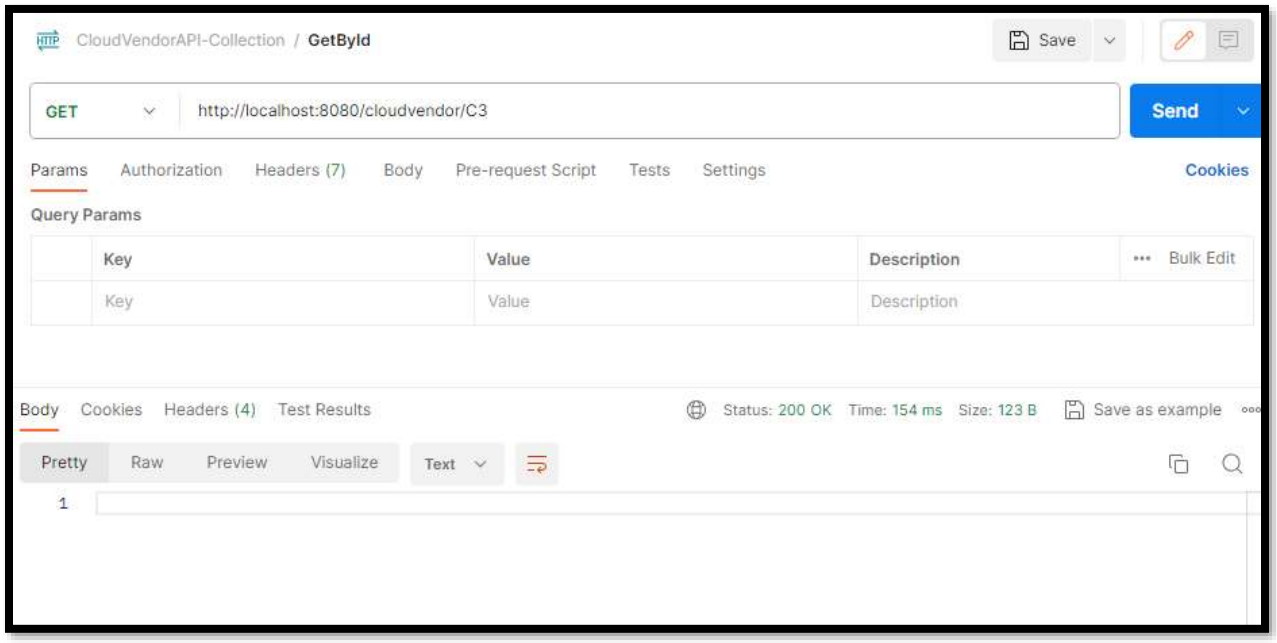
- Now. Let's Delete by Id. I want to delete C3.



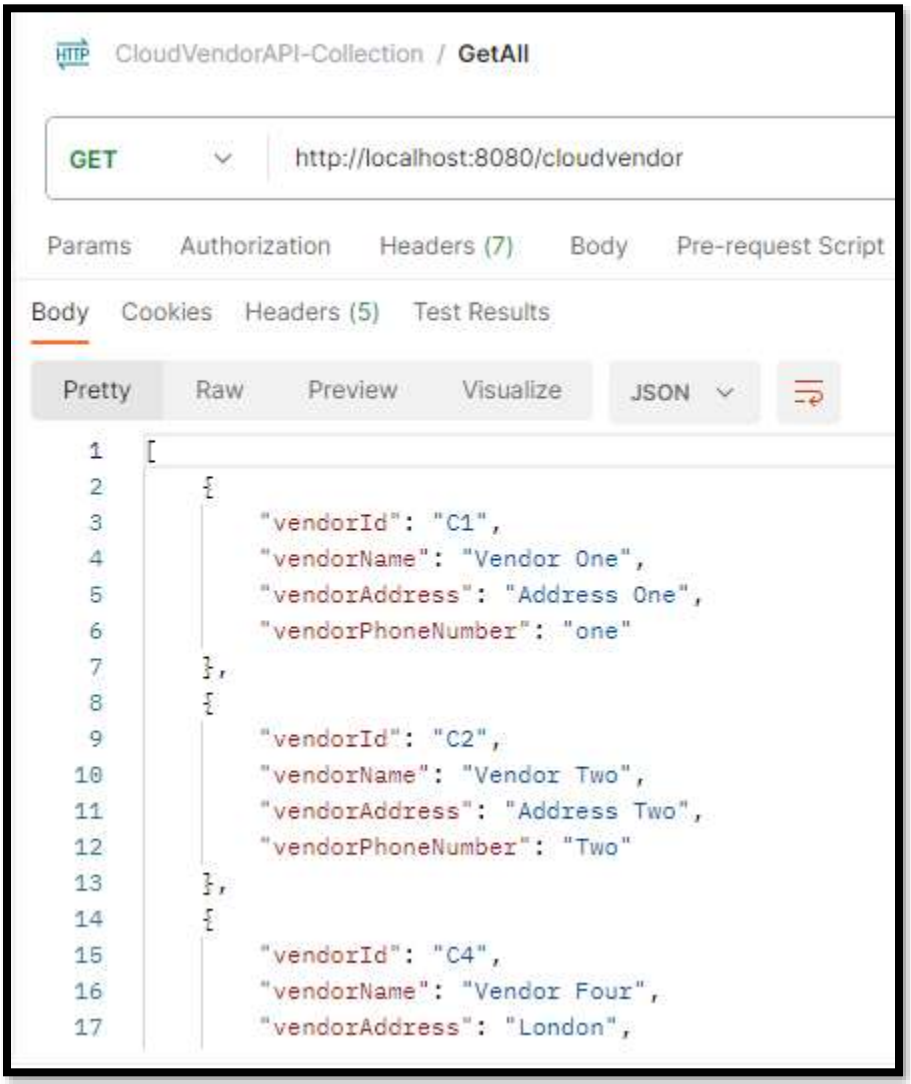
- Go back to database and see what happened.



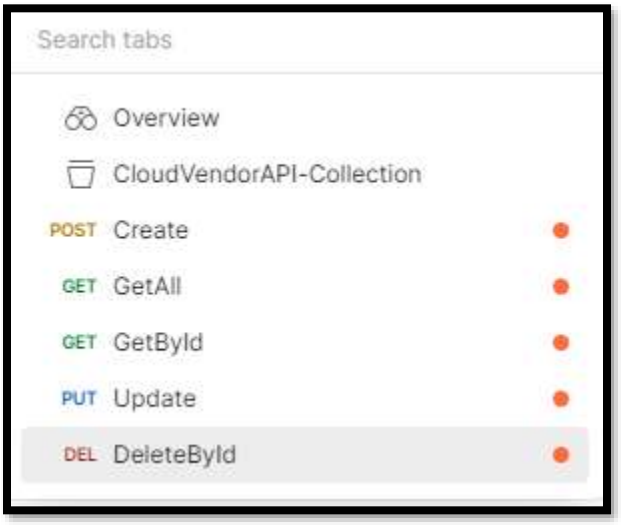
- Test with Postman as well, you will get nothing or timestamp error.



- In get All there would be no C3.



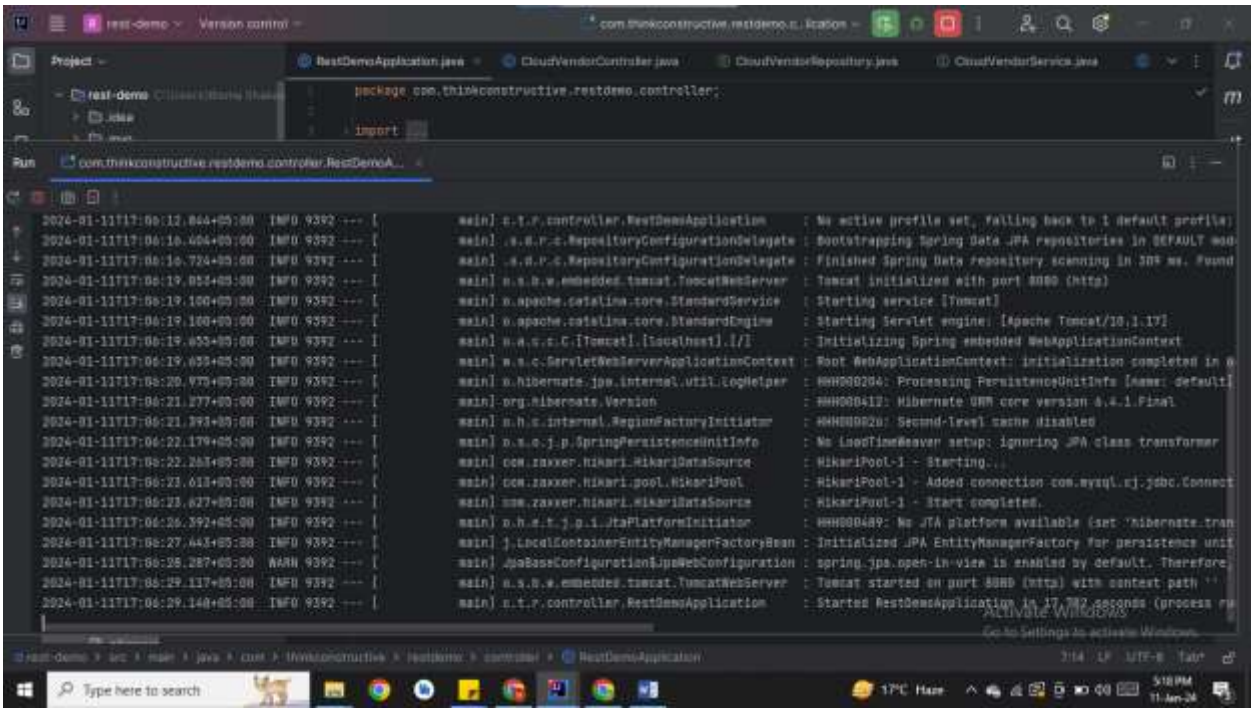
- So, we have tested all the CRUD Operations.



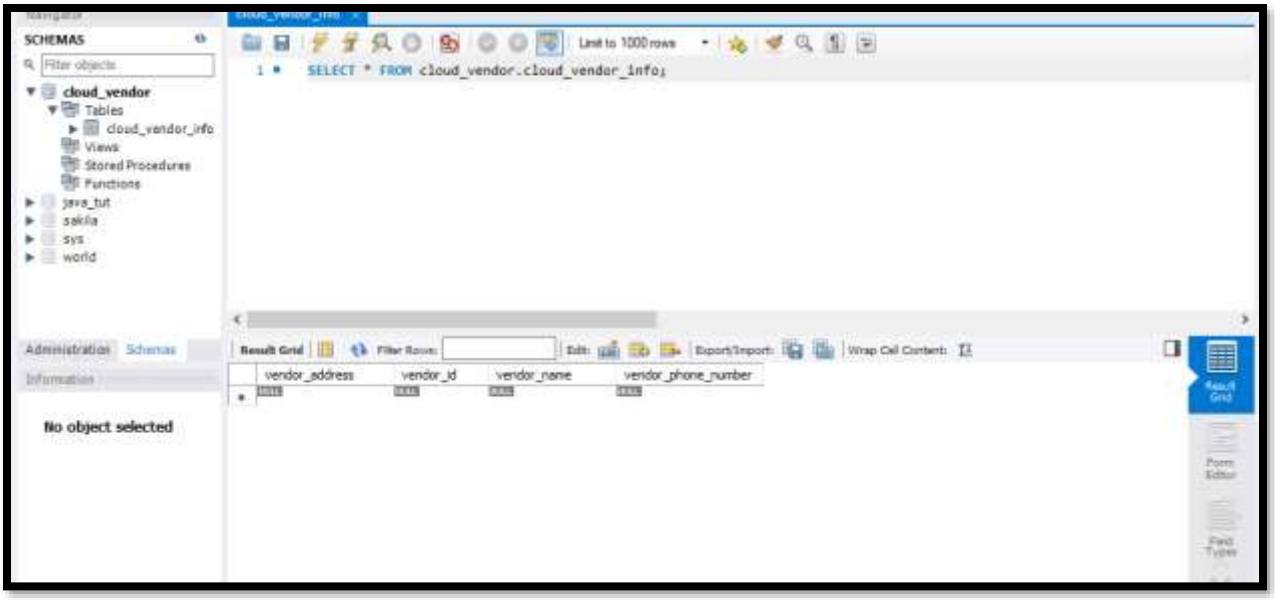
Session# 03

Exception Handling in Spring Boot REST API

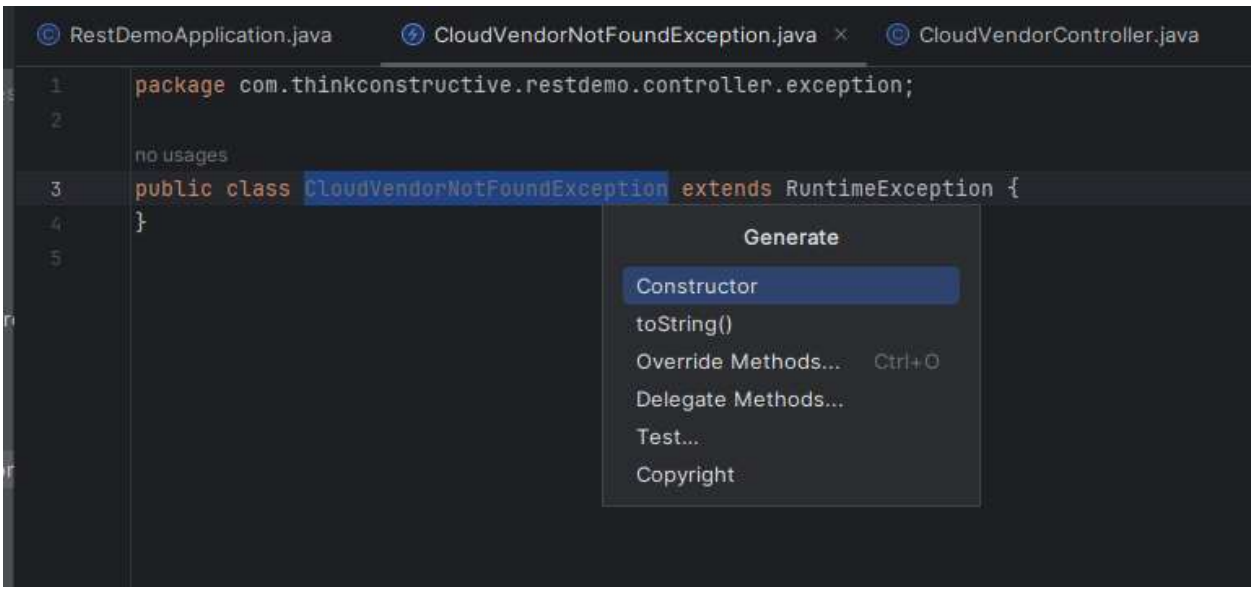
- Run the project created in last two sessions and check whether it runs successfully or not?
- It is running successfully on port 8080.

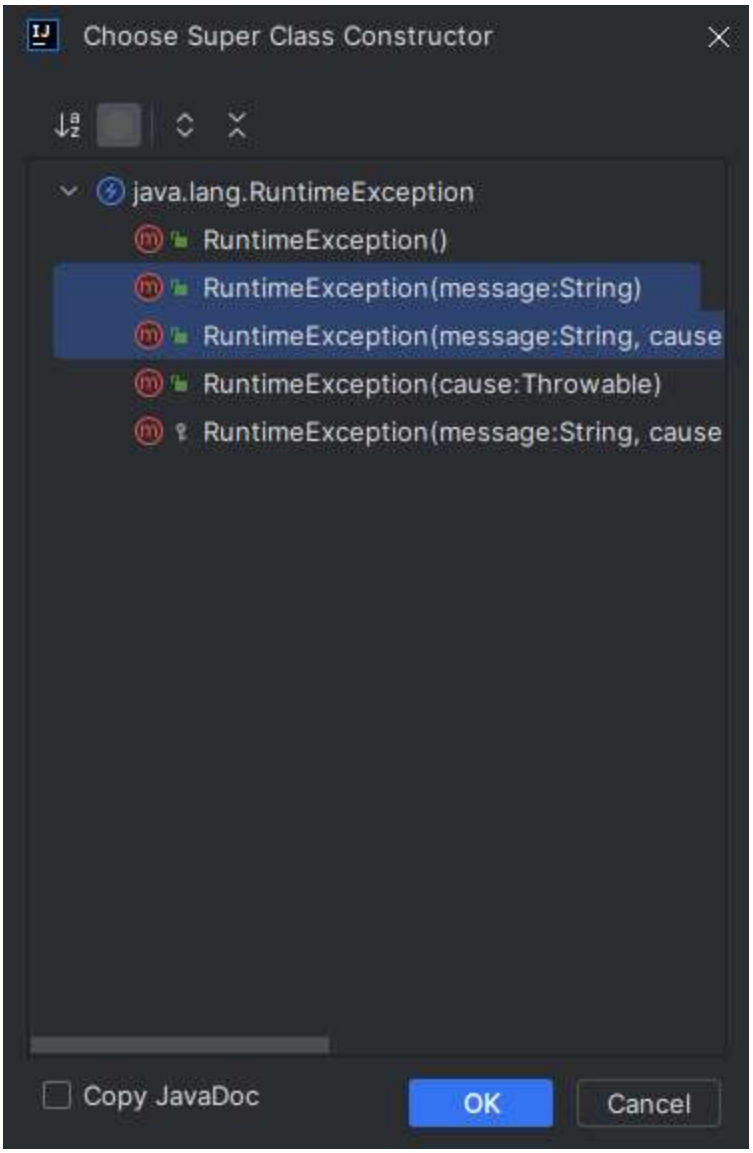


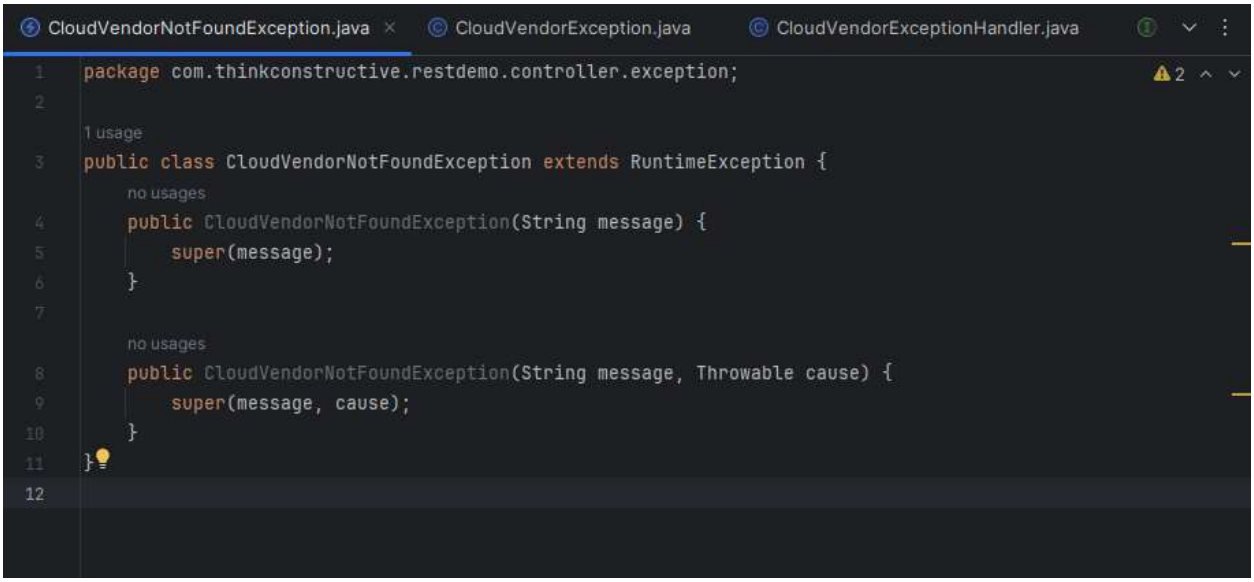
- Now, go to the MySQL workbench.
- There must be table created but currently that table should be empty because I have recently started my application.
- See, it does not have any data now.
- There is no cloud vendor existing in my cloud_vendor_info table.



- How to get a proper error message that actually tells what problem has happened?



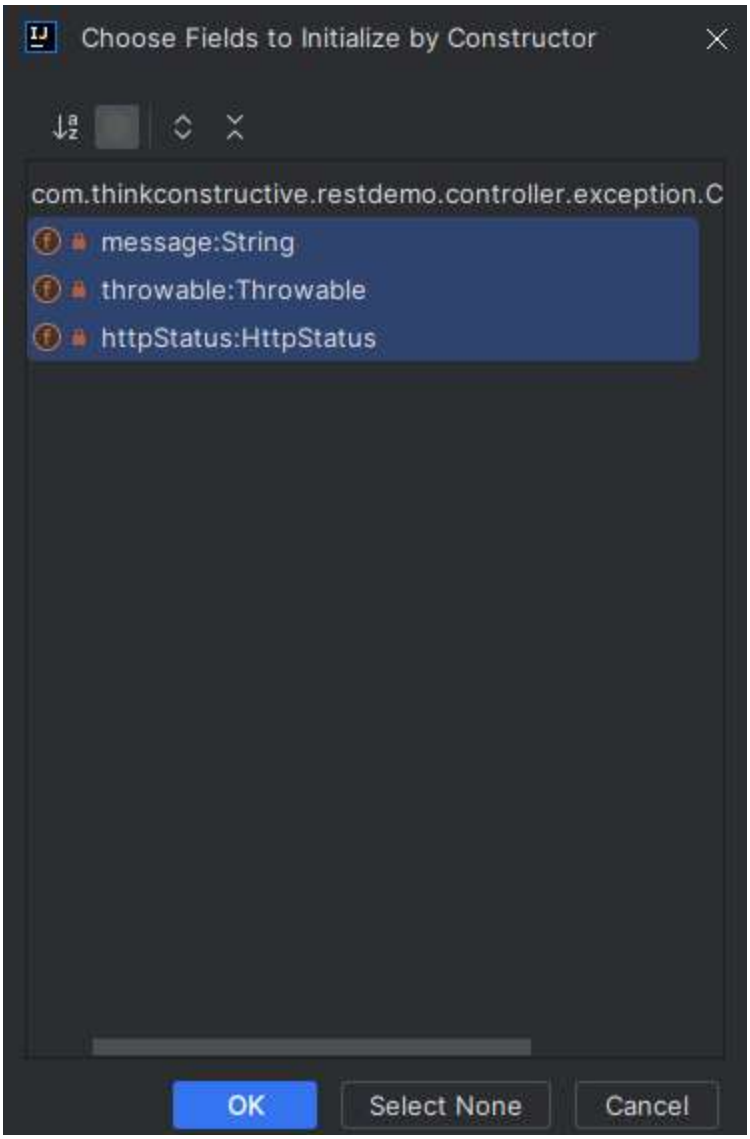

Choose Super Class Constructor
java.lang.RuntimeException
RuntimeException()
RuntimeException(message:String)
RuntimeException(message:String, cause
RuntimeException(cause:Throwable)
RuntimeException(message:String, cause
Copy JavaDoc
OK
Cancel

