



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

<i>Spring Boot Project</i>	1
<i>Session# 01</i>	4
“Java REST API with Spring Boot Tutorial REST API CRUD Implementation”	4
<i>Session# 02</i>	27
Creating Java REST API with Spring Boot, Spring Data JPA and MySQL REST API CRUD Operations.	27
<i>Session# 03</i>	59
Exception Handling in Spring Boot REST API	59
<i>Session# 04</i>	69
Java Spring Boot REST API JSON Response Handling	69
<i>Session# 05</i>	75
Master Unit Testing Java Spring Boot REST API Application in One Shot.....	75

Session# 01

“Java REST API with Spring Boot Tutorial | REST API CRUD Implementation”

- Creating basic spring boot application.

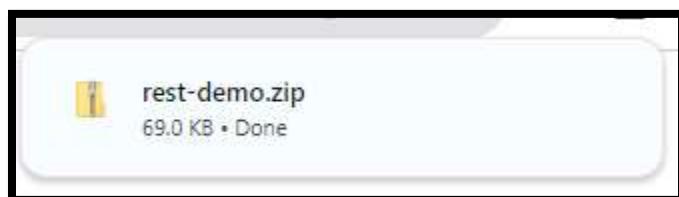
The screenshot shows the Spring Initializr web interface at start.spring.io. The configuration is set up for a Maven-based Spring Boot project:

- Project:** Maven (selected)
- Language:** Java (selected)
- Spring Boot:** 3.2.1 (selected)
- Project Metadata:**
 - Group: com.thinkconstructive
 - Artifact: rest-demo
 - Name: rest-demo
 - Description: Demo project for Spring Boot
 - Package name: com.thinkconstructive.rest-demo
 - Packaging: Jar (selected)
- Java:** 21 (selected)

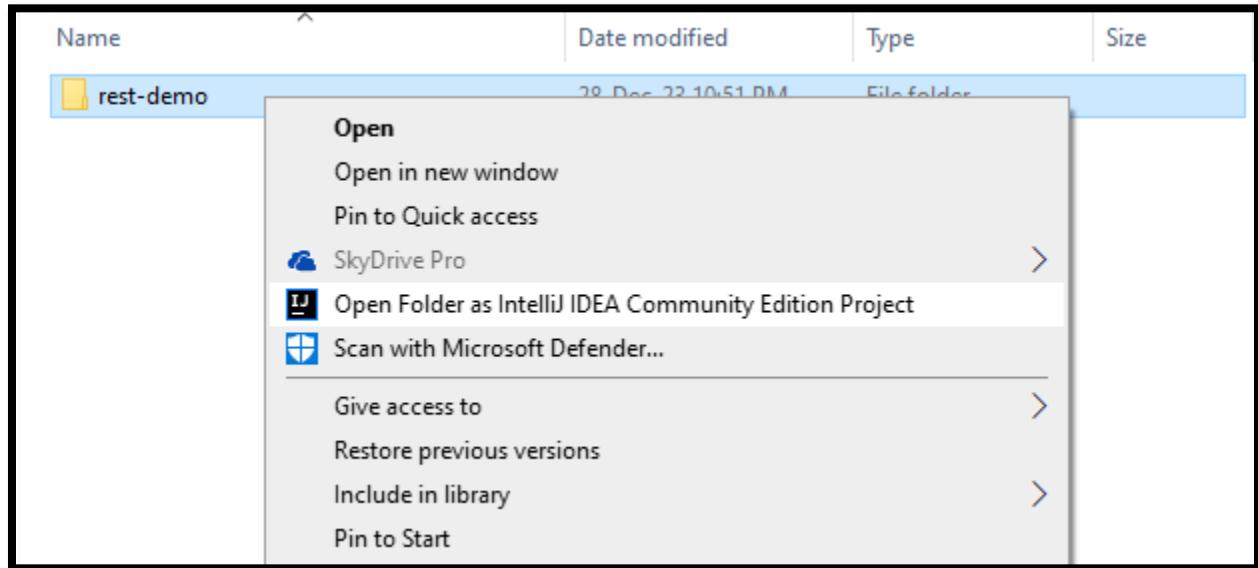
The screenshot shows the Spring Initializr web application. On the left, there are sections for Project (Grade: Groovy, Grade: Kotlin, Maven), Spring Boot (3.2.2 (SNAPSHOT), 3.2.1, 3.1.8 (SNAPSHOT), 3.1.7), Project Metadata (Group: com.thinkconstructive, Artifact: rest-demo, Name: rest-demo, Description: Demo project for Spring Boot, Package name: com.thinkconstructive.rest-demo, Packaging: Jar, Java: 21, 17), and Dependencies (Spring Web). A green bar at the top highlights "Spring Web WEB". A button on the right says "ADD DEPENDENCIES... CTRL + B".

This screenshot shows the same Spring Initializr interface but with different configurations. The Spring Boot version is set to 3.2.1. The dependencies section shows "Spring Web WEB" selected. The bottom right corner has an "Activate Windows" message.

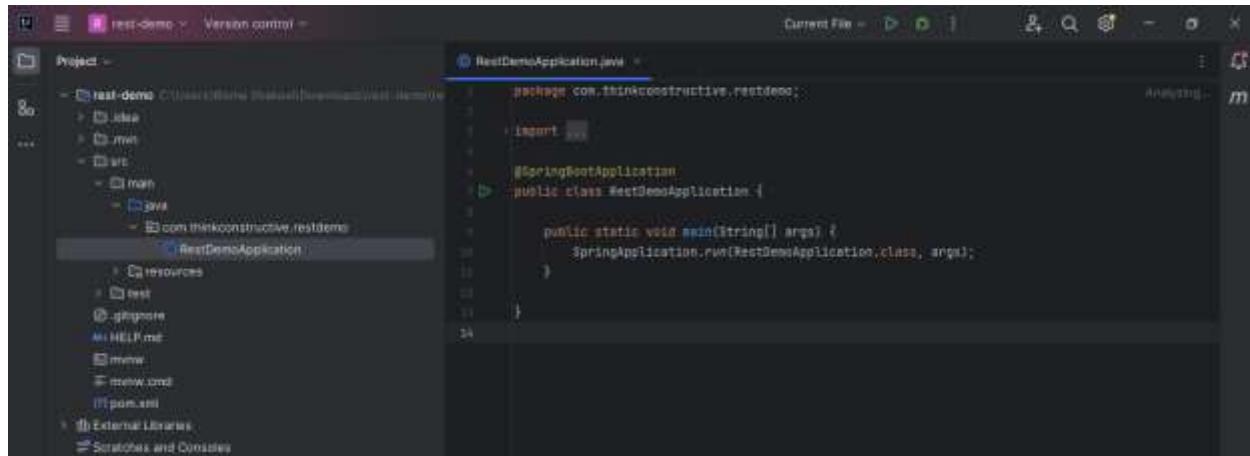
- Now Click on generate



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

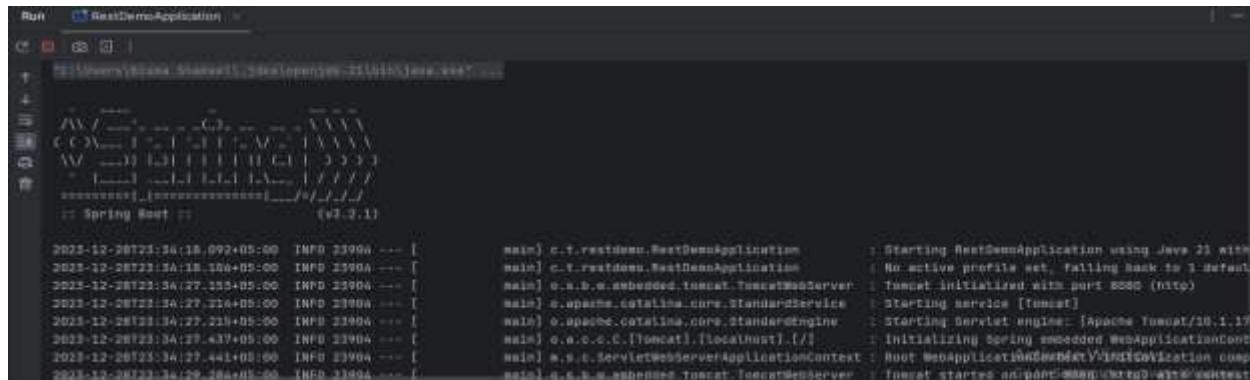


- Rest Demo Application



Output

- My Application is successfully started.

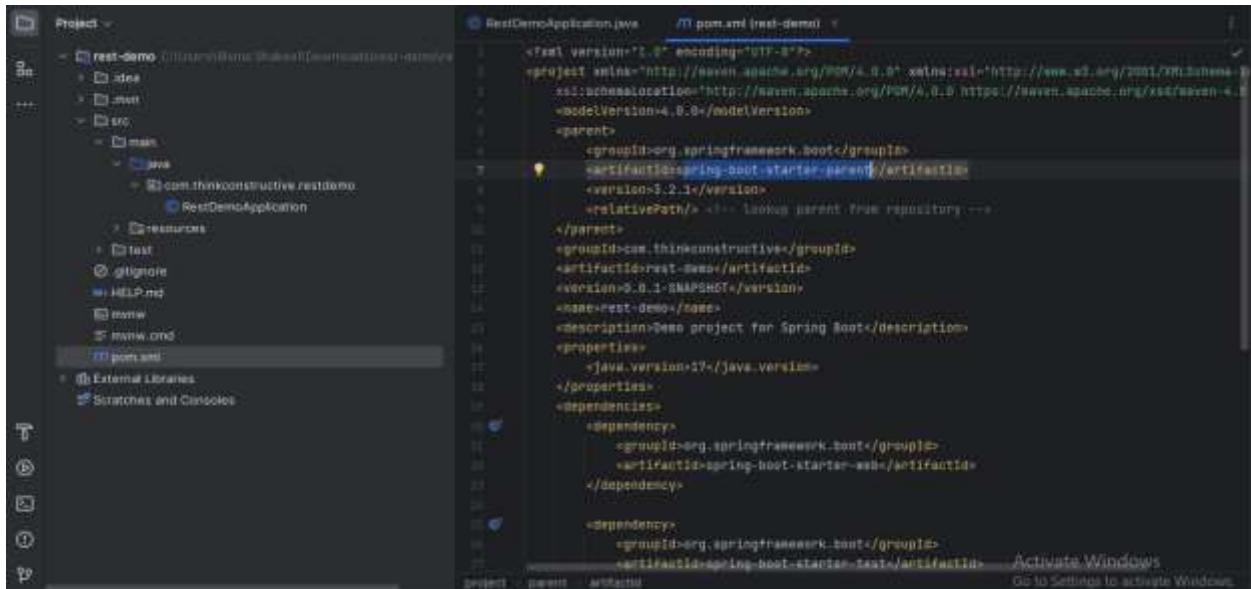


```

main] c.t.restdemo.RestDemoApplication   : Starting RestDemoApplication using Java 25 with PID 23984 (C:\Users\Bina\Downloads\rest-demo)
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.[localhost].[]           : Initializing Spring embedded WebApplicationContext
main] w.s.c.WebApplicationContext        : Root WebApplicationContext: initialization completed in 8613 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 15.895 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

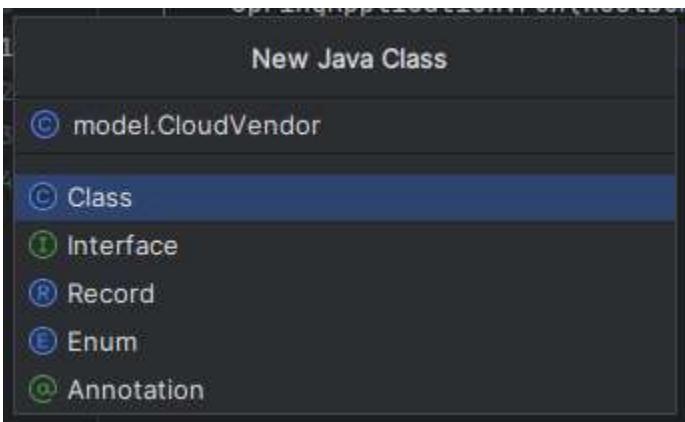
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Thu Dec 28 23:40:10 PKT 2023

There was an unexpected error (type=Not Found, status=404).

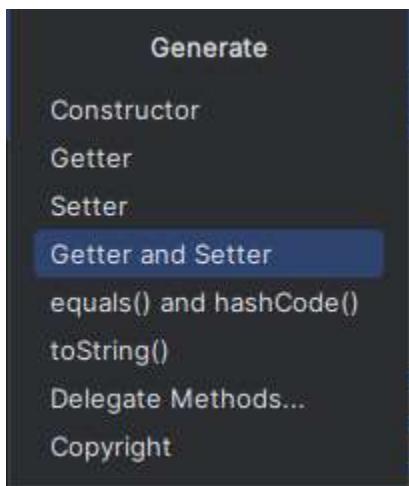
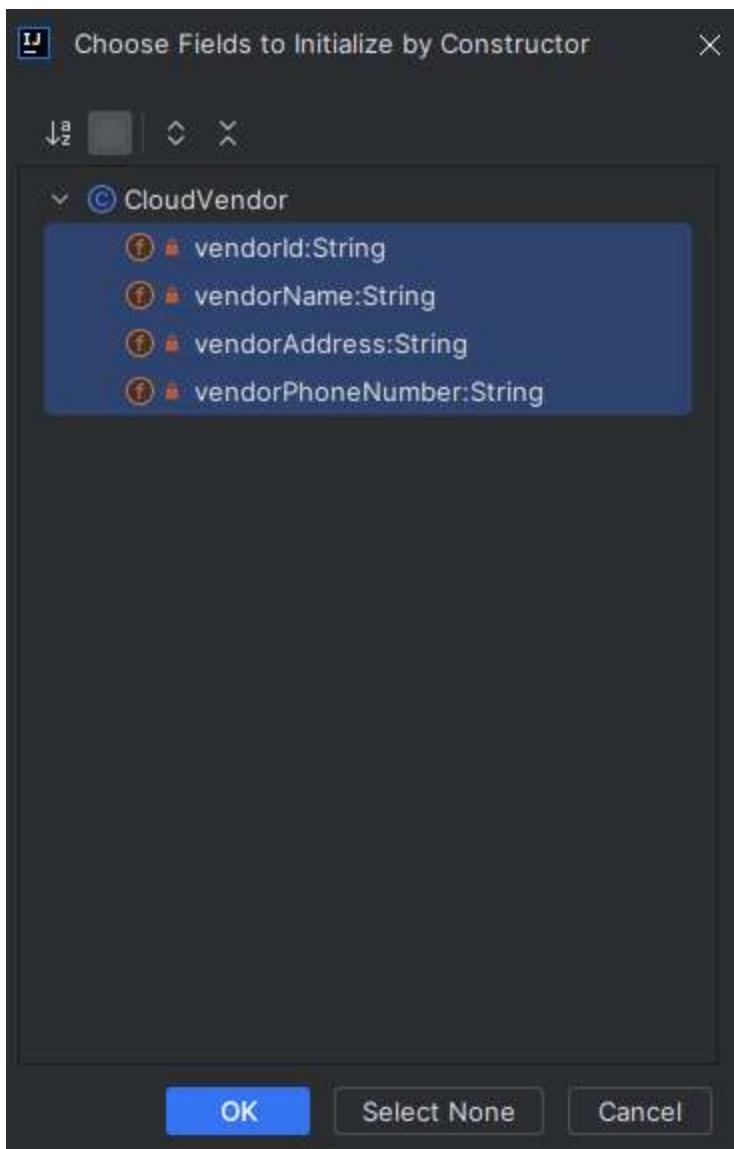
Making Rest API's

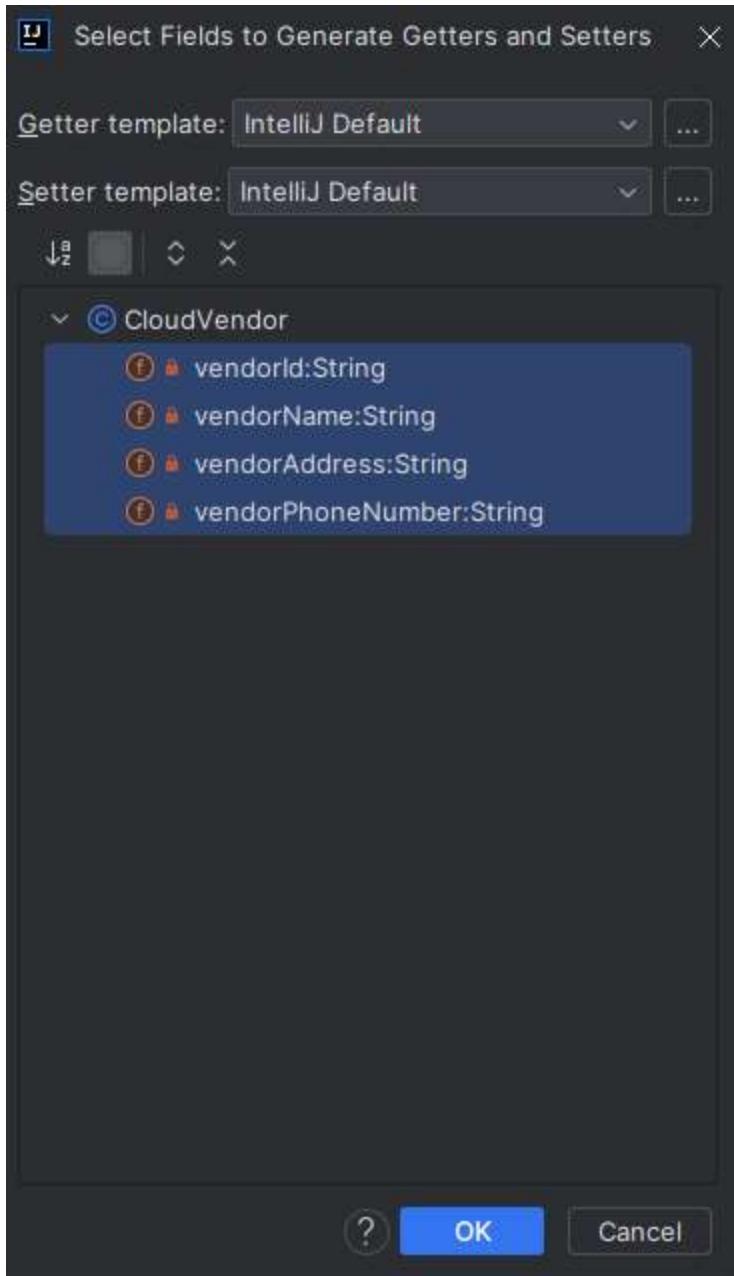
Cloud vendor API



```
1 public class CloudVendor {  
2  
3     private String vendorId;  
4     private String vendorName;  
5     private String vendorAddr;  
6     private String vendorPhon;  
7  
8 }
```

The code editor displays a Java class named 'CloudVendor' with four private string fields: 'vendorId', 'vendorName', 'vendorAddr', and 'vendorPhon'. A context menu is open over the first constructor line ('private String vendorId;'), showing options: 'Generate', 'Constructor' (which is highlighted), 'Getter', 'Setter', 'Getter and Setter', 'equals() and hashCode()', 'toString()', 'Delegate Methods...', and 'Copyright'. The status bar at the bottom right shows 'OFF'.





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;
}

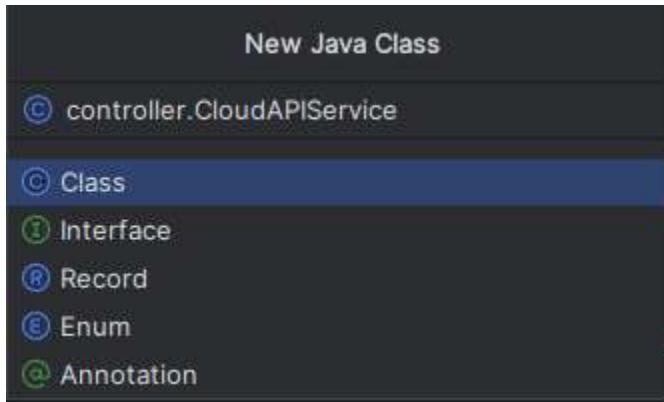
public String getVendorPhoneNumber() {
    return vendorPhoneNumber;
}

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)
1 package com.thinkconstructive.restdemo.controller.controller;
2
3
4 import com.thinkconstructive.restdemo.controller.CloudVendor;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RestController;
8
9 no usages
10 @RestController
11 @RequestMapping("/cloudvendor")
12 public class CloudAPIService {
13     no usages
14     @GetMapping("{vendorId}")
15     public CloudVendor getCloudVendorDetails(String vendorId)
16     {
17         return new CloudVendor( vendorId: "C1", vendorName: "Vendor 1",
18             vendorAddress: "Address One", vendorPhoneNumber: "XXXXXX");
19     }
20 }
```

```

2023-12-29T09:39:58.961+05:00 INFO 3458 --- [main] c.t.r.controller.RestDemoApplication : Starting RestDemoApplication using Java 21 with PID 3458
2023-12-29T09:39:58.961+05:00 INFO 3458 --- [main] c.t.r.controller.RestDemoApplication : No active profile set, falling back to: default
2023-12-29T09:39:58.188+05:00 INFO 3458 --- [main] o.s.w.e.EmbeddedTomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2023-12-29T09:39:58.214+05:00 INFO 3458 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-12-29T09:39:58.256+05:00 INFO 3458 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.17]
2023-12-29T09:39:58.443+05:00 INFO 3458 --- [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2023-12-29T09:39:58.449+05:00 INFO 3458 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 7 ms
2023-12-29T09:39:59.026+05:00 INFO 3458 --- [main] o.s.w.e.EmbeddedTomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path [rest-demo]
2023-12-29T09:39:59.058+05:00 INFO 3458 --- [main] c.t.r.controller.RestDemoApplication : Started RestDemoApplication in 7.096 seconds (process completed in 714 ms)

```

Cloud Vendor Properties:

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

```
{"vendorId": "C1", "vendorName": "Vendor 1", "vendorAddress": "Address One", "vendorPhoneNumber": "xxxxx"}
```

```
{"vendorId": "C1", "vendorName": "Vendor 1", "vendorAddress": "Address One", "vendorPhoneNumber": "xxxxx"}
```

Postman

- Go to Postman

The screenshot shows the Postman application interface. On the left, there's a sidebar titled "My Workspace" with options like "Collections", "Environments", and "History". A collection named "Rest-Demo1" is selected, indicated by a blue arrow icon. The main area is titled "Rest-Demo1" and contains the message "This collection is empty. Add a request to start working.". Below this, there are tabs for "Overview", "Authorization", "Pre-request Script", "Tests", "Variables", and "Runs". In the center, there's a button labeled "View complete documentation →". On the right side, there are sections for "Created by" (with a profile picture of "You") and "Created on" (29 Dec 2023, 9:46 AM). At the bottom, there's a "Send" button and some status indicators.

This screenshot shows the "Req1" request configuration in Postman. The method is set to "GET" and the URL is "http://localhost:8080/cloudVendor/C1". The "Params" tab is selected, showing a table for "Query Params" with columns for "Key", "Value", "Description", and "Edit". There are no entries in this table. Other tabs visible include "Authorization", "Headers (0)", "Body", "Pre-request Script", "Tests", "Settings", and "Cookies".

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/cloudvendor/C1`. The response is a 200 OK status with a response body containing the following JSON:

```
1  {"vendorId": "C1", "vendorName": "Vendor 1", "vendorAddress": "Address One", "vendorPhoneNumber": "xxxxxx"}
```

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", "xxxxxx");
    }
    @PostMapping
    public String createCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
    {
        this.cloudVendor = cloudVendor;
        return "Cloud Vendor Created Successfully";
    }
}
```

```
Run com.thinkconstructive.restdemo.controller.RestDemo4...  
[INFO] [main] 2023-12-29T10:02:22.025+05:00 INFO 25504 --- [main] c.t.r.controller.RestDemoApplication : Starting RestDemoApplication using Java 21 with  
[INFO] [main] 2023-12-29T10:02:22.058+05:00 INFO 25504 --- [main] c.t.r.controller.RestDemoApplication : No active profile set, falling back to the default  
[INFO] [main] 2023-12-29T10:02:27.462+05:00 INFO 25504 --- [main] o.s.d.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)  
[INFO] [main] 2023-12-29T10:02:27.548+05:00 INFO 25504 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]  
[INFO] [main] 2023-12-29T10:02:27.351+05:00 INFO 25504 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.17]  
[INFO] [main] 2023-12-29T10:02:27.943+05:00 INFO 25504 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext  
[INFO] [main] 2023-12-29T10:02:27.946+05:00 INFO 25504 --- [main] w.s.e.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 11.58 seconds  
[INFO] [main] 2023-12-29T10:02:31.977+05:00 INFO 25504 --- [main] o.s.d.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context  
[INFO] [main] 2023-12-29T10:02:32.003+05:00 INFO 25504 --- [main] c.t.r.controller.RestDemo4Application : Started RestDemo4Application in 11.58 seconds (pr  
Activate Windows  
Go to Settings to activate Windows.
```

Rest-Demo1 / Req1

POST <http://localhost:8080/cloudvendor> Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

Body (Pretty, Raw, Preview, Visualize, JSON)

```
1 {  
2   "vendorId": "C2",  
3   "vendorName": "Vendor Two",  
4   "vendorAddress": "Address Two",  
5   "vendorPhoneNumber": "xxxxxx"  
}
```

Body Cookies Headers (5) Test Results 200 OK 603 ms 263 B Save as example

```
1 {  
2   "vendorId": "C1",  
3   "vendorName": "Vendor 1",  
4   "vendorAddress": "Address One",  
5   "vendorPhoneNumber": "xxxxxx"  
}
```

Activate Windows
Go to Settings to activate Windows.

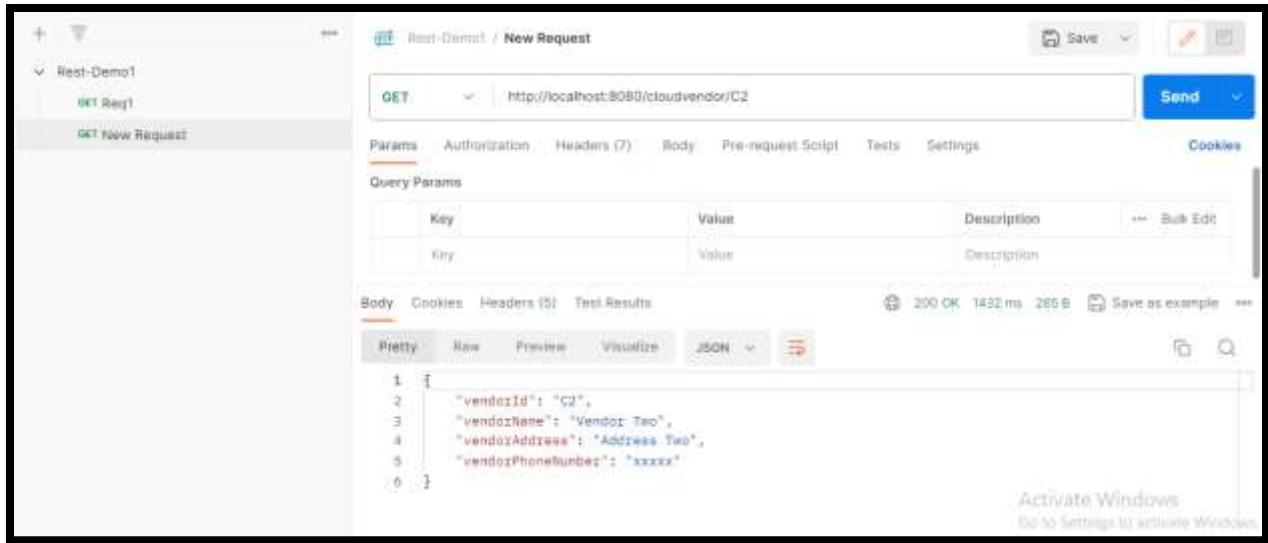
The screenshot shows the Postman interface with a successful API call. The request method is POST, directed to `http://localhost:8080/cloudvendor`. The request body is a JSON object:

```
1 {
2   "vendorId": "C2",
3   "vendorName": "Vendor Two",
4   "vendorAddress": "Address Two",
5   "vendorPhoneNumber": "xxxxx"
}
```

The response status is 200 OK, with a response time of 1836 ms and a response size of 197 B. The response body contains the message: "1 Cloud Vendor Created Successfully".

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/cloudvendor/C2`. The request includes a "Query Params" section with two entries:

Key	Value	Description
Key	Value	Description



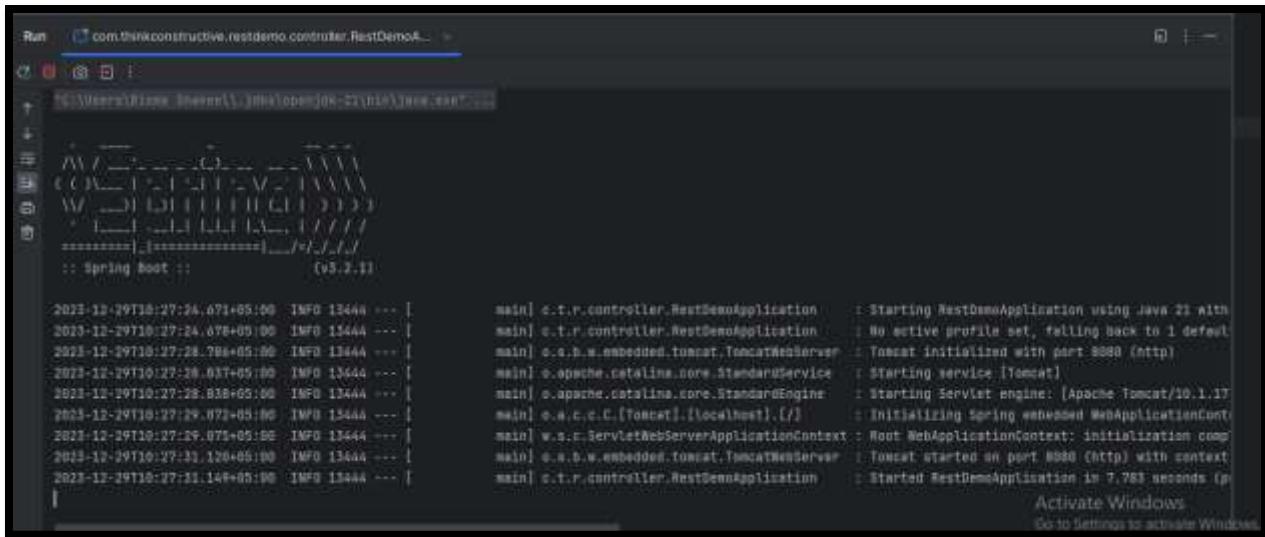
Code:

```
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", "xxxxxx");
    }
    @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:

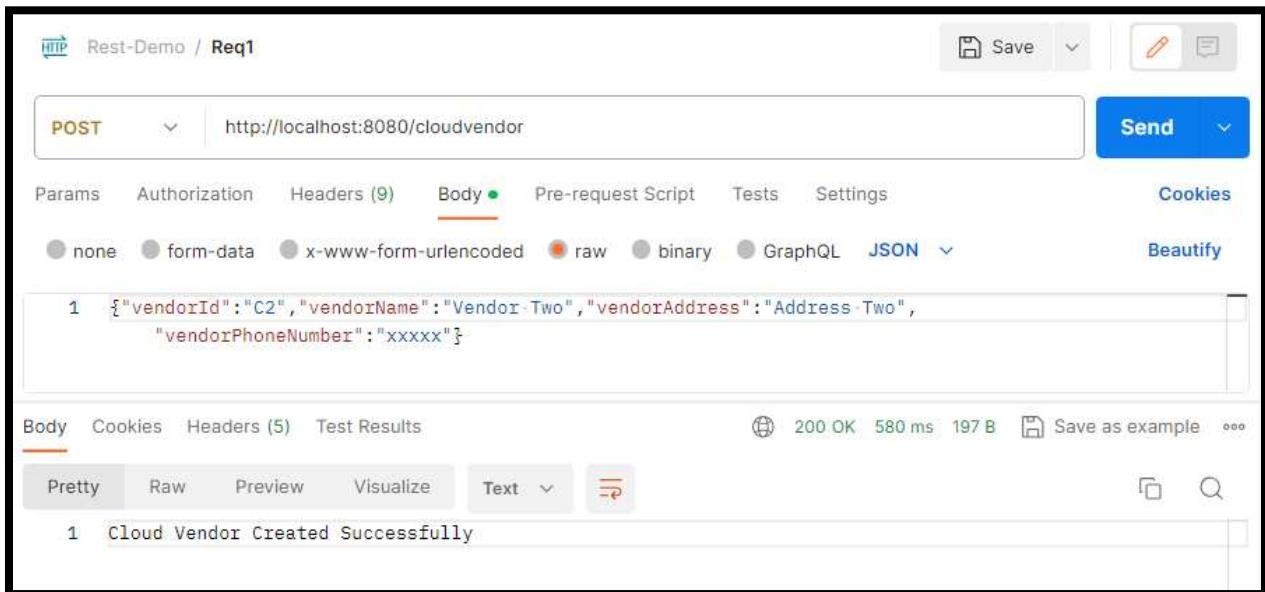


```
Run com.thinkconstructive.restdemo.controller.RestDemoA... >

C:\Users\Utsava\OneDrive\Documents\NetBeansProjects\RestDemoA>cd ..\..\bin&open(jdk-21\bin\java.exe)

:: Spring Boot :: (v3.2.11)

2023-12-29T10:27:24.673+05:00 INFO 13444 --- [main] c.t.r.controller.RestDemoApplication : Starting RestDemoApplication using Java 21 with
2023-12-29T10:27:24.678+05:00 INFO 13444 --- [main] c.t.r.controller.RestDemoApplication : No active profile set, falling back to 1 default
2023-12-29T10:27:28.798+05:00 INFO 13444 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port: 8080 (http)
2023-12-29T10:27:29.037+05:00 INFO 13444 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-12-29T10:27:29.838+05:00 INFO 13444 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.17]
2023-12-29T10:27:29.872+05:00 INFO 13444 --- [main] o.a.c.c.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-12-29T10:27:29.875+05:00 INFO 13444 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 7 ms
2023-12-29T10:27:31.126+05:00 INFO 13444 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context
2023-12-29T10:27:31.146+05:00 INFO 13444 --- [main] c.t.r.controller.RestDemoApplication : Started RestDemoApplication in 7.783 seconds (jpa)
```



HTTP Rest-Demo / Req1

POST <http://localhost:8080/cloudvendor> Send

Params Authorization Headers (9) Body **Pre-request Script** Tests Settings Cookies Beautify

Body

```
1 {"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

```
1 Cloud Vendor Created Successfully
```

HTTP Rest-Demo / New Request

GET http://localhost:8080/cloudvendor/C2

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

Pretty Raw Preview Visualize JSON

```
1 "vendorId": "C2",
2 "vendorName": "Vendor Two",
3 "vendorAddress": "Address Two",
4 "vendorPhoneNumber": "xxxxx"
```

Activate Windows

+ Rest-Demo / New Request

PUT http://localhost:8080/cloudvendor

Params Authorization Headers (0) Body Pre-request Script Tests Settings Cookies

Body

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beauty

```
1 "vendorId": "C2",
2 "vendorName": "Vendor Two UPDATED",
3 "vendorAddress": "Address Two",
4 "vendorPhoneNumber": "xxxxx"
```

Response

The screenshot shows the Postman interface with a 'PUT' request to `http://localhost:8080/cloudvendor`. The request body contains the following JSON:

```

1  {
2     "vendorId": "C2",
3     "vendorName": "Vendor Two UPDATED",
4     "vendorAddress": "Address Two",
5     "vendorPhoneNumber": "xxxxx"
6   }

```

The response status is 200 OK, and the message is "Cloud Vendor Updated Successfully".

- Now We will get the updated one.

The screenshot shows the Postman interface with a 'GET' request to `http://localhost:8080/cloudvendor/C2`. The response status is 200 OK, and the message is "Cloud Vendor Updated Successfully".

Creating Delete API

Code

```

package com.thinkconstructive.restdemo.controller.controller;

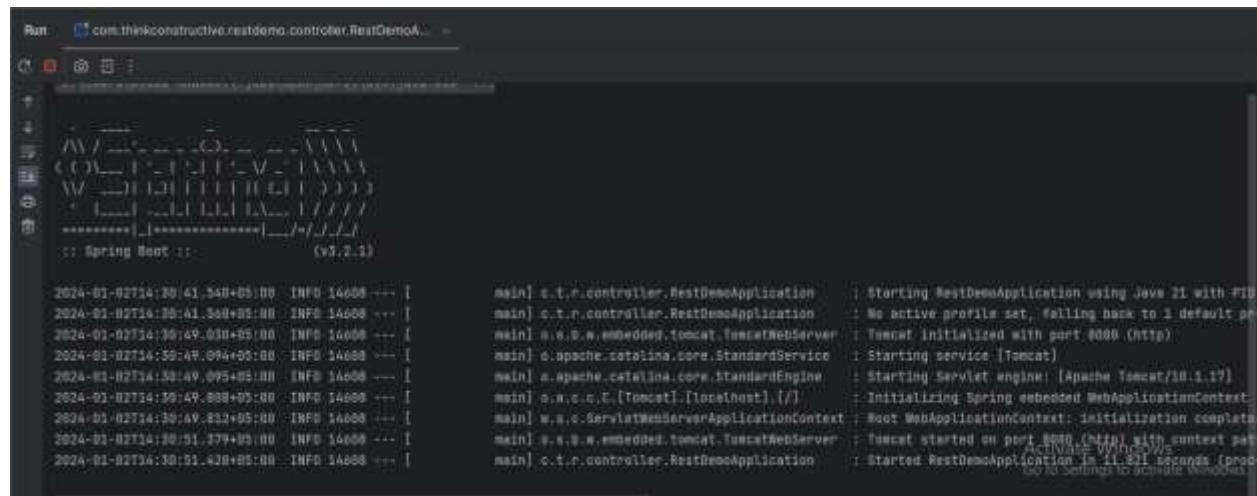
import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/cloudvendor")

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

```

```
public CloudVendor getCloudVendorDetails(String vendorId)
{
    return cloudVendor;
    //new CloudVendor("C1", "Vendor 1",
    //"Address One", "xxxxxx");
}
@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

HTTP Rest-Demo / Req1

POST http://localhost:8080/cloudvendor

Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body (none) form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {"vendorId": "C3",
2 "vendorName": "Vendor Three",
3 "vendorAddress": "Address Three",
4 "vendorPhoneNumber": "yyyyy"
5 }
```

Body Cookies Headers (5) Test Results

200 OK 243 ms 197 B Save as example

Pretty Raw Preview Visualize Text

1 Cloud Vendor Created Successfully

Activate Windows

This screenshot shows a POST request in Postman. The URL is http://localhost:8080/cloudvendor. The request body is a JSON object with fields: vendorId (C3), vendorName (Vendor Three), vendorAddress (Address Three), and vendorPhoneNumber (yyyyy). The response status is 200 OK, and the message is "Cloud Vendor Created Successfully".

POST Req1

GET New Request

HTTP Rest-Demo / New Request

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

```
1 {
2     "vendorId": "C3",
3     "vendorName": "Vendor Three",
4     "vendorAddress": "Address Three",
5     "vendorPhoneNumber": "yyyyy"
6 }
```

Activate Windows

This screenshot shows a GET request in Postman. The URL is http://localhost:8080/cloudvendor/C3. The response status is 200 OK, and the returned JSON object matches the one sent in the previous POST request.

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

POST Req1 GET New Request GET New Request DEL New Request + No Environment

HTTP Rest-Demo / New Request

DELETE 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 16 ms 197 B Save as example ...

Pretty Raw Preview Visualize Text `1 Cloud Vendor Deleted Successfully`

DELETE 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 16 ms 197 B Save as example ...

Pretty Raw Preview Visualize Text `1 Cloud Vendor Deleted Successfully`

200 OK

Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action.

- 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.

The screenshot shows the Postman application interface. At the top, there are tabs for 'POST Req1' (red), 'GET New Request' (green), 'GET New Request' (green), 'DEL New Request' (red), a '+' button, and 'No Environment'. Below the tabs, the URL is set to 'Rest-Demo / New Request' and the method is 'GET'. The request URL is 'http://localhost:8080/cloudvendor/C3'. On the right, there are 'Save' and 'Edit' buttons. The 'Params' tab is selected, showing a table for 'Query Params' with two rows: 'Key' and 'Value'. Below the table, there are tabs for 'Body', 'Cookies', 'Headers (4)', and 'Test Results'. The 'Test Results' tab is active, showing a response status of '200 OK' with '35 ms' latency and '123 B' size. Below the tabs, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', 'Text', and 'Copy'. The main content area displays the response body, which is currently empty.

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.*;

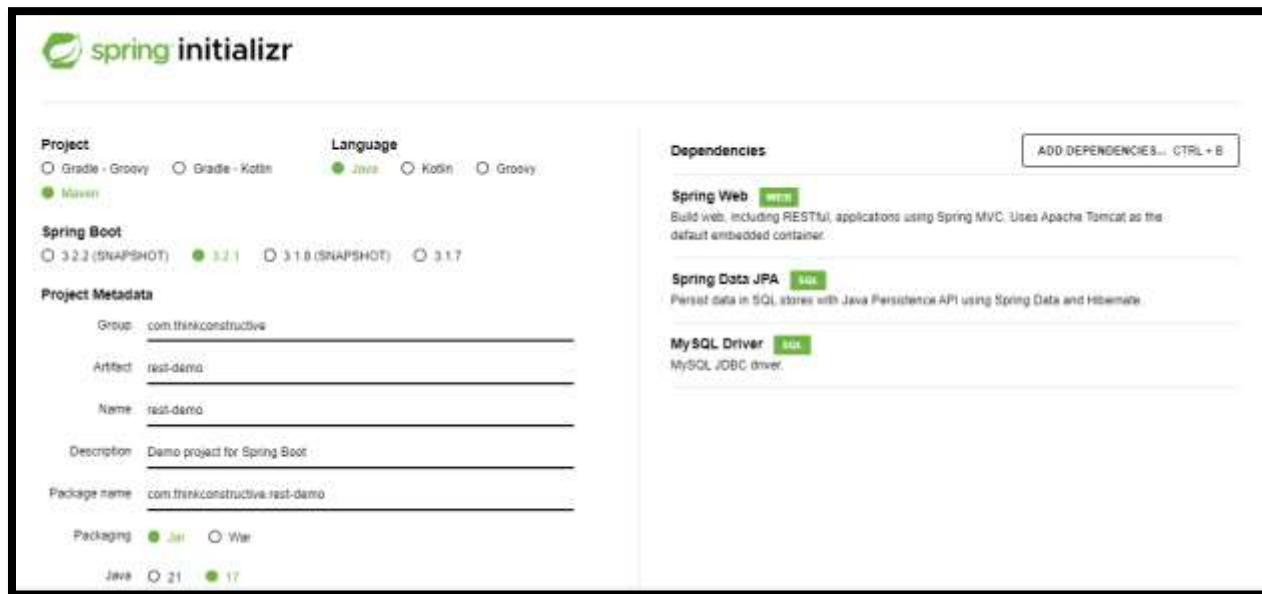
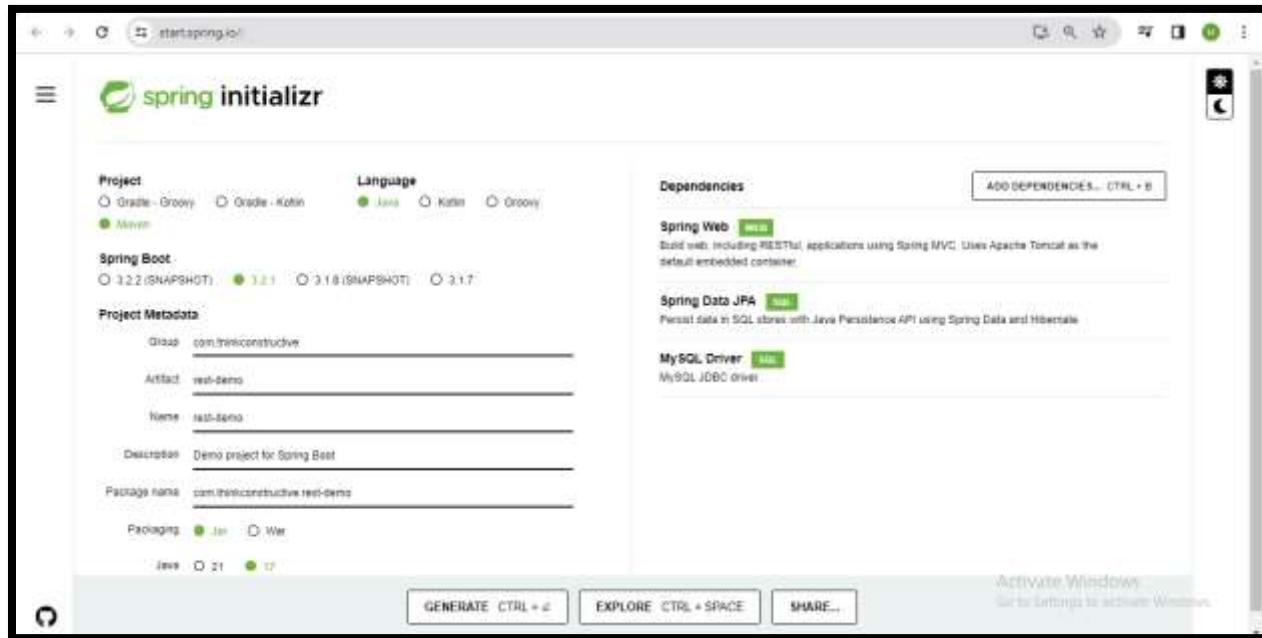
@RestController
@RequestMapping("/cloudvendor")

public class CloudAPIService
{
    CloudVendor cloudVendor;
    @GetMapping("/{vendorId}")
    public CloudVendor getCloudVendorDetails(String vendorId)
    {
        return cloudVendor;
        //new CloudVendor("C1","Vendor 1",
        //"Address One","xxxxxx");
    }
    @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.

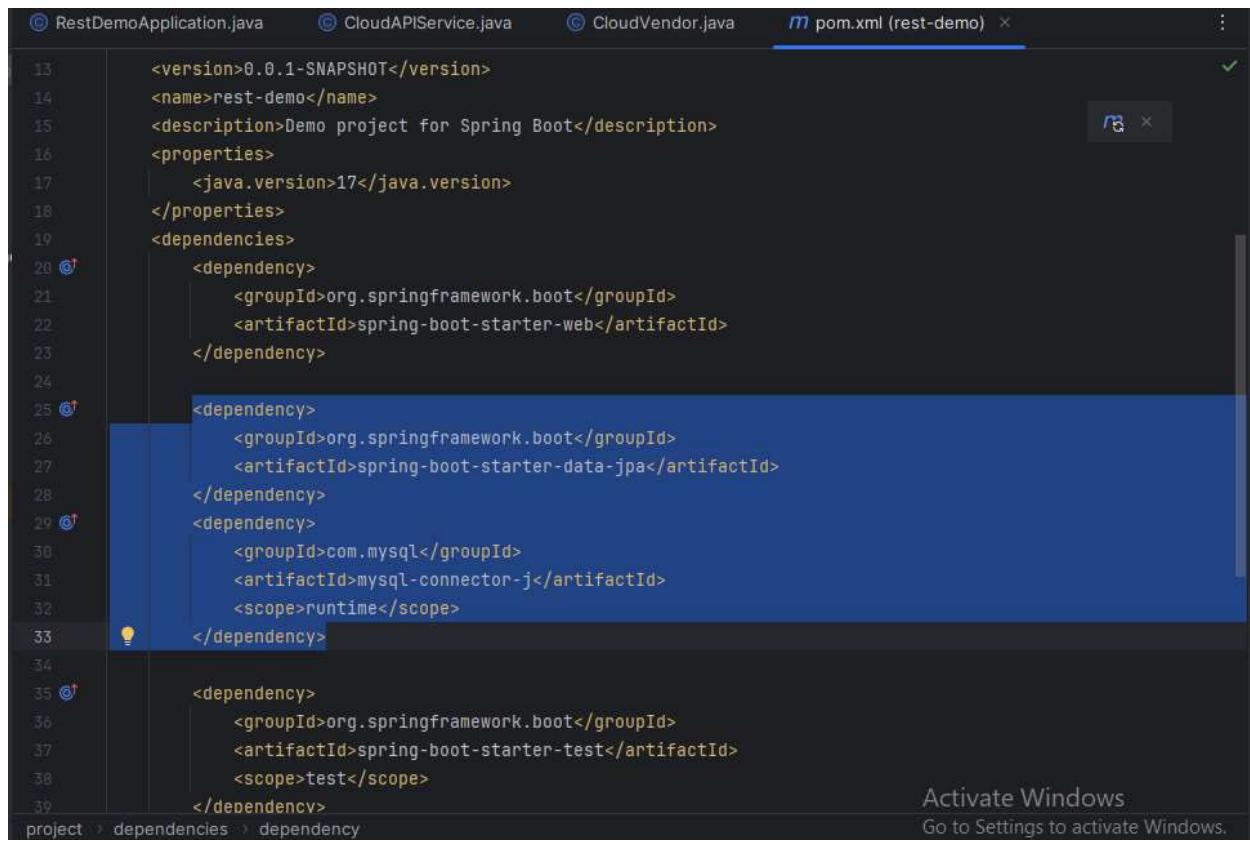


- Now, Open the previous session Rest-Demo project in intelliJ IDEA, in which we have created the controller layer. And add dependencies in Pom.xml
- Currently we have these dependencies.

```
15 <description>Demo project for Spring Boot</description>
16 <properties>
17   <java.version>17</java.version>
18 </properties>
19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-web</artifactId>
23   </dependency>
24
25   <dependency>
26     <groupId>org.springframework.boot</groupId>
27     <artifactId>spring-boot-starter-test</artifactId>
28     <scope>test</scope>
29   </dependency>
30 </dependencies>
```

The screenshot shows the IntelliJ IDEA interface with the 'pom.xml (rest-demo)' tab selected. The code editor displays the XML configuration for the project, specifically the dependencies section. It includes two dependency entries: one for 'spring-boot-starter-web' and another for 'spring-boot-starter-test' with a 'test' scope. Line numbers 15 through 30 are visible on the left side of the code editor.

- Add Jpa and sql dependency in it.



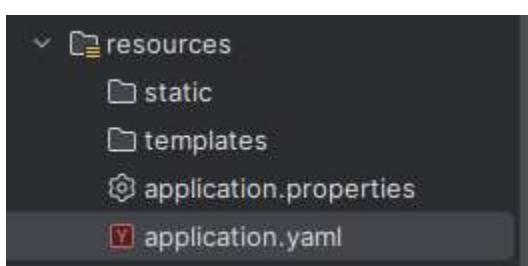
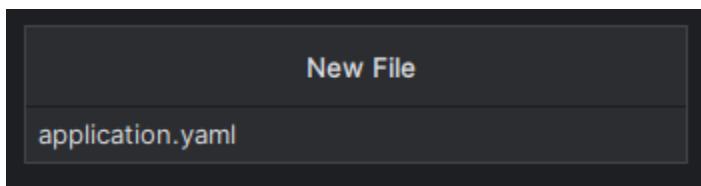
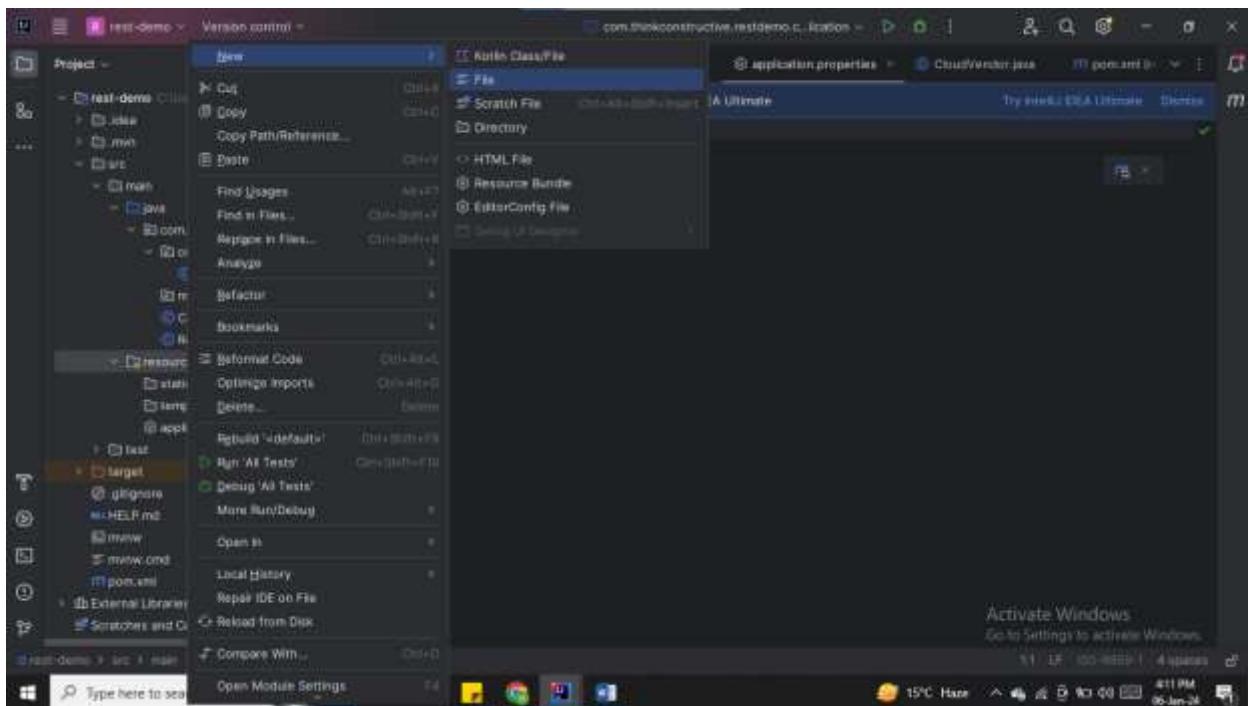
```

13 <version>0.0.1-SNAPSHOT</version>
14 <name>rest-demo</name>
15 <description>Demo project for Spring Boot</description>
16 <properties>
17   <java.version>17</java.version>
18 </properties>
19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-web</artifactId>
23   </dependency>
24
25   <dependency>
26     <groupId>org.springframework.boot</groupId>
27     <artifactId>spring-boot-starter-data-jpa</artifactId>
28   </dependency>
29   <dependency>
30     <groupId>com.mysql</groupId>
31     <artifactId>mysql-connector-j</artifactId>
32     <scope>runtime</scope>
33   </dependency>
34
35   <dependency>
36     <groupId>org.springframework.boot</groupId>
37     <artifactId>spring-boot-starter-test</artifactId>
38     <scope>test</scope>
39   </dependency>