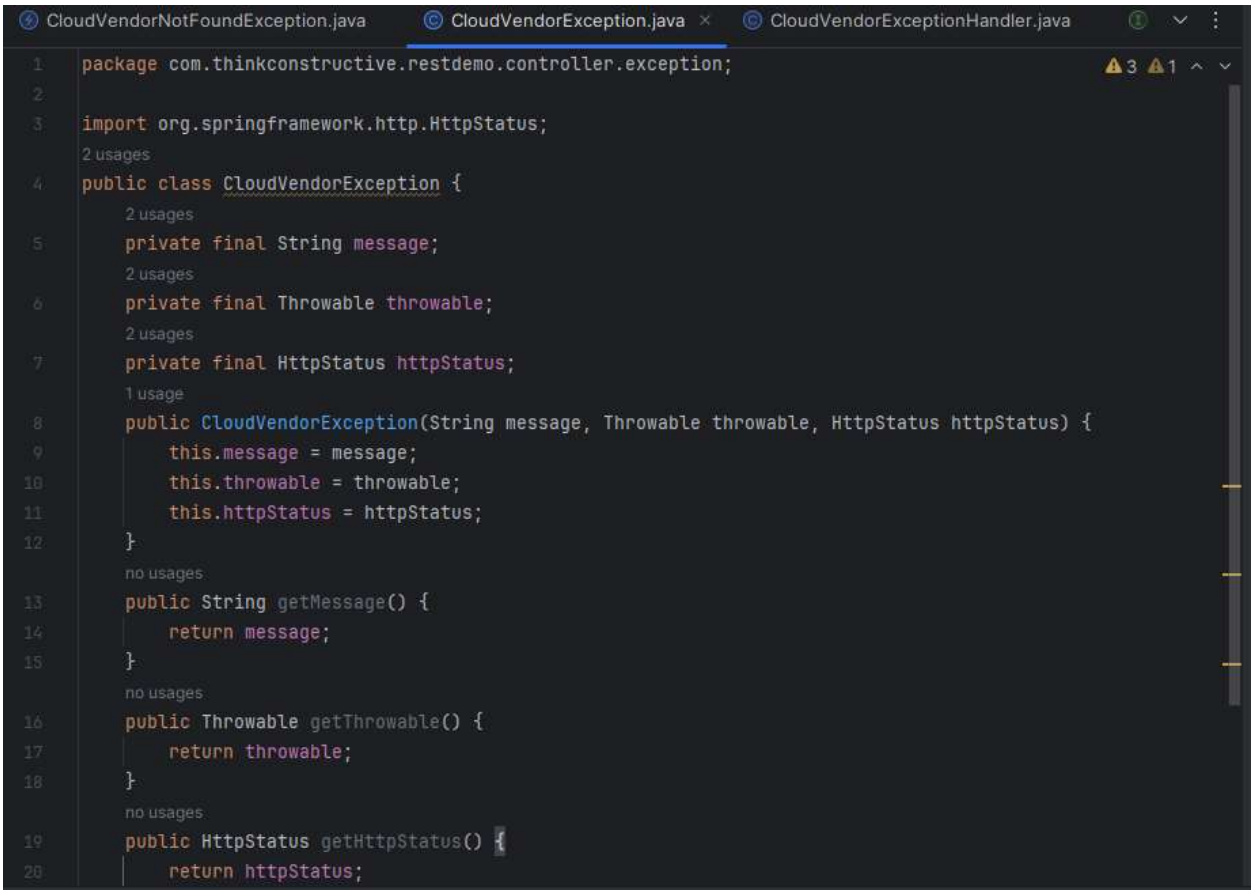


Code:

```
package com.thinkconstructive.restdemo.controller.exception;

public class CloudVendorNotFoundException extends RuntimeException {
    public CloudVendorNotFoundException(String message) {
        super(message);
    }

    public CloudVendorNotFoundException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

- Create another class.

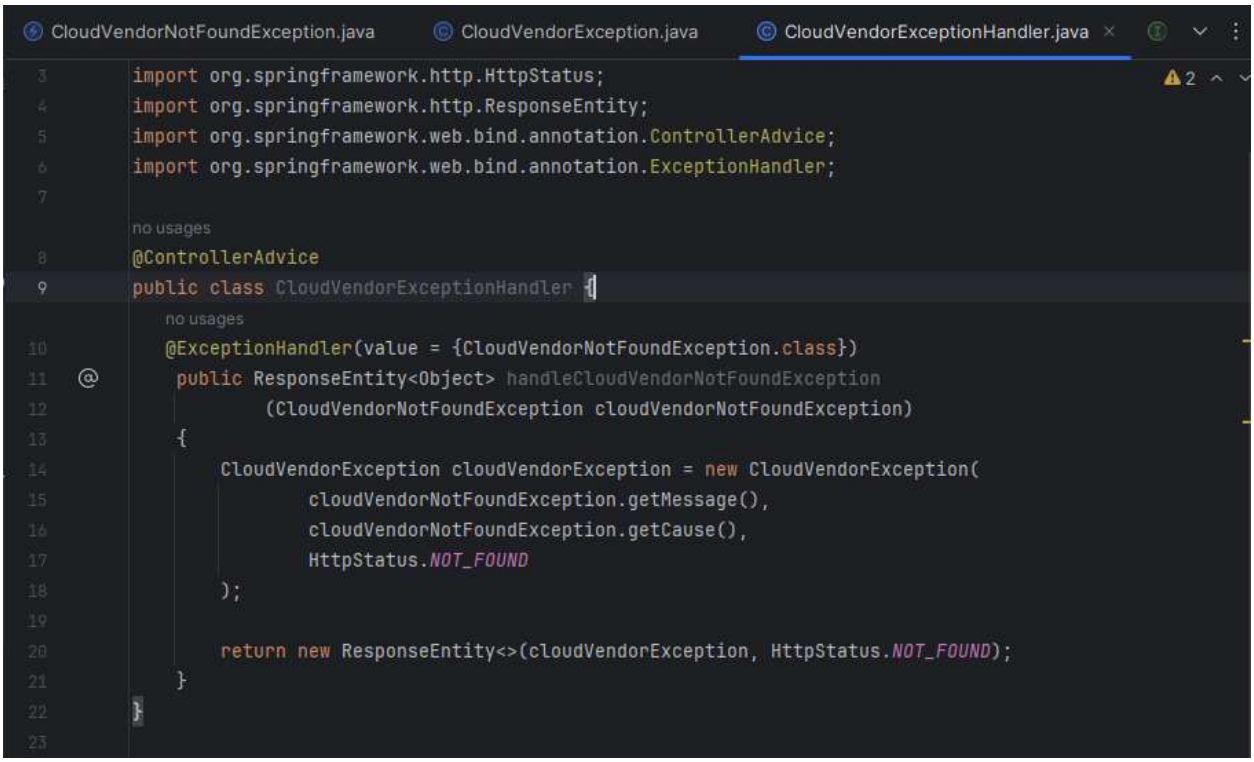

Choose Fields to Initialize by Constructor
com.thinkconstructive.restdemo.controller.exception.C
message:String
throwable:Throwable
httpStatus:HttpStatus
OK
Select None
Cancel



Code:

```
package com.thinkconstructive.restdemo.controller.exception;

import org.springframework.http.HttpStatus;

public class CloudVendorException {
    private final String message;
    private final Throwable throwable;
    private final HttpStatus httpStatus;

    public CloudVendorException(String message, Throwable throwable,
HttpStatus httpStatus) {
        this.message = message;
        this.throwable = throwable;
        this.httpStatus = httpStatus;
    }

    public String getMessage() {
        return message;
    }

    public Throwable getThrowable() {
        return throwable;
    }

    public HttpStatus getHttpStatus() {
```

```
        return httpStatus;
    }


}
```

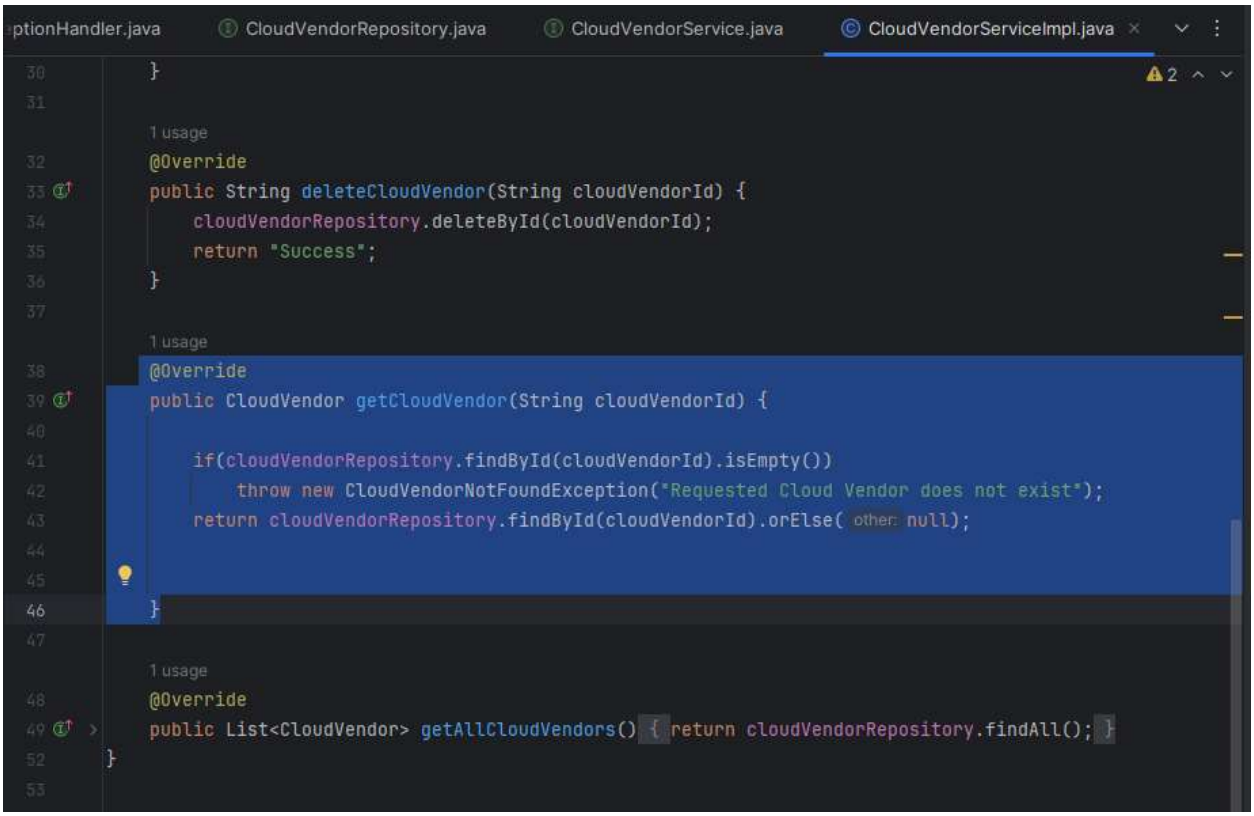


Code:

```
package com.thinkconstructive.restdemo.controller.exception;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
```

```
public class CloudVendorExceptionHandler {
   @ExceptionHandler(value = {CloudVendorNotFoundException.class})
    public ResponseEntity<Object> handleCloudVendorNotFoundException
            (CloudVendorNotFoundException cloudVendorNotFoundException)
    {
        CloudVendorException cloudVendorException = new CloudVendorException(
                cloudVendorNotFoundException.getMessage(),
                cloudVendorNotFoundException.getCause(),
                HttpStatus.NOT_FOUND
        );

        return new ResponseEntity<>(cloudVendorException,
HttpStatus.NOT_FOUND);
    }
}
```

- Now Go to the service layer and getCloudVendor Method and make changes in it.

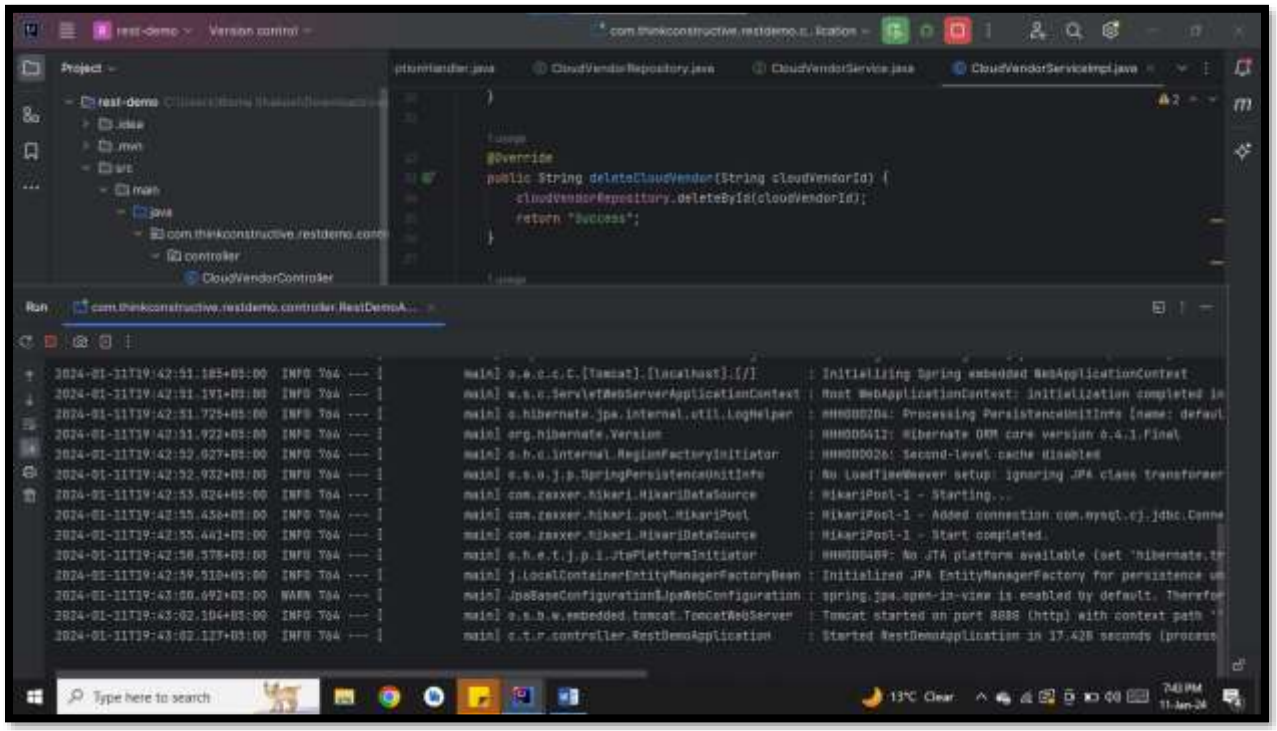


- Do these changes

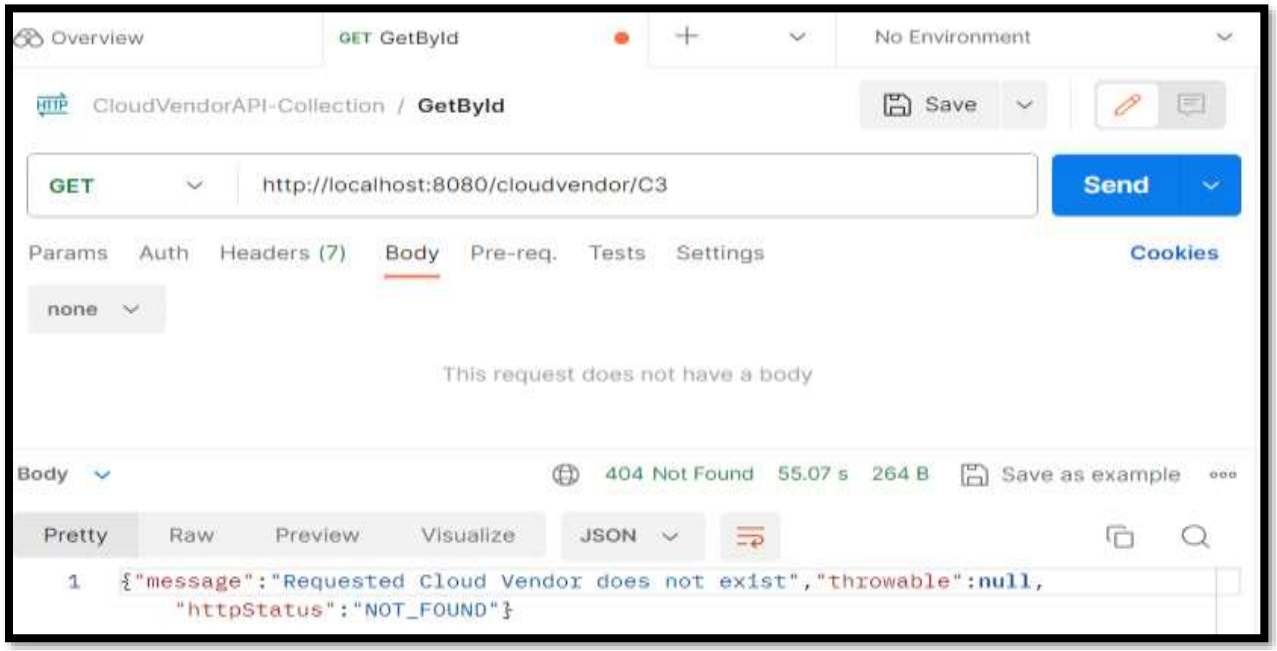
```
@Override
public CloudVendor getCloudVendor(String cloudVendorId) {