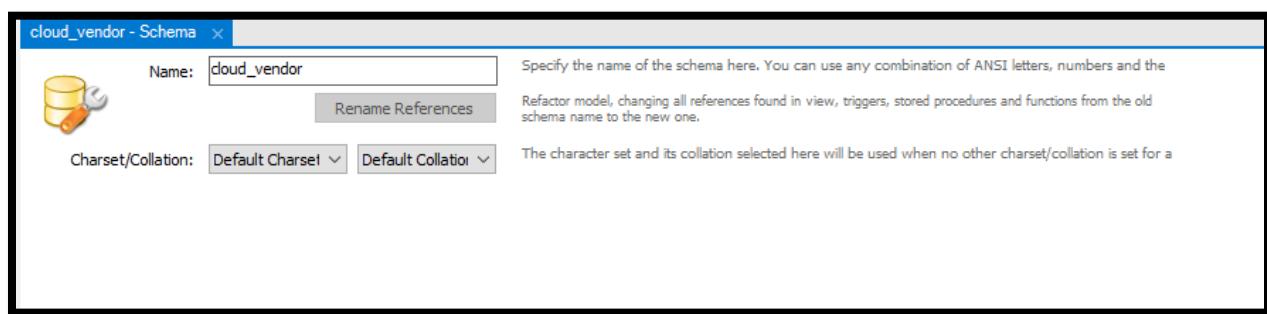
```

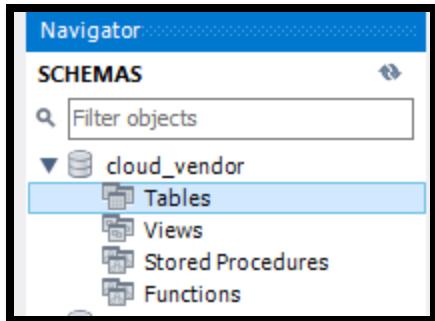
Activate Windows
Go to Settings to activate Windows.

- 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.

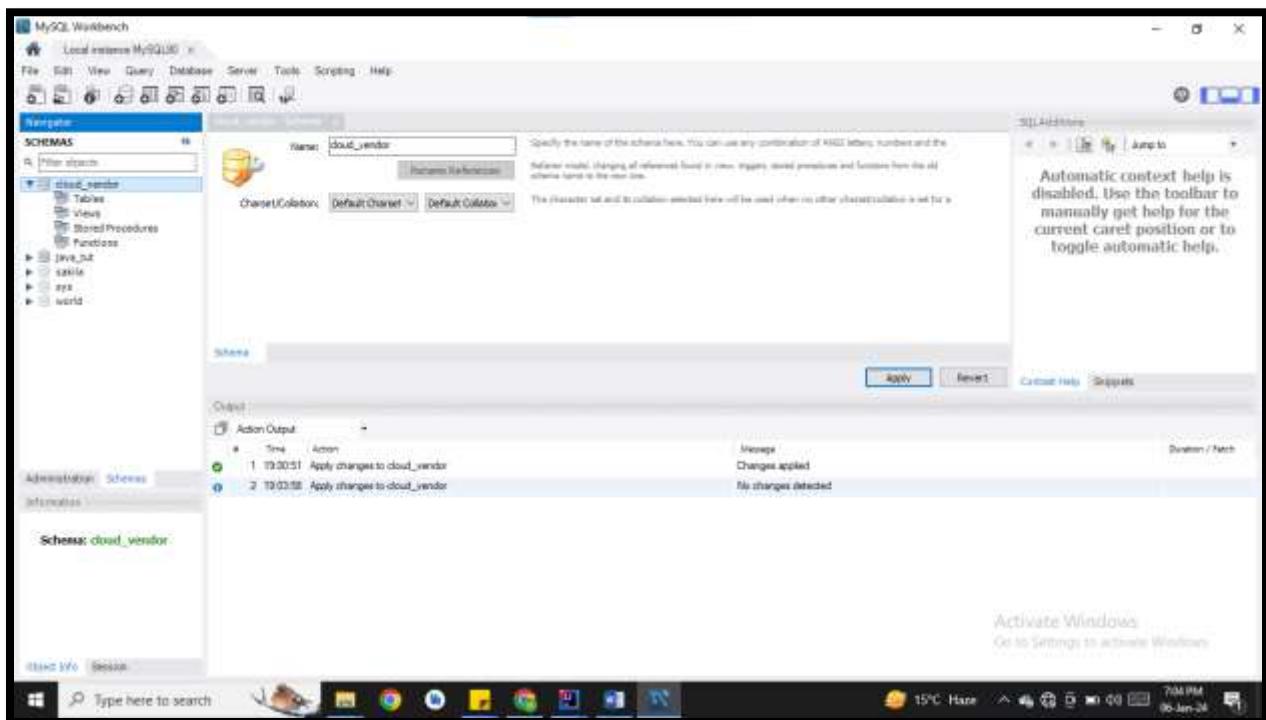


- Making new schema in MySQL Workbench





- 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.

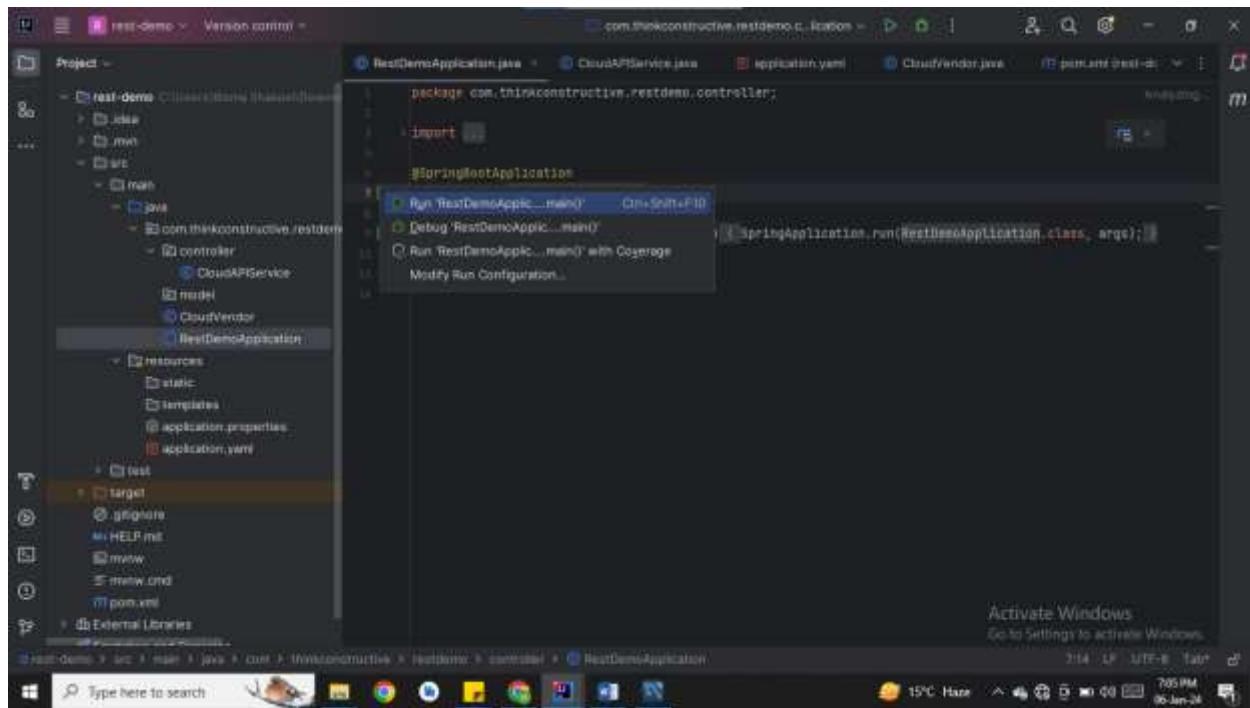


```

1 spring:
2   datasource:
3     url: jdbc:mysql://localhost:3306/cloud_vendor?useSSL=false
4     username: root
5     password: root
6
7 #JPA Settings
8 jpa.hibernate.ddl_auto: create

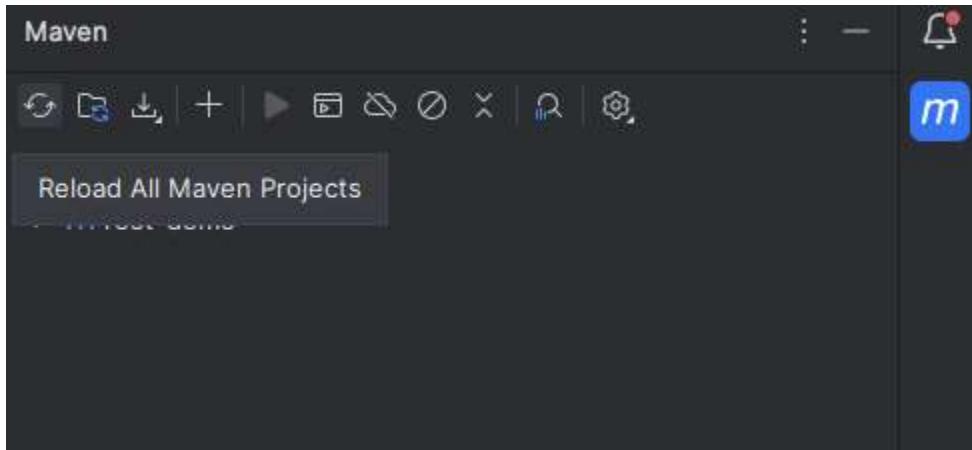
```

- 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.

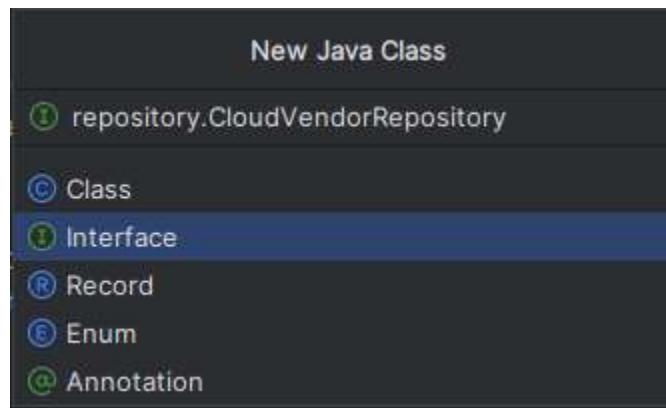


- Do these changes in CloudVendor.java Class

emoApplication.java CloudAPIService.java application.yaml CloudVendor.java pom.xml (rest-demo) : 10

```
1 package com.thinkconstructive.restdemo.controller;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.Id;
5 import jakarta.persistence.Table;
6
7 5 usages
8 @Entity
9 @Table(name="cloud_vendor_info")
10
11 public class CloudVendor {
12
13     3 usages
14     @Id
15     private String vendorId;
16     3 usages
17     private String vendorName;
18     3 usages
19     private String vendorAddress;
20     3 usages
21     private String vendorPhoneNumber;
22
23     no usages
24     public String getVendorId() {
25         return vendorId;
26     }
27
28     no usages
29 }
```

- Create new repository



```
emoApplication.java  CloudAPIService.java  CloudVendorRepository.java  application.yaml  CloudVendor.java  : 1 package com.thinkconstructive.restdemo.controller.repository; 2 3 import com.thinkconstructive.restdemo.controller.CloudVendor; 4 import org.springframework.data.jpa.repository.JpaRepository; 5 6 no usages 7 public interface CloudVendorRepository extends JpaRepository<CloudVendor, String> { 8 } 9 10
```

```

package com.thinkconstructive.restdemo.controller.repository;

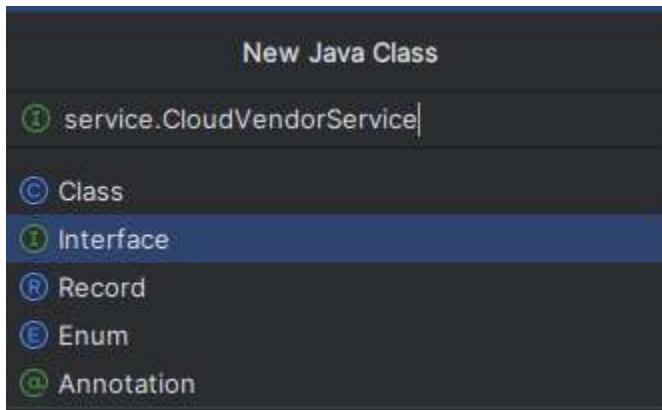
import com.thinkconstructive.restdemo.controller.CloudVendor;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CloudVendorRepository extends JpaRepository<CloudVendor,
String> {

}

```

- Create New Repository



- Cloud vendor service interface is defined.

The screenshot shows a code editor with several tabs at the top: 'demoApplication.java', 'CloudAPIService.java', 'CloudVendorRepository.java', 'CloudVendorService.java' (which is currently active and has a blue underline), and 'application.properties'. The 'CloudVendorService.java' tab contains the following code:

```

package com.thinkconstructive.restdemo.controller.service;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import java.util.List;

no usages
public interface CloudVendorService {
    no usages
    public String createCloudVendor(CloudVendor cloudVendor);
    no usages
    public String updateCloudVendor(CloudVendor cloudVendor);
    no usages
    public String deleteCloudVendor(String cloudVendorId);
    no usages
    public CloudVendor getCloudVendor(String cloudVendorId);
    no usages
    public List<CloudVendor> getAllCloudVendors();
}

```

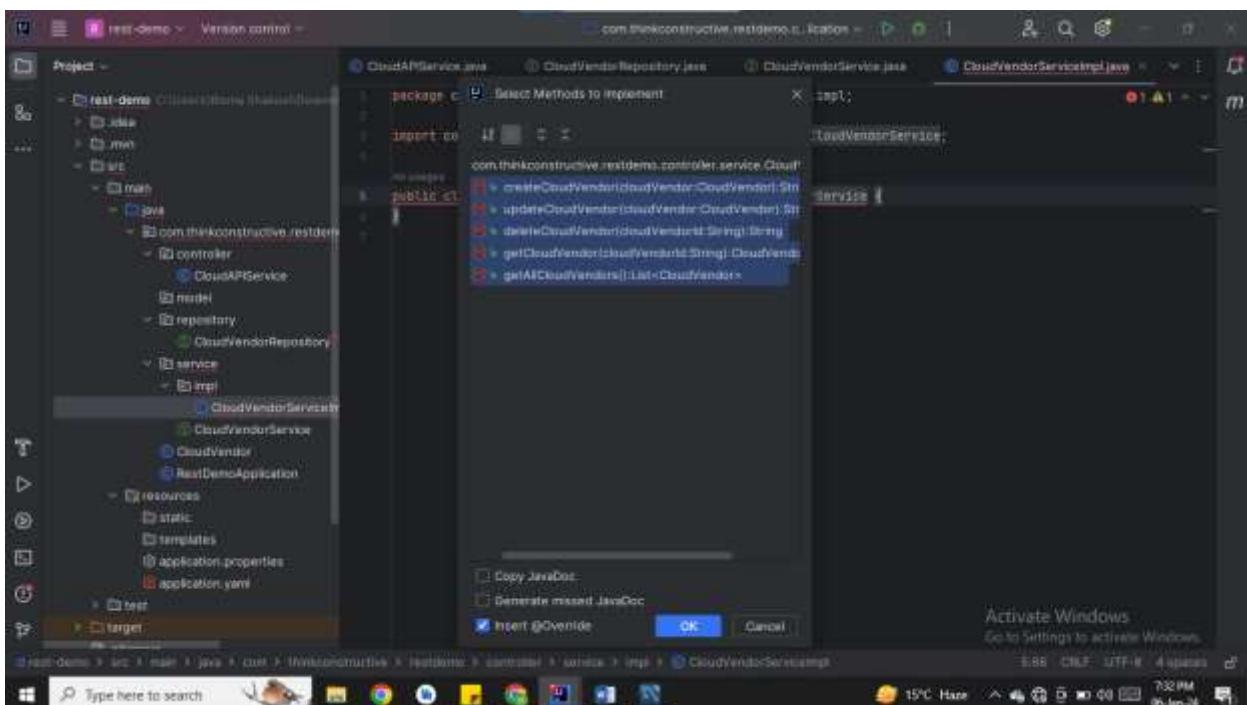
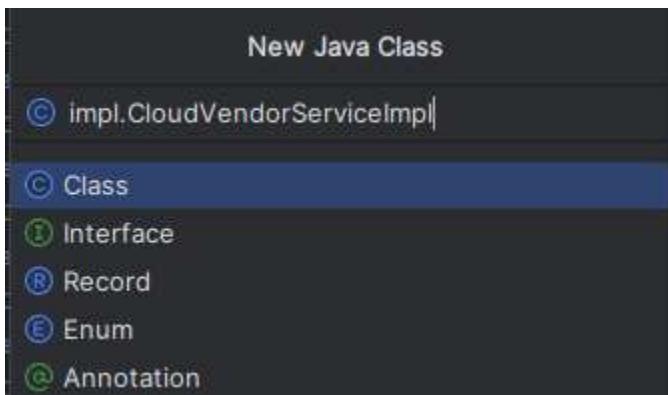
```

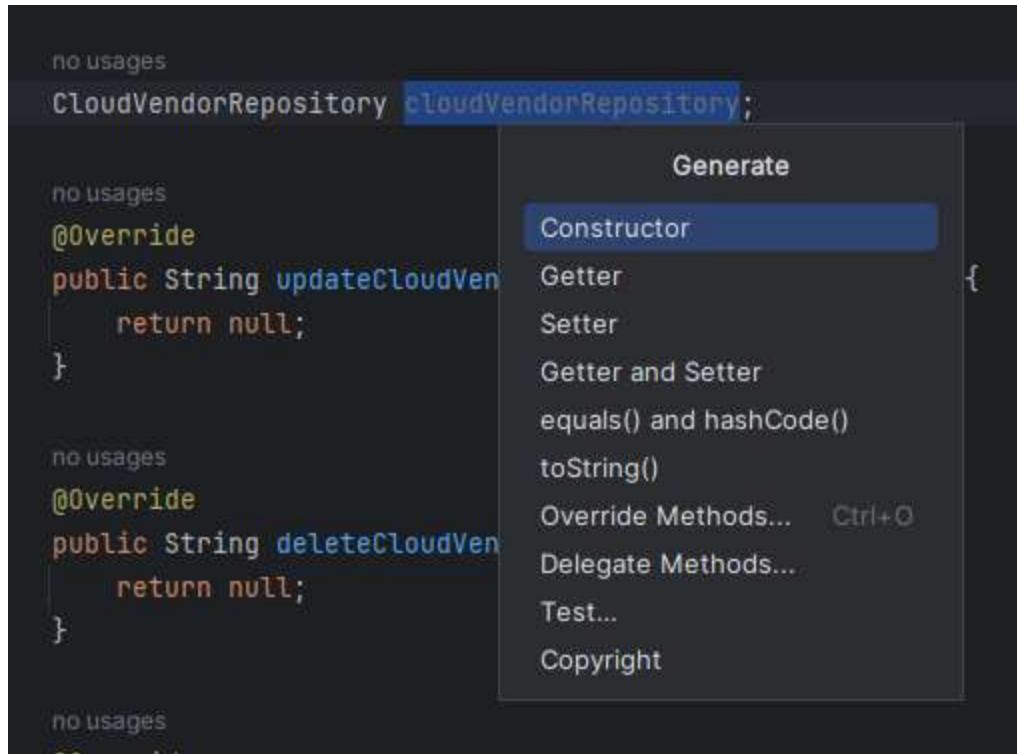
package com.thinkconstructive.restdemo.controller.service;

import com.thinkconstructive.restdemo.controller.CloudVendor;
import java.util.List;

```

```
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();
        }
    }
}

```

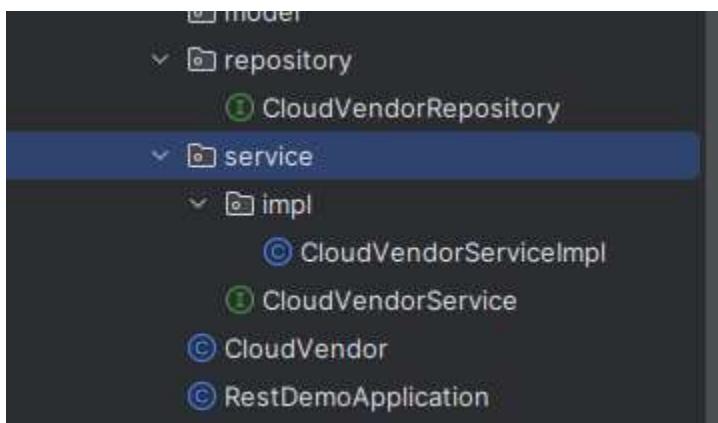
```

CloudAPIService.java   CloudVendorRepository.java   CloudVendorService.java   CloudVendorServiceImpl.java
1 package com.thinkconstructive.restdemo.controller.service.impl;
2
3 import com.thinkconstructive.restdemo.controller.CloudVendor;
4 import com.thinkconstructive.restdemo.controller.repository.CloudVendorRepository;
5 import com.thinkconstructive.restdemo.controller.service.CloudVendorService;
6 import org.springframework.stereotype.Service;
7
8 import java.util.List;
9
10 no usages
11 @Service
12
13 public class CloudVendorServiceImpl implements CloudVendorService {
14
15     6 usages
16     CloudVendorRepository cloudVendorRepository;
17     no usages
18     public CloudVendorServiceImpl(CloudVendorRepository cloudVendorRepository) {
19         this.cloudVendorRepository = cloudVendorRepository;
20     }
21
22     no usages
23     @Override
24     public String createCloudVendor(CloudVendor cloudVendor) {
25         cloudVendorRepository.save(cloudVendor);
26         return "Success";
27     }
28 }

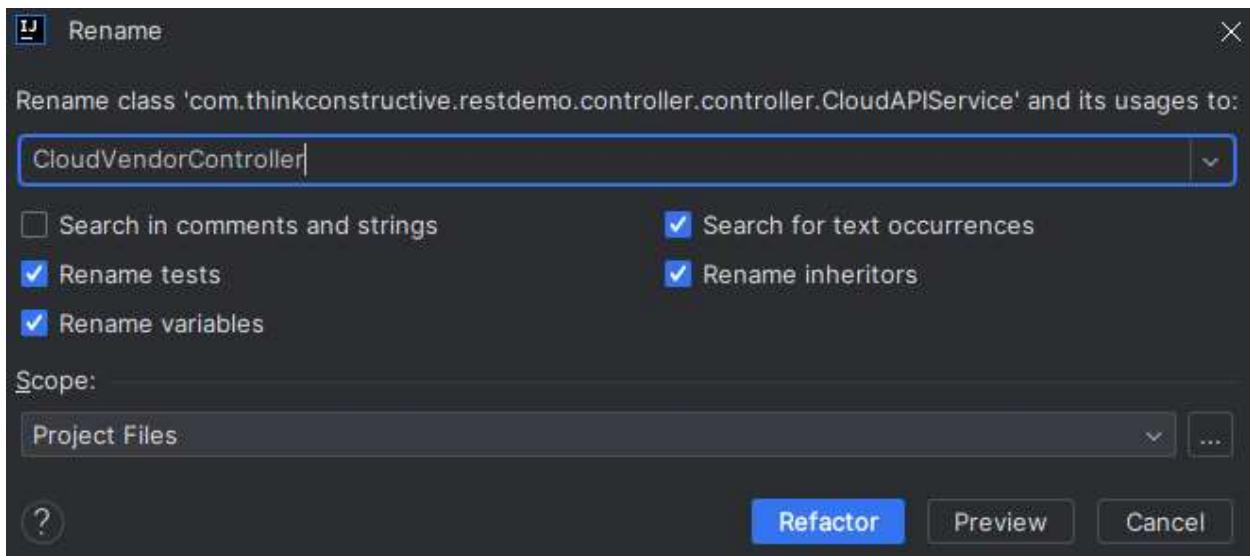
```

Activate Windows

- 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;
    }
}

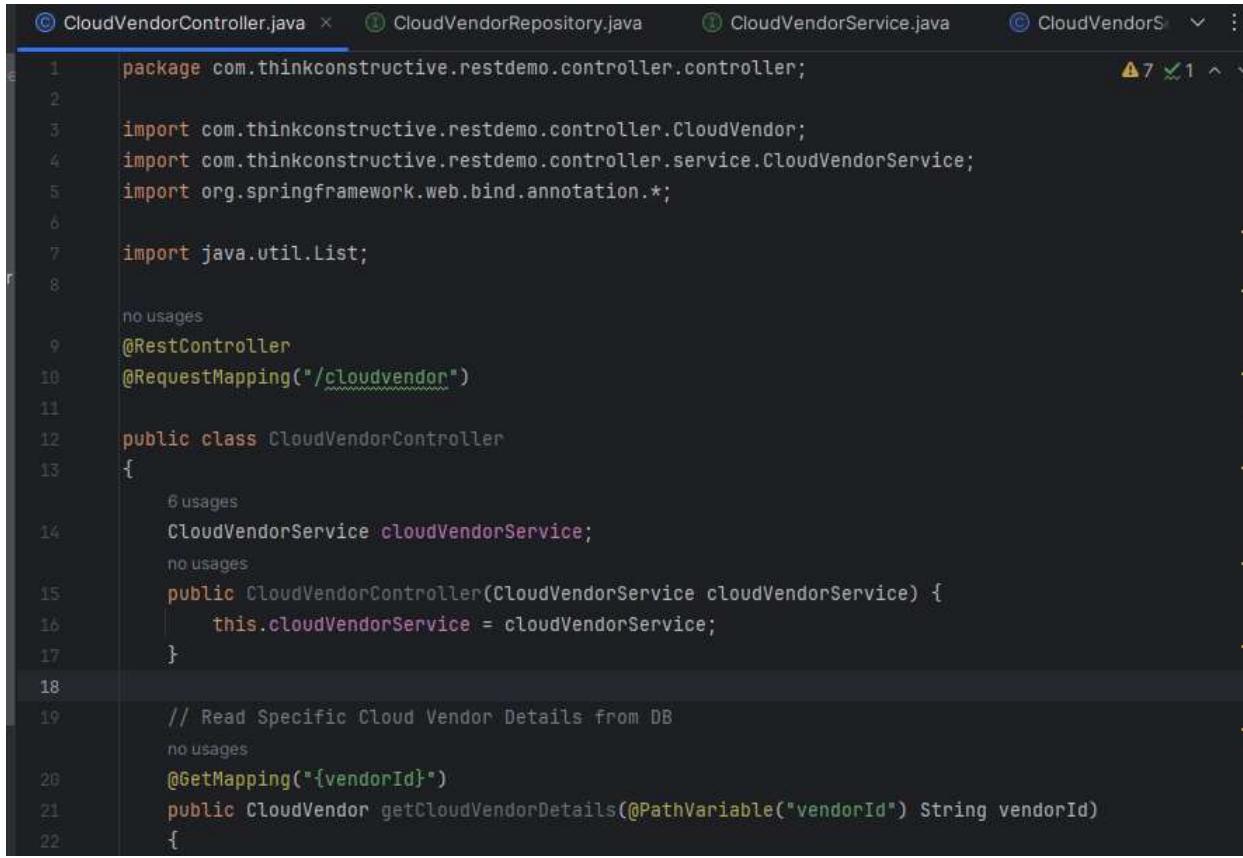
```

```
// 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();

}

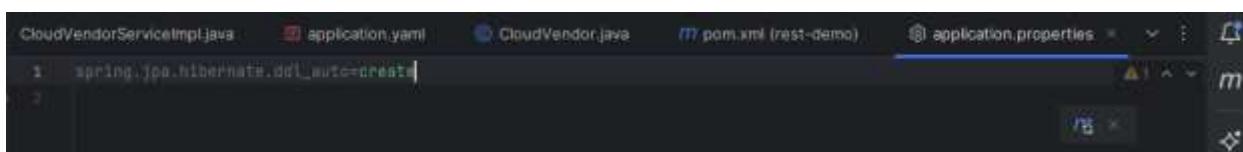
@PostMapping
public String createCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
{
    cloudVendorService.createCloudVendor(cloudVendor);
    return "Cloud Vendor Created Successfully";
}
@PutMapping
public String updateCloudVendorDetails(@RequestBody CloudVendor
cloudVendor)
{
    cloudVendorService.updateCloudVendor(cloudVendor);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



```

1 package com.thinkconstructive.restdemo.controller;
2
3 import com.thinkconstructive.restdemo.controller.CloudVendor;
4 import com.thinkconstructive.restdemo.controller.service.CloudVendorService;
5 import org.springframework.web.bind.annotation.*;
6
7 import java.util.List;
8
9 no usages
10 @RestController
11 @RequestMapping("/cloudvendor")
12
13 public class CloudVendorController {
14
15     6 usages
16     CloudVendorService cloudVendorService;
17     no usages
18     public CloudVendorController(CloudVendorService cloudVendorService) {
19         this.cloudVendorService = cloudVendorService;
20     }
21
22     // Read Specific Cloud Vendor Details from DB
23     no usages
24     @GetMapping("{vendorId}")
25     public CloudVendor getCloudVendorDetails(@PathVariable("vendorId") String vendorId)
26     {
27

```

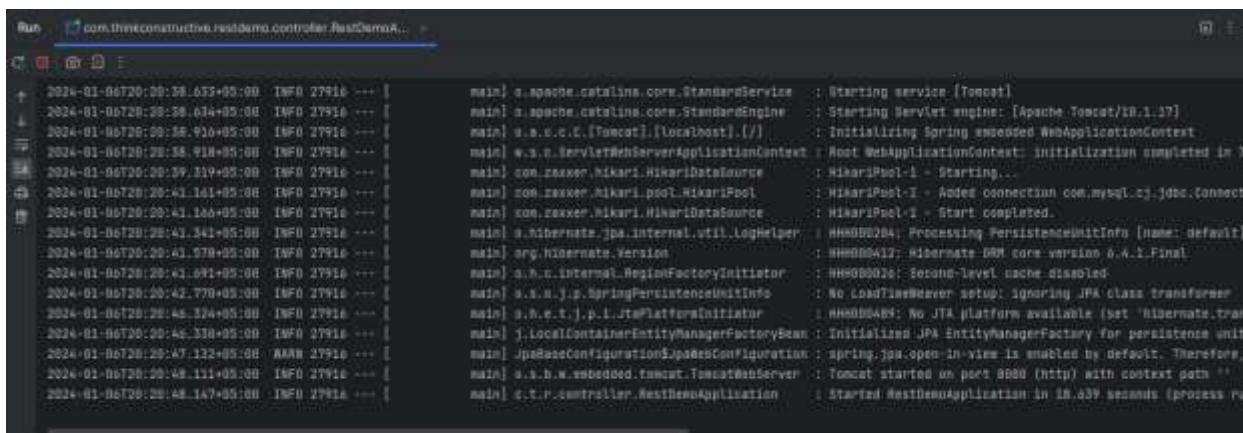


```

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.

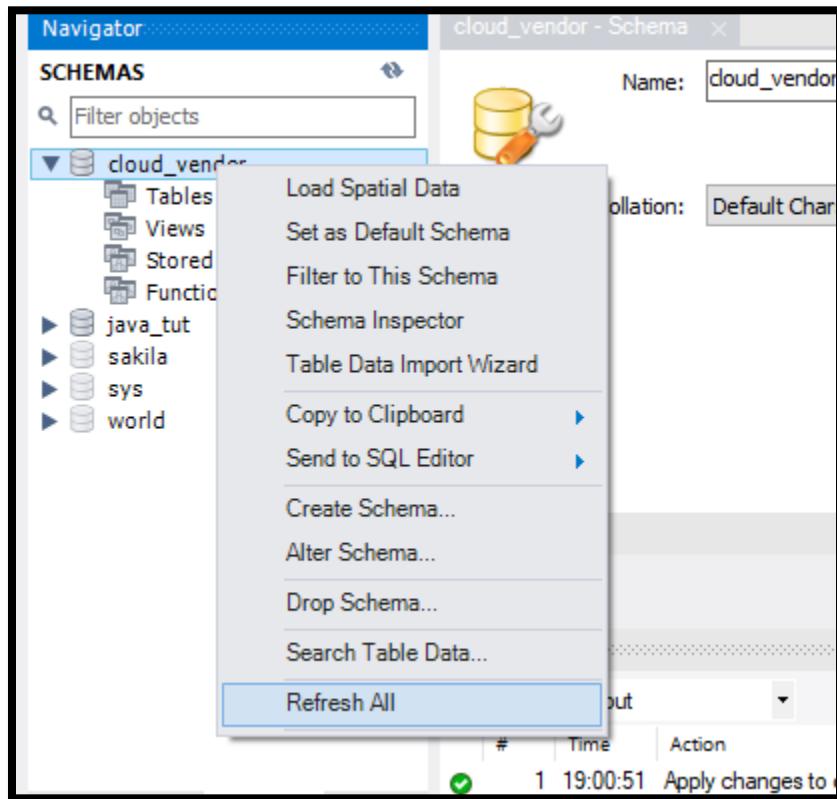


```

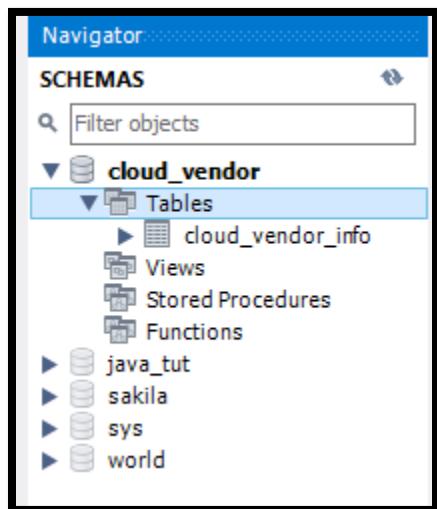
2024-01-06T20:20:38.653+05:00 INFO 27916 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-01-06T20:20:38.654+05:00 INFO 27916 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.27]
2024-01-06T20:20:38.926+05:00 INFO 27916 --- [main] o.a.c.c.t.Tomcat : [localhost] : / : Initializing Spring embedded WebApplicationContext
2024-01-06T20:20:38.918+05:00 INFO 27916 --- [main] w.s.n.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 73 ms
2024-01-06T20:20:39.219+05:00 INFO 27916 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-01-06T20:20:41.161+05:00 INFO 27916 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@34a5f3c
2024-01-06T20:20:41.166+05:00 INFO 27916 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-01-06T20:20:43.341+05:00 INFO 27916 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-01-06T20:20:41.578+05:00 INFO 27916 --- [main] org.hibernate.Version : HHH000412: Hibernate Core version 6.4.1.Final
2024-01-06T20:20:41.691+05:00 INFO 27916 --- [main] o.h.e.internals.RegimeFactoryInitiator : HHH000026: Second-level cache disabled
2024-01-06T20:20:42.770+05:00 INFO 27916 --- [main] o.s.m.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2024-01-06T20:20:46.324+05:00 INFO 27916 --- [main] o.s.e.t.j.p.l.JtaPlatformInitiator : HHH000449: No JTA platform available (set hibernate.transaction.jta.platform)
2024-01-06T20:20:46.338+05:00 INFO 27916 --- [main] j.localContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit
2024-01-06T20:20:47.124+05:00 [WARN] 27916 --- [main] JpaBaseConfiguration$paeserConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries will be executed at every single API endpoint.
2024-01-06T20:20:48.111+05:00 INFO 27916 --- [main] o.s.n.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
2024-01-06T20:20:48.147+05:00 INFO 27916 --- [main] o.s.r.RestController.RestTemplateOptimizat

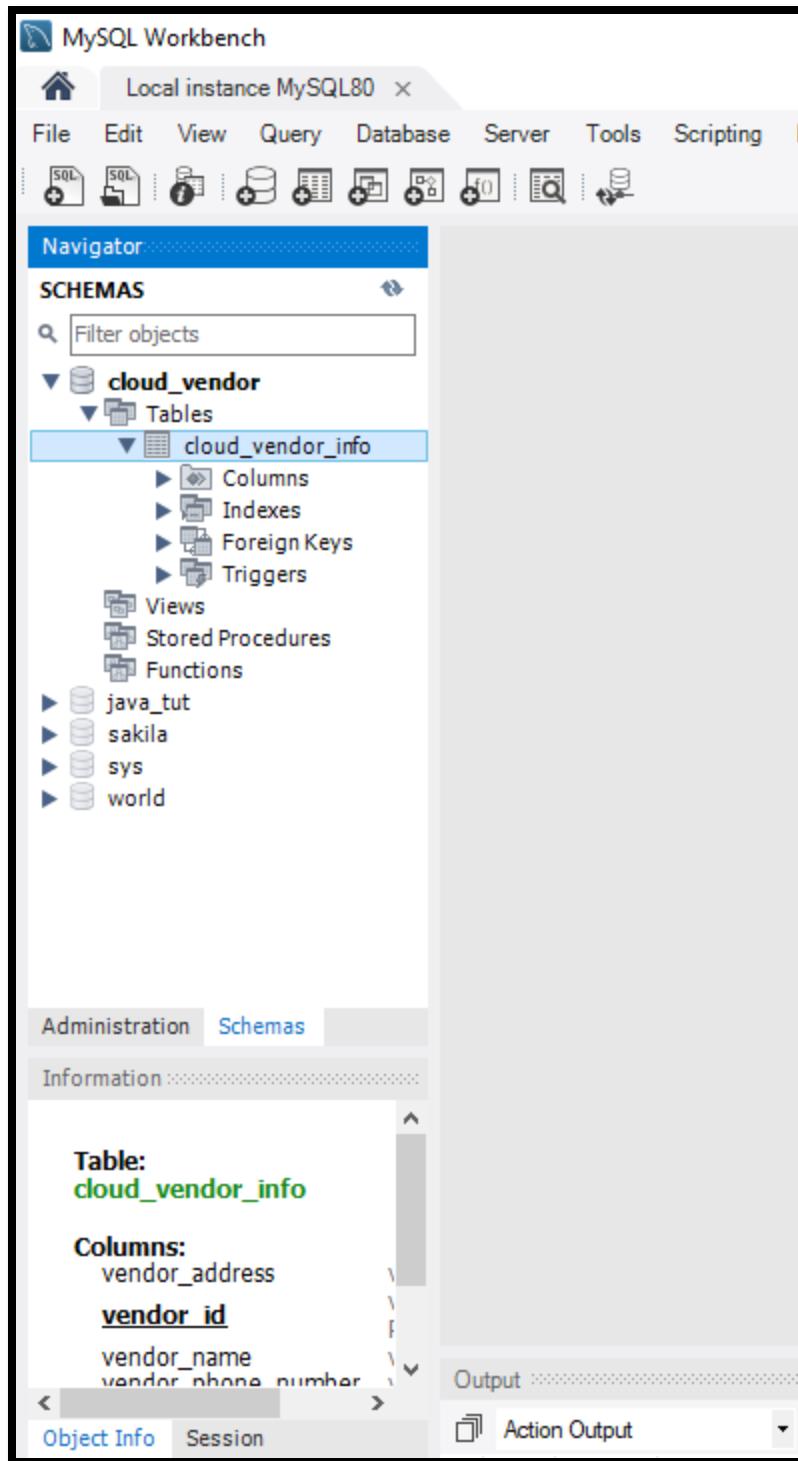
```

- Refresh and check whether any table is created.



- Successfully table is created in the database.





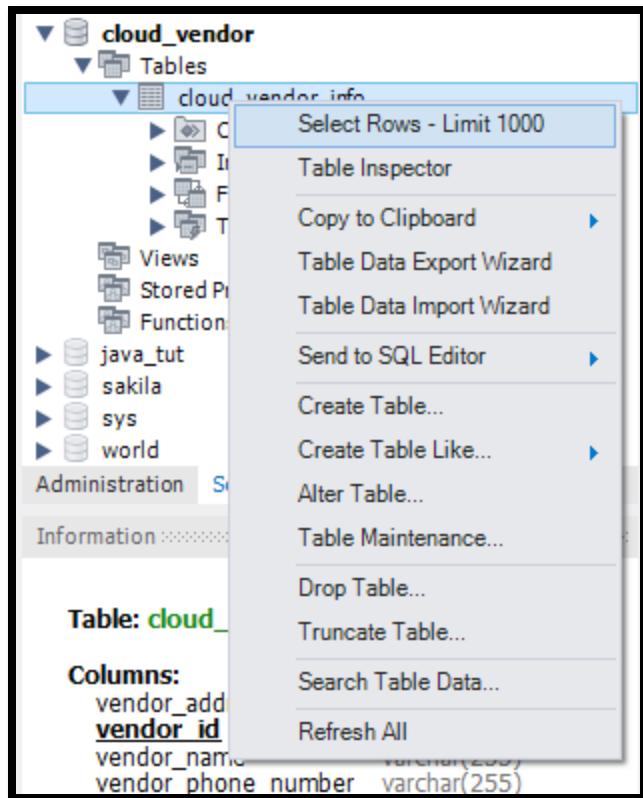
- All four fields are visible here.

The screenshot shows the MySQL Workbench interface. The left pane displays the 'SCHEMAS' tree. Under the 'cloud_vendor' schema, the 'Tables' node is expanded, showing the 'cloud_vendor_info' table. This table has four columns: 'vendor_address' (varchar(255)), '**vendor_id**' (varchar(255) PK), 'vendor_name' (varchar(255)), and 'vendor_phone_number' (varchar(255)). The right pane contains detailed information about the 'cloud_vendor_info' table, including its columns, primary key, and other metadata.

Table: cloud_vendor_info

Columns:

Column Name	Type
vendor_address	varchar(255)
vendor_id	varchar(255) PK
vendor_name	varchar(255)
vendor_phone_number	varchar(255)



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

The screenshot shows the MySQL Workbench interface with the results grid open for the 'cloud_vendor_info' table. The query executed is:

```
1 *   SELECT * FROM cloud_vendor.cloud_vendor_info;
```

The results grid displays the following data:

vendor_address	vendor_id	vendor_name	vendor_phone_number

- All four fields are here without any values.

A screenshot of a database result grid. The grid has four columns labeled "vendor_address", "vendor_id", "vendor_name", and "vendor_phone_number". There are four rows of data, each containing a single value: "NULL". The interface includes a toolbar at the top with various icons for filtering, editing, and exporting. On the right side, there is a sidebar with icons for "Result Grid", "Form Editor", and "Field Types". The title bar of the window says "Result Grid".

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

A screenshot of the Postman application. It shows a collection named "CloudVendorAPI-Collection" with a "GET Create" endpoint selected. A new POST request is being prepared to the URL "http://localhost:8080/cloudvendor". The "Body" tab is active, showing a JSON payload:

```

1   {
2     "VendorId": "C1",
3     "vendorName": "Vendor One",
4     "vendorAddress": "Address One",
5     "vendorPhoneNumber": "One"
6   }

```

- Create cloud vendor request is successfully sent.

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/cloudvendor`. The request body is a JSON object:

```

1 {
2   "vendorId": "C1",
3   "vendorName": "Vendor One",
4   "vendorAddress": "Address One",
5   "vendorPhoneNumber": "one"
6 }

```

The response status is 200 OK with a message: "Cloud Vendor Created Successfully".

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

The screenshot shows the MySQL Workbench interface with a query window containing the following SQL statement:

```
1 • SELECT * FROM cloud_vendor.cloud_vendor_info;
```

The result grid displays the following data:

	vendor_address	vendor_id	vendor_name	vendor_phone_number
▶	Address One	C1	Vendor One	one
*	NULL	NULL	NULL	NULL

	vendor_address	vendor_id	vendor_name	vendor_phone_number
▶	Address One	C1	Vendor One	one
*	NULL	NULL	NULL	NULL

- Check what should we get.

The screenshot shows the Postman interface with a collection named "CloudVendorAPI-Collection". A GET request is selected with the URL "http://localhost:8080/cloudvendor". The "Params" tab is active, showing a table for "Query Params" which is currently empty. Other tabs include Authorization, Headers (7), Body, Pre-request Script, Tests, Settings, and Cookies.

- We got all the four fields.

The screenshot shows the Postman interface with the same GET request. The "Body" tab is active, displaying the response in JSON format:

```

1 [ {"vendorId": "C1", "vendorName": "Vendor One", "vendorAddress": "Address One", "vendorPhoneNumber": "one"} ]

```

The response status is 200 OK with a 542 ms duration and 265 B size. Buttons for Pretty, Raw, Preview, Visualize, and JSON are visible at the bottom of the response pane.

- Check Get By Id for C1.

HTTP CloudVendorAPI-Collection / GetById

GET http://localhost:8080/cloudvendor/C1

Send

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

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Response

HTTP CloudVendorAPI-Collection / GetById

GET http://localhost:8080/cloudvendor/C1

Save

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 134 ms 263 B Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "vendorId": "C1",
3   "vendorName": "Vendor One",
4   "vendorAddress": "Address One",
5   "vendorPhoneNumber": "one"
6 }
```

Activate Windows
Go to Settings to activate Windows

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

POST http://localhost:8080/cloudvendor

Body (JSON)

```
{
  "vendorId": "C2",
  "vendorName": "Vendor Two",
  "vendorAddress": "Address Two",
  "vendorPhoneNumber": "Two"
}
```

200 OK 2.01 s 197 B Save as example

Cloud Vendor Created Successfully

- Check in database.

cloud_vendor_info

cloud_vendor

vendor_address	vendor_id	vendor_name	vendor_phone_number
Address One	C1	Vendor One	one
Address Two	C2	Vendor Two	Two

- Now Check some get by id for C2.

HTTP CloudVendorAPI-Collection / GetById

GET http://localhost:8080/cloudvendor/C2

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

Pretty Raw Preview Visualize JSON

```

1 {
2   "vendorId": "C2",
3   "vendorName": "Vendor Two",
4   "vendorAddress": "Address Two",
5   "vendorPhoneNumber": "Two"
6 }
```

Activate Windows

- In get all, I should get two records.

HTTP CloudVendorAPI-Collection / GetAll

GET http://localhost:8080/cloudvendor

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

Body Cookies Headers (5) Test Results

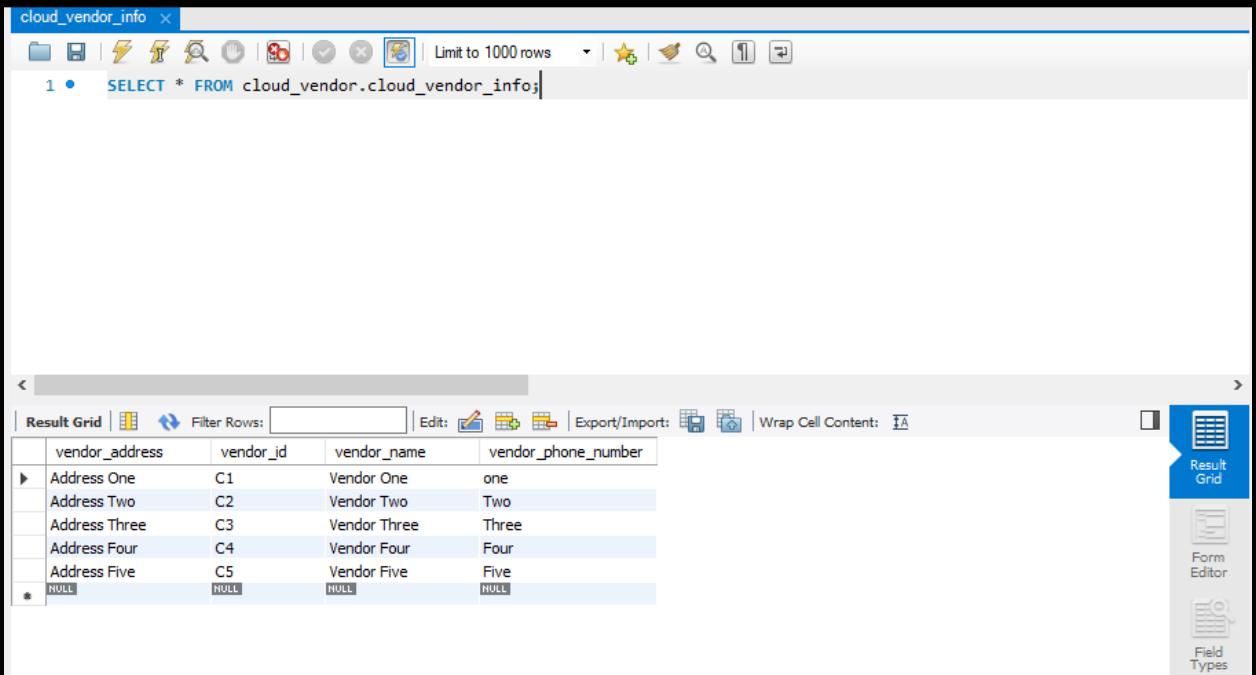
Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "vendorId": "C1",
4     "vendorName": "Vendor One",
5     "vendorAddress": "Address One",
6     "vendorPhoneNumber": "one"
7   },
8   {
9     "vendorId": "C2",
10    "vendorName": "Vendor Two",
11    "vendorAddress": "Address Two",
12    "vendorPhoneNumber": "Two"
13  }
14 ]
```

Activate Windows
Go to Settings to activate Windows

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



The screenshot shows the MySQL Workbench interface with a query editor window titled "cloud_vendor_info". The query is:

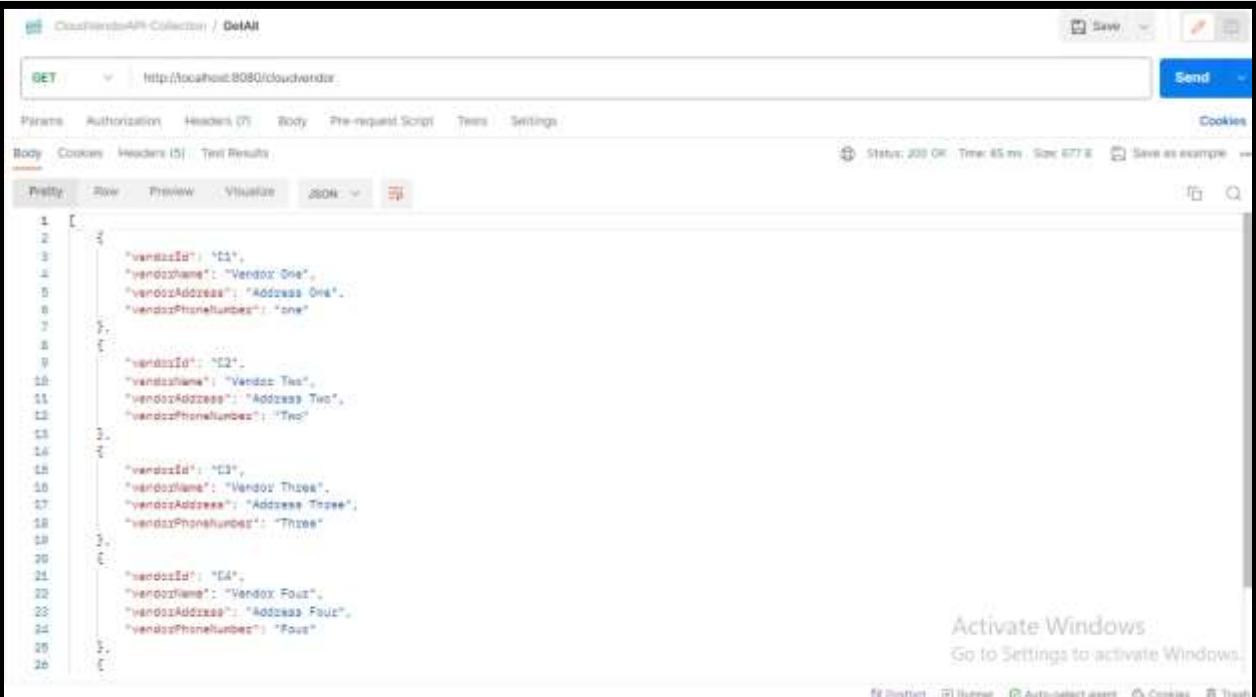
```
1 •  SELECT * FROM cloud_vendor.cloud_vendor_info;
```

The results are displayed in a "Result Grid" tab, showing the following data:

	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
	Address Four	C4	Vendor Four	Four
	Address Five	C5	Vendor Five	Five
*	NULL	NULL	NULL	NULL

On the right side of the interface, there is a vertical toolbar with three buttons: "Result Grid" (selected), "Form Editor", and "Field Types".

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



The screenshot shows a Postman collection named "CloudVendorAPI-Collection" with a "Detail" view. A GET request is made to `http://localhost:8080/cloudvendors`. The response body is displayed in "Pretty" format as JSON:

```

1 [
2   {
3     "vendorId": "C1",
4     "vendorName": "Vendor One",
5     "vendorAddress": "Address One",
6     "vendorPhoneNumber": "One"
7   },
8   {
9     "vendorId": "C2",
10    "vendorName": "Vendor Two",
11    "vendorAddress": "Address Two",
12    "vendorPhoneNumber": "Two"
13  },
14  {
15    "vendorId": "C3",
16    "vendorName": "Vendor Three",
17    "vendorAddress": "Address Three",
18    "vendorPhoneNumber": "Three"
19  },
20  {
21    "vendorId": "C4",
22    "vendorName": "Vendor Four",
23    "vendorAddress": "Address Four",
24    "vendorPhoneNumber": "Four"
25  }
26]
```

The status bar at the bottom indicates: Status: 200 OK Time: 45 ms Size: 677 B. There is also an "Activate Windows" watermark.

```

27     "vendorId": "C5",
28     "vendorName": "Vendor Five",
29     "vendorAddress": "Address Five",
30     "vendorPhoneNumber": "Five"
31   }
32 ]

```

- Now, when I'll check for get by id for C4, I will get the following.

HTTP CloudVendorAPI-Collection / GetById

GET http://localhost:8080/cloudvendor/C4

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

Pretty Raw Preview Visualize JSON

```

1 {
2   "vendorId": "C4",
3   "vendorName": "Vendor Four",
4   "vendorAddress": "Address Four",
5   "vendorPhoneNumber": "Four"
6 }

```

Status: 200 OK Time: 211 ms Size: 266 B Save as example ...