    if(cloudVendorRepository.findById(cloudVendorId).isEmpty())
        throw new CloudVendorNotFoundException("Requested Cloud Vendor does
not exist");
    return cloudVendorRepository.findById(cloudVendorId).orElse(null);
```

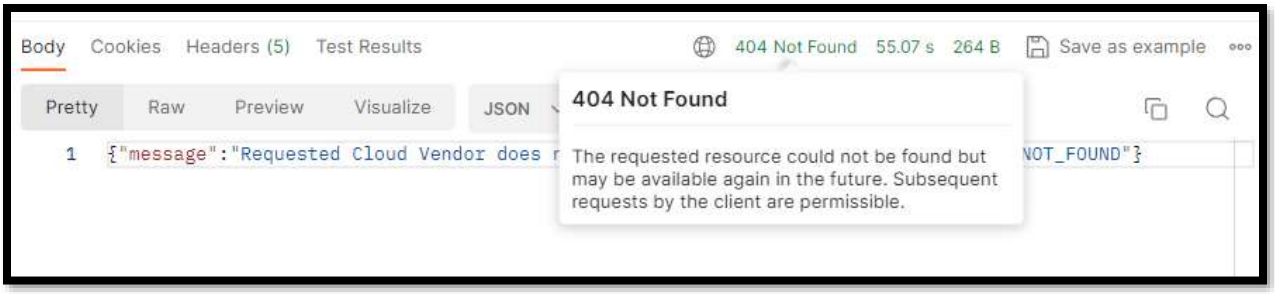
```
}
```

- Now, again run the application.

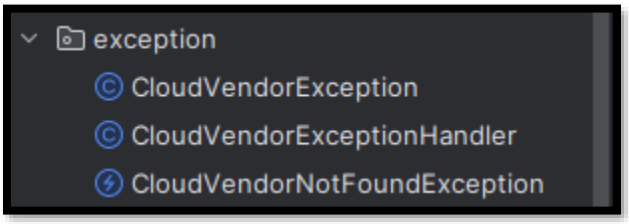


- Go to Postman. Instead of getting timestamp error message you will get the following.





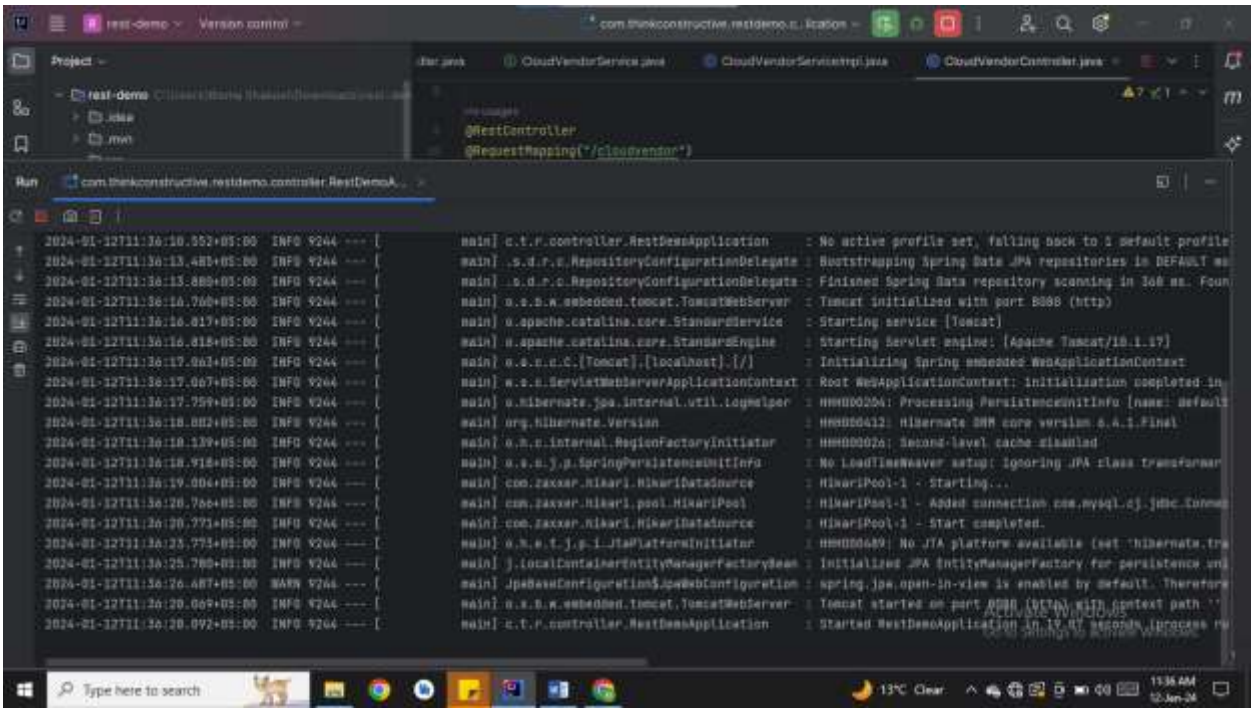
- This is how exception handling could be done with spring boot rest API Application.
- We have added the custom exception handling.



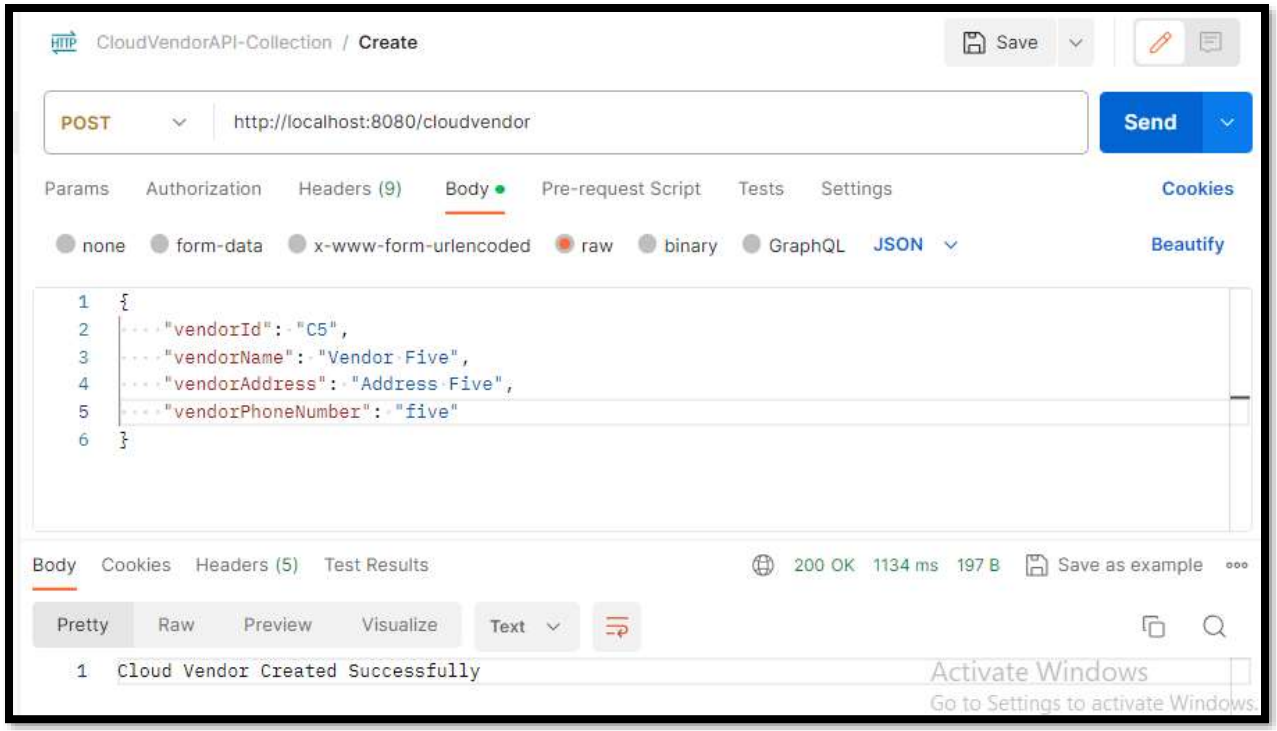
Session# 04

Java Spring Boot REST API JSON Response Handling

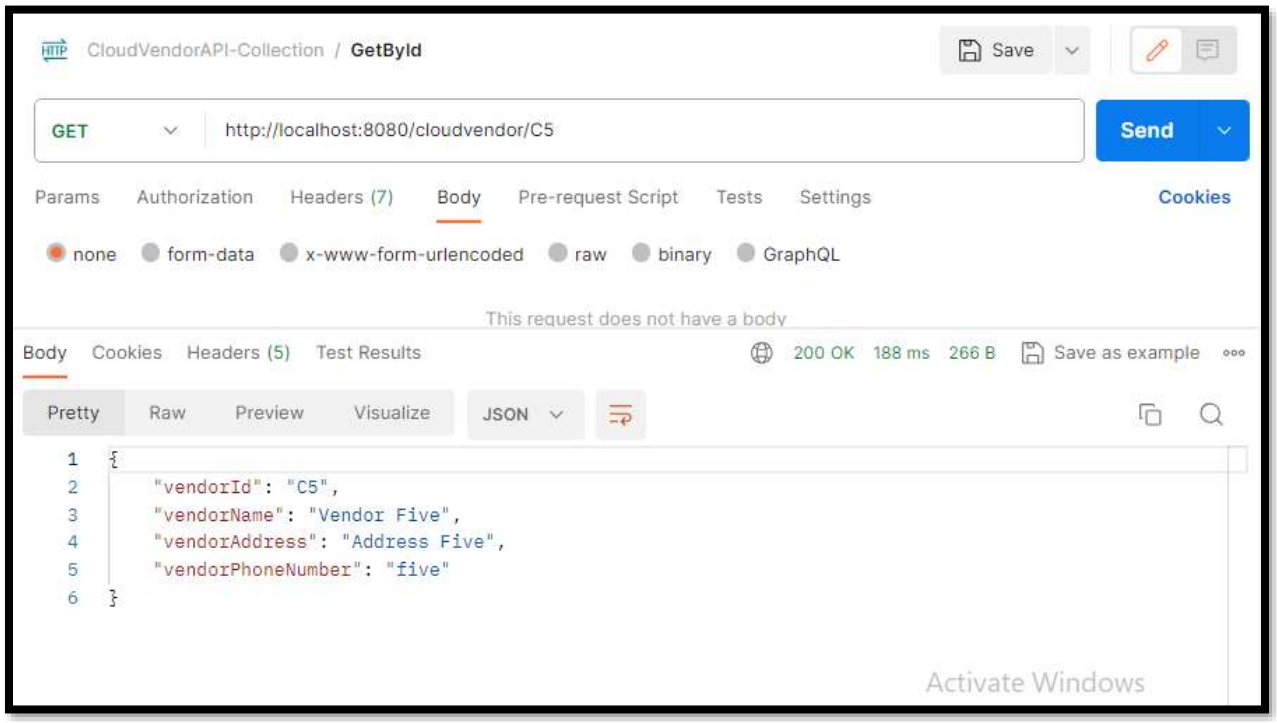
- Run the Application.



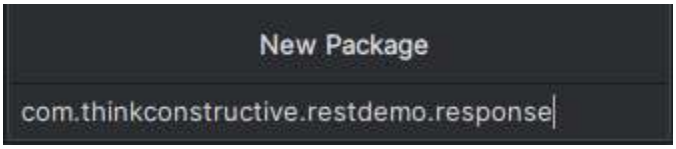
- Now, go to postman, first we will create one cloud vendor.



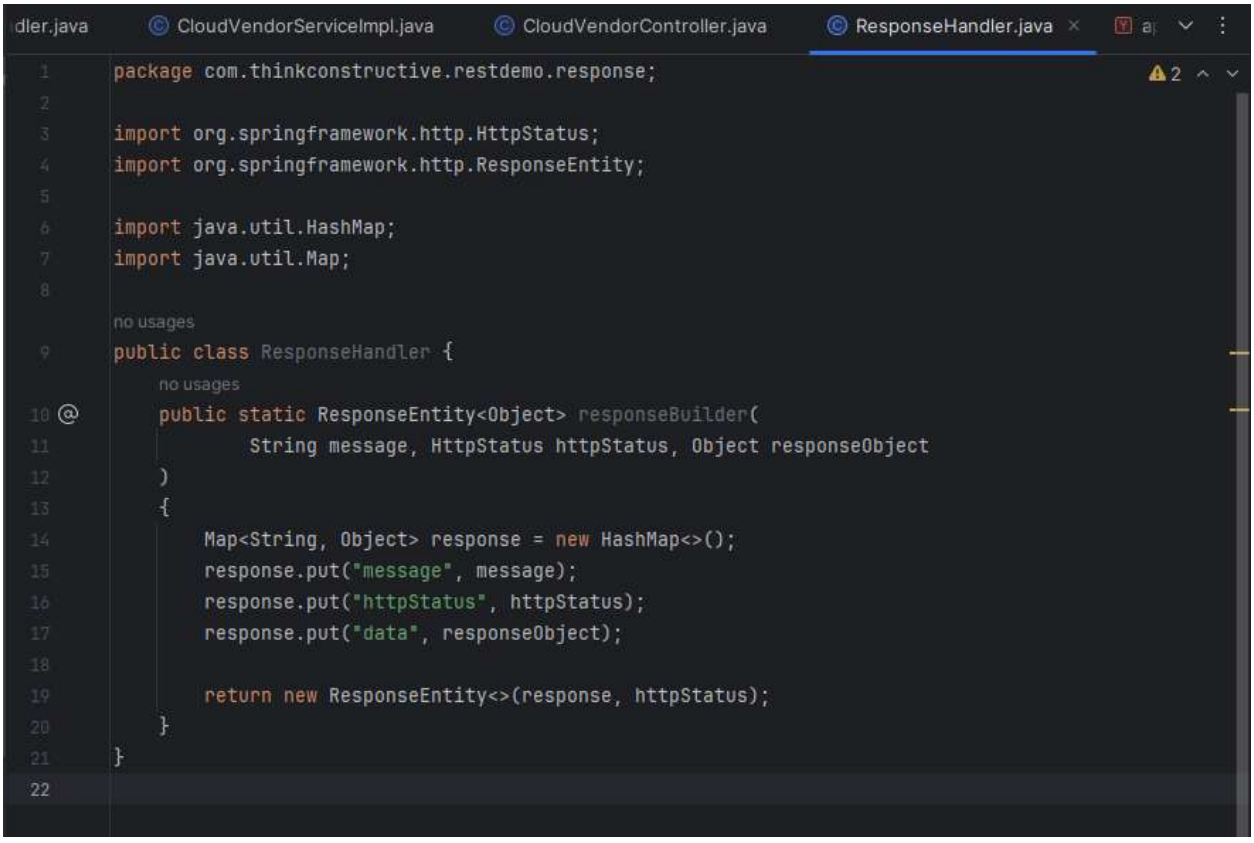
- Query C5 Get by ID.



- In addition to the above result I would also want to capture Http status code and a proper message. In order to do this let's do some code changes.
- In order to get the custom response, we need to add a package.



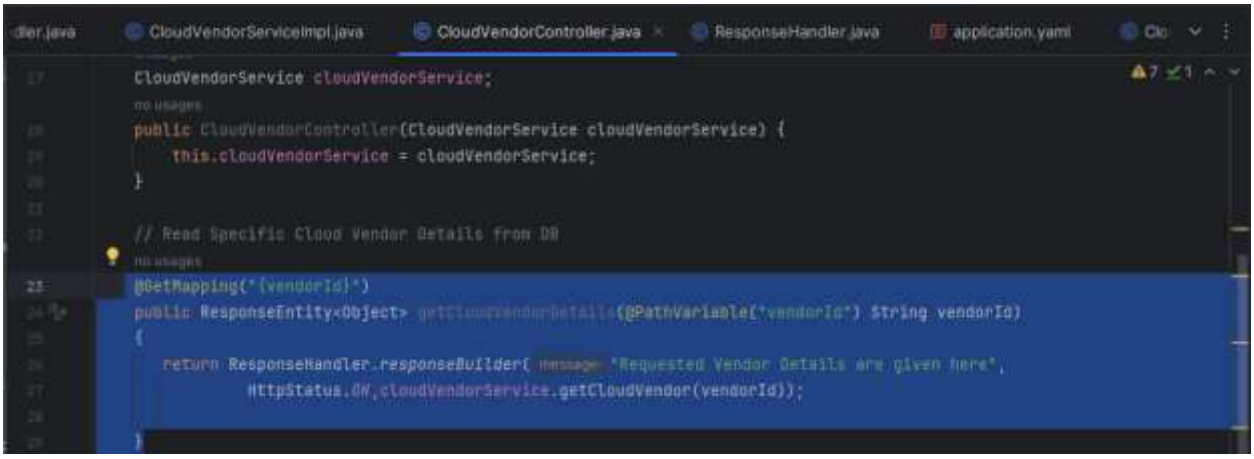
- Inside this response add a class.



Code:

```
package com.thinkconstructive.restdemo.response;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import java.util.HashMap;
import java.util.Map;

public class ResponseHandler {
    public static ResponseEntity<Object> responseBuilder(
            String message, HttpStatus httpStatus, Object responseObject
    )
    {
        Map<String, Object> response = new HashMap<>();
        response.put("message", message);
        response.put("httpStatus", httpStatus);
        response.put("data", responseObject);

        return new ResponseEntity<>(response, httpStatus);
    }
}
```

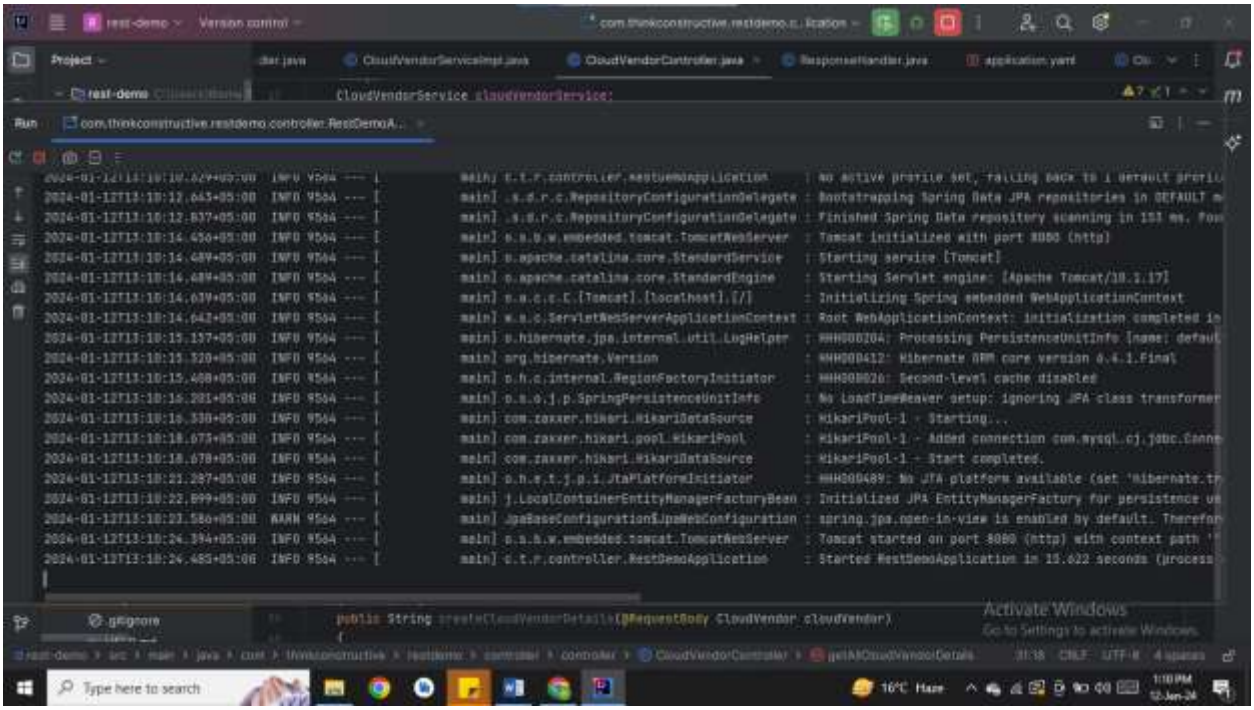
- Now call the ResponseBuilder Method in controller layer.



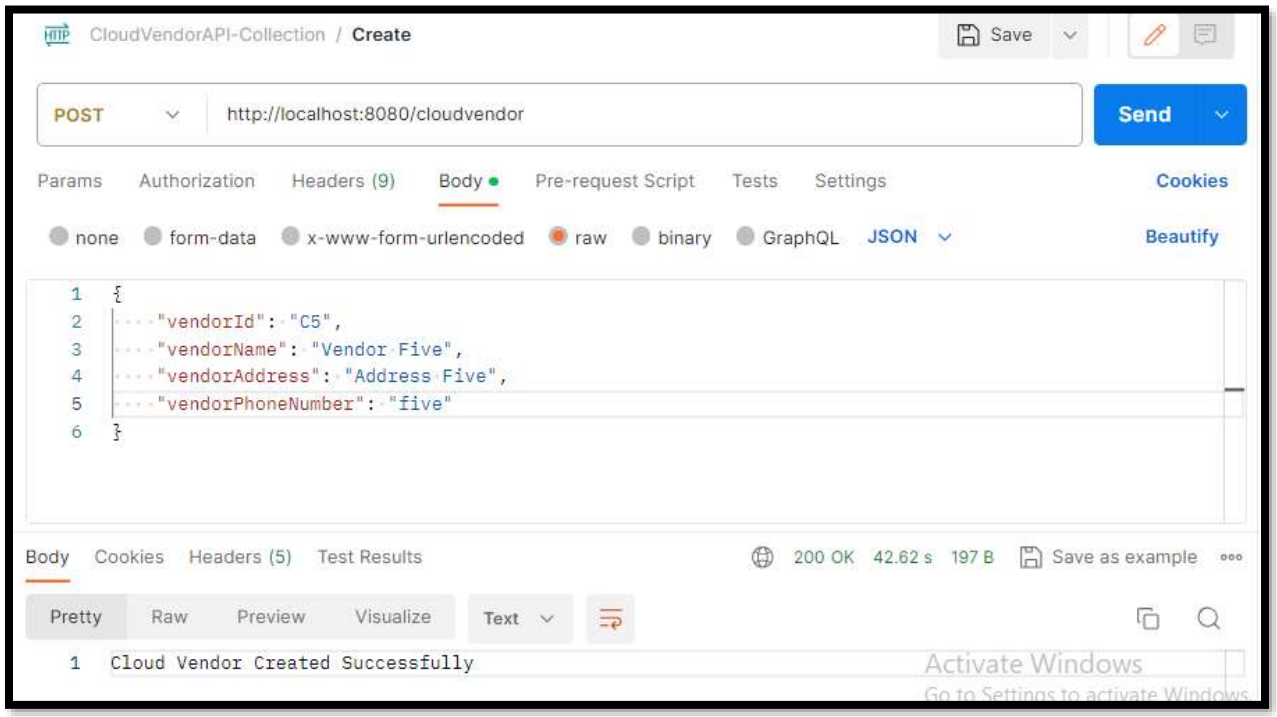
Code:

```
@GetMapping("{vendorId}")
public ResponseEntity<Object> getCloudVendorDetails(@PathVariable("vendorId")
String vendorId)
{
   return ResponseHandler.responseBuilder("Requested Vendor Details are given
here",
            HttpStatus.OK,cloudVendorService.getCloudVendor(vendorId));