Activate Windows

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

HTTP CloudVendorAPI-Collection / Update

PUT http://localhost:8080/cloudvendor

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```

2 ...
3 ...
4 ...
5 ...
6 ...

```

Response

The screenshot shows a POST request in Postman to `http://localhost:8080/cloudvendor`. The request body is a JSON object:

```
2   ... "vendorId": "C4",
3   ... "vendorName": "Vendor Four",
4   ... "vendorAddress": "London",
5   ... "vendorPhoneNumber": "Four"
6 }
```

The response status is 200 OK, and the response body is:

```
1 Cloud Vendor Updated Successfully
```

- 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
	HULL	NUL	NUL	NUL

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

GET <http://localhost:8080/cloudvendor/C4>

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

Query Params

	Key	Value	Description
	Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK Time: 1142 ms

Pretty Raw Preview Visualize JSON `JSON`

```

1 {
2   "vendorId": "C4",
3   "vendorName": "Vendor Four",
4   "vendorAddress": "London",
5   "vendorPhoneNumber": "Four"
6 }
```

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

HTTP CloudVendorAPI-Collection / **DeleteById** Save [Edit](#) [Copy](#)

DELETE <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	

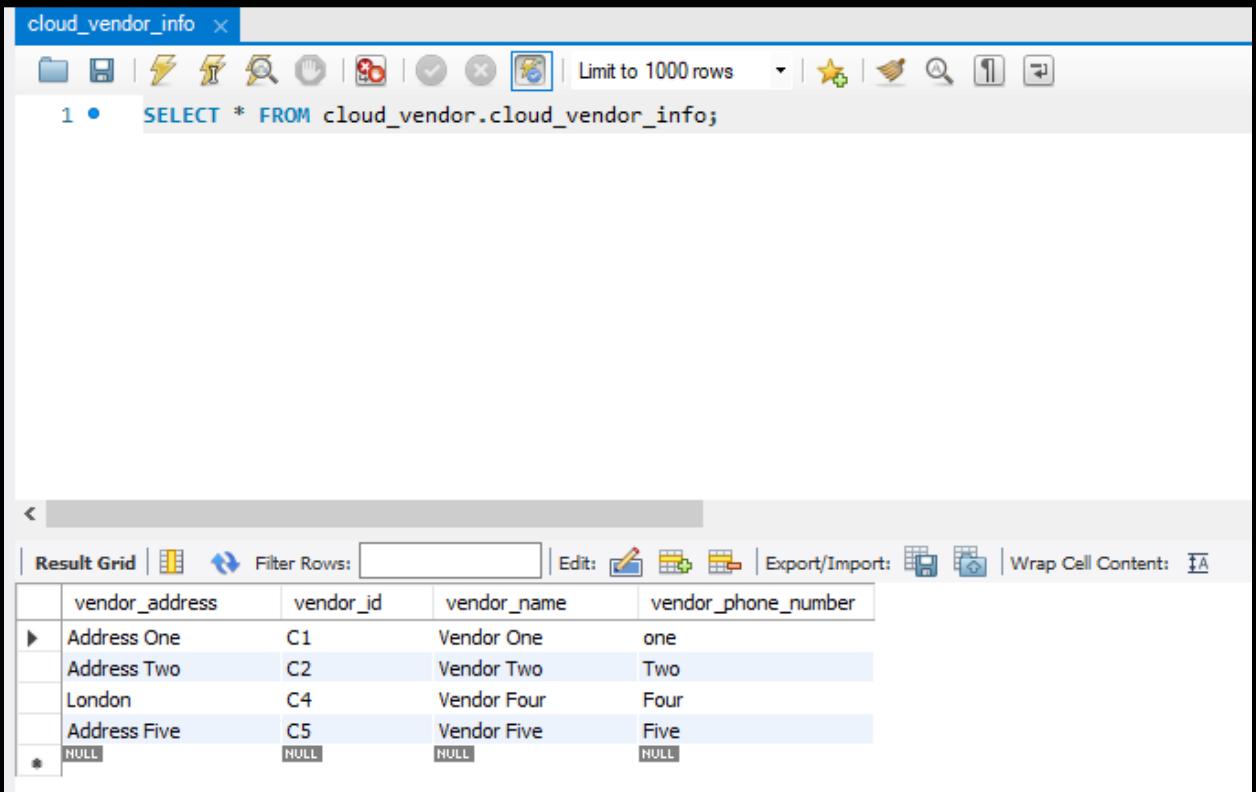
Body Cookies Headers (5) Test Results Status: 200 OK Time: 7.45 s Size: 197 B Save as example [...](#)

Pretty Raw Preview Visualize Text `Text`

```

1 Cloud Vendor Deleted Successfully
```

- Go back to database and see what happened.



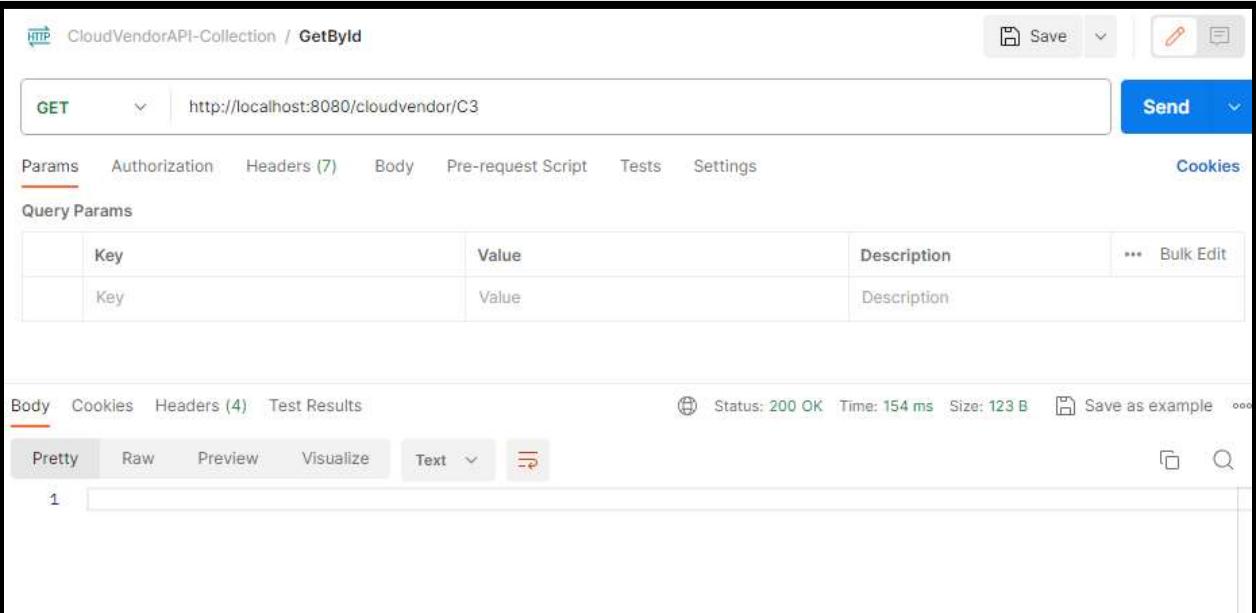
The screenshot shows a MySQL Workbench interface with a query editor window titled "cloud_vendor_info". The query entered is:

```
1 •  SELECT * FROM cloud_vendor.cloud_vendor_info;
```

The result grid displays the following data:

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

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



The screenshot shows a Postman collection named "CloudVendorAPI-Collection" with a selected endpoint "GetById". The request method is "GET" and the URL is "http://localhost:8080/cloudvendor/C3".

The "Params" tab is selected, showing a table for "Query Params" with one entry:

Key	Value	Description	Bulk Edit
Key	Value	Description	

The "Body" tab is selected, showing the response body:

```
1
```

The status bar at the bottom indicates: Status: 200 OK Time: 154 ms Size: 123 B Save as example

- In get All there would be no C3.

HTTP CloudVendorAPI-Collection / GetAll

GET http://localhost:8080/cloudvendor

Params Authorization Headers (7) Body Pre-request Script

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ↻

```

1 [
2   {
3     "vendorId": "C1",
4     "vendorName": "Vendor One",
5     "vendorAddress": "Address One",
6     "vendorPhoneNumber": "one"
7   },
8   {
9     "vendorId": "C2",
10    "vendorName": "Vendor Two",
11    "vendorAddress": "Address Two",
12    "vendorPhoneNumber": "Two"
13  },
14  {
15    "vendorId": "C4",
16    "vendorName": "Vendor Four",
17    "vendorAddress": "London",

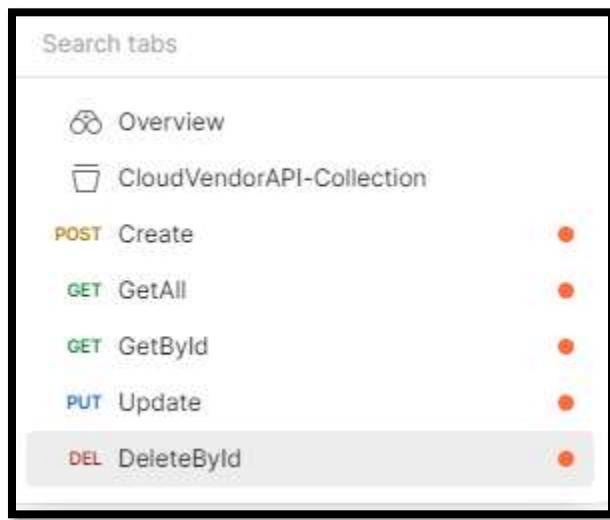
```

```

18   "vendorPhoneNumber": "Four"
19 },
20 {
21   "vendorId": "C5",
22   "vendorName": "Vendor Five",
23   "vendorAddress": "Address Five",
24   "vendorPhoneNumber": "Five"
25 }
26 ]

```

- So, we have tested all the CRUD Operations.



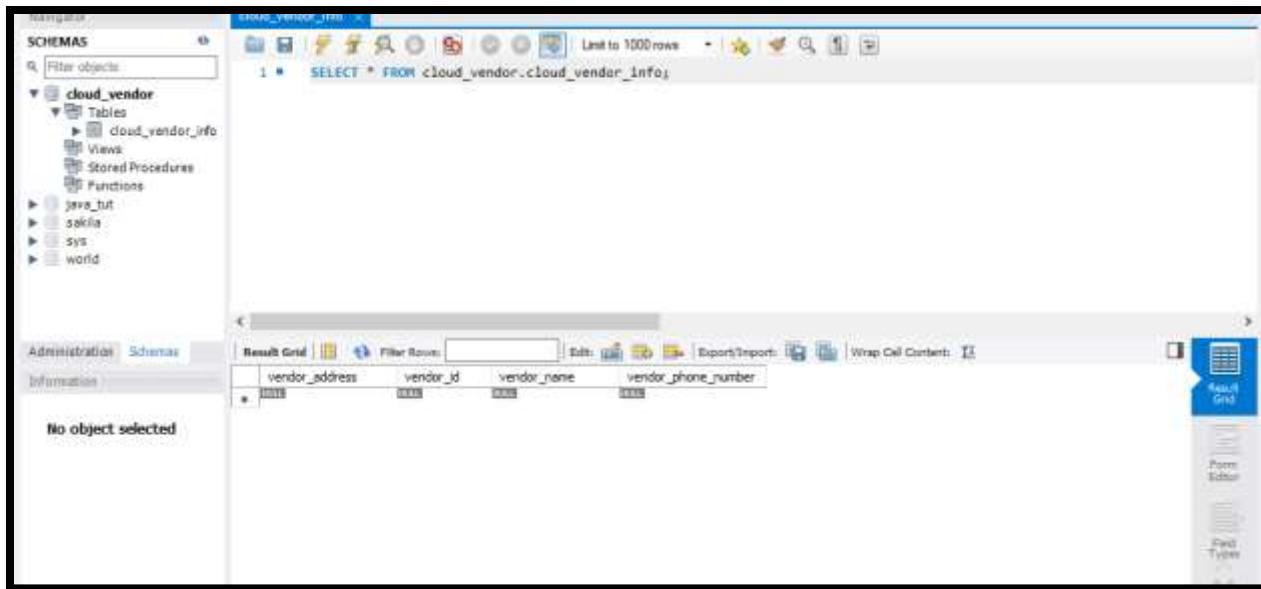
Session# 03

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.

```
2024-01-17T17:00:12.804+05:00 INFO 9392 --- [main] c.t.r.controller.RestDemoApplication : No active profile set, falling back to 1 default profile
2024-01-17T17:00:16.404+05:00 INFO 9392 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-01-17T17:00:16.724+05:00 INFO 9392 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 305 ms. Found 0 repository interfaces.
2024-01-17T17:00:19.054+05:00 INFO 9392 --- [main] o.s.d.b.o.embedded.TomcatWebServer : Tomcat initialized with port 8080 (http)
2024-01-17T17:00:19.100+05:00 INFO 9392 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-01-17T17:00:19.100+05:00 INFO 9392 --- [main] o.apache.catalina.core.StandardEngine : Starting servlet engine: [Apache Tomcat/10.1.17]
2024-01-17T17:00:19.153+05:00 INFO 9392 --- [main] o.a.c.c.TomcatWebappContext : Root WebApplicationContext: initialization completed in 0 ms
2024-01-17T17:00:19.153+05:00 INFO 9392 --- [main] o.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 0 ms
2024-01-17T17:00:19.177+05:00 INFO 9392 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-01-17T17:00:21.393+05:00 INFO 9392 --- [main] o.h.e.int_.RegionFactoryInitiator : HHH000412: Hibernate ORM core version 6.4.1.Final
2024-01-17T17:00:22.179+05:00 INFO 9392 --- [main] o.m.e.j.p.jp.PersistenceInfo : HHH000020: Second-level cache disabled
2024-01-17T17:00:22.265+05:00 INFO 9392 --- [main] com.zaxxer.hikari.HikariDataSource : No LruCacheReaper setup; ignoring JPA class transformer
2024-01-17T17:00:23.013+05:00 INFO 9392 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Starting...
2024-01-17T17:00:23.027+05:00 INFO 9392 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Added connection com.mysql.cj.jdbc.Connection
2024-01-17T17:00:26.392+05:00 INFO 9392 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to 'true')
2024-01-17T17:00:27.145+05:00 INFO 9392 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit
2024-01-17T17:00:28.287+05:00 WARN 9392 --- [main] .jaBaseConfigurationTypeWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, non thread-safe JDBC operations should not be performed during rendering
2024-01-17T17:00:29.137+05:00 INFO 9392 --- [main] o.s.d.b.o.embedded.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
2024-01-17T17:00:29.140+05:00 INFO 9392 --- [main] c.t.r.controller.RestDemoApplication : Started RestDemoApplication in 17.782 seconds (process info)
  Started RestDemoApplication in 17.782 seconds (process info)
  Activate Windows
```

- 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?

The screenshot shows a Java code editor with three tabs: RestDemoApplication.java, CloudVendorNotFoundException.java (selected), and CloudVendorController.java. The code in CloudVendorNotFoundException.java is:

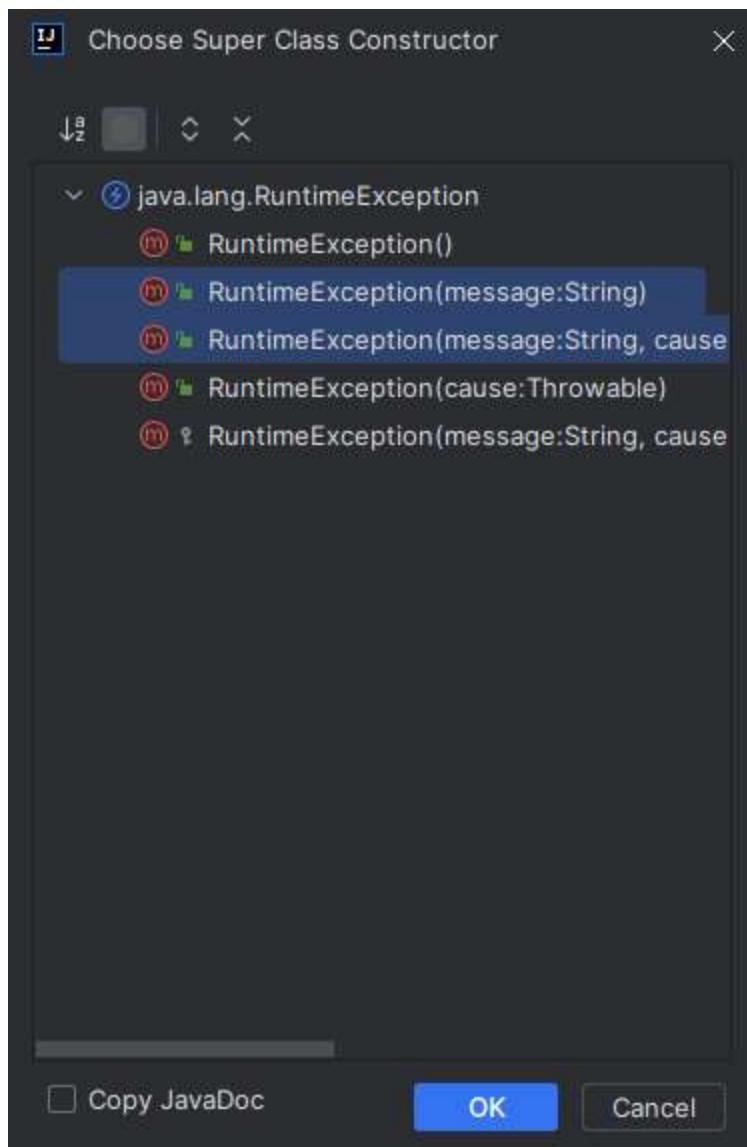
```
package com.thinkconstructive.restdemo.controller.exception;

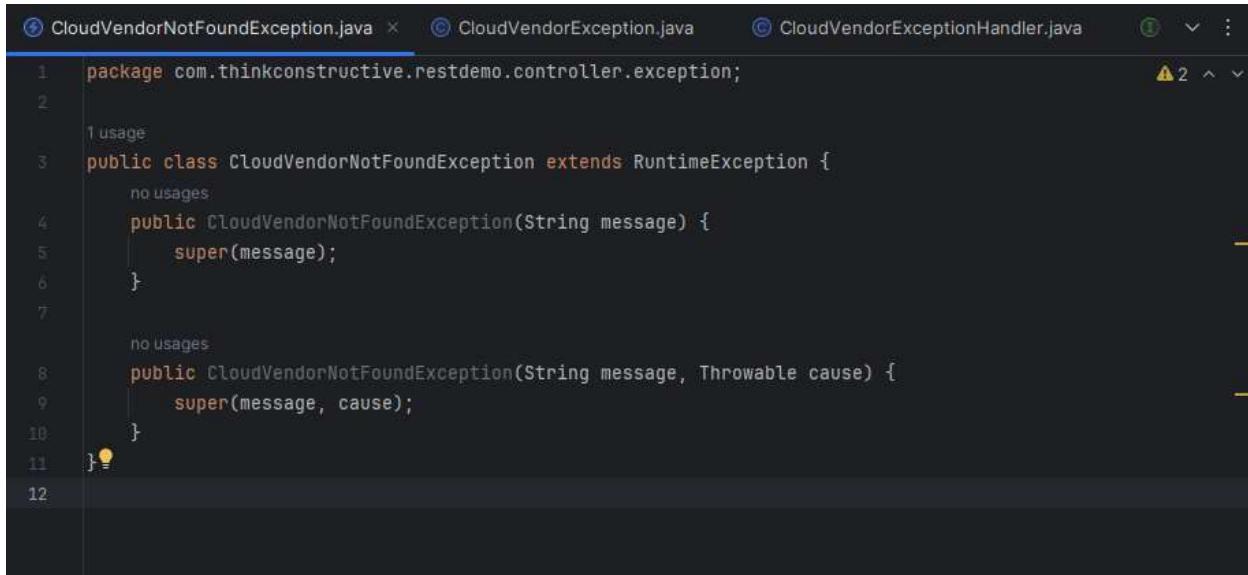
no usages

public class CloudVendorNotFoundException extends RuntimeException {
```

A context menu is open over the closing brace of the class definition, with 'Generate' selected. A submenu is displayed:

- Constructor
- toString()
- Override Methods... Ctrl+O
- Delegate Methods...
- Test...
- Copyright





```
CloudVendorNotFoundException.java x CloudVendorException.java CloudVendorExceptionHandler.java
1 package com.thinkconstructive.restdemo.controller.exception;
2
3 1 usage
4 public class CloudVendorNotFoundException extends RuntimeException {
5     no usages
6     public CloudVendorNotFoundException(String message) {
7         super(message);
8     }
9
10    no usages
11    public CloudVendorNotFoundException(String message, Throwable cause) {
12        super(message, cause);
13    }
14 }
```

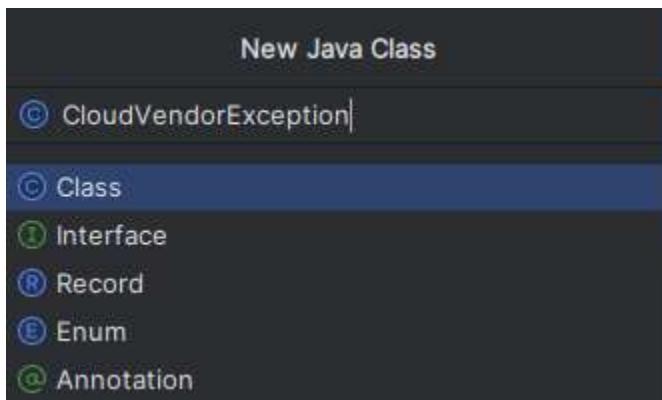
Code:

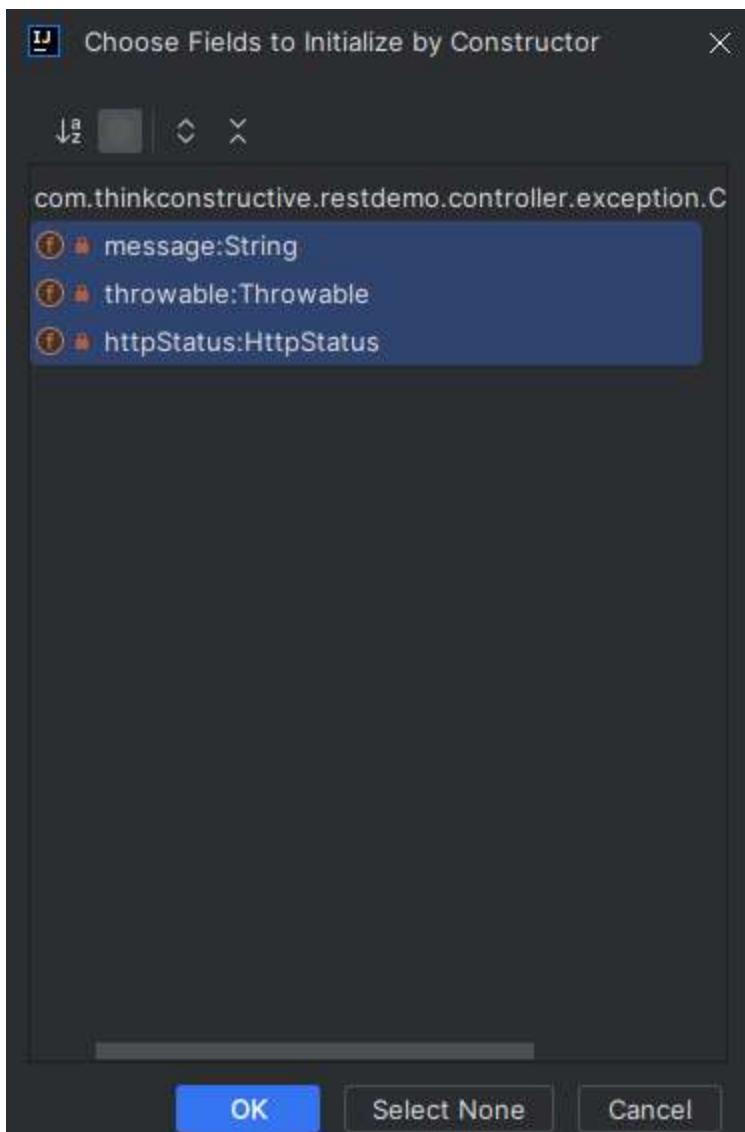
```
package com.thinkconstructive.restdemo.controller.exception;

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

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

- Create another class.





```
1 package com.thinkconstructive.restdemo.controller.exception;
2
3 import org.springframework.http.HttpStatus;
4
5 public class CloudVendorException {
6     private final String message;
7     private final Throwable throwable;
8     private final HttpStatus httpStatus;
9
10    public CloudVendorException(String message, Throwable throwable, HttpStatus httpStatus) {
11        this.message = message;
12        this.throwable = throwable;
13        this.httpStatus = httpStatus;
14    }
15
16    public String getMessage() {
17        return message;
18    }
19
20    public Throwable getThrowable() {
21        return throwable;
22    }
23
24    public HttpStatus getHttpStatus() {
25        return httpStatus;
26    }
27}
```

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;
    }

}

```



```

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

no usages
@ControllerAdvice
public class CloudVendorExceptionHandler {

    no usages
    @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);
    }
}

```

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

```

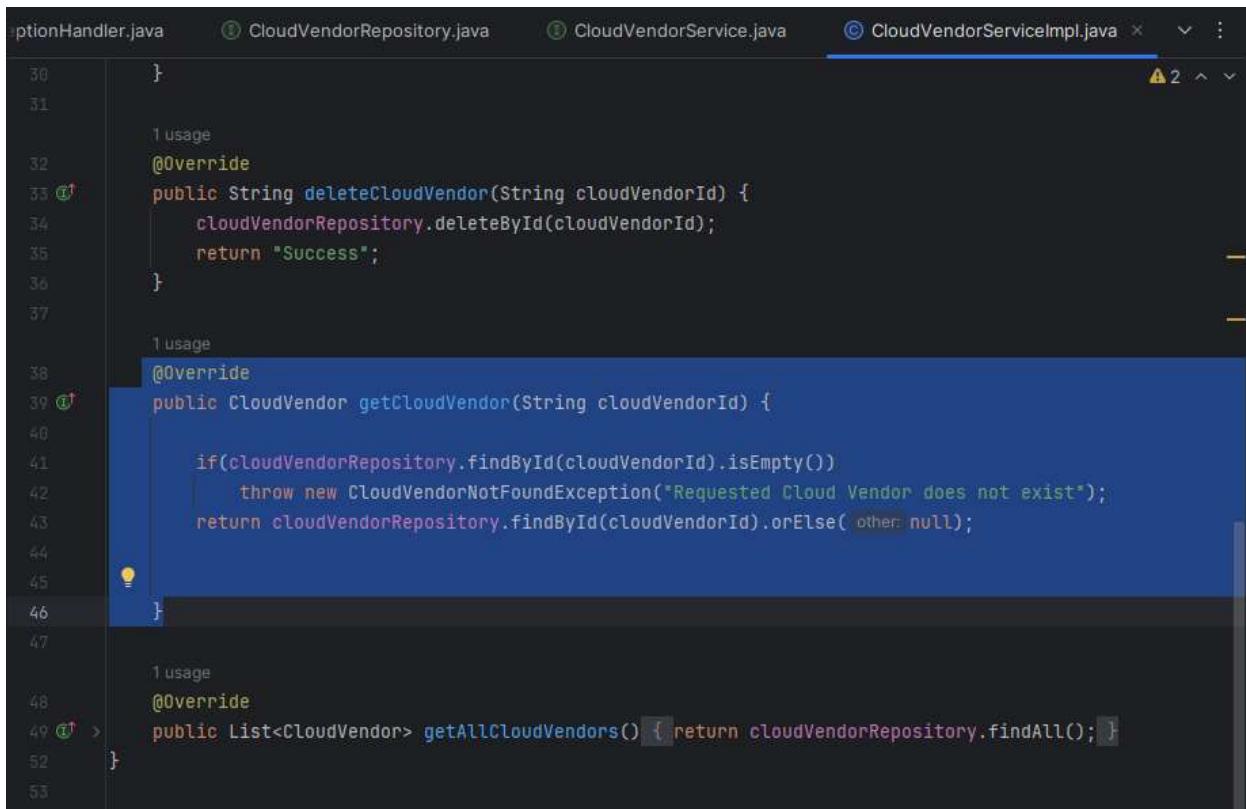
```

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.



```

CloudVendorHandler.java   CloudVendorRepository.java   CloudVendorService.java   CloudVendorServiceImpl.java
30
31
32     1 usage
33     @Override
34     public String deleteCloudVendor(String cloudVendorId) {
35         cloudVendorRepository.deleteById(cloudVendorId);
36         return "Success";
37     }
38
39     1 usage
40     @Override
41     public CloudVendor getCloudVendor(String cloudVendorId) {
42
43         if(cloudVendorRepository.findById(cloudVendorId).isEmpty())
44             throw new CloudVendorNotFoundException("Requested Cloud Vendor does not exist");
45         return cloudVendorRepository.findById(cloudVendorId).orElse(null);
46     }
47
48     1 usage
49     @Override
50     public List<CloudVendor> getAllCloudVendors() { return cloudVendorRepository.findAll(); }
51
52 }
53

```

- 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.

The screenshot shows an IDE interface with the project structure on the left and code editor on the right. The code in the editor is:

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

Below the code, there is a terminal window displaying application logs. The logs show the application starting up and connecting to a MySQL database using HikariCP. The logs end with the message "Started RestDemoApplication in 37.428 seconds (process time)".

```
2024-01-11T19:42:51.185+01:00 INFO 704 --- [main] o.s.e.c.c.C$Formatting>[] : Initializing Spring embedded WebApplicationContext
2024-01-11T19:42:51.191+01:00 INFO 704 --- [main] o.s.w.s.GenericWebServerApplicationContext : Root WebApplicationContext: initialization completed in 0 ms
2024-01-11T19:42:51.725+01:00 INFO 704 --- [main] o.h.b.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-01-11T19:42:52.922+01:00 INFO 704 --- [main] org.hibernate.Version : HHH000512: Hibernate ORM core version 6.4.1.Final
2024-01-11T19:42:52.927+01:00 INFO 704 --- [main] o.h.b.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2024-01-11T19:42:52.927+01:00 INFO 704 --- [main] o.s.o.o.j.p.SpringPersistenceCapableInfo : No LoadTimeWeaver setup; ignoring JPA class transformer
2024-01-11T19:42:53.024+01:00 INFO 704 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-01-11T19:42:55.456+01:00 INFO 704 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@500...
2024-01-11T19:42:55.461+01:00 INFO 704 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-01-11T19:42:56.578+01:00 INFO 704 --- [main] o.h.e.t.j.p.l.JtaPlatformInitiator : HHH000409: No JTA platform available (set 'hibernate.transaction.jta.platform' to 'true' if you want to use JBoss Seam's JtaPlatform)
2024-01-11T19:42:59.510+01:00 INFO 704 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-01-11T19:43:00.692+01:00 WARN 704 --- [main] JobCacheConfiguration$UpdatableConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be executed during a flush.
2024-01-11T19:43:02.104+01:00 INFO 704 --- [main] org.apache.catalina.core.StandardService : Tomcat started on port 8080 (http) with context path ''
2024-01-11T19:48:02.127+01:00 INFO 704 --- [main] o.s.t.c.RestDemoApplication : Started RestDemoApplication in 37.428 seconds (process time)
```

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

The screenshot shows a Postman collection named "CloudVendorAPI-Collection". A GET request is defined with the URL "http://localhost:8080/cloudvendor/C3". The "Body" tab is selected, showing the response body:

```
{"message": "Requested Cloud Vendor does not exist", "throwable": null, "httpStatus": "NOT_FOUND"}
```

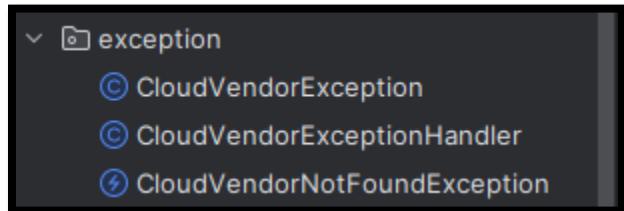
The response status is 404 Not Found, and the time taken was 55.07 s.

A screenshot of a REST API response in a browser. The top navigation bar includes tabs for Body, Cookies, Headers (5), and Test Results. Below the tabs, there are buttons for Pretty, Raw, Preview, Visualize, and JSON. The JSON button is selected. The main content area shows a 404 Not Found error with the following JSON response:

```
1 { "message": "Requested Cloud Vendor does not exist" }
```

A tooltip for the message field provides the following text: "The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible." To the right of the response, there is a sidebar with icons for copy, search, and refresh, and the text "NOT_FOUND" is visible.

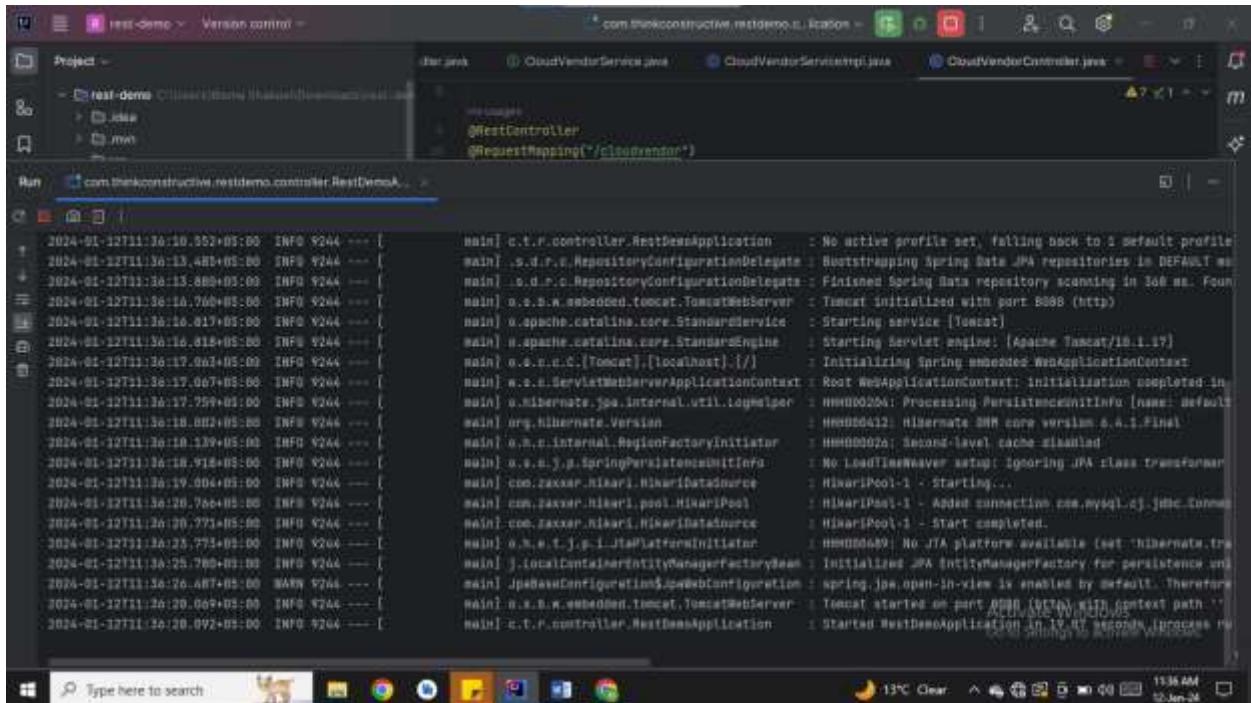
- This is how exception handling could be done with spring boot rest API Application.
- We have added the custom exception handling.



Session# 04

Java Spring Boot REST API JSON Response Handling

- Run the Application.



```
2024-01-12T11:36:10.552+05:00 INFO 9244 --- [main] c.t.r.controller.RestDemoApplication : No active profile set, falling back to 1 default profile
2024-01-12T11:36:13.481+05:00 INFO 9244 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode
2024-01-12T11:36:13.889+05:00 INFO 9244 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 3d6 ms. Found 0 repository interfaces
2024-01-12T11:36:14.700+05:00 INFO 9244 --- [main] o.s.d.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2024-01-12T11:36:16.817+05:00 INFO 9244 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-01-12T11:36:16.818+05:00 INFO 9244 --- [main] o.apache.catalina.core.StandardEngine : Starting servlet engine: [Apache Tomcat/10.1.17]
2024-01-12T11:36:17.064+05:00 INFO 9244 --- [main] o.a.n.e.c.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-01-12T11:36:17.067+05:00 INFO 9244 --- [main] w.e.e.ServletContextListenerApplicationContex : Root WebApplicationContext: initialization completed in 1 ms
2024-01-12T11:36:17.759+05:00 INFO 9244 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-01-12T11:36:18.002+05:00 INFO 9244 --- [main] org.hibernate.Version : HHH000421: Hibernate ORM version 5.6.1.Final
2024-01-12T11:36:18.139+05:00 INFO 9244 --- [main] o.s.o.i.internal.RegionFactoryInitiator : HHH000025: second-level cache enabled
2024-01-12T11:36:18.918+05:00 INFO 9244 --- [main] o.s.o.j.p.SpringPersistenceInitializer : No LoadTimeWeaver setup; ignoring JPA class transformer
2024-01-12T11:36:19.004+05:00 INFO 9244 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-01-12T11:36:20.764+05:00 INFO 9244 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.Connection
2024-01-12T11:36:20.771+05:00 INFO 9244 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-01-12T11:36:23.771+05:00 INFO 9244 --- [main] o.h.e.t.j.p.JtaPlatformInitiator : HHH000469: No JTA platform available (set hibernate.transaction.jta.platform)
2024-01-12T11:36:25.780+05:00 INFO 9244 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit
2024-01-12T11:36:26.487+05:00 WARN 9244 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during a controller's view rendering. See https://github.com/spring-projects/spring-boot/issues/2793 for more information.
2024-01-12T11:36:29.069+05:00 INFO 9244 --- [main] o.s.d.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
2024-01-12T11:36:29.092+05:00 INFO 9244 --- [main] c.t.r.controller.RestDemoApplication : Started RestDemoApplication in 10.115 seconds (process info)
```

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

HTTP CloudVendorAPI-Collection / Create

POST http://localhost:8080/cloudvendor Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body (JSON)

```
1 {
2     "vendorId": "C5",
3     "vendorName": "Vendor Five",
4     "vendorAddress": "Address Five",
5     "vendorPhoneNumber": "five"
6 }
```

Body Cookies Headers (5) Test Results

200 OK 1134 ms 197 B Save as example

Pretty Raw Preview Visualize Text

1 Cloud Vendor Created Successfully

Activate Windows Go to Settings to activate Windows

This screenshot shows a POST request to 'http://localhost:8080/cloudvendor' in Postman. The request body is a JSON object with fields: vendorId ('C5'), vendorName ('Vendor Five'), vendorAddress ('Address Five'), and vendorPhoneNumber ('five'). The response status is 200 OK, and the message is 'Cloud Vendor Created Successfully'.

- Query C5 Get by ID.

HTTP CloudVendorAPI-Collection / GetById

GET http://localhost:8080/cloudvendor/C5 Send

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

Body (Raw)

This request does not have a body

Body Cookies Headers (5) Test Results

200 OK 188 ms 266 B Save as example

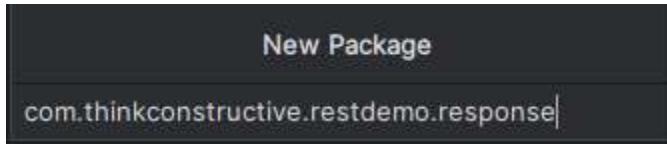
Pretty Raw Preview Visualize JSON

```
1 {
2     "vendorId": "C5",
3     "vendorName": "Vendor Five",
4     "vendorAddress": "Address Five",
5     "vendorPhoneNumber": "five"
6 }
```

Activate Windows

This screenshot shows a GET request to 'http://localhost:8080/cloudvendor/C5' in Postman. The response status is 200 OK, and the returned JSON object matches the one sent in the previous screenshot, indicating successful retrieval of the vendor with ID C5.

- 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.



```

dler.java      CloudVendorServiceImpl.java      CloudVendorController.java      ResponseHandler.java      a: 2 ^ v :
1 package com.thinkconstructive.restdemo.response;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.http.ResponseEntity;
5
6 import java.util.HashMap;
7 import java.util.Map;
8
9 no usages
10 public class ResponseHandler {
11     no usages
12     @
13     public static ResponseEntity<Object> responseBuilder(
14         String message, HttpStatus httpStatus, Object responseObject
15     ) {
16         Map<String, Object> response = new HashMap<>();
17         response.put("message", message);
18         response.put("httpStatus", httpStatus);
19         response.put("data", responseObject);
20
21         return new ResponseEntity<>(response, httpStatus);
22     }
23 }

```

Code:

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

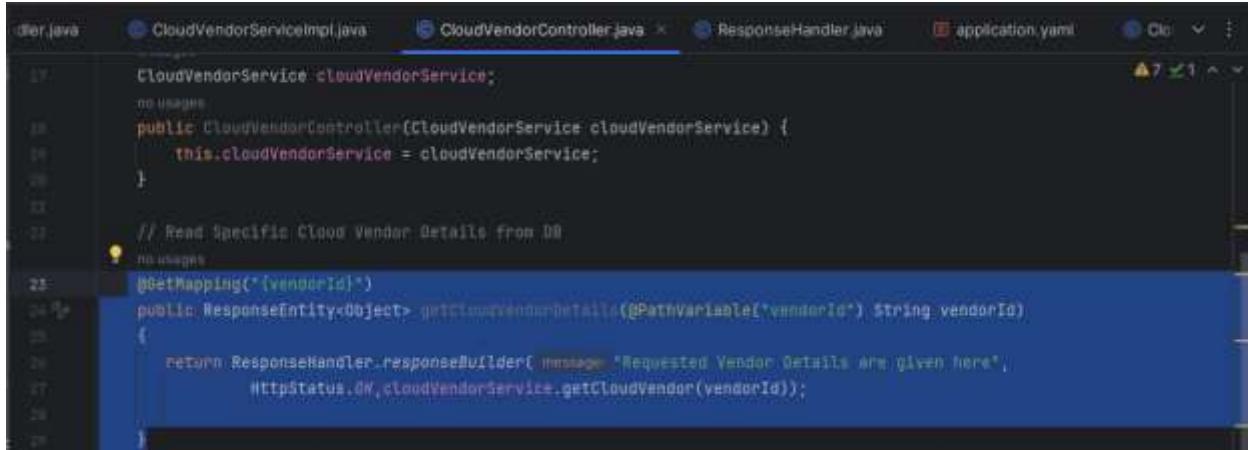
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

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.



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

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

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.

```

2024-01-12T13:10:10.629+05:00 INFO 9564 --- [main] o.s.t.c.RestControllerApplication : No active profile set, falling back to a default profile: 'default'
2024-01-12T13:10:12.665+05:00 INFO 9564 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-01-12T13:10:12.837+05:00 INFO 9564 --- [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 157 ms. Found 0 repository interfaces.
2024-01-12T13:10:14.450+05:00 INFO 9564 --- [main] o.a.w.e.embedded.tomcat.TomcatWebServer : Tomcat initialized with port: 8080 (http)
2024-01-12T13:10:14.489+05:00 INFO 9564 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-01-12T13:10:14.489+05:00 INFO 9564 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.17]
2024-01-12T13:10:14.639+05:00 INFO 9564 --- [main] s.a.c.ServletWebServerApplicationContext : Initializing Spring embedded WebApplicationContext
2024-01-12T13:10:15.157+05:00 INFO 9564 --- [main] s.a.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 10ms
2024-01-12T13:10:19.320+05:00 INFO 9564 --- [main] org.hibernate.Version : HHH000404: Processing PersistenceUnitInfo [name: default]
2024-01-12T13:10:19.408+05:00 INFO 9564 --- [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2024-01-12T13:10:19.408+05:00 INFO 9564 --- [main] o.m.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup; ignoring JPA class transformer
2024-01-12T13:10:19.530+05:00 INFO 9564 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-01-12T13:10:19.673+05:00 INFO 9564 --- [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@401...
2024-01-12T13:10:19.679+05:00 INFO 9564 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-01-12T13:10:21.287+05:00 INFO 9564 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to 'LocalContainerEntityManagerFactoryBean')
2024-01-12T13:10:22.899+05:00 INFO 9564 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-01-12T13:10:23.580+05:00 WARN 9564 --- [main] jpaBaseConfiguration.EntityManagerConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during a view's lifecycle. Explicitly configure open-in-view if this behavior is desired.
2024-01-12T13:10:24.394+05:00 INFO 9564 --- [main] o.a.w.e.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
2024-01-12T13:10:24.485+05:00 INFO 9564 --- [main] s.t.c.RestControllerApplication : Started RestDemoApplication in 15.622 seconds (process time: 15.622)

```

- Create again

HTTP CloudVendorAPI-Collection / Create

POST http://localhost:8080/cloudvendor

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Body

```

1 {
2   "vendorId": "C5",
3   "vendorName": "Vendor Five",
4   "vendorAddress": "Address Five",
5   "vendorPhoneNumber": "five"
6 }

```

Body Cookies Headers (5) Test Results

200 OK 42.62 s 197 B Save as example

Pretty Raw Preview Visualize Text

1 Cloud Vendor Created Successfully

- Query C5 with get by Id.

HTTP CloudVendorAPI-Collection / GetById

GET http://localhost:8080/cloudvendor/C5

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

Body Cookies Headers (5) Test Results

200 OK 1896 ms 345 B Save as example

```

1 {
2   "data": {
3     "vendorId": "C5",
4     "vendorName": "Vendor Five",
5     "vendorAddress": "Address Five",
6     "vendorPhoneNumber": "five"
7   },
8   "httpStatus": "OK",
9   "message": "Requested Vendor Details are given here"
10 }

```

Activate Windows
Go to Settings to activate Windows.

```

Pretty Raw Preview Visualize JSON
1 {
2   "data": {
3     "vendorId": "C5",
4     "vendorName": "Vendor Five",
5     "vendorAddress": "Address Five",
6     "vendorPhoneNumber": "five"
7   },
8   "httpStatus": "OK",
9   "message": "Requested Vendor Details are given here"
10 }

```

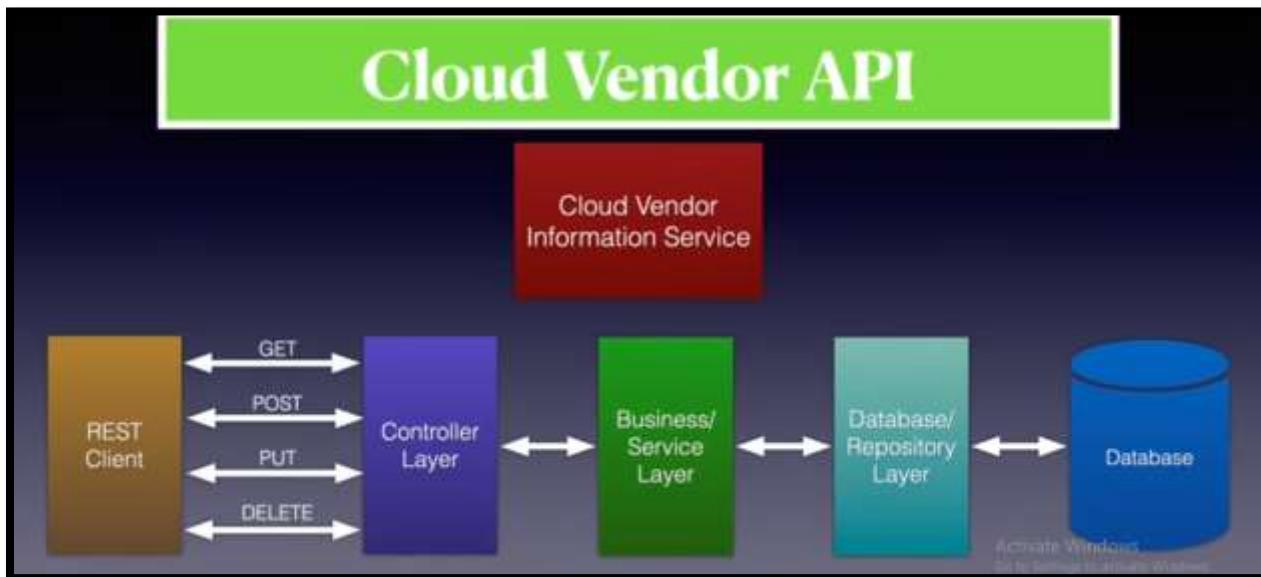
- Transformed the generic response to custom response successfully.

Session# 05

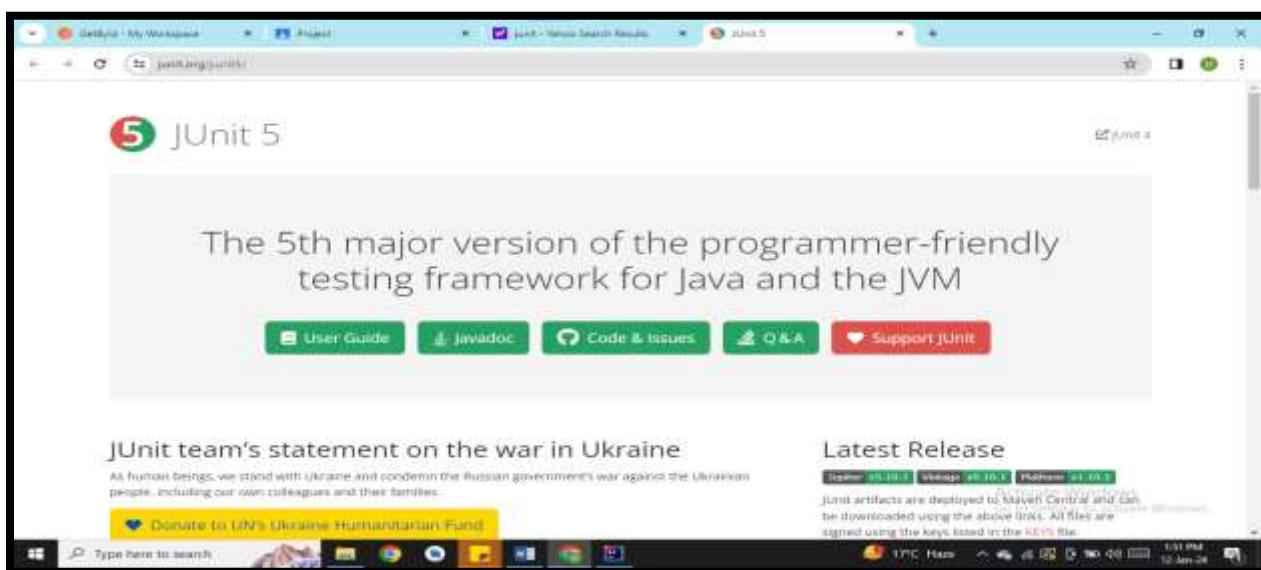
Master Unit Testing Java Spring Boot REST API Application in One Shot

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

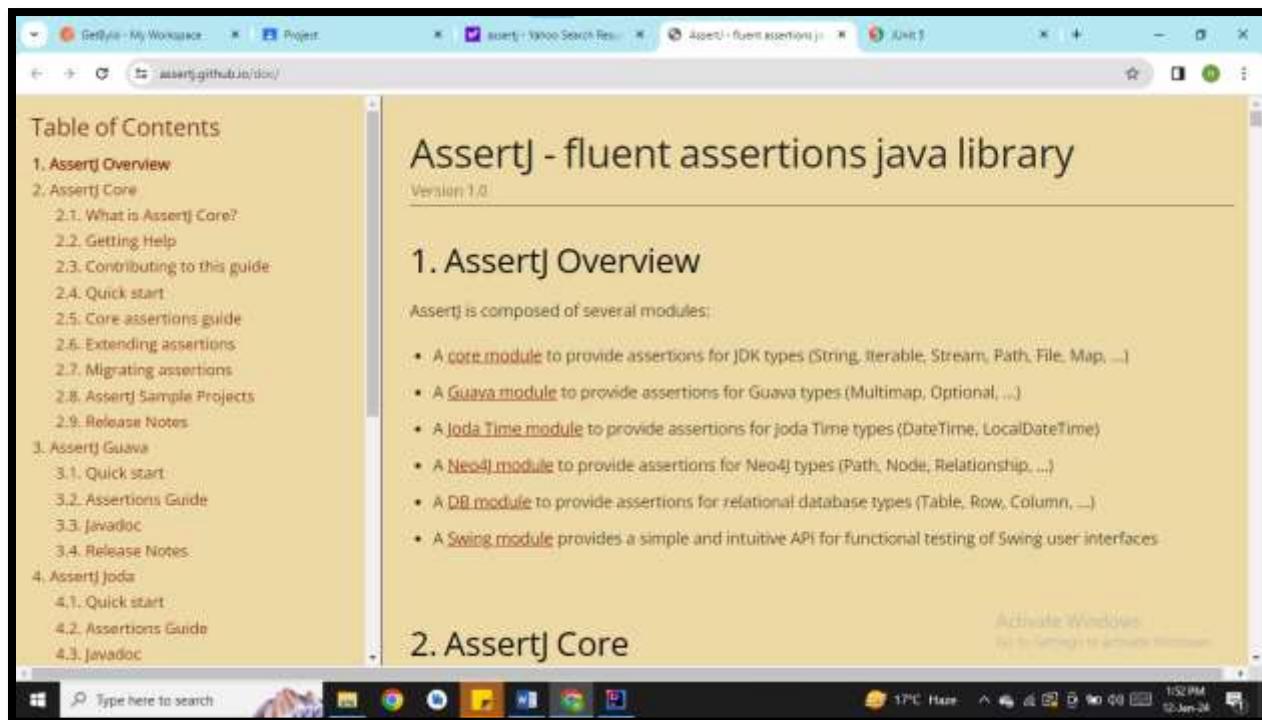
Unit Testing:



Junit 5 Framework



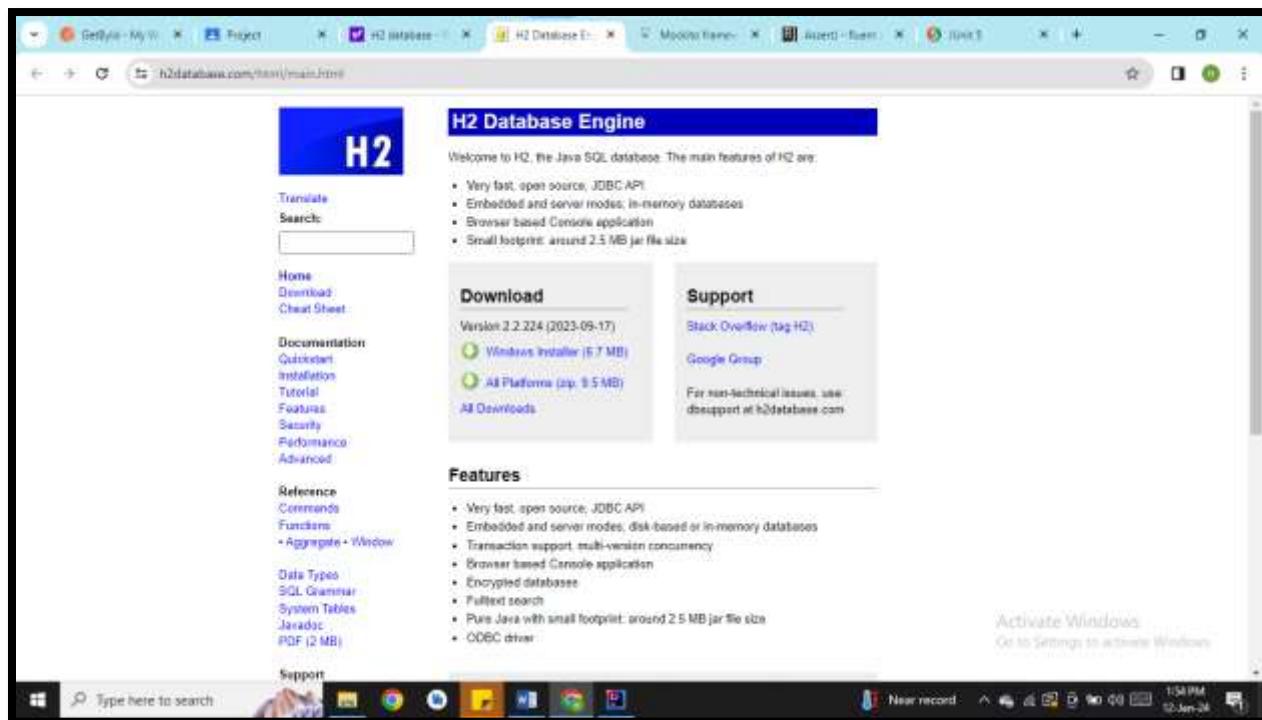
AssertJ Framework (java library)



Mockito Framework



H2 Database

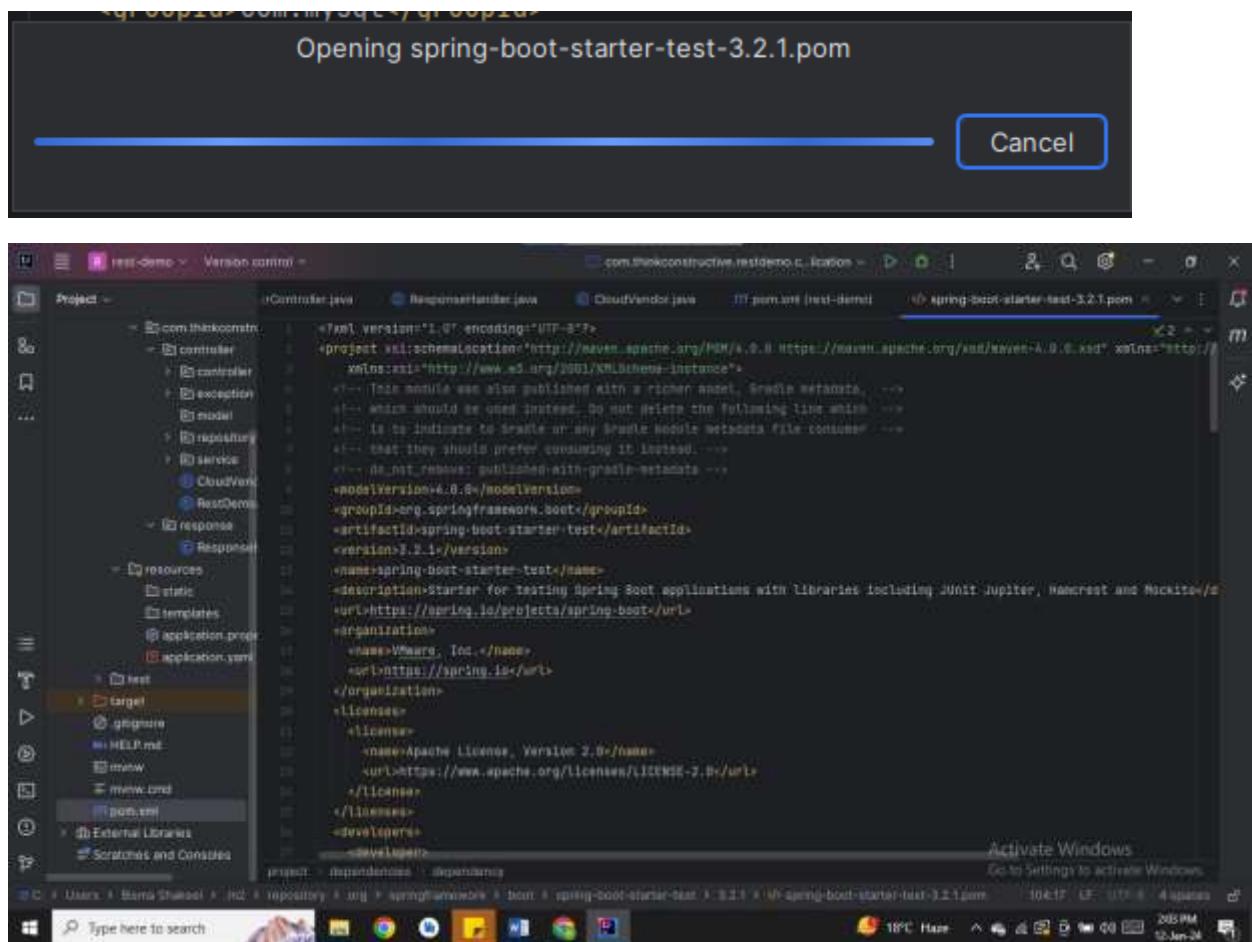


Open IntelliJ IDEA Editor Spring Boot API Project.

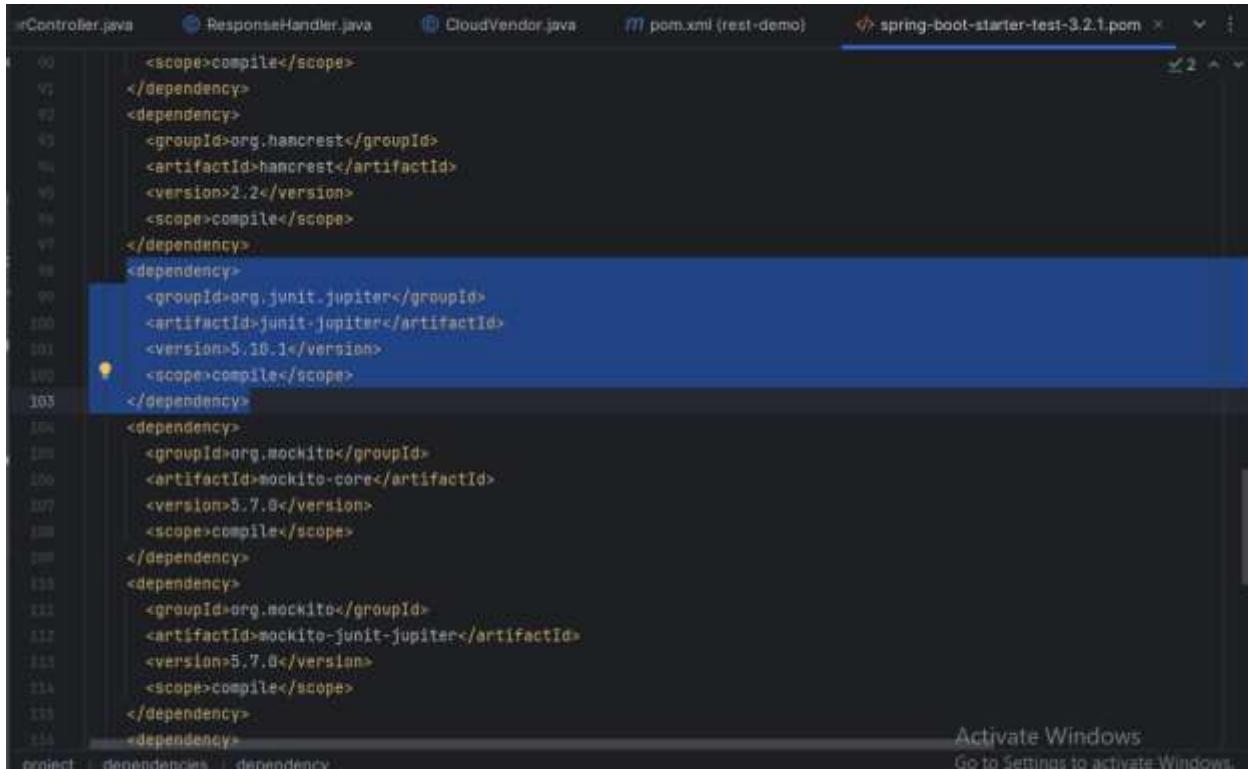
- Check in pom.xml that you have test dependency.

A screenshot of the IntelliJ IDEA code editor showing the contents of a pom.xml file. The file contains XML configuration for a Spring Boot project, specifically defining dependencies for the application. The code includes sections for Spring Boot starters (web, data-jpa), MySQL connector, and a test dependency for Spring Boot's starter-test. The editor interface shows tabs for other files like CloudVendorController.java, ResponseHandler.java, application.yaml, and CloudVendor.java.

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

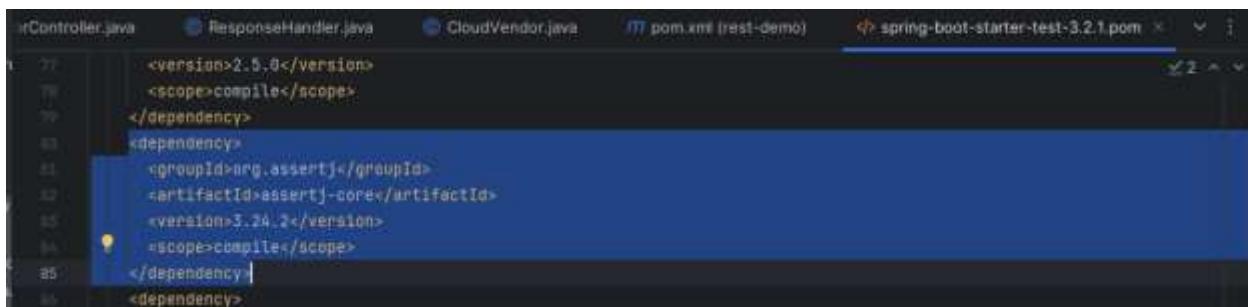


- Junit dependency is already included.