}
```

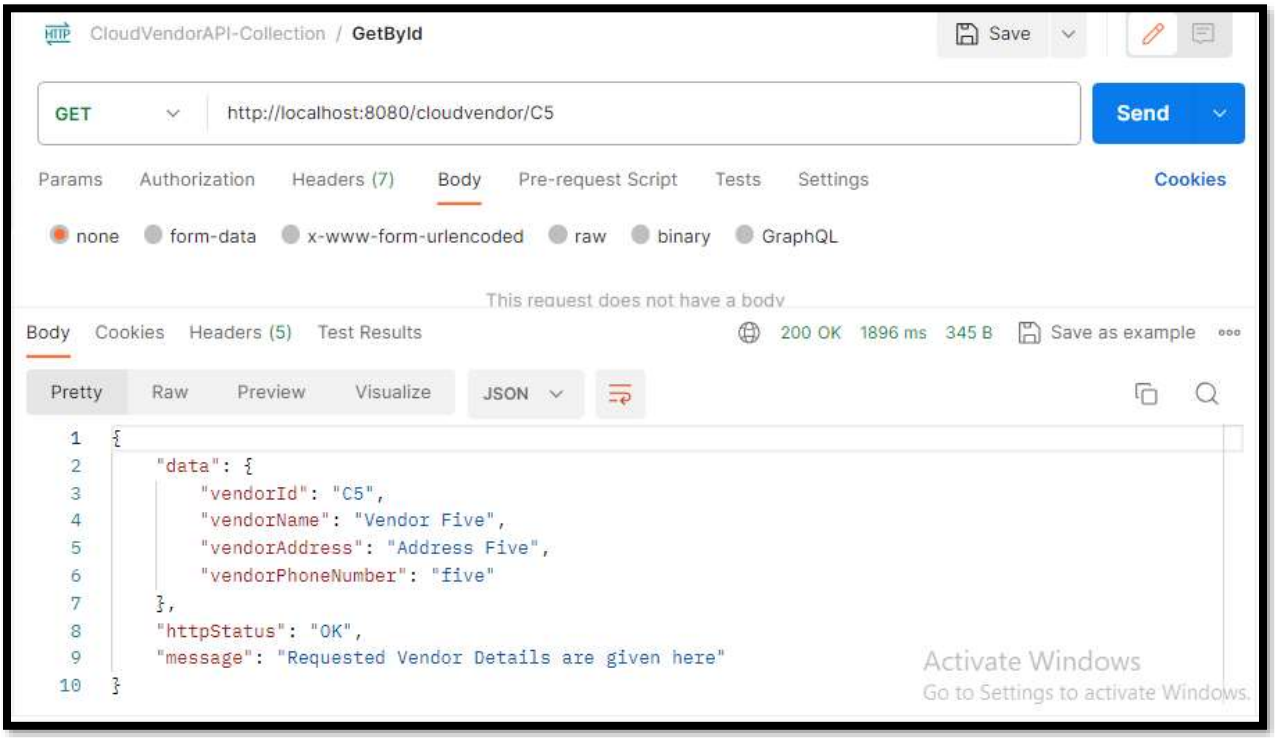
- Start the spring boot application.



- Create again



- Query C5 with get by Id.



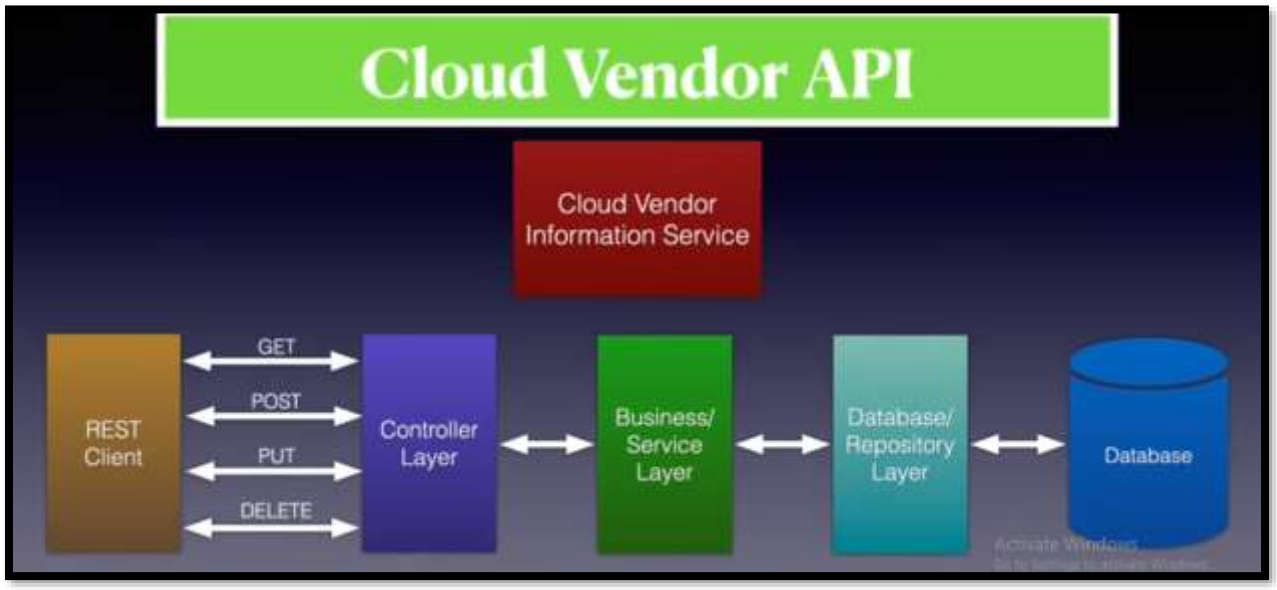
- Transformed the generic response to custom response successfully.

Session# 05

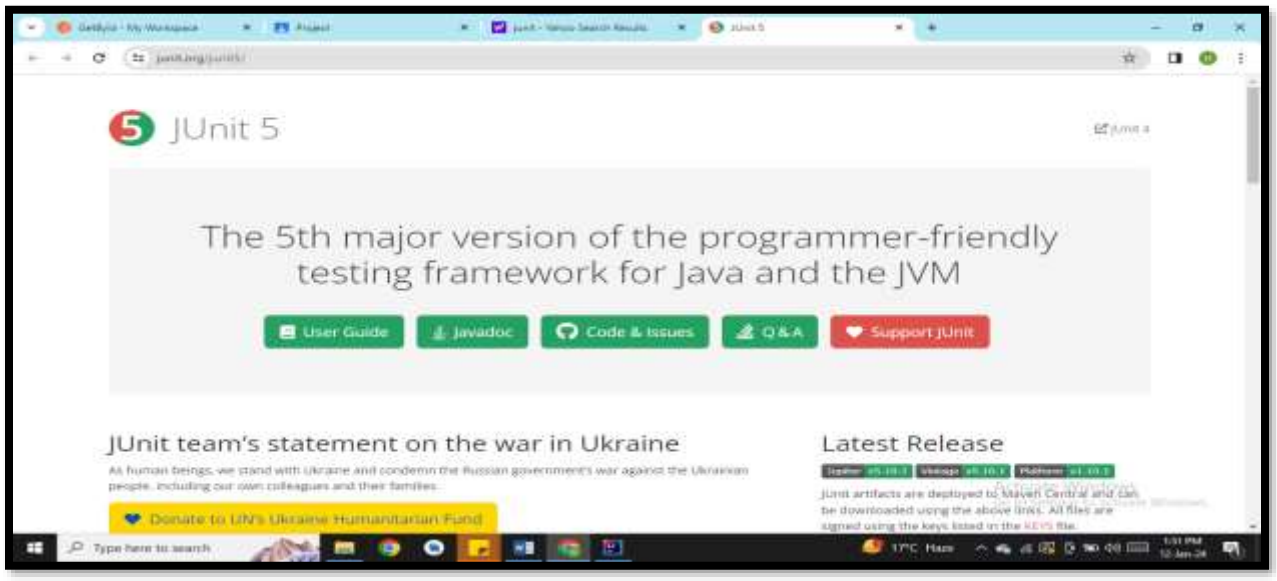
Master Unit Testing Java Spring Boot REST API Application in One Shot

- Unit testing java spring boot rest api application **using Junit , AssertJ , Mockito , H2Database , Springframework test package libraries / frameworks**.

Unit Testing:



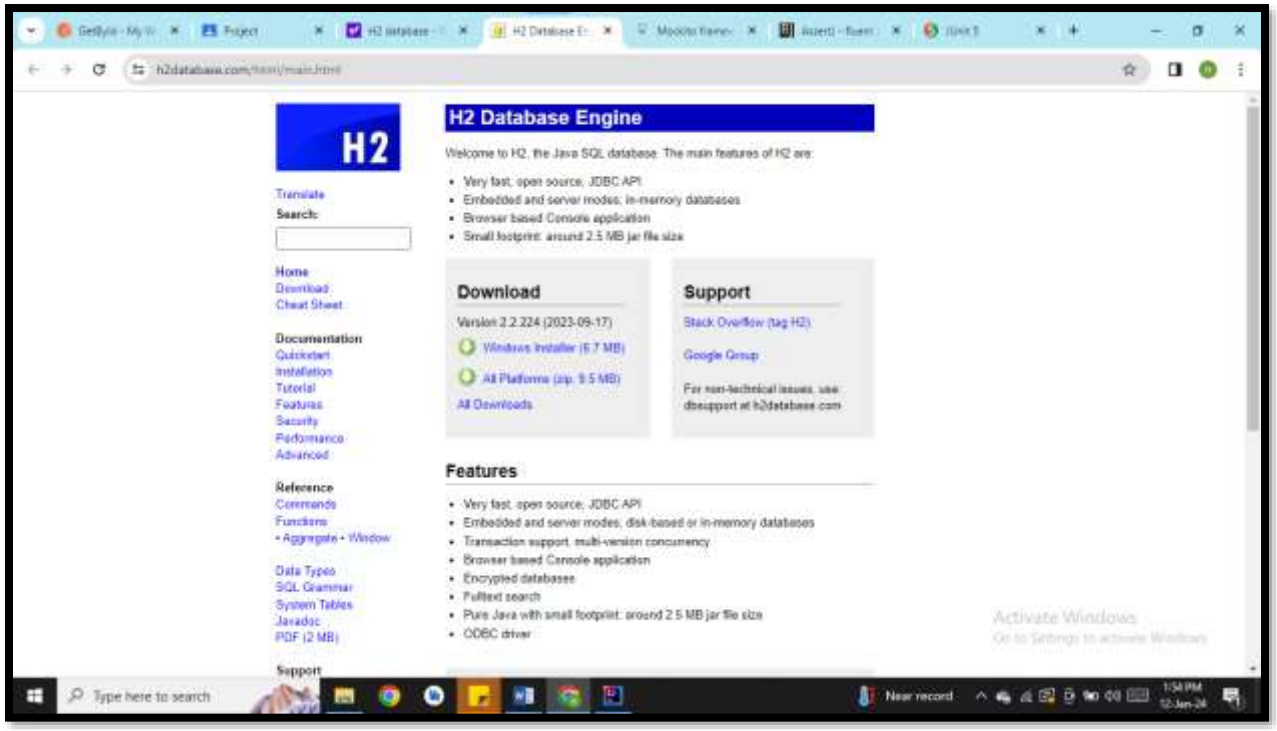
Junit 5 Framework



AssertJ Framework (java library)

Mockito Framework

H2 Database

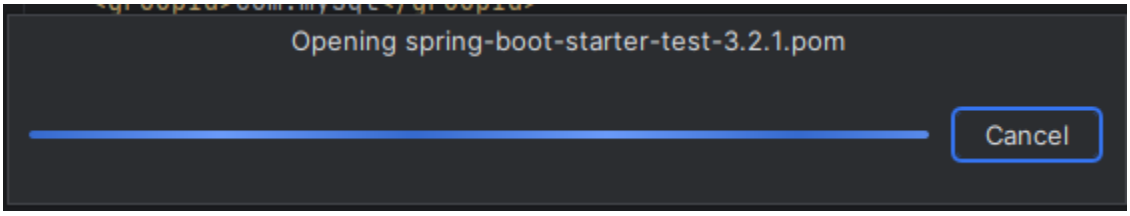


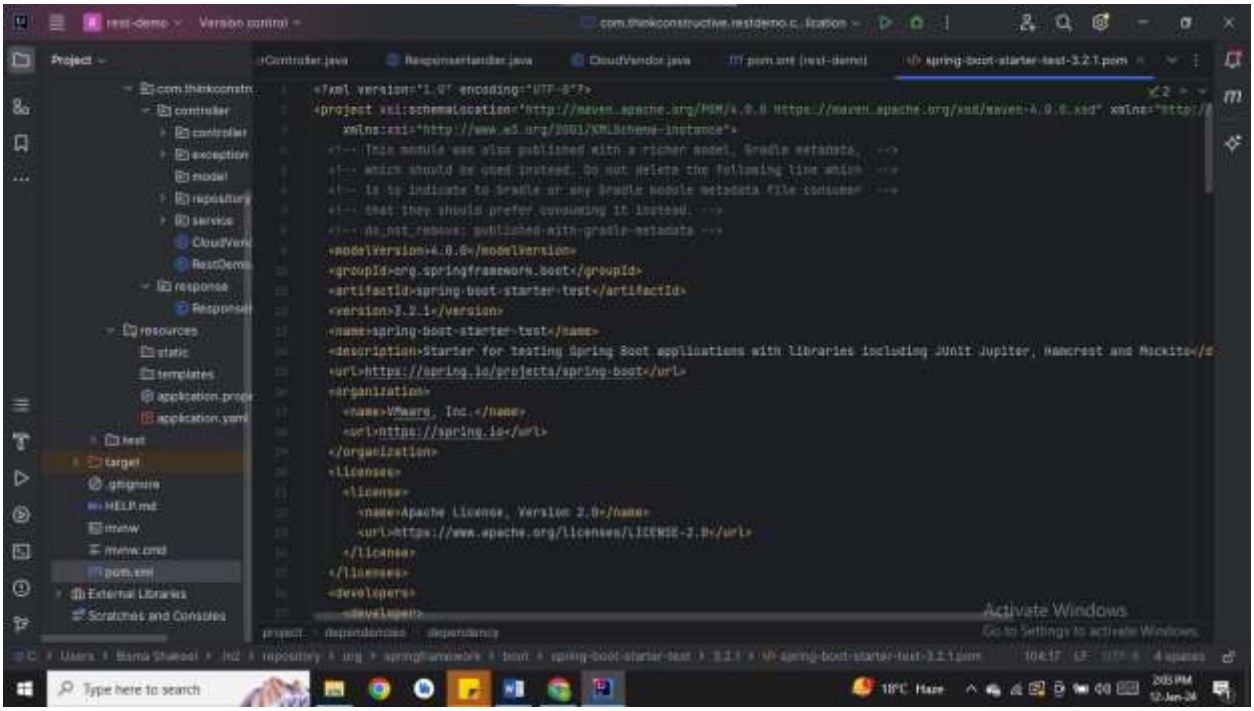
Open IntelliJ IDEA Editor Spring Boot API Project.

- Check in pom.xml that you have test dependency.

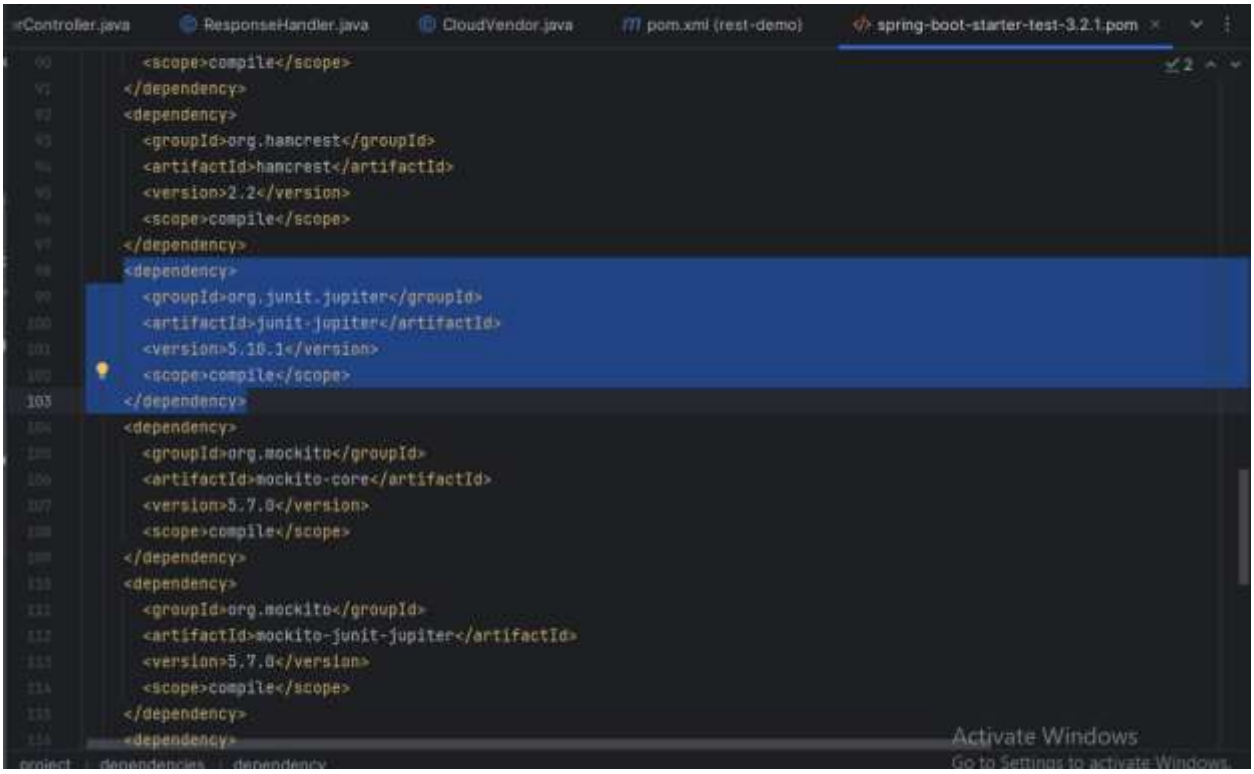
```
CloudVendorController.java  ResponseHandler.java  application.yaml  CloudVendor.java  pom.xml (rest-demo)

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

- Press ctrl and click spring-boot-starter-test.

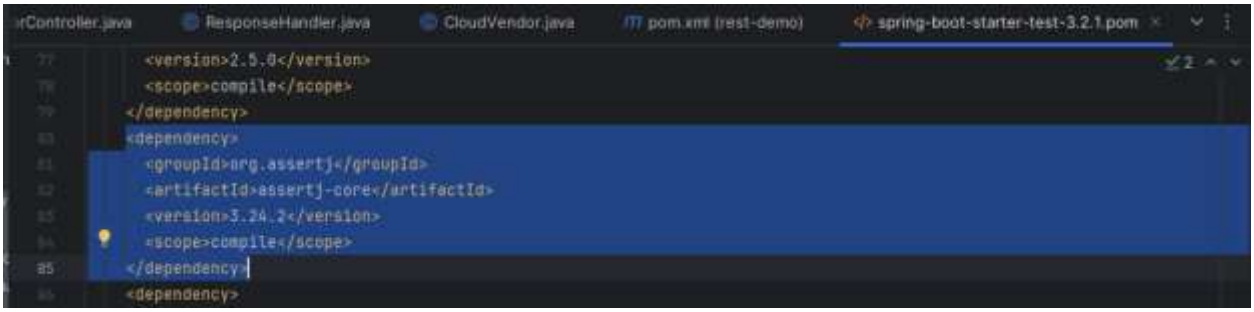




- Junit dependency is already included.

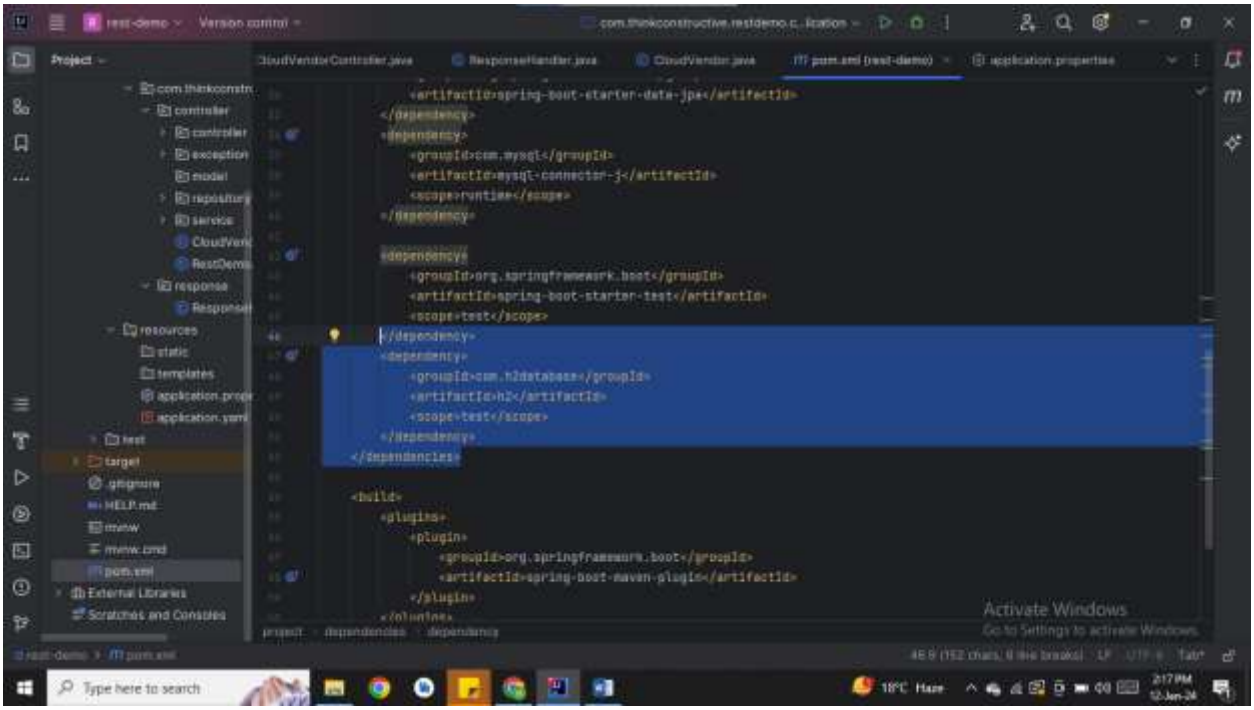


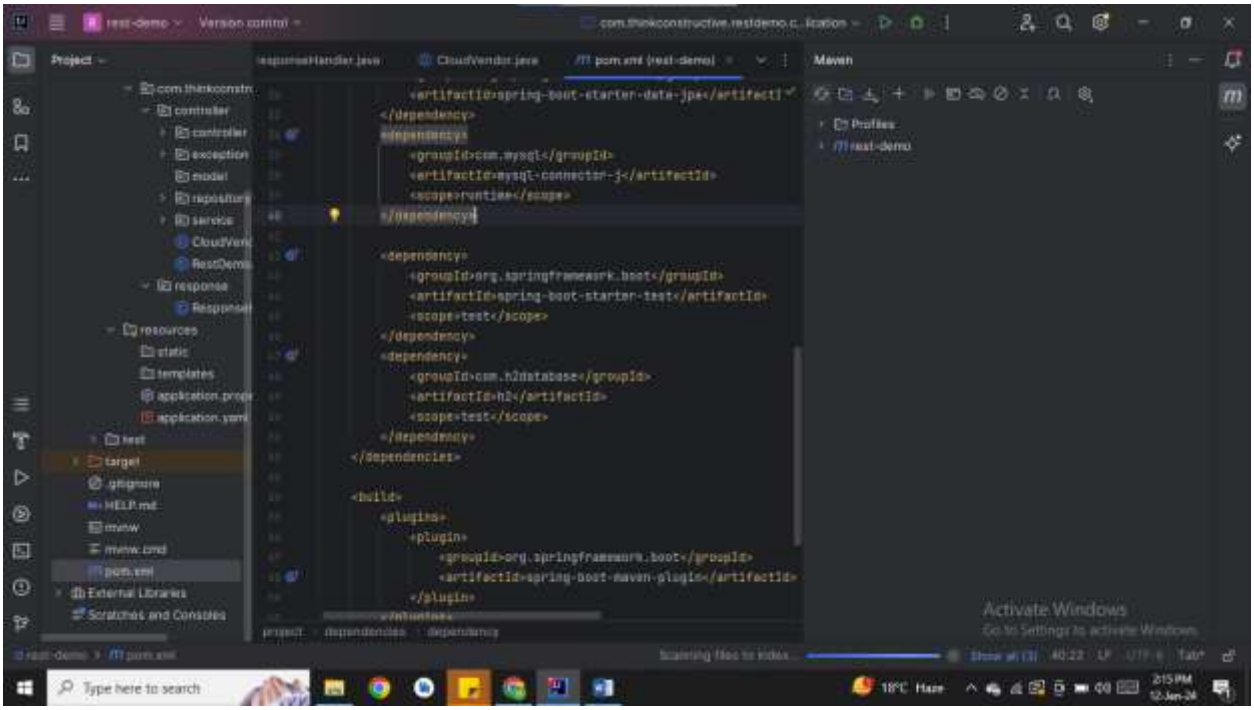
- Assertj dependency is already included.



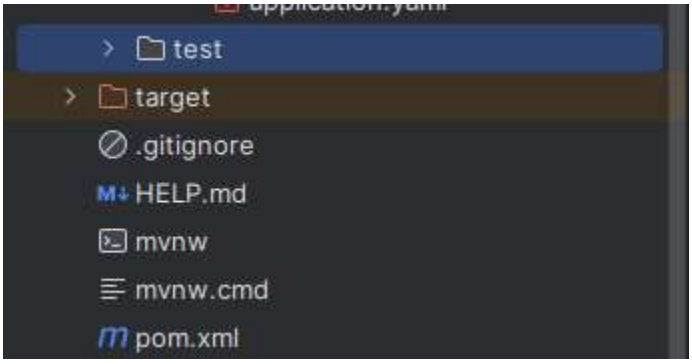
- Mockito dependency is already included.

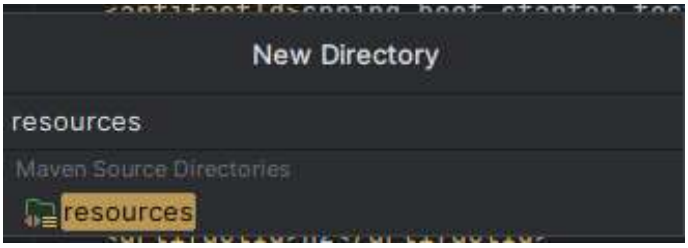
- Add the H2 database dependency in pom.xml, and refresh maven.

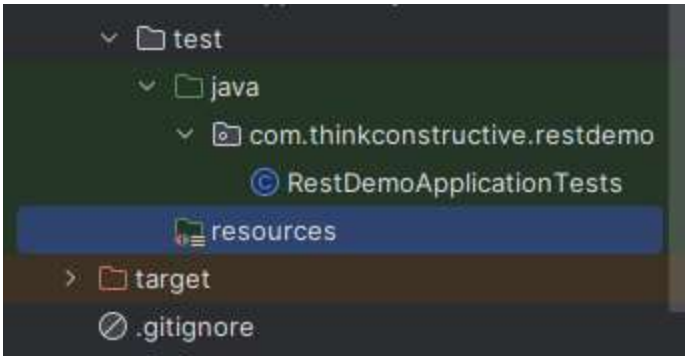




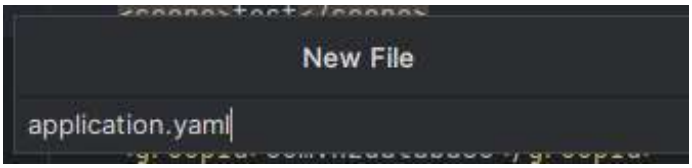
- Create Resources folder in Test.







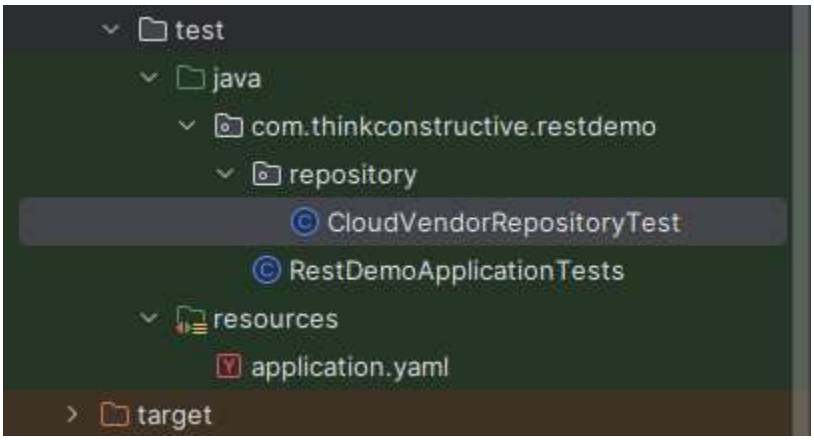
- Inside these resources create a new file.



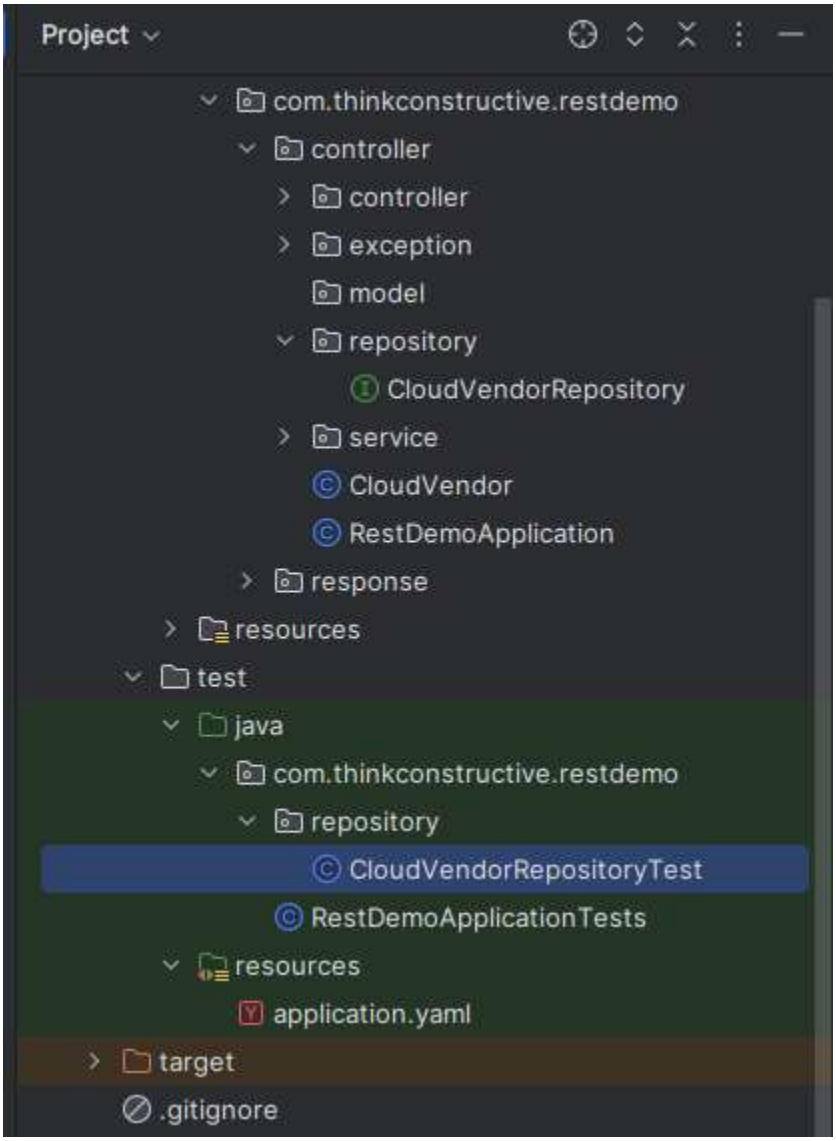

```
spring:
  datasource:
    url: jdbc:h2://mem:db;DB_CLOSE_DELAY=-1
    username: sa
    password: sa
    driver-class-name: org.h2.Driver

# The SQL dialect makes Hibernate generate better SQL for the chosen database
jpa:
  properties:
    hibernate:
      dialect: org.hibernate.dialect.H2Dialect
  show-sql: true

#JPA settings
jpa.hibernate.ddl-auto: create-drop
```

Test Cases for Repository Layer:

- Start creating test cases for Repository.