```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.10.1</version>
    <scope>compile</scope>
</dependency>
```

- Assertj dependency is already included.



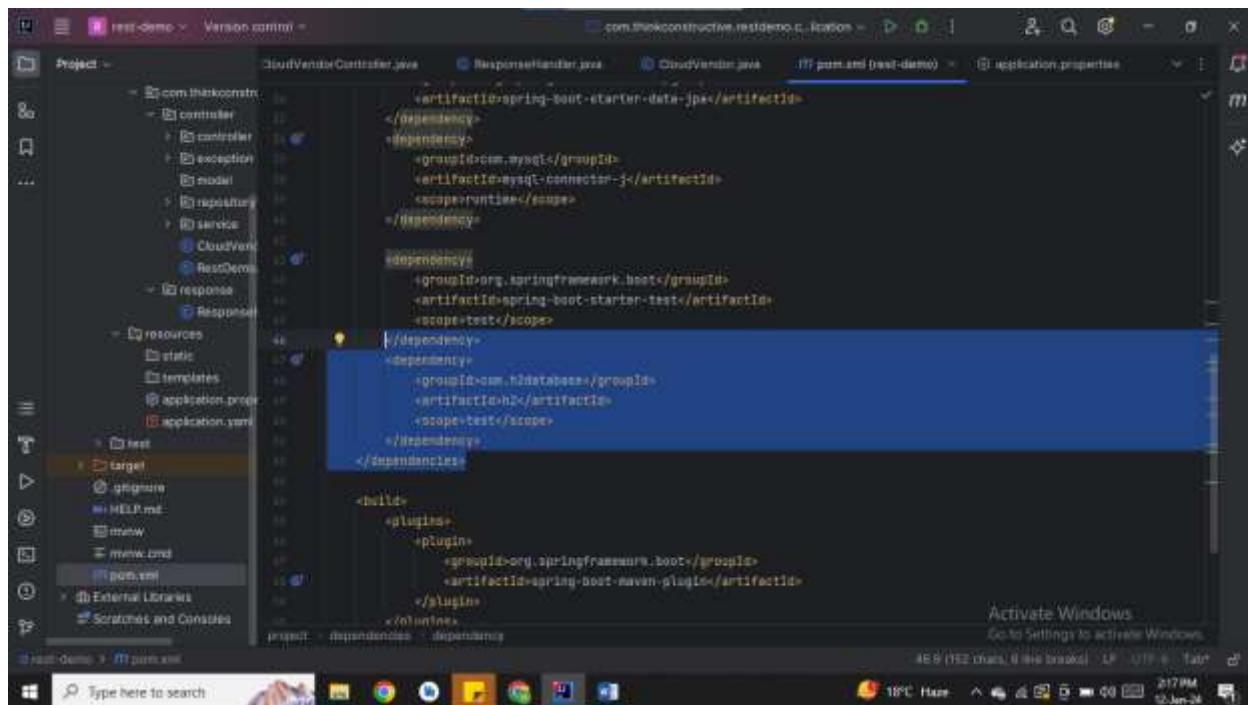
```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>3.7.0</version>
    <scope>compile</scope>
</dependency>
```

- Mockito dependency is already included.



```
<dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest-core</artifactId>
    <version>2.2.0</version>
    <scope>compile</scope>
</dependency>
```

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



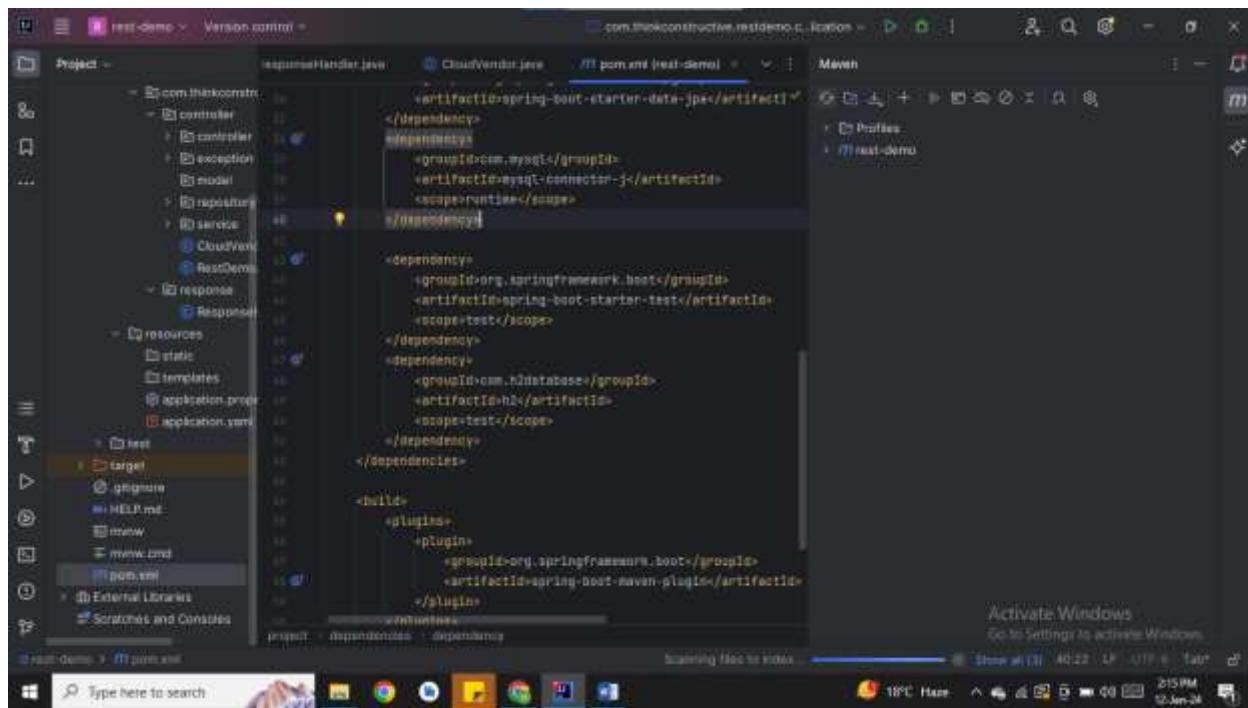
```

<dependency>
    <groupId>com.mysql> </groupId>
    <artifactId>mysql-connector-j </artifactId>
    <scope>runtime</scope>
</dependency>

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

<dependency>
    <groupId>com.h2database </groupId>
    <artifactId>h2 </artifactId>
    <scope>test</scope>
</dependency>

```



```

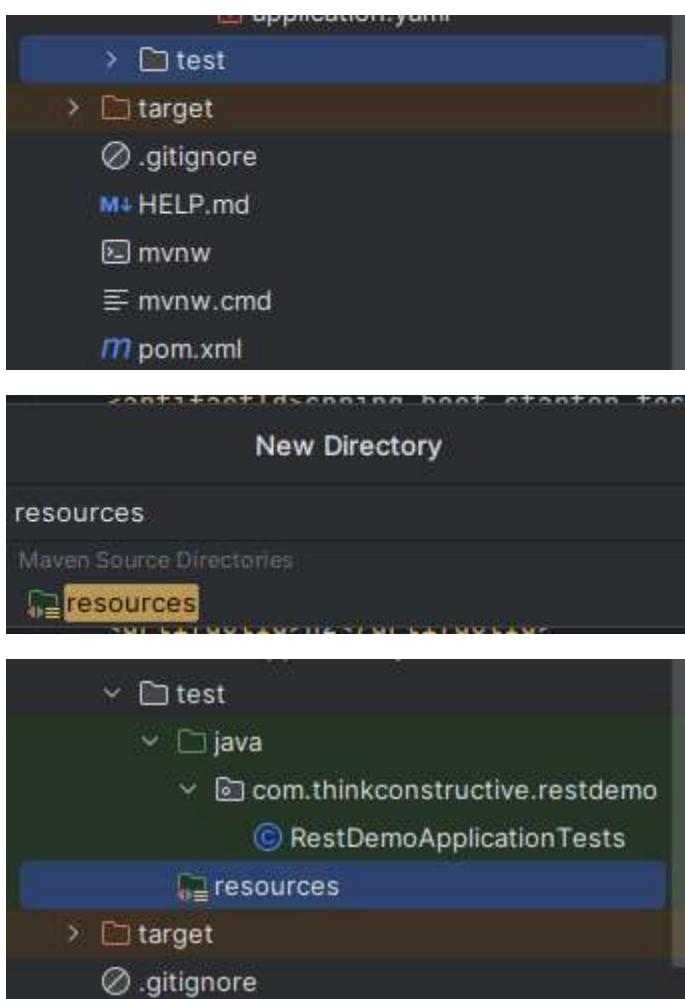
<dependency>
    <groupId>com.mysql </groupId>
    <artifactId>mysql-connector-j </artifactId>
    <scope>runtime</scope>
</dependency>

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

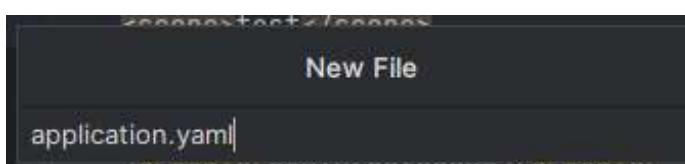
<dependency>
    <groupId>com.h2database </groupId>
    <artifactId>h2 </artifactId>
    <scope>test</scope>
</dependency>

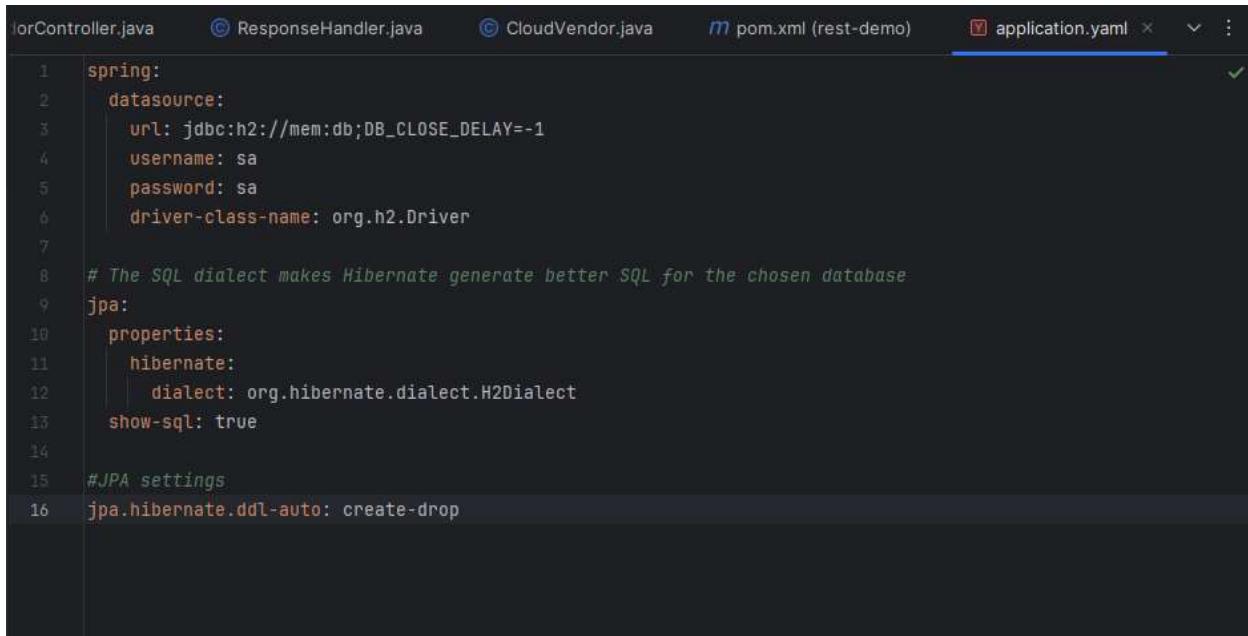
```

- Create Resources folder in Test.



- Inside these resources create a new file.





```
1 spring:
2   datasource:
3     url: jdbc:h2://mem:db;DB_CLOSE_DELAY=-1
4     username: sa
5     password: sa
6     driver-class-name: org.h2.Driver
7
8   # The SQL dialect makes Hibernate generate better SQL for the chosen database
9   jpa:
10    properties:
11      hibernate:
12        dialect: org.hibernate.dialect.H2Dialect
13        show-sql: true
14
15   #JPA settings
16   jpa.hibernate.ddl-auto: create-drop
```

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

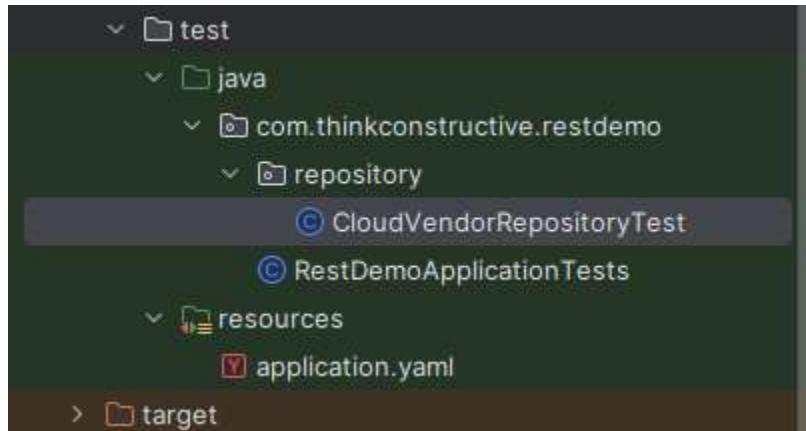
  # The SQL dialect makes Hibernate generate better SQL for the chosen database
  jpa:
    properties:
      hibernate:
        dialect: org.hibernate.dialect.H2Dialect
    show-sql: true

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

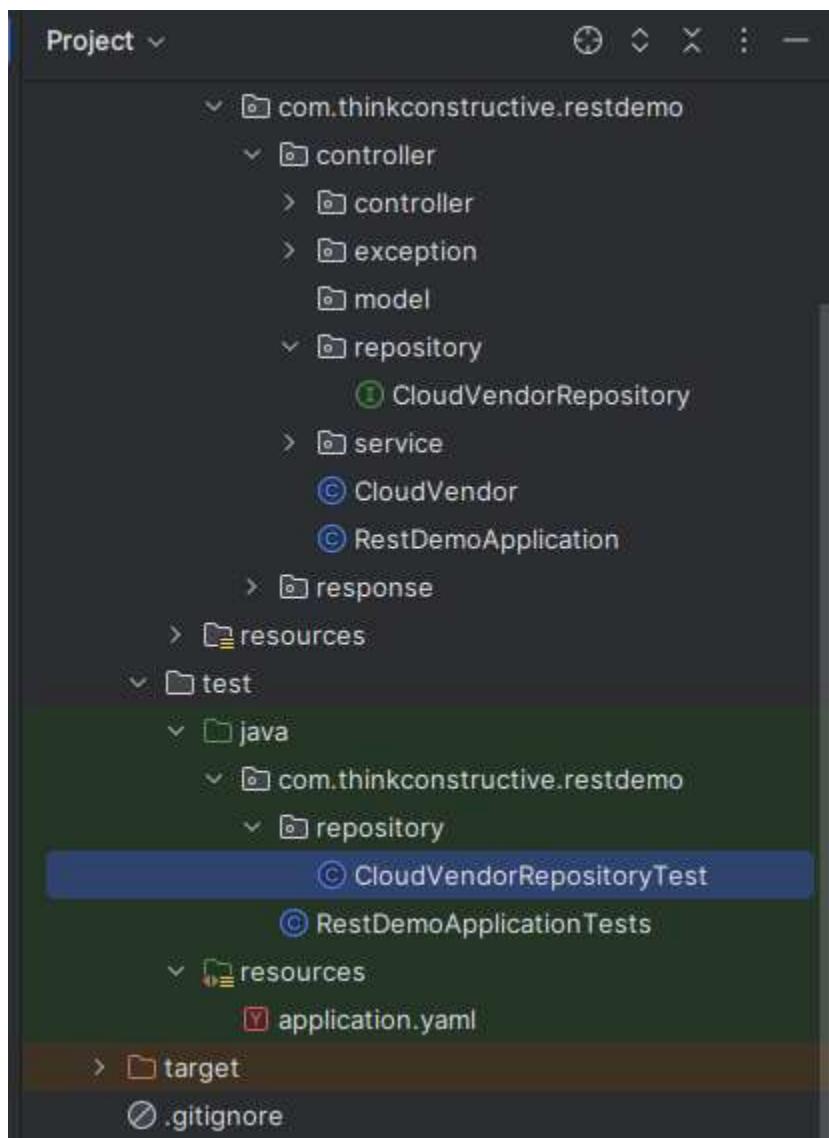
Test Cases for Repository Layer:

- 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 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);
}

```

```

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);
}

```

- Test case SUCCESS**

```

@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.getVendorId());
    assertThat(cloudVendorList.get(0).getVendorAddress())
        .isEqualTo(cloudVendor.getVendorAddress());
}

```

```

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

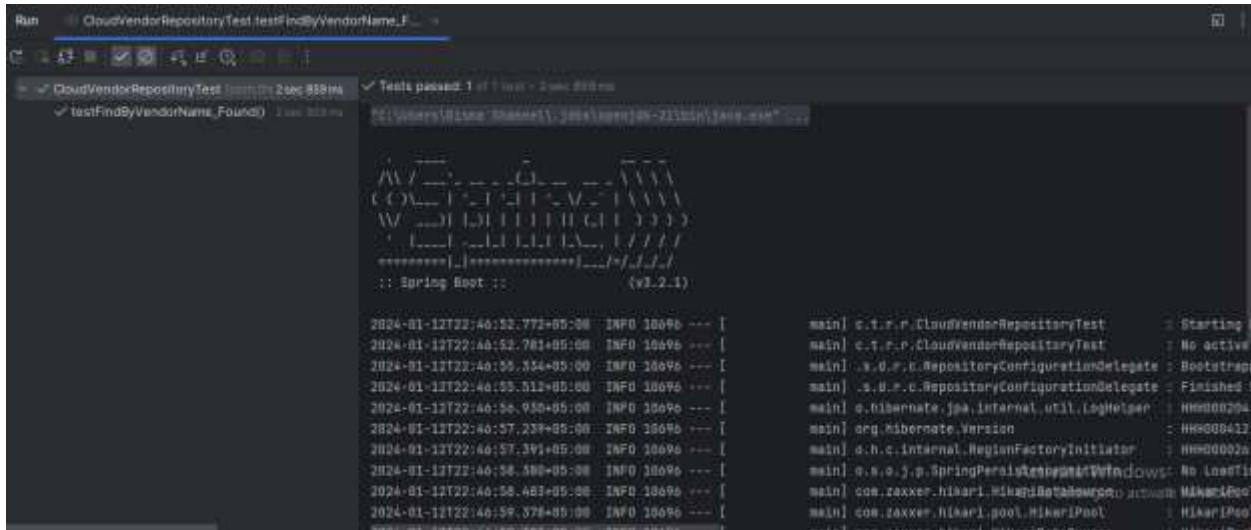
```

```

        assertThat(cloudVendorList).isNotEmpty();

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