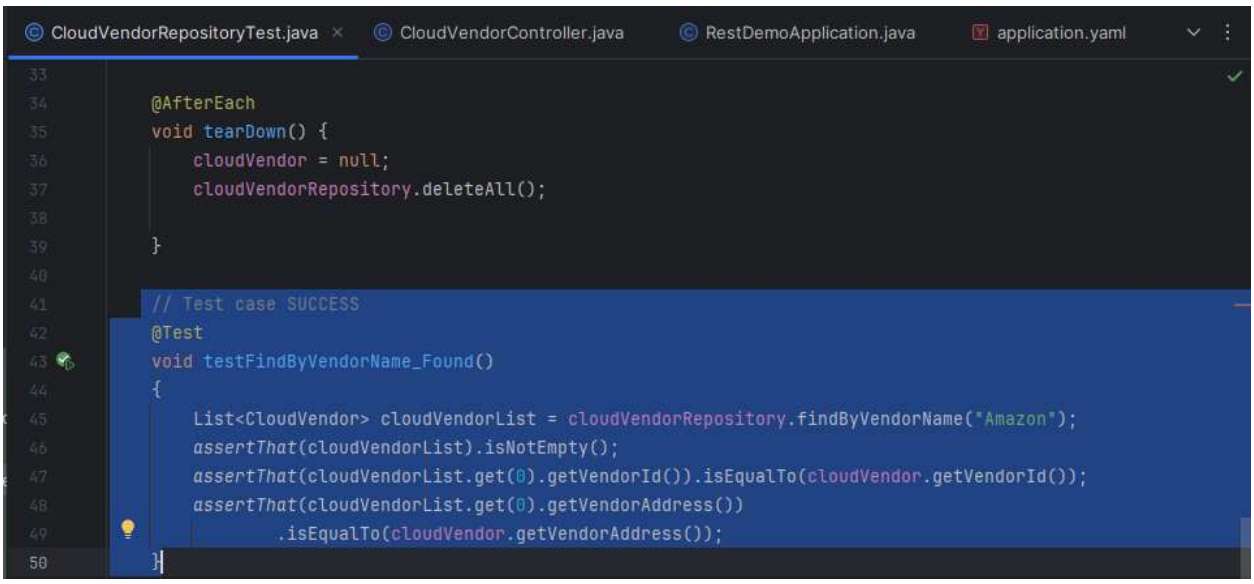
- Exact same structure.



- Writing the test case for the vendor name because there can be same name for two vendors.

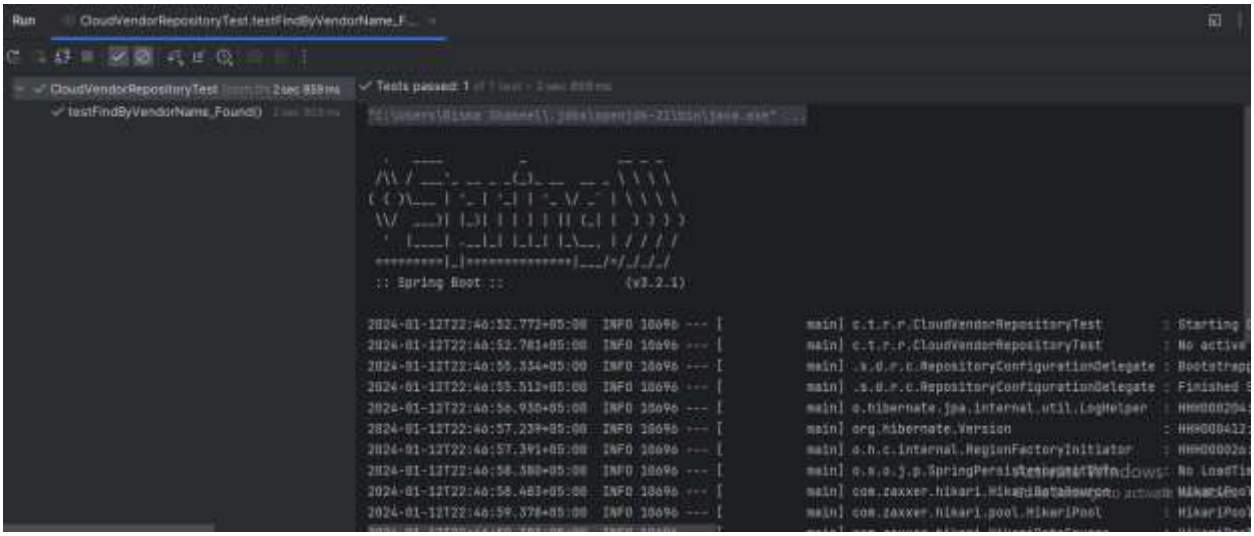

```
package com.thinkconstructive.restdemo.controller.repository;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.data.jpa.repository.JpaRepository;

import java.awt.*;
import java.util.List;

public interface CloudVendorRepository extends JpaRepository<CloudVendor,
String> {
    List<CloudVendor> findByVendorName(String vendorName);
}
```

- **<u>Test case SUCCESS</u>**

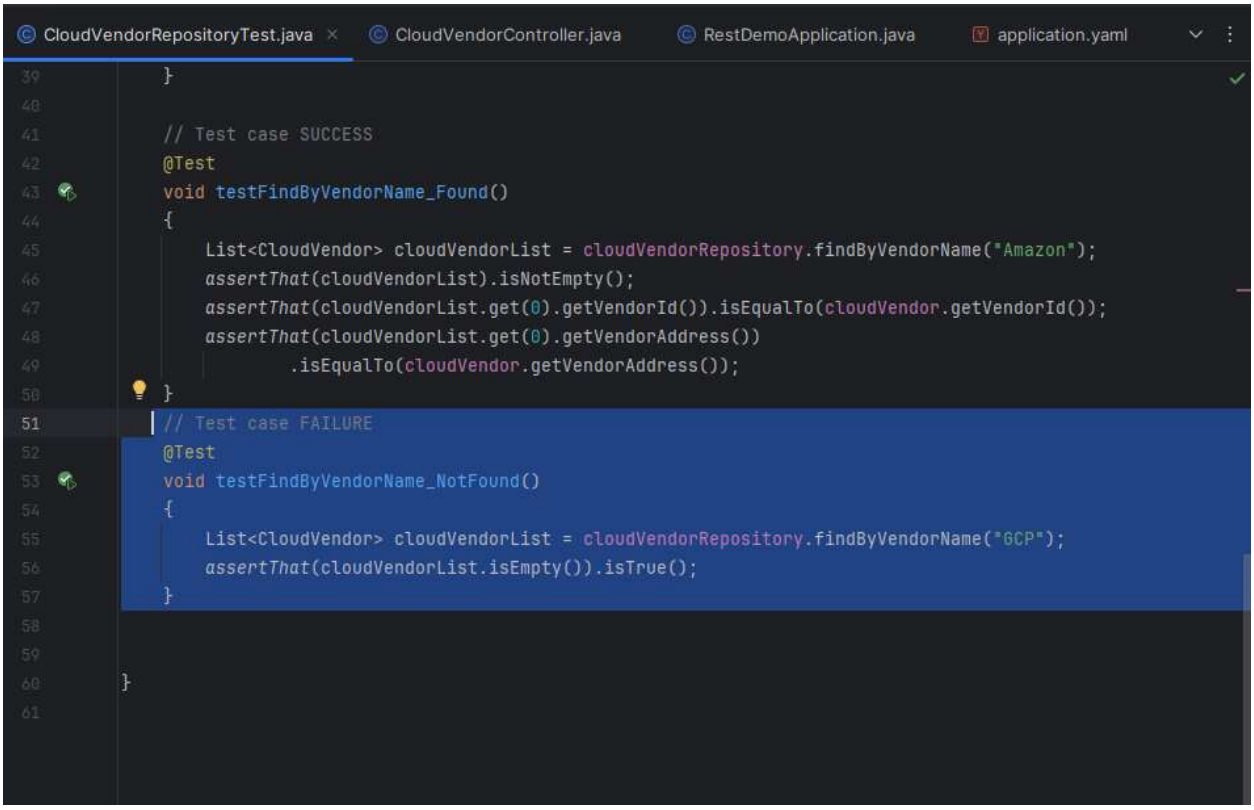



```
// Test case SUCCESS
@Test
void testFindByVendorName_Found()
{
    List<CloudVendor> cloudVendorList =
cloudVendorRepository.findByVendorName("Amazon");
```

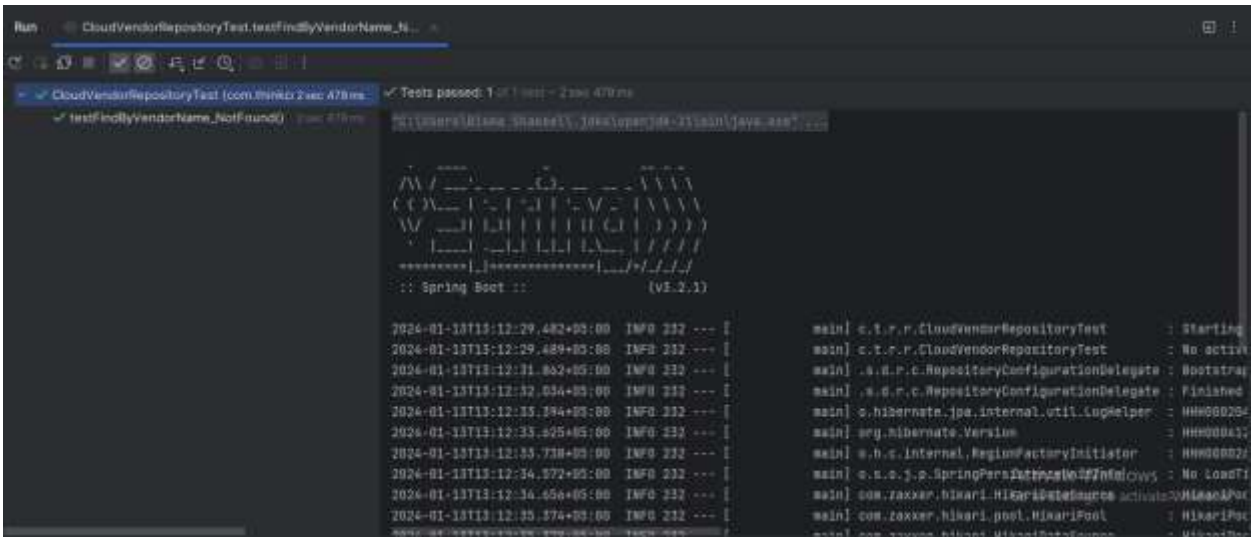
```
    assertThat(cloudVendorList).isNotEmpty();

assertThat(cloudVendorList.get(0).getVendorId()).isEqualTo(cloudVendor.getVen
dorId());
    assertThat(cloudVendorList.get(0).getVendorAddress())
            .isEqualTo(cloudVendor.getVendorAddress());
}
```



- Test case FALIURE



```
// Test case FAILURE
@Test
void testFindByVendorName_NotFound()
{
    List<CloudVendor> cloudVendorList =
cloudVendorRepository.findByVendorName("GCP");
    assertThat(cloudVendorList.isEmpty()).isTrue();
}
```

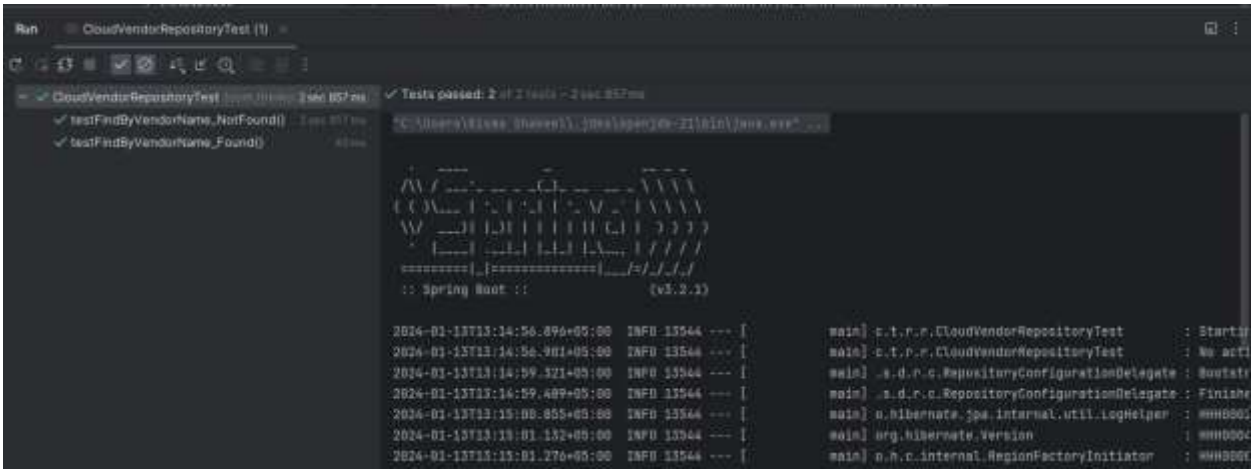


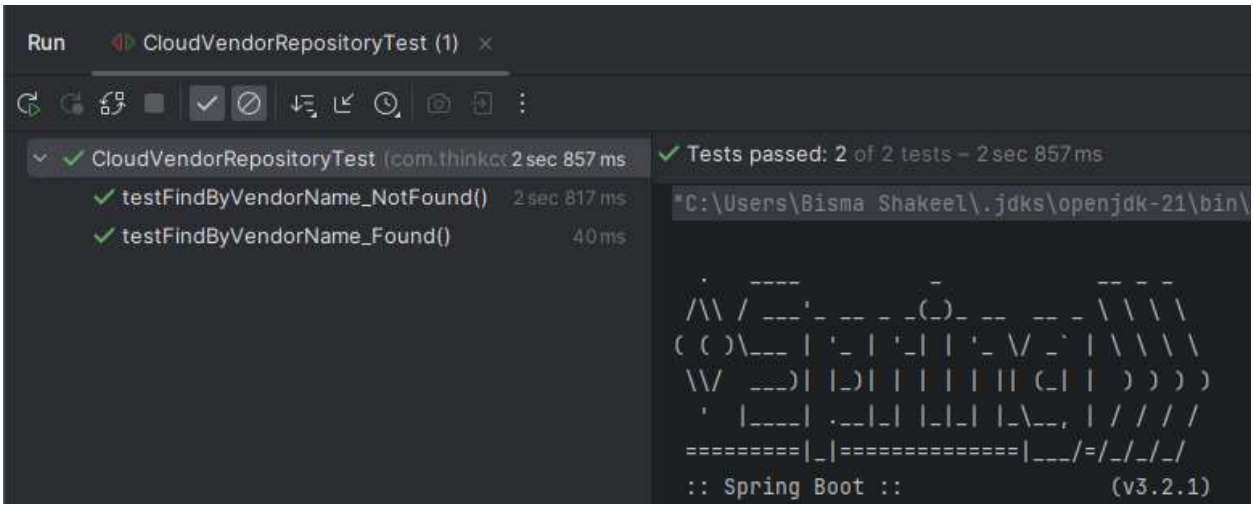
- Let's execute the whole test repository.

Code:

```
package com.thinkconstructive.restdemo.repository;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import com.thinkconstructive.restdemo.controller.RestDemoApplication;
import
com.thinkconstructive.restdemo.controller.repository.CloudVendorRepository;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;


import java.util.List;

import static org.assertj.core.api.Assertions.assertThat;

@SpringBootTest(classes = RestDemoApplication.class)
public class CloudVendorRepositoryTest {

   //given - when - then

    @Autowired
    public CloudVendorRepository cloudVendorRepository;
```

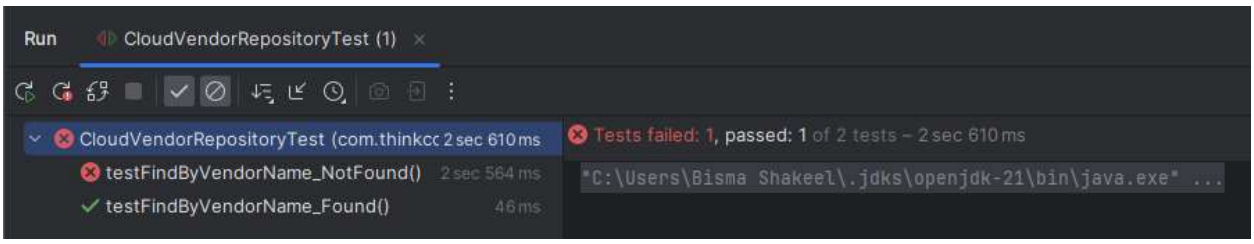
```
    CloudVendor cloudVendor;

    @BeforeEach
    void setUp() {
        cloudVendor = new CloudVendor("1","Amazon",
                      "USA","12345");
        cloudVendorRepository.save(cloudVendor);
    }

    @AfterEach
    void tearDown() {
        cloudVendor = null;
        cloudVendorRepository.deleteAll();

    }

    // Test case SUCCESS
    @Test
    void testFindByVendorName_Found()
    {
        List<CloudVendor> cloudVendorList =
cloudVendorRepository.findByVendorName("Amazon");
        assertThat(cloudVendorList).isNotEmpty();

assertThat(cloudVendorList.get(0).getVendorId()).isEqualTo(cloudVendor.getVen
dorId());
        assertThat(cloudVendorList.get(0).getVendorAddress())
                .isEqualTo(cloudVendor.getVendorAddress());
    }
    // Test case FAILURE
    @Test
    void testFindByVendorName_NotFound()
    {
        List<CloudVendor> cloudVendorList =
cloudVendorRepository.findByVendorName("GCP");
        assertThat(cloudVendorList.isEmpty()).isTrue();
    }


}
```





- Now, If I write isFalse() in Test case failure, I will get Test case failed.

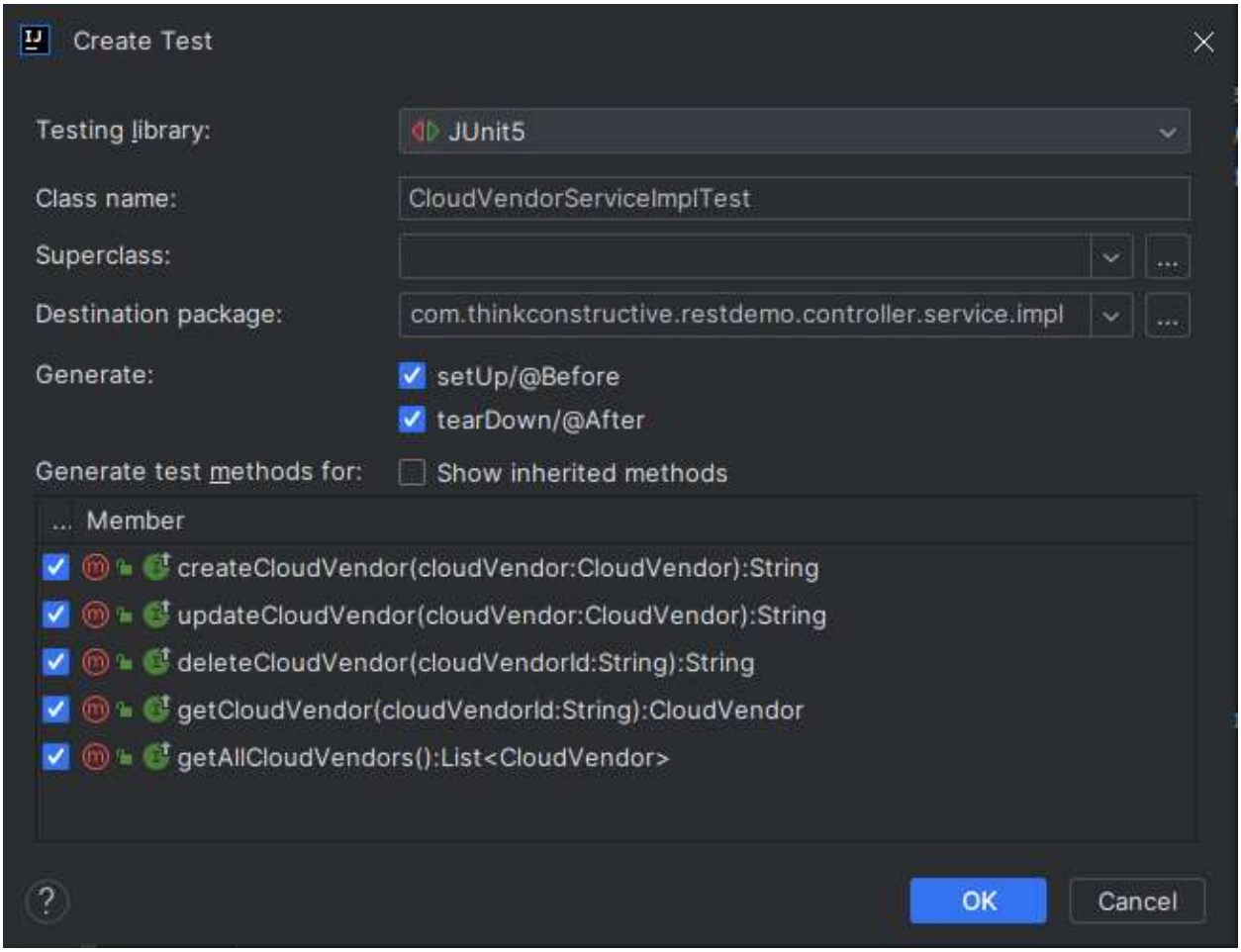
```
// Test case FAILURE
@Test
void testFindByVendorName_NotFound()
{
    List<CloudVendor> cloudVendorList =
cloudVendorRepository.findByVendorName("GCP");
    assertThat(cloudVendorList.isEmpty()).isFalse();
}
```



So, my cloud vendor repository test class is ready.

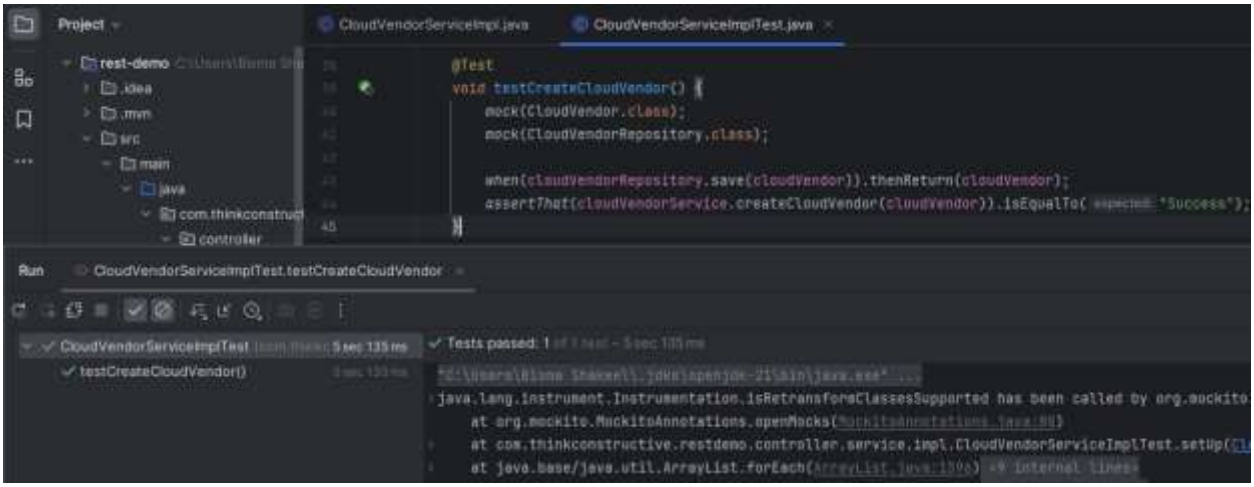
Test Cases for Service Layer:

- Writing CloudVendorServiceImpl Layer Test cases.



- testCreateCloudVendor Test Case:

```
@Test
void testCreateCloudVendor() {
    mock(CloudVendor.class);
    mock(CloudVendorRepository.class);

    when(cloudVendorRepository.save(cloudVendor)).thenReturn(cloudVendor);

assertThat(cloudVendorService.createCloudVendor(cloudVendor)).isEqualTo("Succ
ess");
}
```



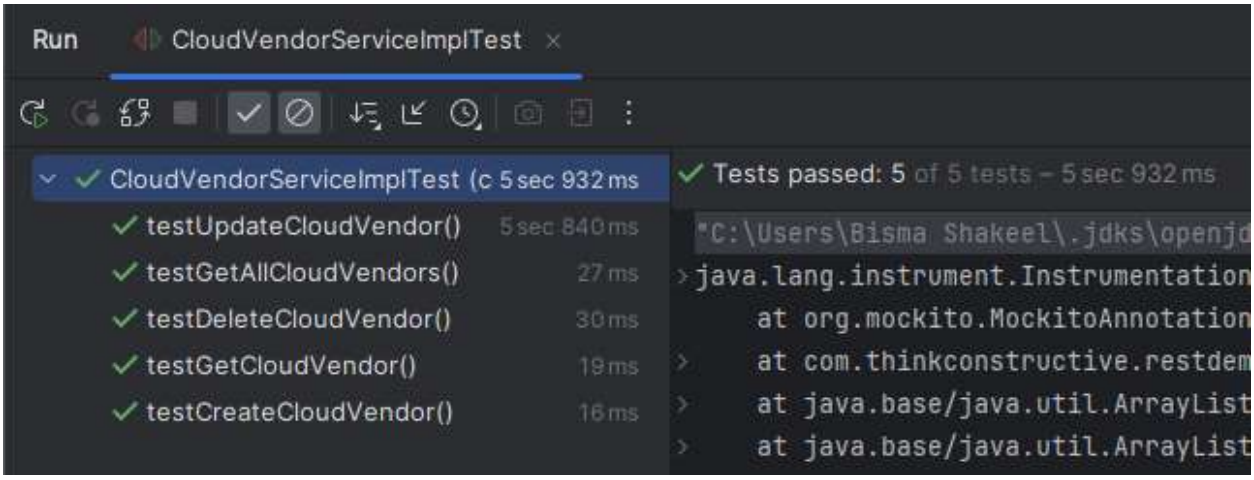
Full Code with all test cases:

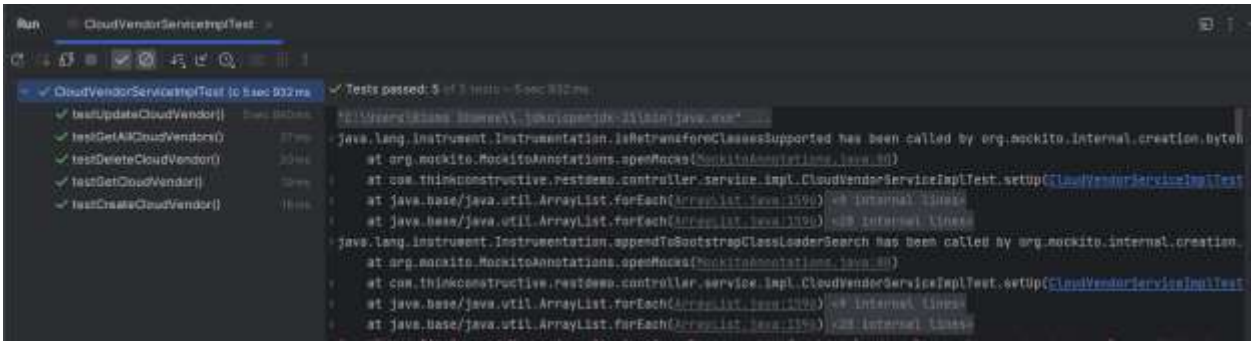
Code:

```
package com.thinkconstructive.restdemo.controller.service.impl;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import
com.thinkconstructive.restdemo.controller.repository.CloudVendorRepository;
import com.thinkconstructive.restdemo.controller.service.CloudVendorService;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.Answers;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Optional;

import static org.assertj.core.api.AssertionsForClassTypes.assertThat;
import static org.mockito.Mockito.*;

class CloudVendorServiceImplTest {

    @Mock
    private CloudVendorRepository cloudVendorRepository;
    private CloudVendorService cloudVendorService;
    AutoCloseable autoCloseable;
    CloudVendor cloudVendor;


    @BeforeEach
    void setUp() {
        autoCloseable = MockitoAnnotations.openMocks(this);
        cloudVendorService = new
CloudVendorServiceImpl(cloudVendorRepository);
```

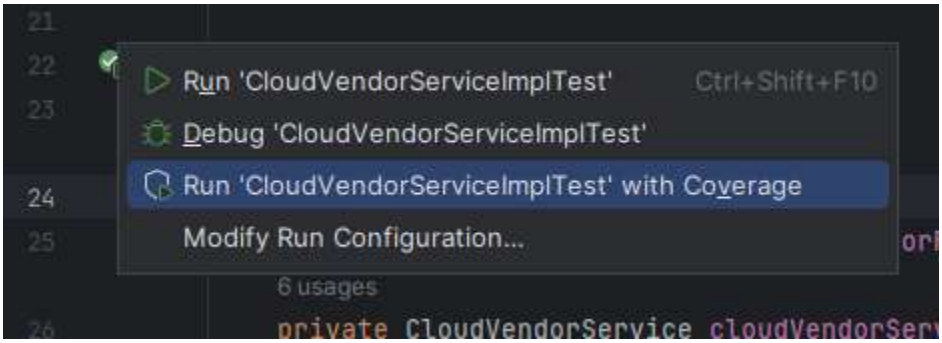
```
        cloudVendor = new CloudVendor("1","Amazon",
                            "USA","xxxxx");
    }

    @AfterEach
    void tearDown() throws Exception{
        autoCloseable.close();
    }

    @Test
    void testCreateCloudVendor() {
        mock(CloudVendor.class);
        mock(CloudVendorRepository.class);


when(cloudVendorRepository.save(cloudVendor)).thenReturn(cloudVendor);

assertThat(cloudVendorService.createCloudVendor(cloudVendor)).isEqualTo("Succ
ess");
    }

    @Test
    void testUpdateCloudVendor() {
        mock(CloudVendor.class);
        mock(CloudVendorRepository.class);


when(cloudVendorRepository.save(cloudVendor)).thenReturn(cloudVendor);

assertThat(cloudVendorService.updateCloudVendor(cloudVendor)).isEqualTo("Succ
ess");

    }

    @Test
    void testGetCloudVendor() {
        mock(CloudVendor.class);
        mock(CloudVendorRepository.class);

        when(cloudVendorRepository.findById("1")).thenReturn(
                Optional.ofNullable(cloudVendor)
        );

        assertThat(cloudVendorService.getCloudVendor("1").getVendorName())
                  .isEqualTo(cloudVendor.getVendorName());
    }

    @Test
    void testGetAllCloudVendors() {
        mock(CloudVendor.class);
        mock(CloudVendorRepository.class);

        when(cloudVendorRepository.findAll()).thenReturn(
                new
ArrayList<CloudVendor>(Collections.singleton(cloudVendor))
        );
```

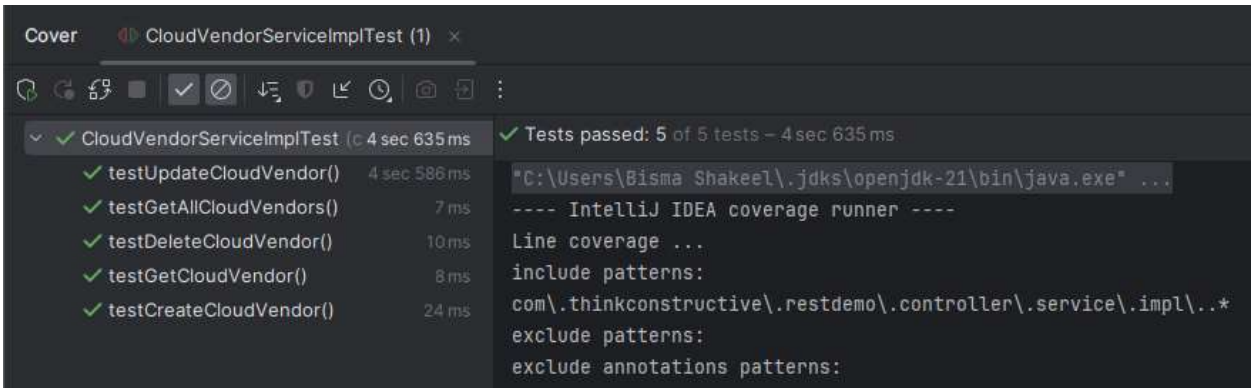
```
assertThat(cloudVendorService.getAllCloudVendors().get(0).getVendorPhoneNumbe
r())
                .isEqualTo(cloudVendor.getVendorPhoneNumber());
    }


    @Test
    void testDeleteCloudVendor() {
        mock(CloudVendor.class);
        mock(CloudVendorRepository.class, Mockito.CALLS_REAL_METHODS);

        doAnswer(Answers.CALLS_REAL_METHODS).when(
                cloudVendorRepository).deleteById(any());


assertThat(cloudVendorService.deleteCloudVendor("1")).isEqualTo("Success");

    }
}
```

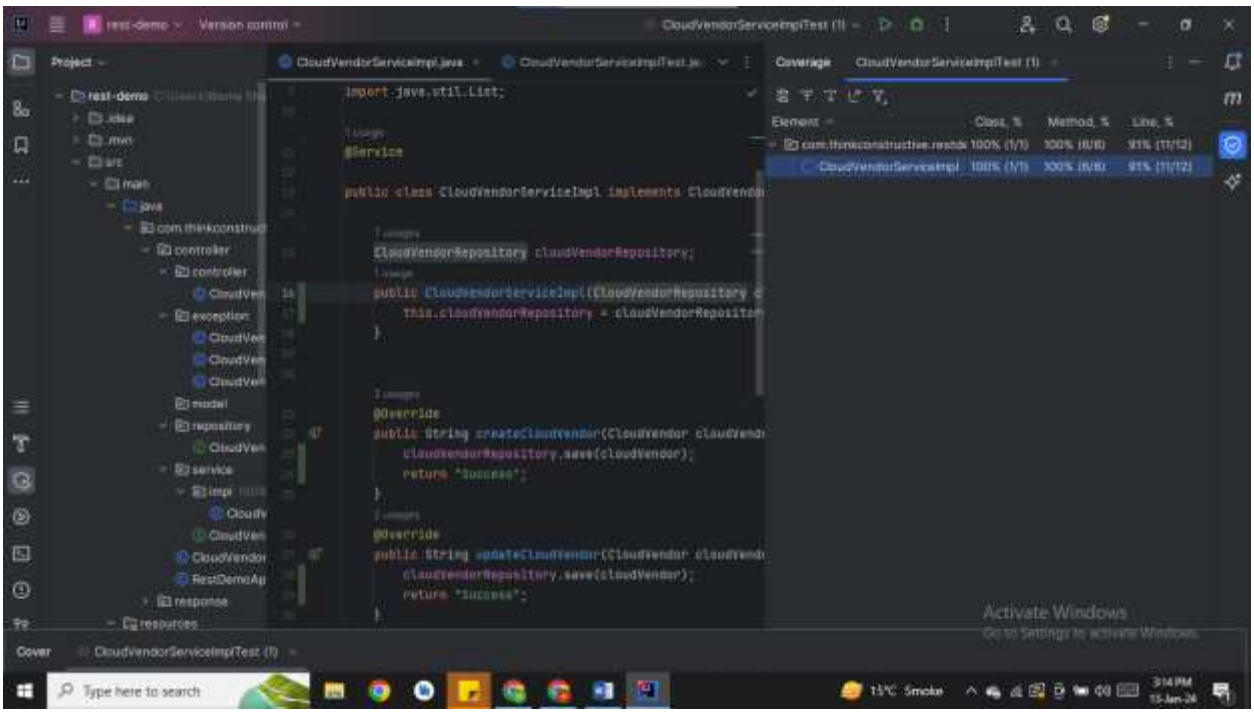




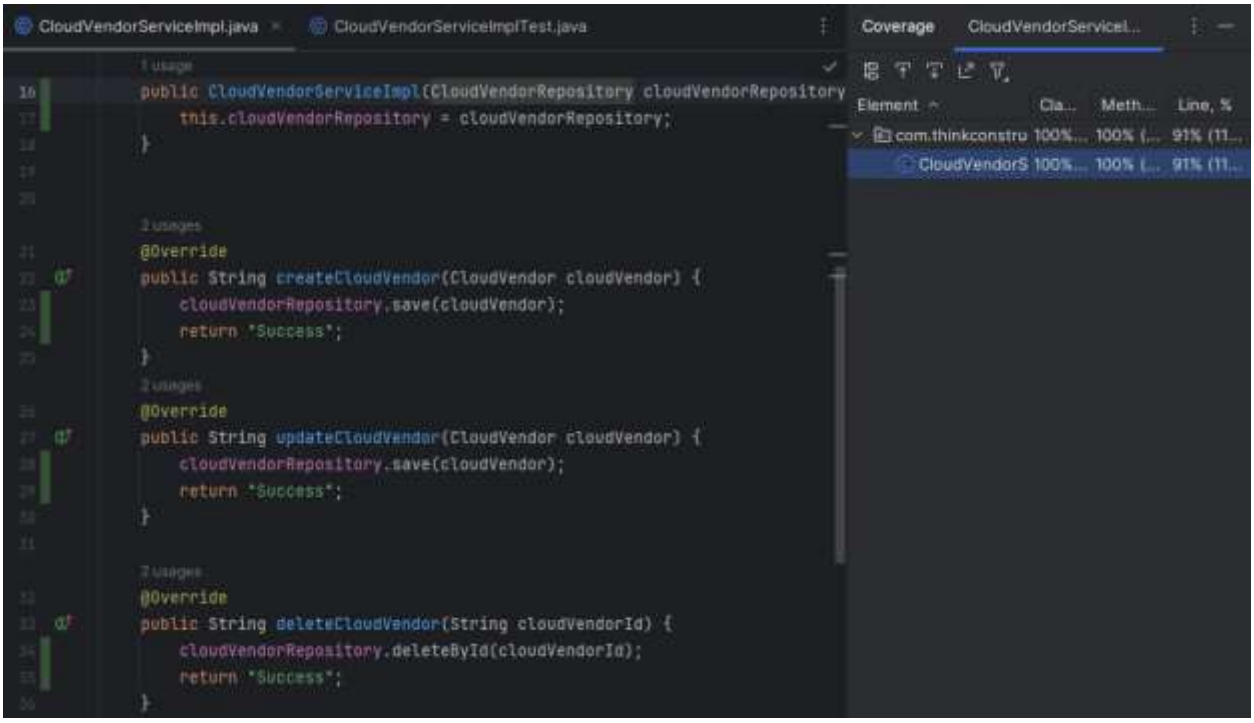
- Now, run with coverage



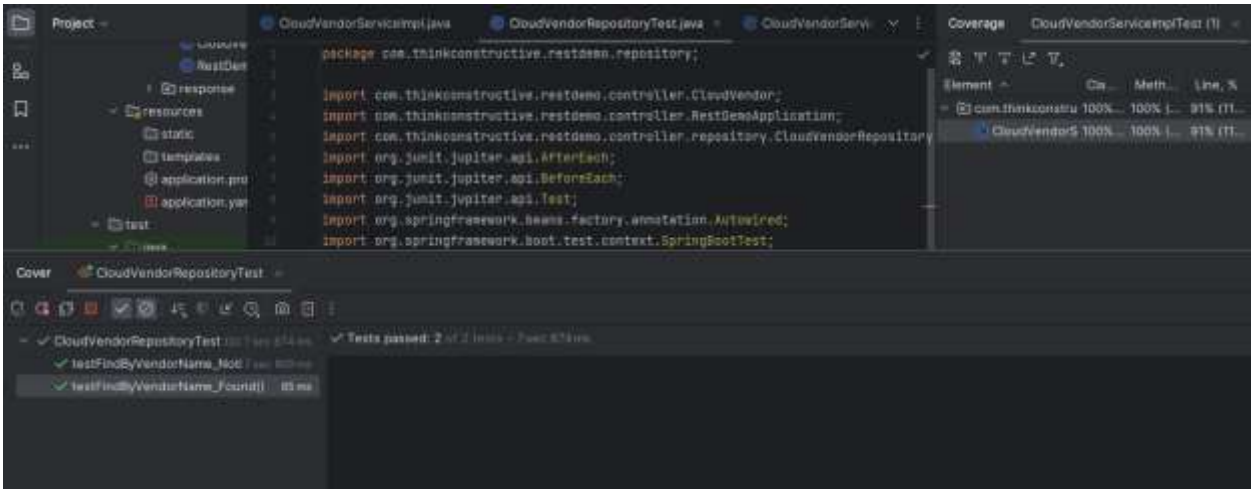




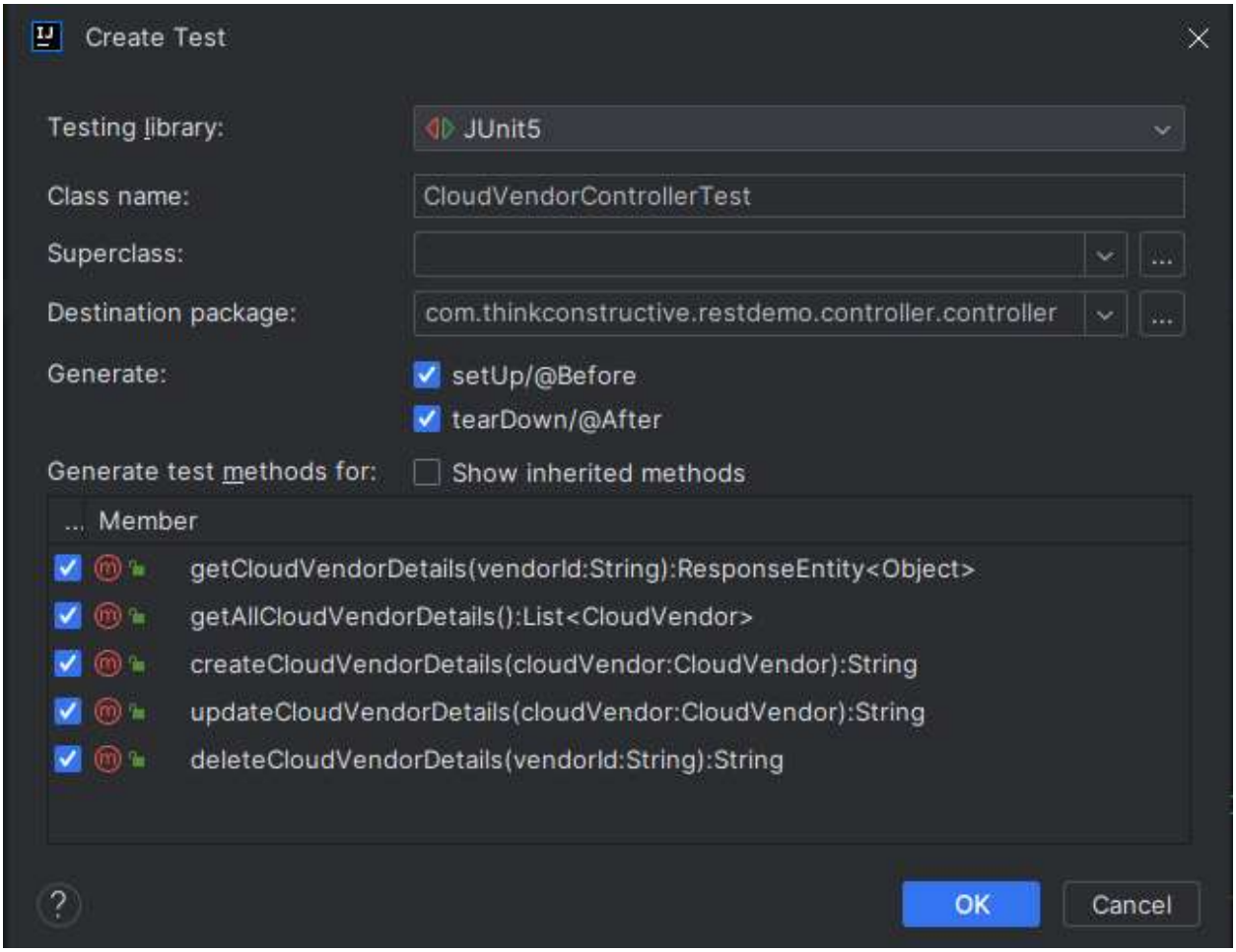
- All green means covered.



- CloudVendorRepositoryTest Run with coverage.

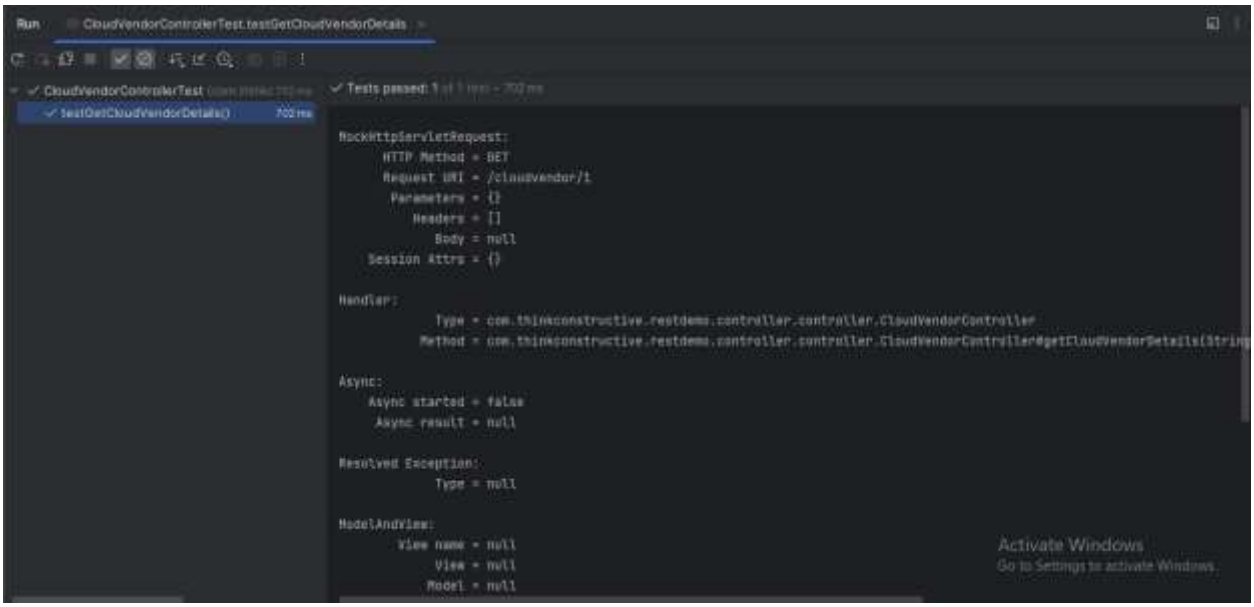


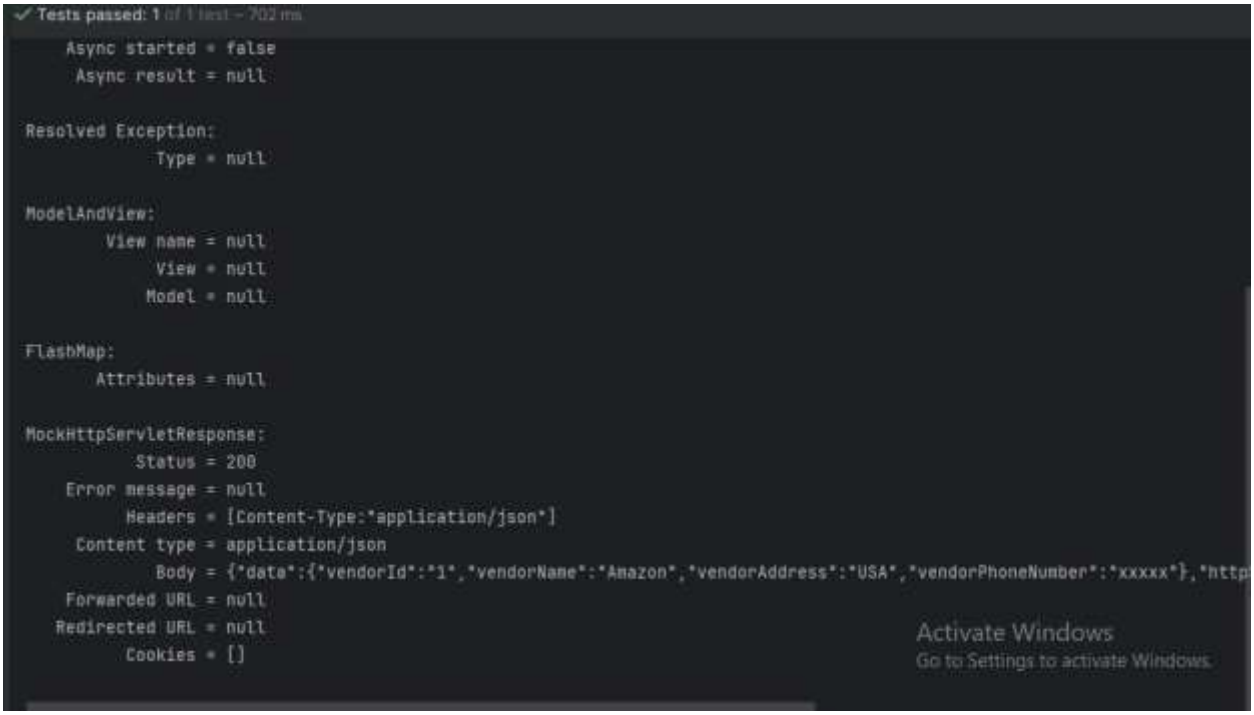
Test Cases for Controller Layer:



- GetCloudVendorDetails Test Case:

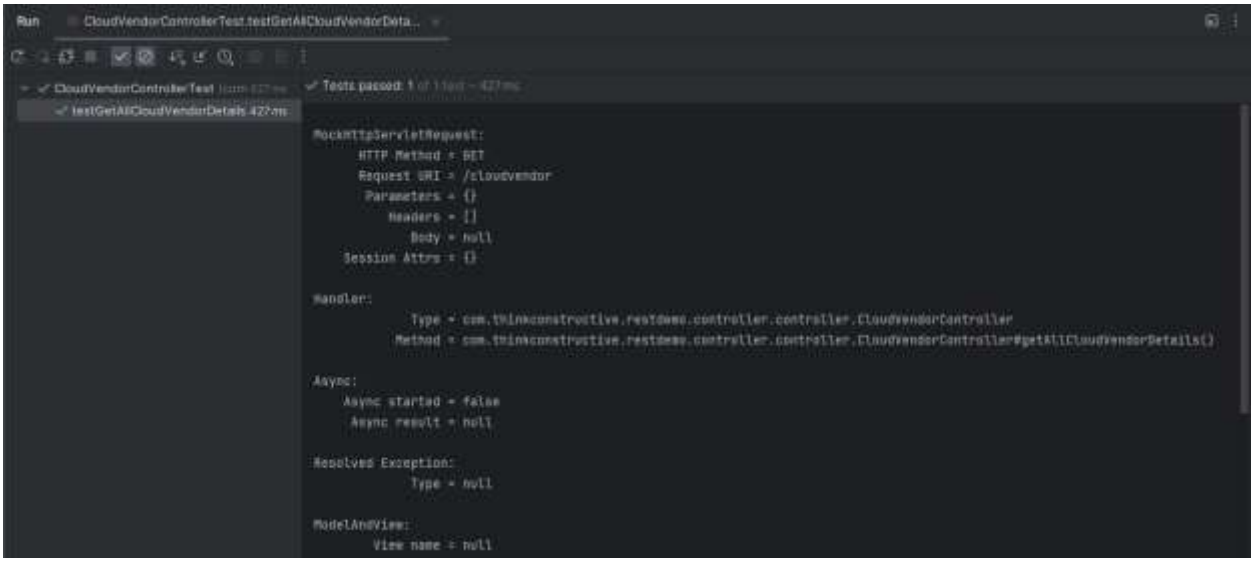
```
@Test
void testGetCloudVendorDetails() throws Exception{
    when(cloudVendorService.getCloudVendor("1"))
            .thenReturn(cloudVendorOne);
    this.mockMvc.perform(get("/cloudvendor/1"))
            .andDo(print()).andExpect(status().isOk());
}
```

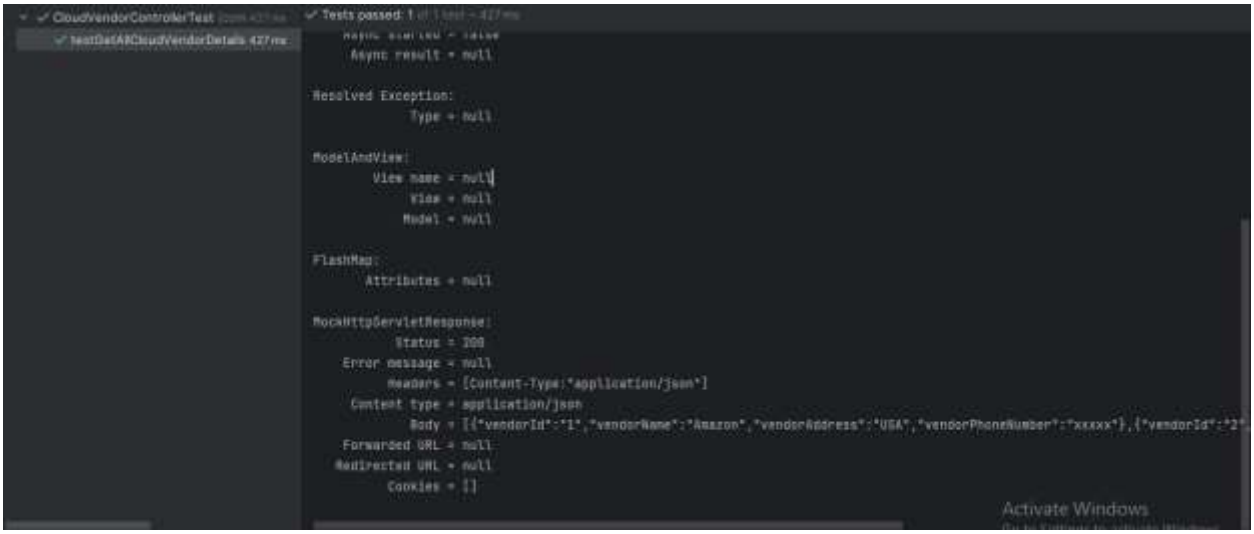

Run
CloudVendorControllerTest.testGetCloudVendorDetails
CloudVendorControllerTest
testGetCloudVendorDetails()
702 ms
Tests passed: 1
MockHttpServletRequest:
HTTP Method = GET
Request URI = /cloudvendor/1
Parameters = {}
Headers = []
Body = null
Session Attrs = {}
Handler:
Type = com.thinkconstructive.restdemo.controller.controller.CloudVendorController
Method = com.thinkconstructive.restdemo.controller.controller.CloudVendorController#getCloudVendorDetails(Strin
Async:
Async started = false
Async result = null
Resolved Exception:
Type = null
ModelAndView:
View name = null
View = null
Model = null
Activate Windows
Go to Settings to activate Windows.


Tests passed: 1 of 1 test – 702 ms
Async started = false
Async result = null
Resolved Exception:
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"application/json"]
Content type = application/json
Body = {"data":{"vendorId":"1","vendorName":"Amazon","vendorAddress":"USA","vendorPhoneNumber":"xxxxx"},"http
Forwarded URL = null
Redirected URL = null
Cookies = []
Activate Windows
Go to Settings to activate Windows.

- GetAllCloudVendorDetails Test Case:

```
@Test
void testGetAllCloudVendorDetails() throws Exception {

    when(cloudVendorService.getAllCloudVendors())
            .thenReturn(cloudVendorList);
    this.mockMvc.perform(get("/cloudvendor"))
            .andDo(print()).andExpect(status().isOk());
}
```

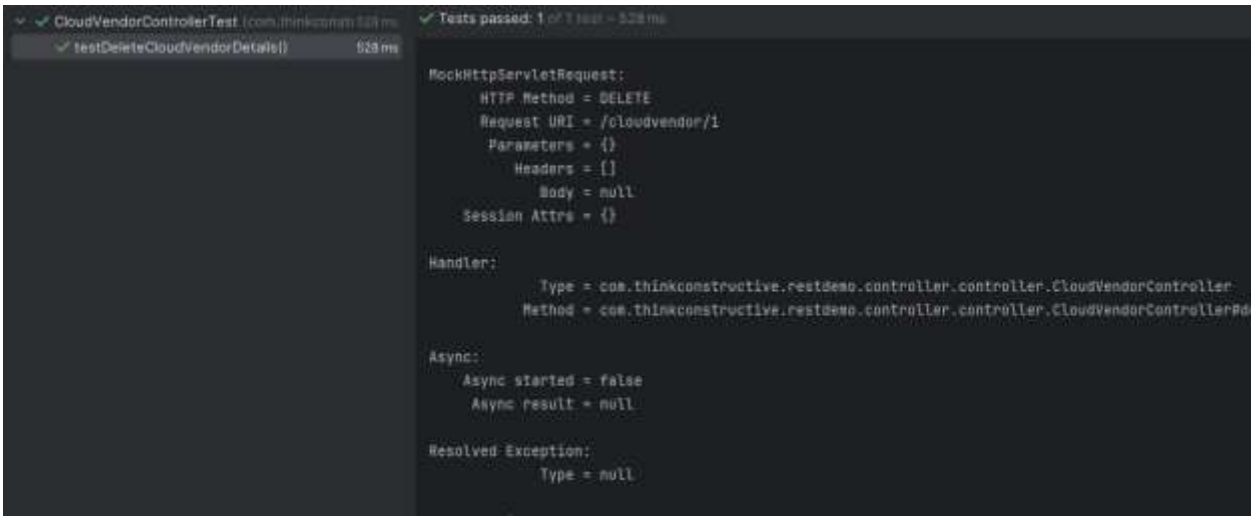




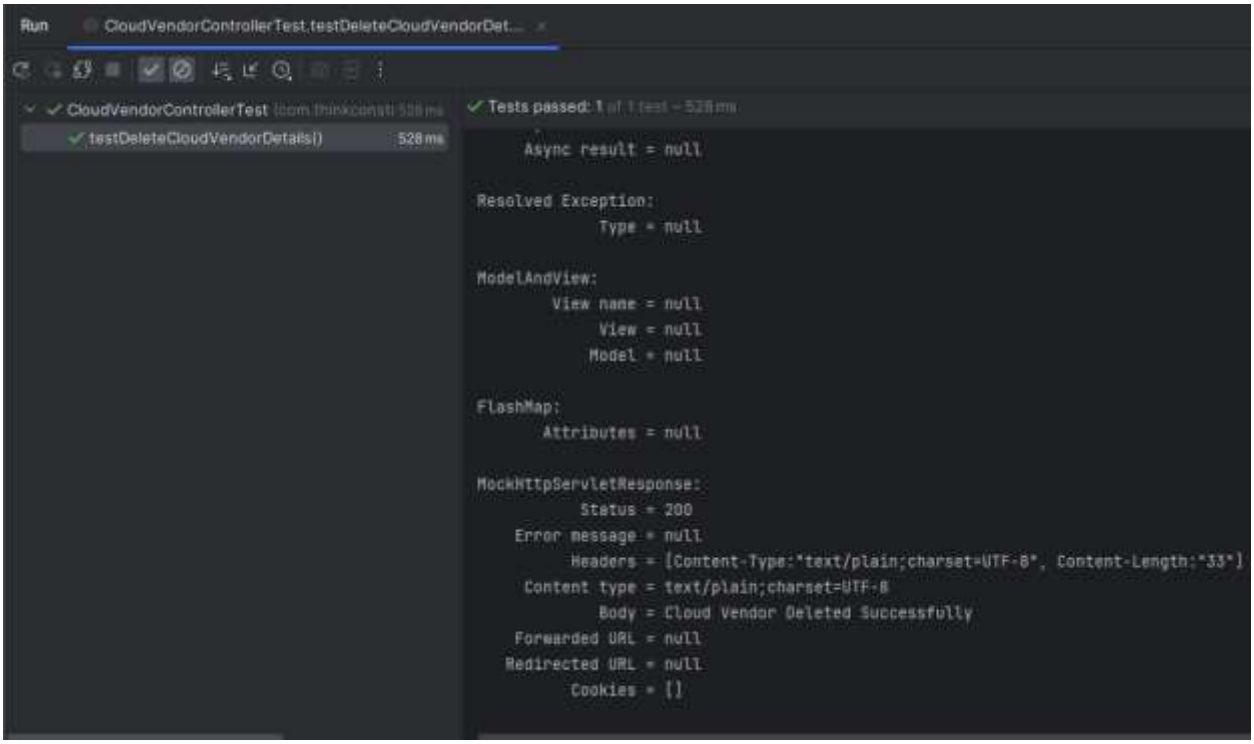
- List has been shown

- DeleteCloudVendorDetails Test Case:

```
@Test
void testDeleteCloudVendorDetails() throws Exception {
    when(cloudVendorService.deleteCloudVendor("1"))
            .thenReturn("Success");
    this.mockMvc.perform(delete("/cloudvendor/1"))
            .andDo(print()).andExpect(status().isOk());
}
```



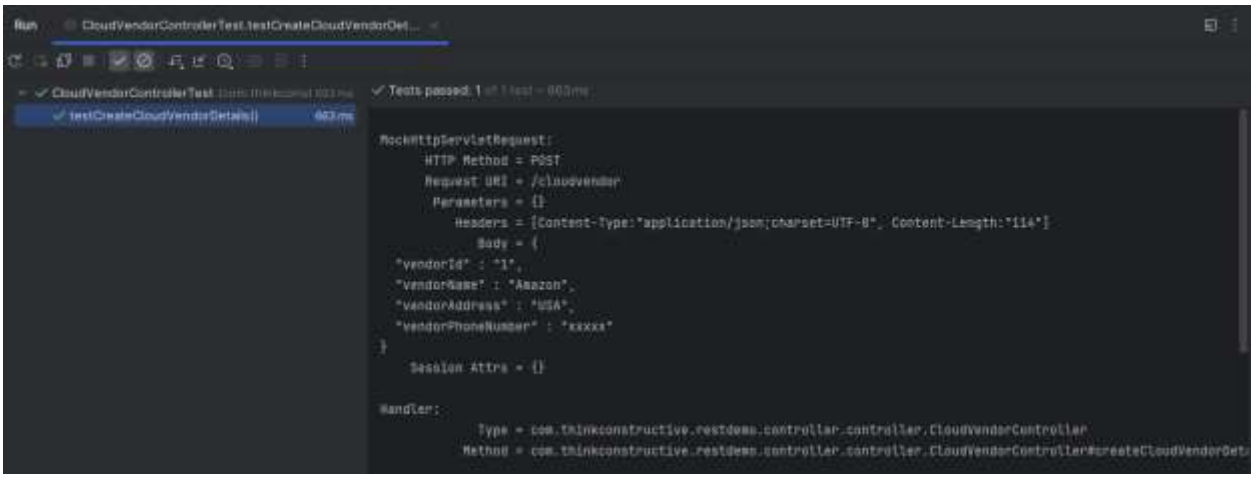
```
MockHttpServletResponse:
           Status = 200
    Error message = null
          Headers = [Content-Type:"text/plain;charset=UTF-8", Content-Length:"33"]
     Content type = text/plain;charset=UTF-8
             Body = Cloud Vendor Deleted Successfully
    Forwarded URL = null
   Redirected URL = null
          Cookies = []
```

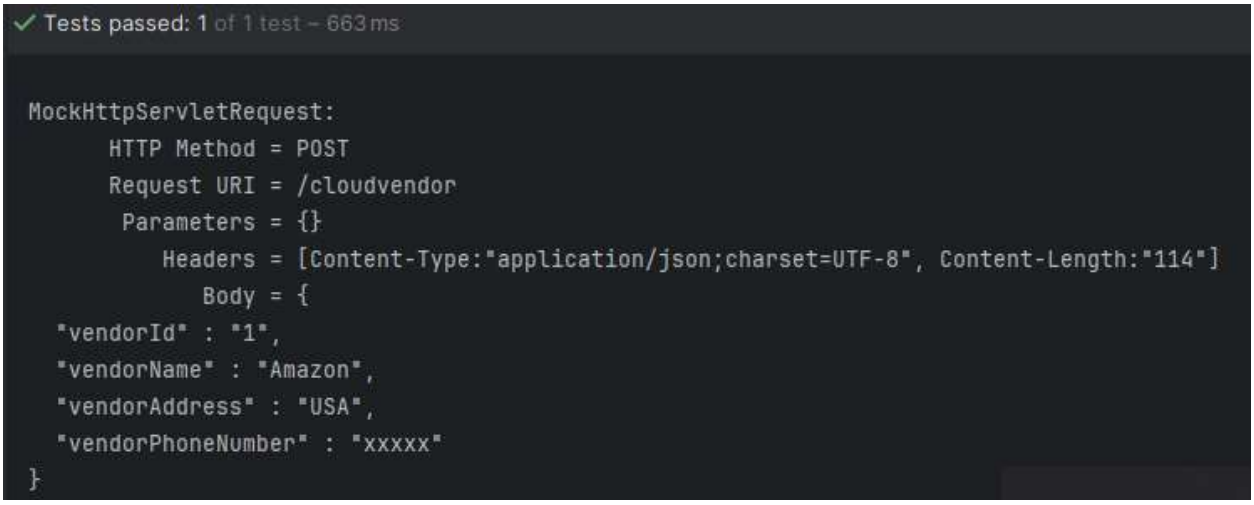


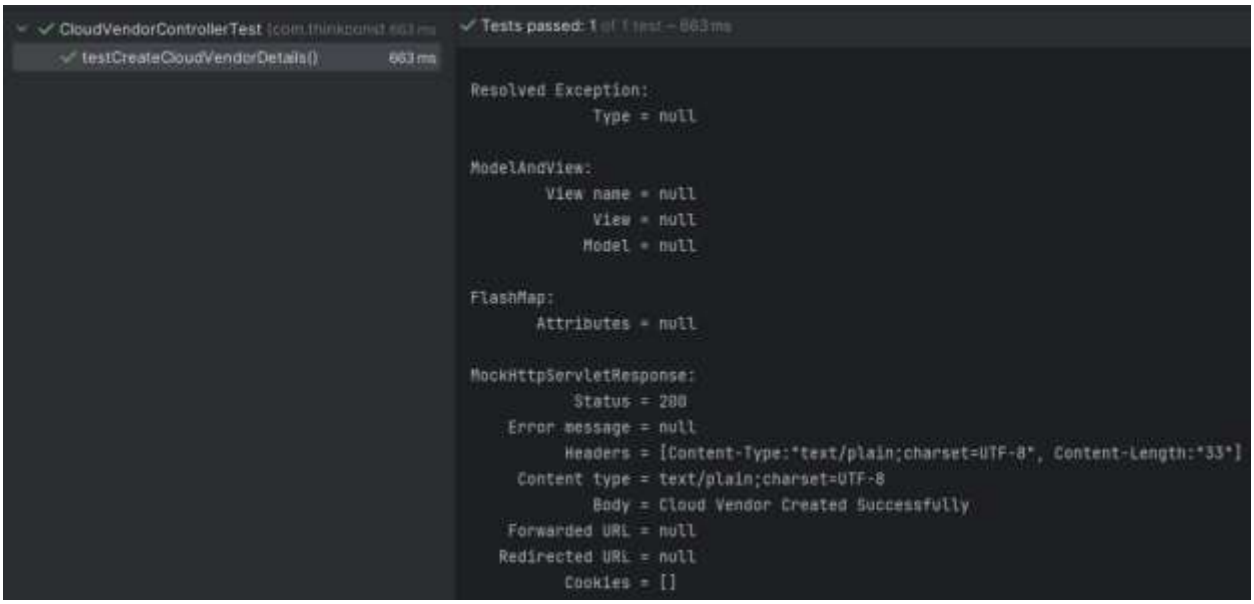
- Create Cloud Vendor Details Test Case:

```
@Test
void testCreateCloudVendorDetails() throws Exception {

    ObjectMapper mapper = new ObjectMapper();
    mapper.configure(SerializationFeature.WRAP_ROOT_VALUE, false);
    ObjectWriter ow = mapper.writer().withDefaultPrettyPrinter();
    String requestJson=ow.writeValueAsString(cloudVendorOne);

    when(cloudVendorService.createCloudVendor(cloudVendorOne))
            .thenReturn("Success");
    this.mockMvc.perform(post("/cloudvendor")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(requestJson))
```

```
                .andDo(print()).andExpect(status().isOk());
}
```

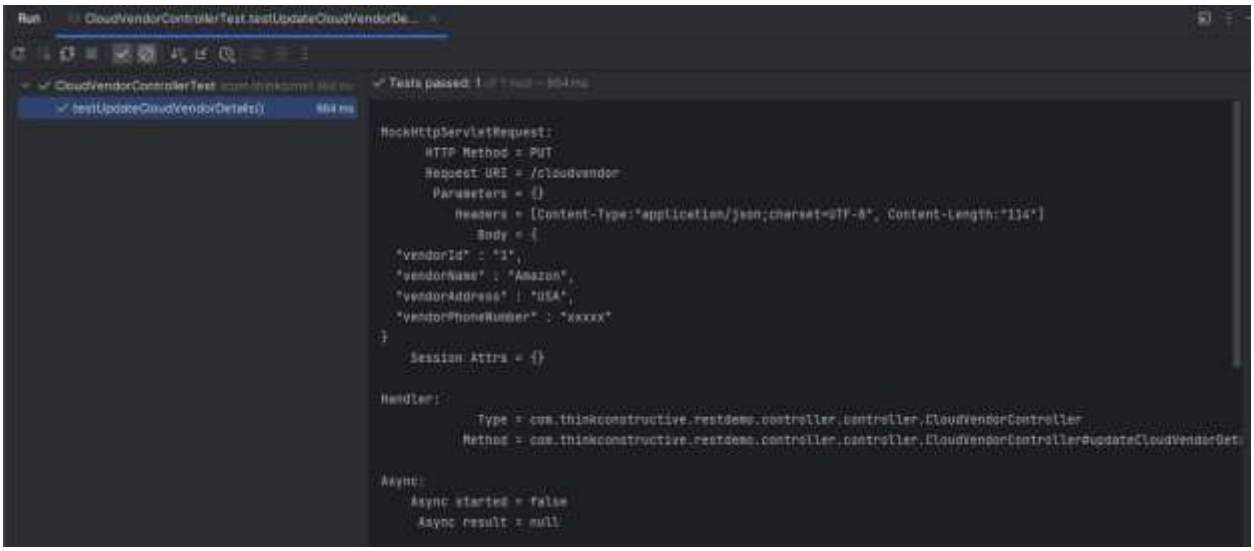






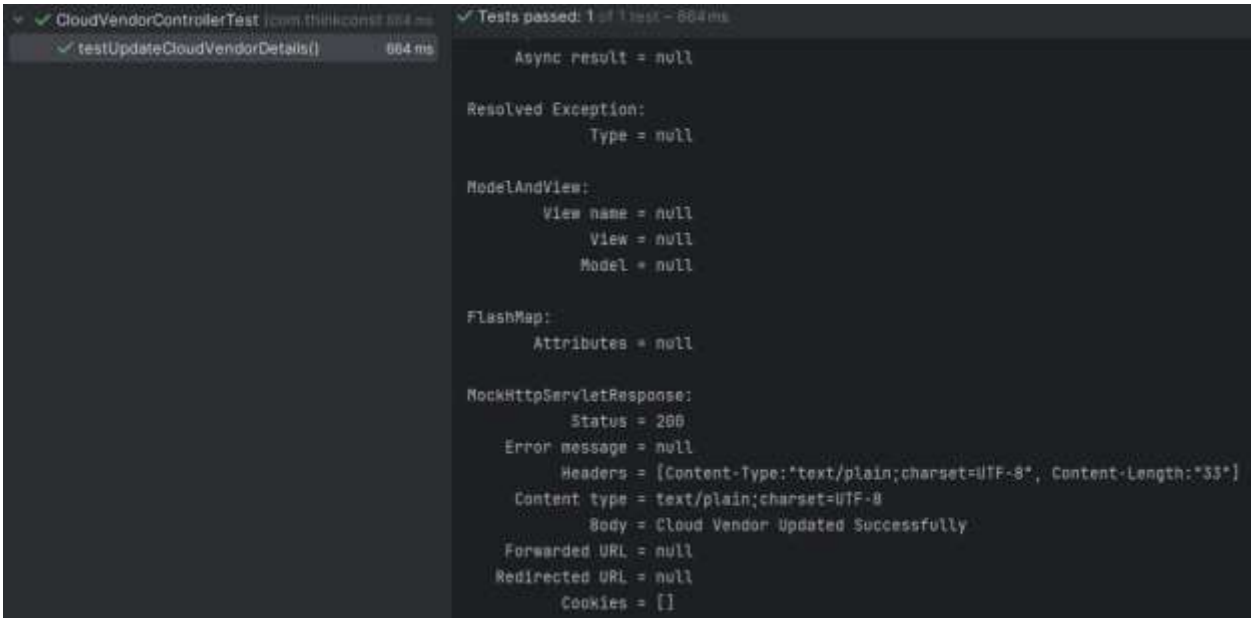
- Update Cloud Vendor Details Test Case:

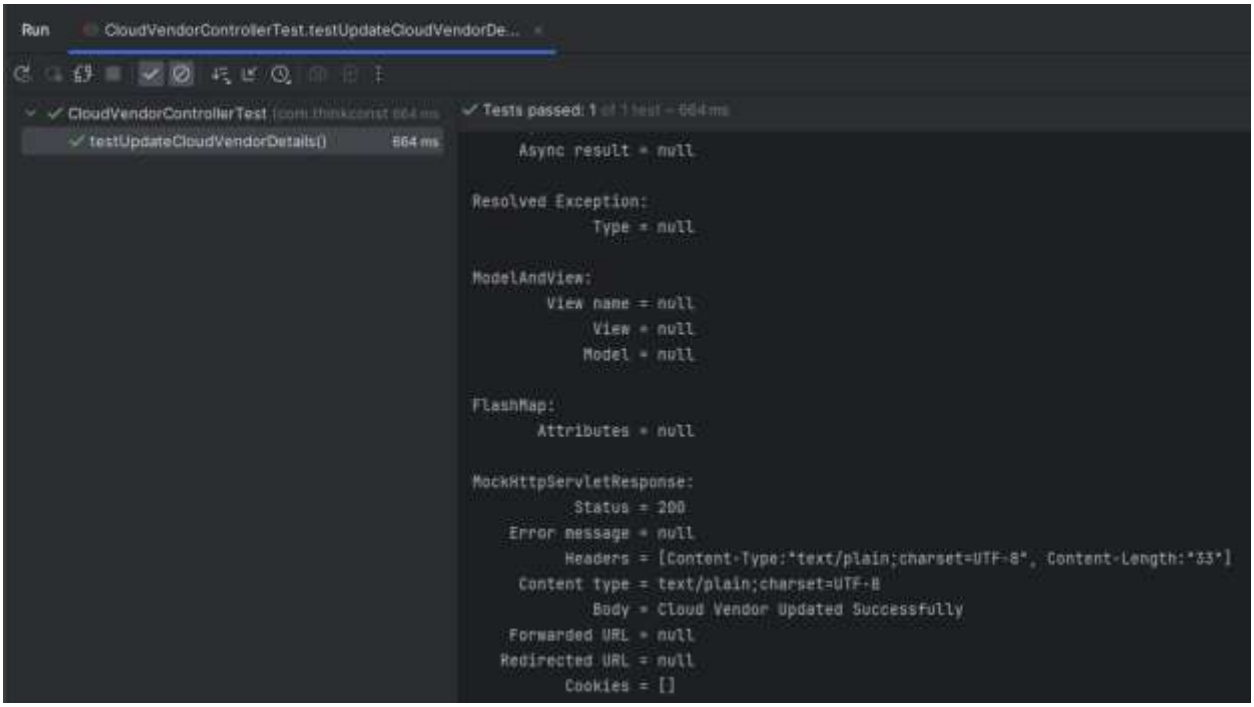
```
@Test
void testUpdateCloudVendorDetails() throws Exception{
    ObjectMapper mapper = new ObjectMapper();
    mapper.configure(SerializationFeature.WRAP_ROOT_VALUE, false);
    ObjectWriter ow = mapper.writer().withDefaultPrettyPrinter();
    String requestJson=ow.writeValueAsString(cloudVendorOne);

    when(cloudVendorService.updateCloudVendor(cloudVendorOne))
            .thenReturn("Success");
    this.mockMvc.perform(put("/cloudvendor")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(requestJson))
            .andDo(print()).andExpect(status().isOk());


}
```



✓ Tests passed: 1 of 1 test – 664 ms

```
MockHttpServletRequest:
      HTTP Method = PUT
      Request URI = /cloudvendor
       Parameters = {}
          Headers = [Content-Type:"application/json;charset=UTF-8", Content-Length:"114"]
             Body = {
  "vendorId" : "1",
  "vendorName" : "Amazon",
  "vendorAddress" : "USA",
  "vendorPhoneNumber" : "xxxxx"
}
    Session Attrs = {}
```


CloudVendorControllerTest
testUpdateCloudVendorDetails()
664 ms
Tests passed: 1
Async result = null
Resolved Exception:
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"text/plain;charset=UTF-8", Content-Length:"33"]
Content type = text/plain;charset=UTF-8
Body = Cloud Vendor Updated Successfully
Forwarded URL = null
Redirected URL = null
Cookies = []

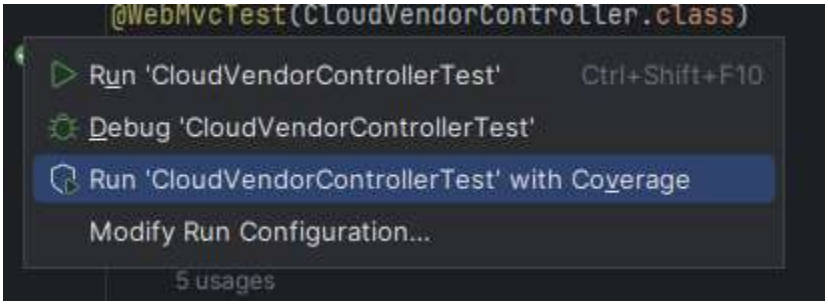

Run
CloudVendorControllerTest.testUpdateCloudVendorDe...
CloudVendorControllerTest
testUpdateCloudVendorDetails()
664 ms
Tests passed: 1
Async result = null
Resolved Exception:
Type = null
ModelAndView:
View name = null
View = null
Model = null
FlashMap:
Attributes = null
MockHttpServletResponse:
Status = 200
Error message = null
Headers = [Content-Type:"text/plain;charset=UTF-8", Content-Length:"33"]
Content type = text/plain;charset=UTF-8
Body = Cloud Vendor Updated Successfully
Forwarded URL = null
Redirected URL = null
Cookies = []

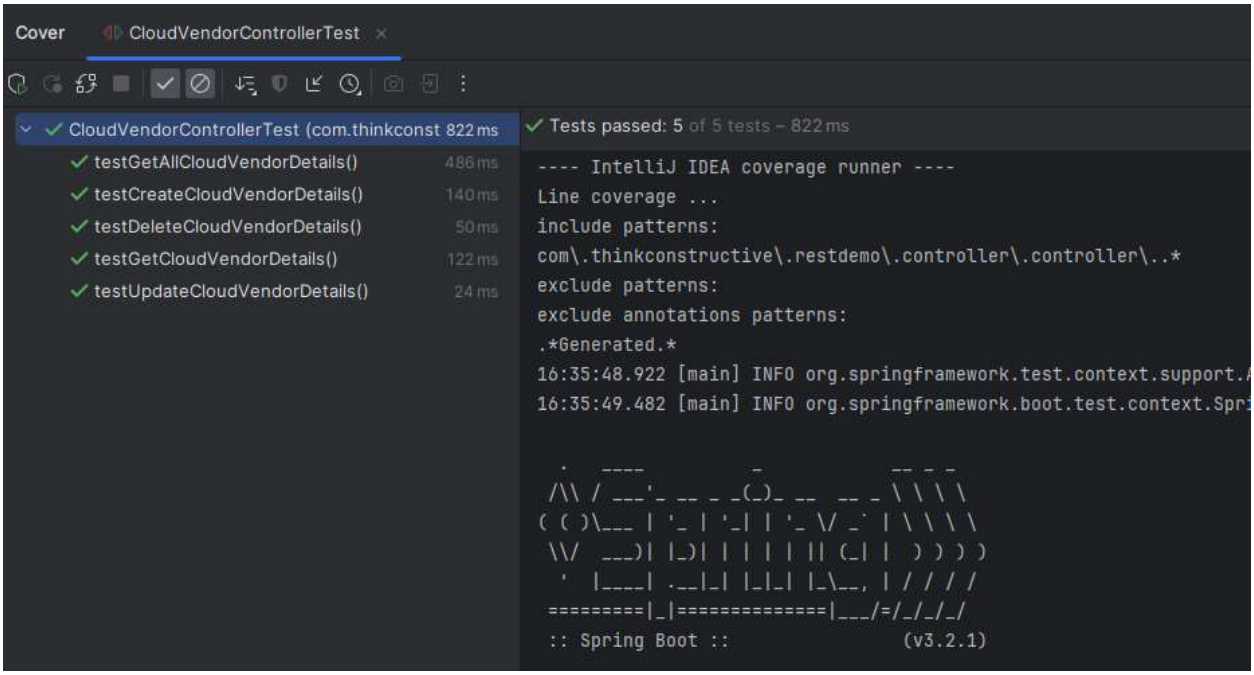
Whole Test Cases Code for Controller Layer.

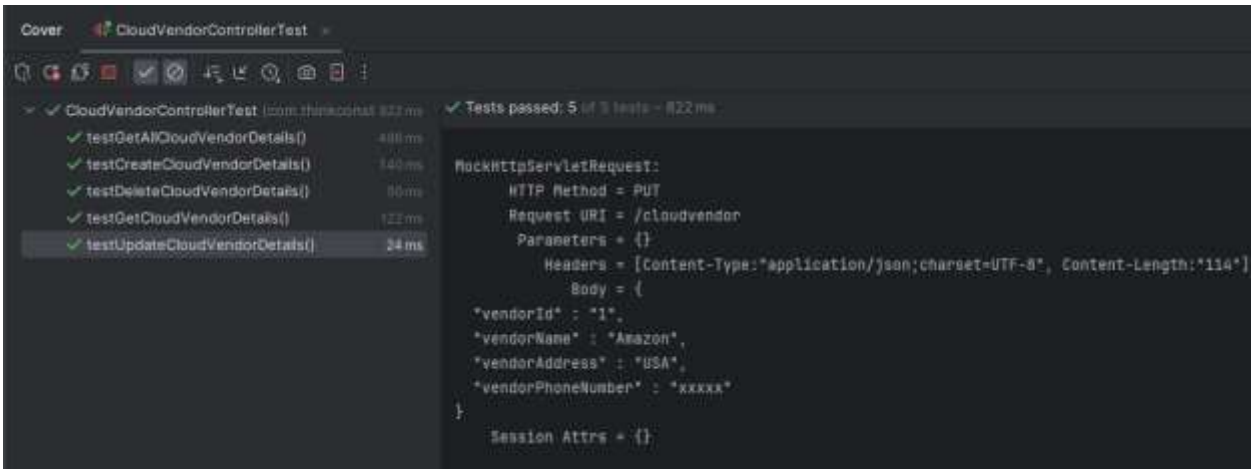
```
package com.thinkconstructive.restdemo.controller.controller;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.thinkconstructive.restdemo.controller.CloudVendor;
import com.thinkconstructive.restdemo.controller.service.CloudVendorService;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.MockMvc;

import java.util.ArrayList;
import java.util.List;

import static org.mockito.Mockito.when;
import static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
import static
org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

@WebMvcTest(CloudVendorController.class)
class CloudVendorControllerTest {

    @Autowired
    private MockMvc mockMvc;
    @MockBean
    private CloudVendorService cloudVendorService;
    CloudVendor cloudVendorOne;
    CloudVendor cloudVendorTwo;
    List<CloudVendor> cloudVendorList = new ArrayList<>();

    @BeforeEach
    void setUp() {
        cloudVendorOne = new CloudVendor("1","Amazon",
                "USA","xxxxx");
        cloudVendorTwo = new CloudVendor("2","GCP",
                "UK","yyyyy");
        cloudVendorList.add(cloudVendorOne);
        cloudVendorList.add(cloudVendorTwo);
    }

    @AfterEach
    void tearDown() {
    }

    @Test
    void testGetCloudVendorDetails() throws Exception{
        when(cloudVendorService.getCloudVendor("1"))
```

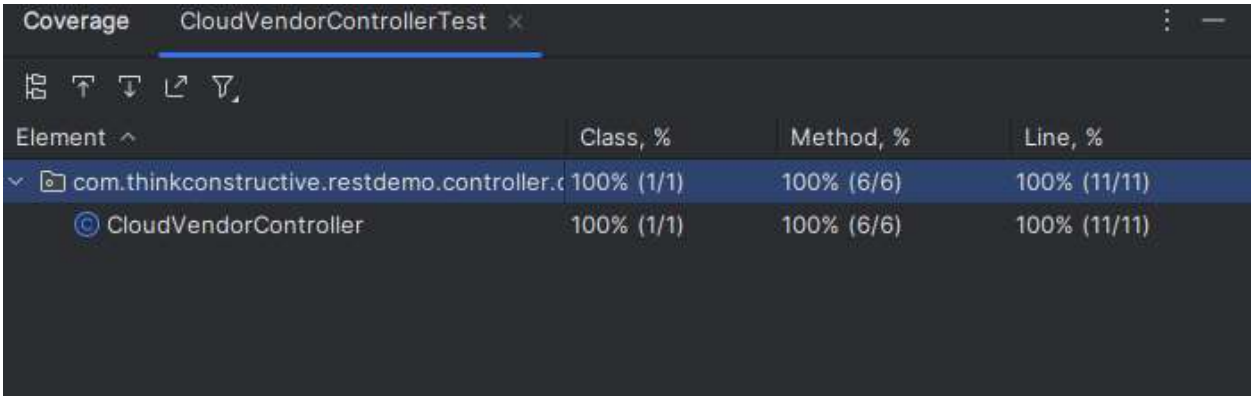
```
                .thenReturn(cloudVendorOne);
        this.mockMvc.perform(get("/cloudvendor/1"))
                .andDo(print()).andExpect(status().isOk());
    }

    @Test
    void testGetAllCloudVendorDetails() throws Exception {

        when(cloudVendorService.getAllCloudVendors())
                .thenReturn(cloudVendorList);
        this.mockMvc.perform(get("/cloudvendor"))
                .andDo(print()).andExpect(status().isOk());
    }

    @Test
    void testCreateCloudVendorDetails() throws Exception {

        ObjectMapper mapper = new ObjectMapper();
        mapper.configure(SerializationFeature.WRAP_ROOT_VALUE, false);
        ObjectWriter ow = mapper.writer().withDefaultPrettyPrinter();
        String requestJson=ow.writeValueAsString(cloudVendorOne);

        when(cloudVendorService.createCloudVendor(cloudVendorOne))
                .thenReturn("Success");
        this.mockMvc.perform(post("/cloudvendor")
                        .contentType(MediaType.APPLICATION_JSON)
                        .content(requestJson))
                .andDo(print()).andExpect(status().isOk());
    }

    @Test
    void testUpdateCloudVendorDetails() throws Exception{
        ObjectMapper mapper = new ObjectMapper();
        mapper.configure(SerializationFeature.WRAP_ROOT_VALUE, false);
        ObjectWriter ow = mapper.writer().withDefaultPrettyPrinter();
        String requestJson=ow.writeValueAsString(cloudVendorOne);

        when(cloudVendorService.updateCloudVendor(cloudVendorOne))
                .thenReturn("Success");
        this.mockMvc.perform(put("/cloudvendor")
                        .contentType(MediaType.APPLICATION_JSON)
                        .content(requestJson))
                .andDo(print()).andExpect(status().isOk());


    }

    @Test
    void testDeleteCloudVendorDetails() throws Exception {
        when(cloudVendorService.deleteCloudVendor("1"))
                .thenReturn("Success");
        this.mockMvc.perform(delete("/cloudvendor/1"))
                .andDo(print()).andExpect(status().isOk());
    }
}
```

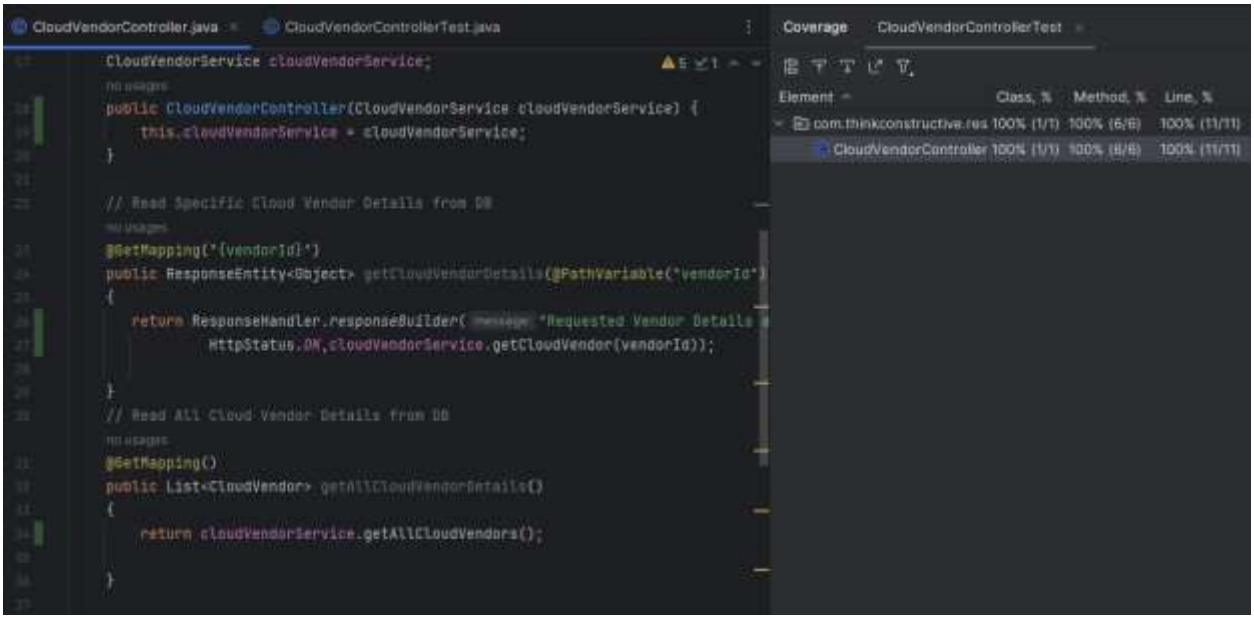
- Execute the Full Controller Test Cases with coverage.

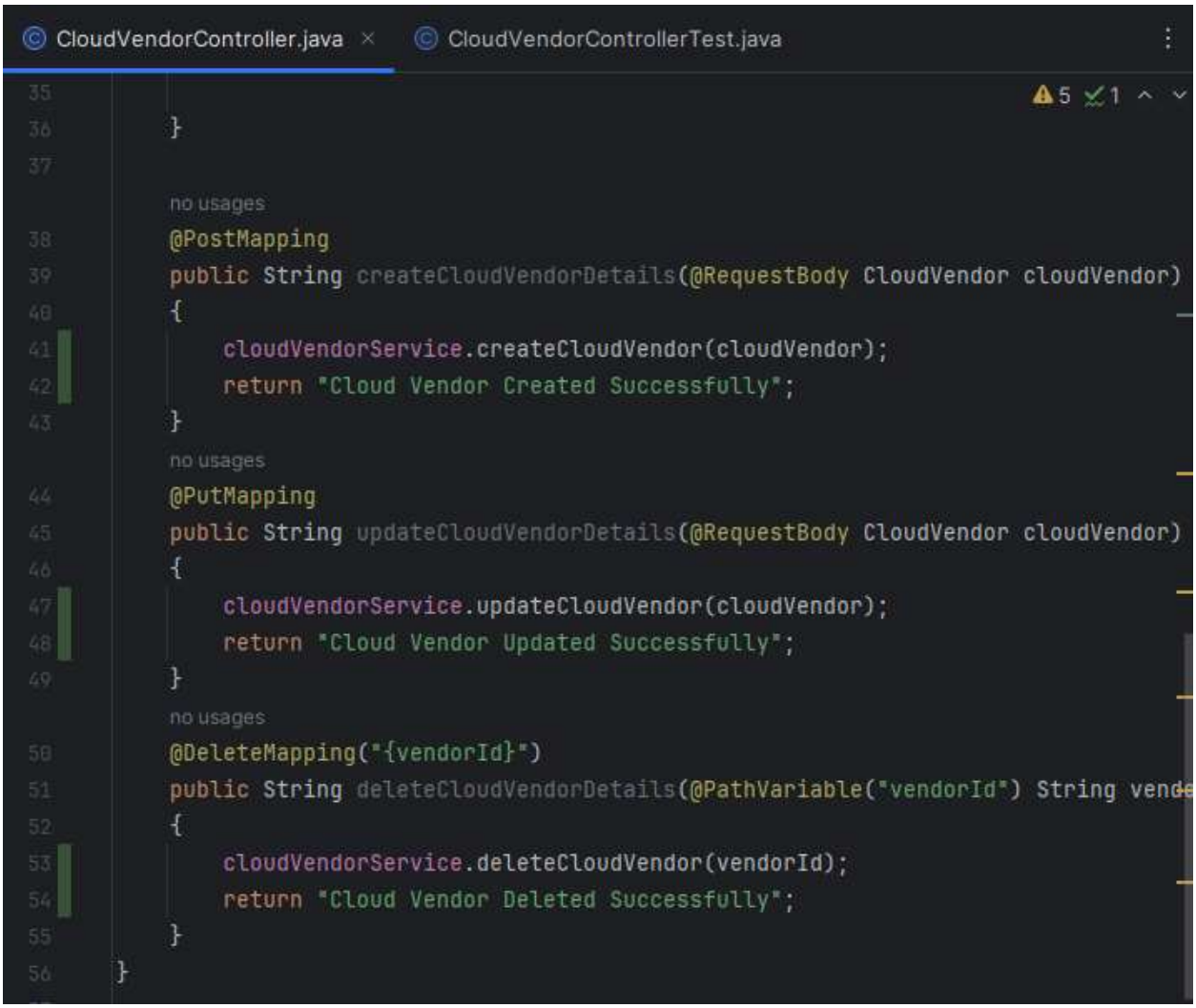







Coverage
CloudVendorControllerTest
Element
Class, %
Method, %
Line, %
com.thinkconstructive.restdemo.controller.
100% (1/1)
100% (6/6)
100% (11/11)
CloudVendorController
100% (1/1)
100% (6/6)
100% (11/11)


CloudVendorController.java
CloudVendorControllerTest.java
CloudVendorService cloudVendorService;
public CloudVendorController(CloudVendorService cloudVendorService) {
this.cloudVendorService = cloudVendorService;
}
// Read Specific Cloud Vendor Details from DB
@GetMapping("{vendorId}")
public ResponseEntity<Object> getCloudVendorDetails(@PathVariable("vendorId")
{
return ResponseHandler.responseBuilder(message: "Requested Vendor Details
HttpStatus.OK,cloudVendorService.getCloudVendor(vendorId));
}
// Read All Cloud Vendor Details from DB
@GetMapping()
public List<CloudVendor> getAllCloudVendorDetails()
{
return cloudVendorService.getAllCloudVendors();
}
Coverage
CloudVendorControllerTest
Element
Class, %
Method, %
Line, %
com.thinkconstructive.res 100% (1/1) 100% (6/6) 100% (11/11)
CloudVendorController 100% (1/1) 100% (6/6) 100% (11/11)



- Above all methods are green that means all method's lines of codes are tested/covered successfully.
- It's really good thing for this project.