```



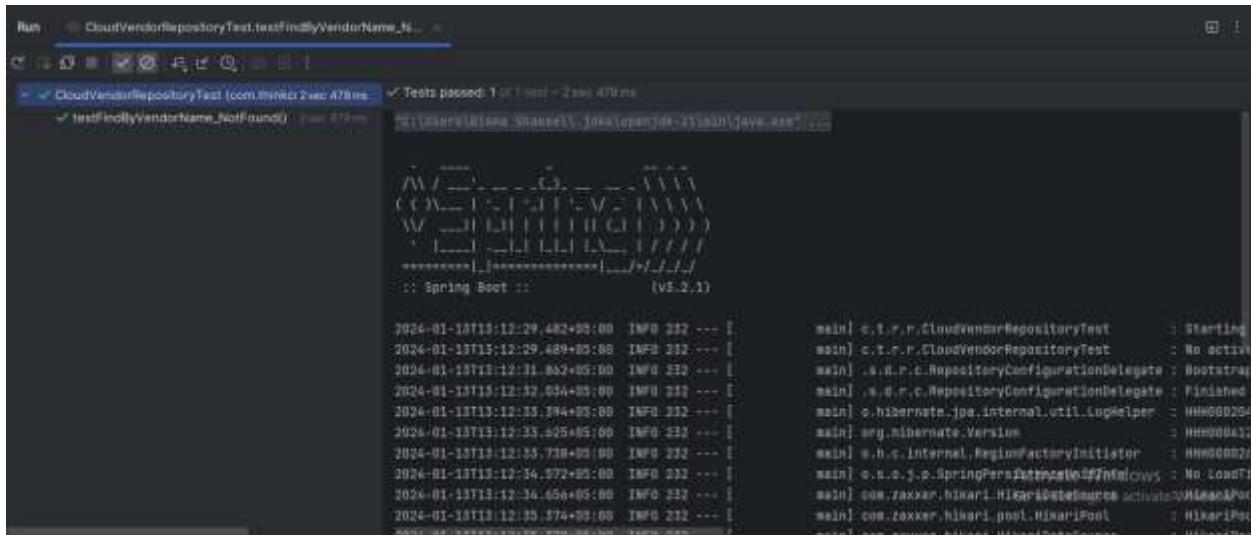
- Test case FAILURE

```

39     }
40
41     // Test case SUCCESS
42     @Test
43     void testFindByVendorName_Found()
44     {
45         List<CloudVendor> cloudVendorList = cloudVendorRepository.findByVendorName("Amazon");
46         assertThat(cloudVendorList).isNotEmpty();
47         assertThat(cloudVendorList.get(0).getVendorId()).isEqualTo(cloudVendor.getVendorId());
48         assertThat(cloudVendorList.get(0).getVendorAddress())
49             .isEqualTo(cloudVendor.getVendorAddress());
50     }
51     // Test case FAILURE
52     @Test
53     void testFindByVendorName_NotFound()
54     {
55         List<CloudVendor> cloudVendorList = cloudVendorRepository.findByVendorName("GCP");
56         assertThat(cloudVendorList.isEmpty()).isTrue();
57     }
58
59
60 }
61

```

```
// 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);
}

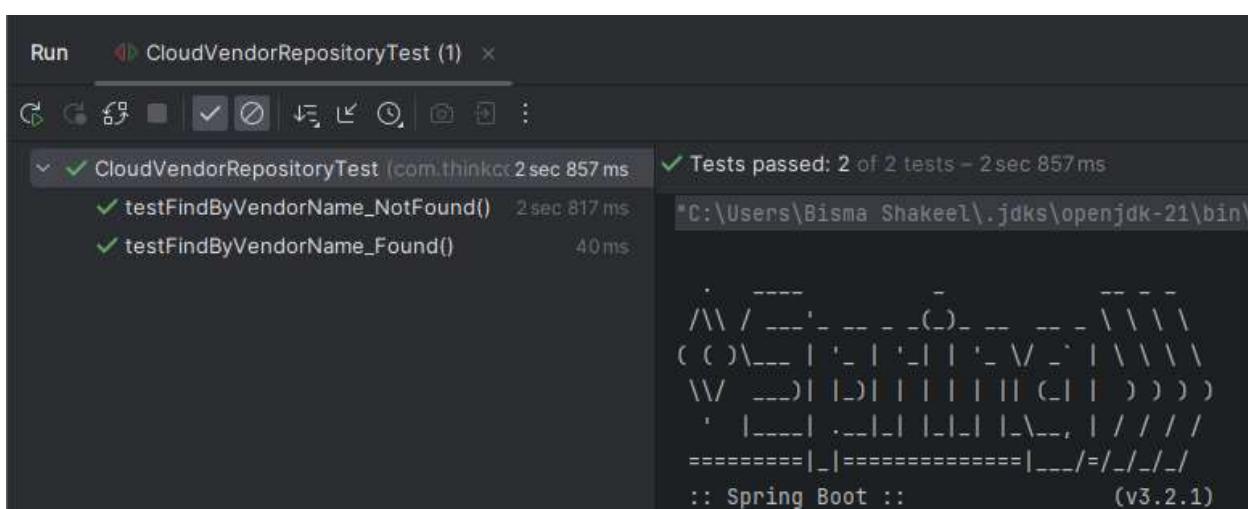
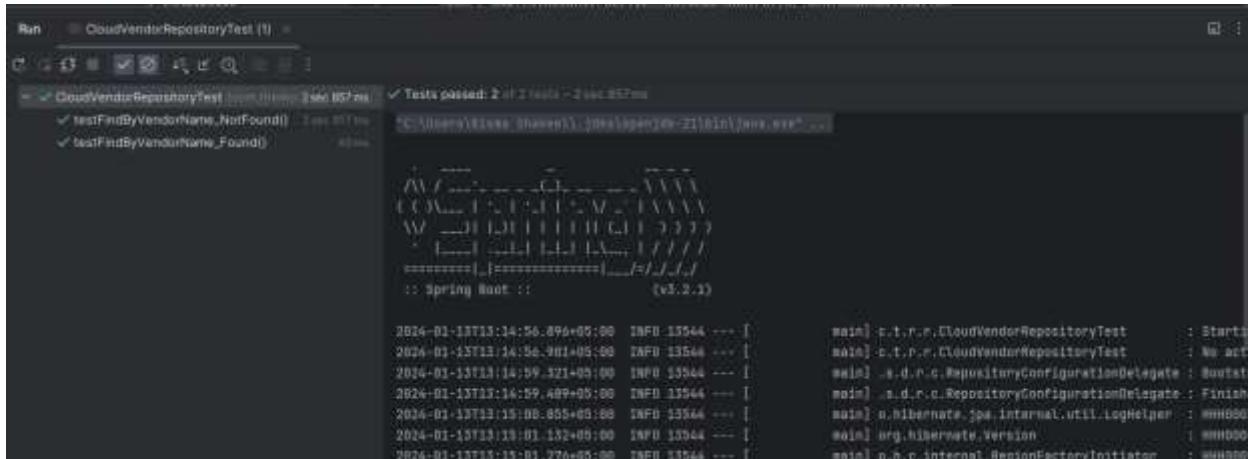
@AfterEach
void tearDown() {
    cloudVendor = null;
    cloudVendorRepository.deleteAll();
}

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

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

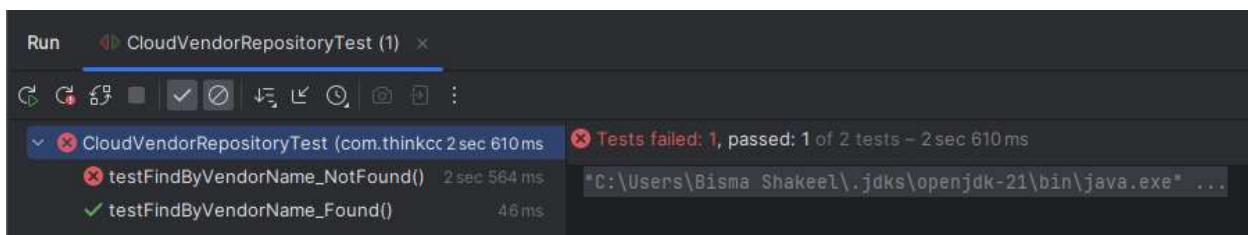
// Test case FAILURE
@Test
void testFindByVendorName_NotFound()
{
    List<CloudVendor> cloudVendorList =
cloudVendorRepository.findByVendorName("GCP");
    assertThat(cloudVendorList.isEmpty()).isTrue();
}
}

```



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

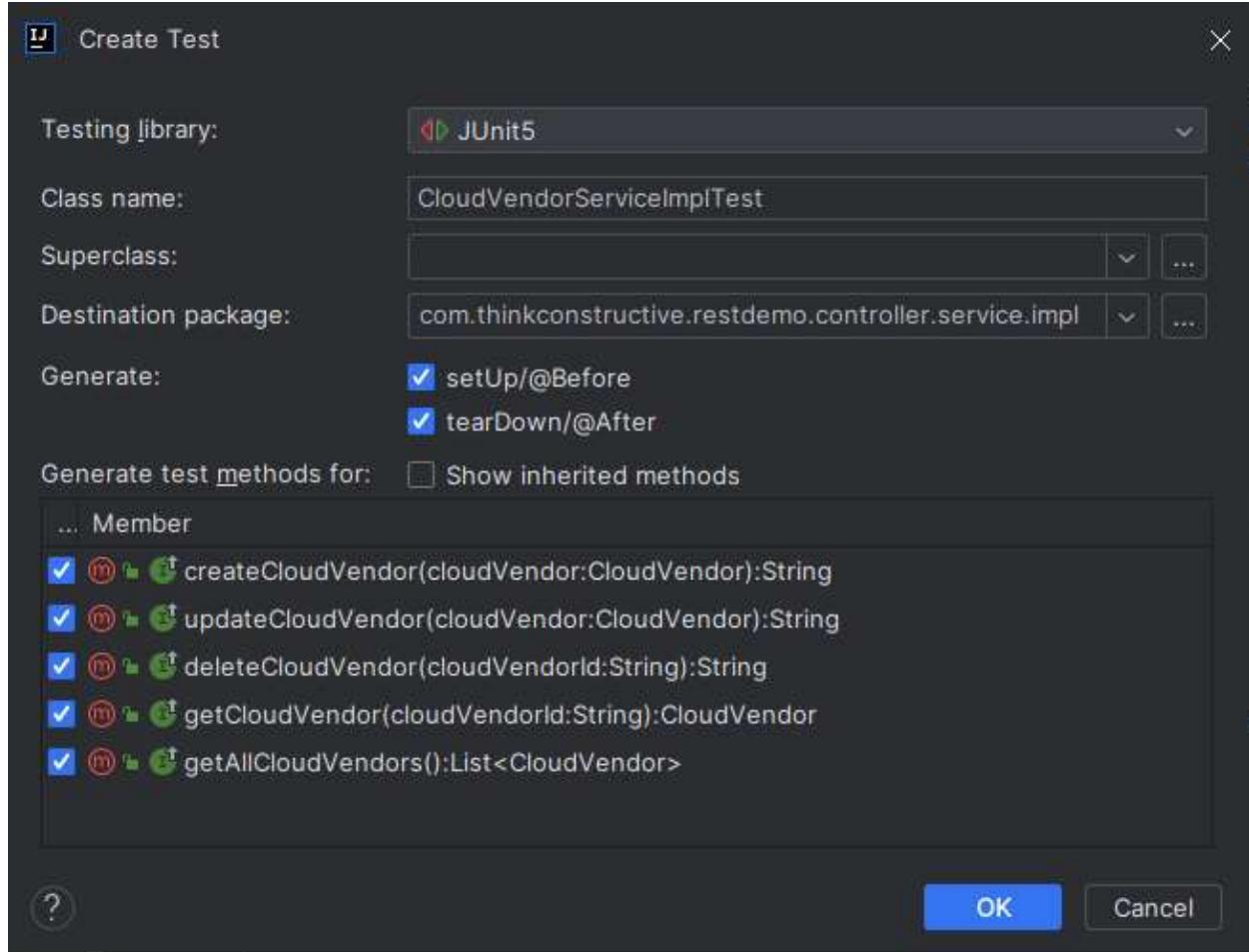
```
// Test case FAILURE
@Test
void testFindByVendorName_NotFound()
{
    List<CloudVendor> cloudVendorList =
cloudVendorRepository.findByVendorName("GCP");
    assertThat(cloudVendorList.isEmpty()).isFalse();
}
```



So, my cloud vendor repository test class is ready.

Test Cases for Service Layer:

- Writing CloudVendorServiceImpl Layer Test cases.

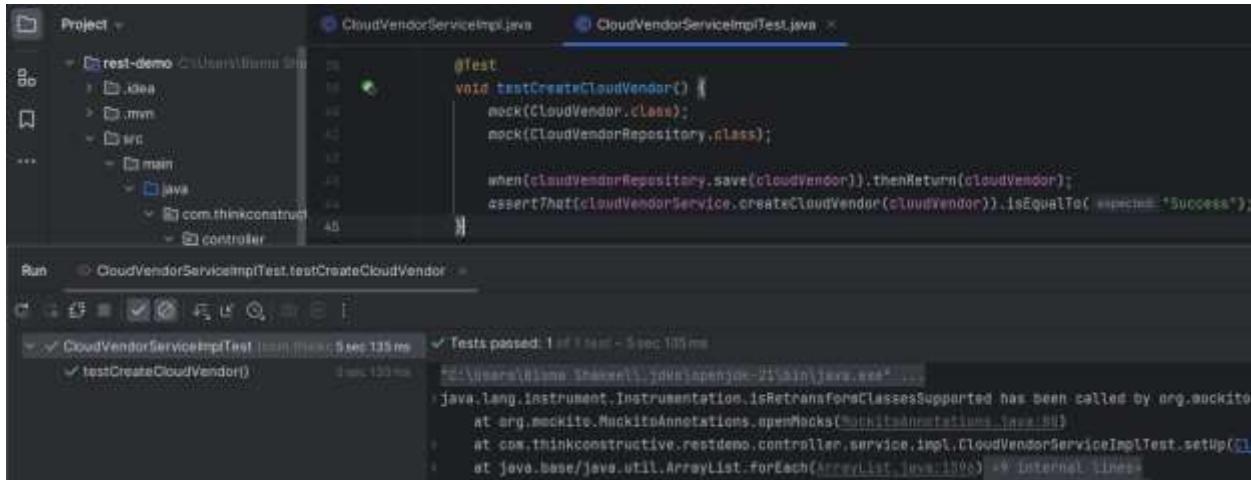


- testCreateCloudVendor Test Case:

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

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

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



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", "xxxxxx");
    }

    @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("Success");
    }

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

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

        assertThat(cloudVendorService.updateCloudVendor(cloudVendor)).isEqualTo("Success");
    }

    @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 test GetAllCloudVendors() {
        mock(CloudVendor.class);
        mock(CloudVendorRepository.class);

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

```

```

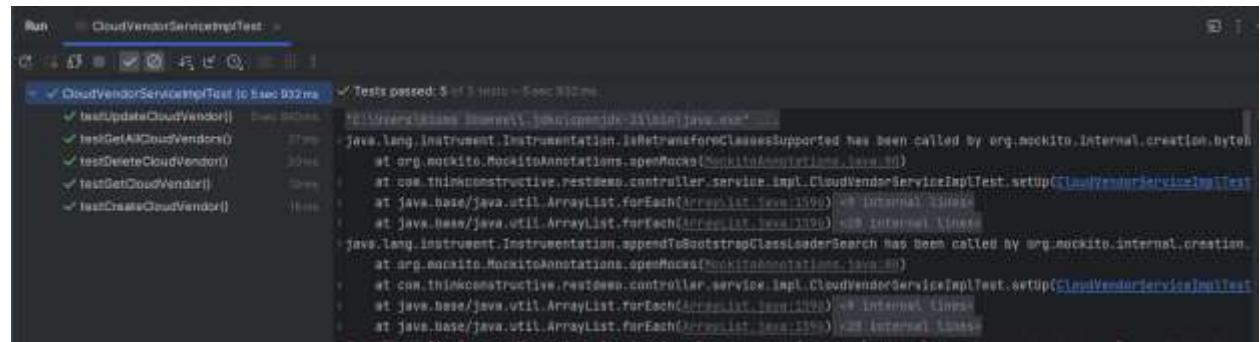
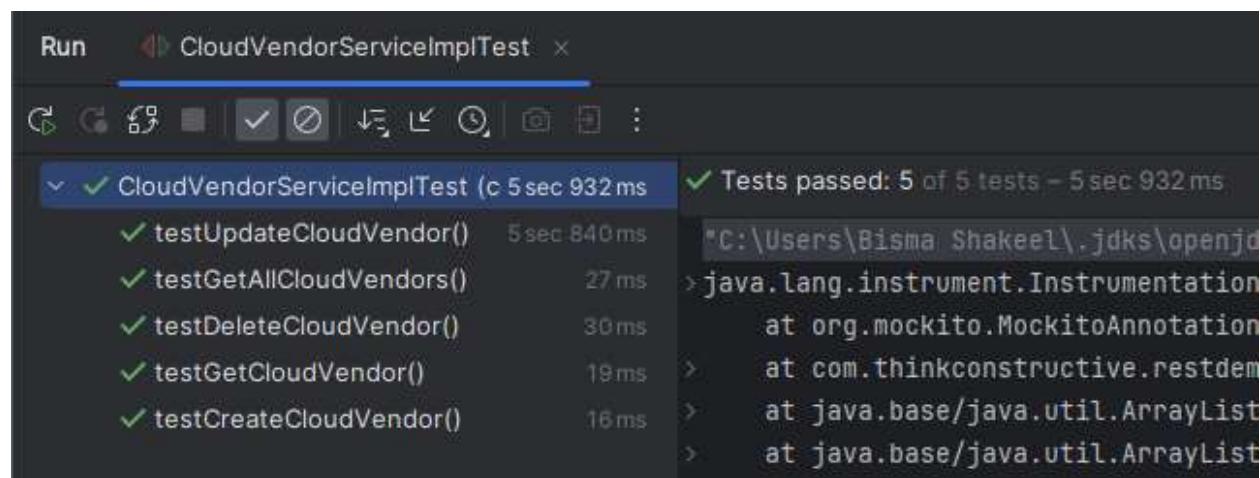
        assertThat(cloudVendorService.getAllCloudVendors().get(0).getVendorPhoneNumber())
            .isEqualTo(cloudVendor.getVendorPhoneNumber());
    }

    @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

The screenshot shows the IntelliJ IDEA interface with the Coverage tool window open. The Coverage window displays the following information:

- CloudVendorServiceImplTest (1)**
- Tests passed: 5 of 5 tests – 4 sec 635 ms**
- Line coverage ...**
- include patterns:** com.\thinkconstructive\.restdemo\.controller\.service\.impl\..*
- exclude patterns:**
- exclude annotations patterns:**

The code editor shows the `CloudVendorServiceImpl.java` file with coverage annotations. The `CloudVendorServiceImpl` class has 100% coverage for its methods. The `CloudVendorServiceImplTest` class also has 100% coverage for its methods.

- All green means covered.

CloudVendorServiceImpl.java

```

16     public CloudVendorServiceImpl(CloudVendorRepository cloudVendorRepository) {
17         this.cloudVendorRepository = cloudVendorRepository;
18     }
19
20
21     3 usages
22     @Override
23     public String createCloudVendor(CloudVendor cloudVendor) {
24         cloudVendorRepository.save(cloudVendor);
25         return "Success";
26     }
27
28     2 usages
29     @Override
30     public String updateCloudVendor(CloudVendor cloudVendor) {
31         cloudVendorRepository.save(cloudVendor);
32         return "Success";
33     }
34
35     2 usages
36     @Override
37     public String deleteCloudVendor(String cloudVendorId) {
38         cloudVendorRepository.deleteById(cloudVendorId);
39         return "Success";
40     }

```

Coverage: 100% (11/11), 100% (11/11), 91% (11/12)

- CloudVendorRepositoryTest Run with coverage.

Project: CloudVendorServiceImpl.java

CloudVendorRepositoryTest.java

```

package com.thinkconstructive.restdemo.repository;
import com.thinkconstructive.restdemo.controller.CloudVendor;
import com.thinkconstructive.restdemo.controller.RestGeoApplication;
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;

```

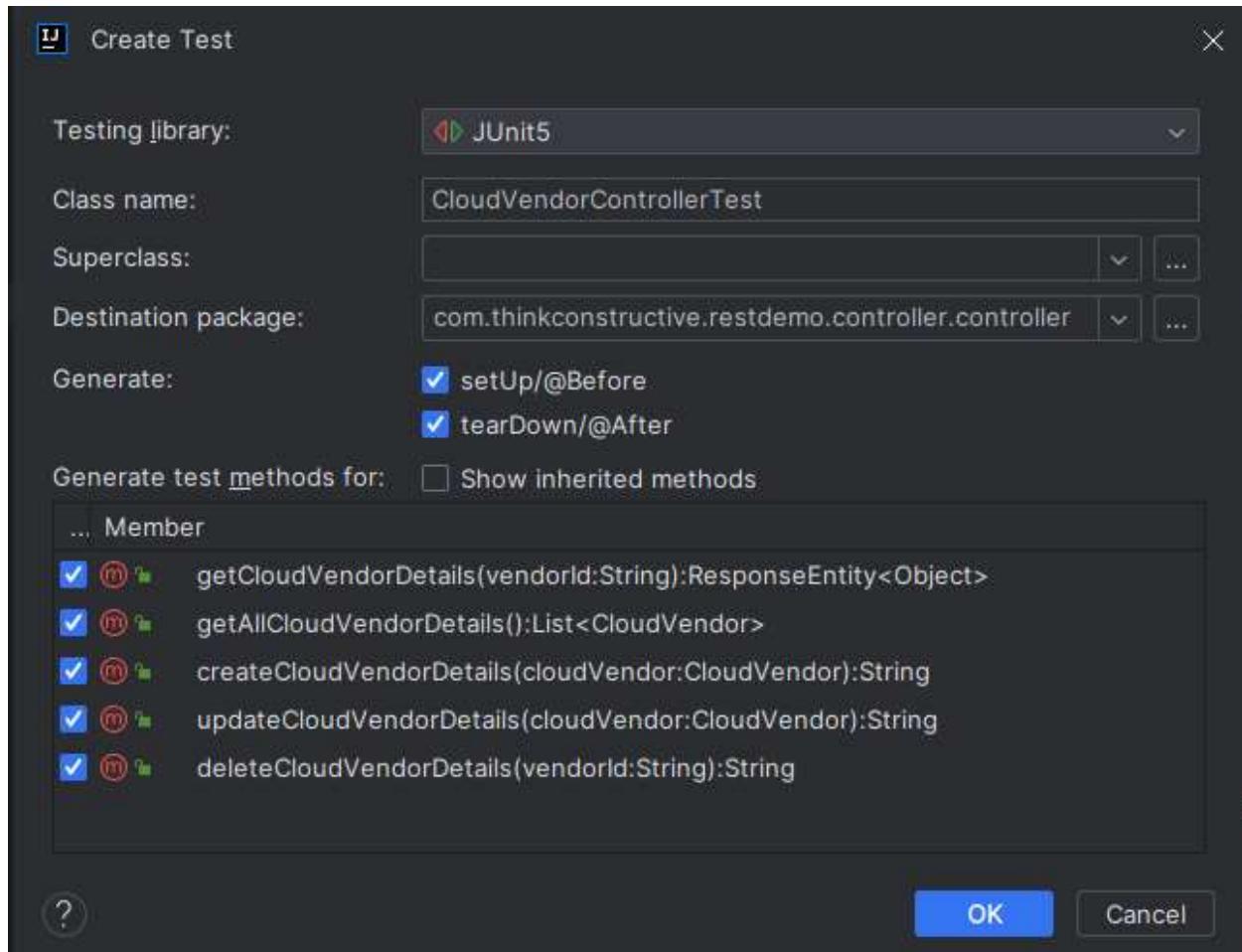
Coverage: 100% (10/10), 100% (10/10), 91% (11/12)

CloudVendorRepositoryTest

CloudVendorRepositoryTest

- CloudVendorRepositoryTest
- testFindByVendorName_NoMatch
- testFindByVendorName_Found

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 >
C F D E V S W M G R P L C I T B S E T A S H E X S E C U R I T Y
= ✓ CloudVendorControllerTest (com.thinkconstructive.controller.CloudVendorController)
  ✓ testGetCloudVendorDetails() 702 ms

MockHttpServletRequest:
    HTTP Method = GET
    Request URI = /clousvendor/t
    Parameters = {}
        Headers = []
        Body = null
    Session Attrs = {}

Handler:
    Type = com.thinkconstructive.testdems.controller.controller.CloudVendorController
    Method = com.thinkconstructive.testdems.controller.controller.CloudVendorController#GetCloudVendorDetails(String)

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"}, "httpForwarded": null}
        Forwarded URL = null
        Redirected URL = null
        Cookies = []

Activate Windows
Go to Settings to activate Windows.
```

```
✓ CloudVendorControllerTest [CloudVendorControllerTest.java:17] Tests passed: 1 of 1 (0ms)
  ✓ testGetAllCloudVendorDetails [CloudVendorControllerTest.java:22] 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","vendorPhoneNumbers":"xxxxx"}, {"httpStatus":100,"message": "Success"}]}
      Forwarded URL = null
      Redirected URL = null
```

- GetAllCloudVendorDetails Test Case:

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

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

```
Run └─ CloudVendorControllerTest.testGetAllCloudVendorDetails...
  ✓ CloudVendorControllerTest [CloudVendorControllerTest.java:17] Tests passed: 1 of 1 (0ms)
    ✓ testGetAllCloudVendorDetails [CloudVendorControllerTest.java:22] MockHttpServletResponse:
      HTTP Method = GET
      Request URI = /cloudvendor
      Parameters = {}
      Headers = {}
      Body = null
      Session Attrs = {}

      Handler:
        Type = com.thinkconstructive.restdemo.controller.controller.CloudVendorController
        Method = com.thinkconstructive.restdemo.controller.controller.CloudVendorController#getAllCloudVendorDetails()

      Async:
        Async started = false
        Async result = null

      Resolved Exception:
        Type = null

      ModelAndView:
        View name = null
```

```

CloudVendorControllerTest ... ✓ Tests passed: 1 of 1 (total - 427 ms)
  ✓ testGetAllCloudVendorDetails (427 ms)
    MockHttpServletRequest:
     getAttributeNames = [id]
      getAttribute(id) = 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 = [{"vendorId":1,"vendorName":"Amazon","vendorAddress":"USA","vendorPhoneNumber":"xxxxx"}, {"vendorId":2,"vendorName":"Bata","vendorAddress":"India","vendorPhoneNumber":"xxxxx"}]
        forwarded URL = null
        redirected URL = null
        cookies = []

```

- List has been shown

```

MockHttpServletResponse:
  status = 200
  error message = null
  headers = [{Content-Type:"application/json"}]
  content type = application/json
  body = [{"vendorId":1,"vendorName":"Amazon","vendorAddress":"USA","vendorPhoneNumber":"xxxxx"}, {"vendorId":2,"vendorName":"Bata","vendorAddress":"India","vendorPhoneNumber":"xxxxx"}]
  forwarded URL = null
  redirected URL = null
  cookies = []

```

- 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());
}

```

```

CloudVendorControllerTest (com.thinkconstructive.restdemo) ... ✓ Tests passed: 1 of 1 (total - 528 ms)
  ✓ testDeleteCloudVendorDetails() 528 ms
    MockHttpServletRequest:
      HTTP Method = DELETE
      Request URI = /cloudvendor/1
      Parameters = {}
      Headers = {}
      Body = null
      Session Attrs = {}

    Handler:
      Type = com.thinkconstructive.restdemo.controller.CloudVendorController
      Method = com.thinkconstructive.restdemo.controller.CloudVendorController#deleteCloudVendor

      Async:
        Async started = false
        Async result = null

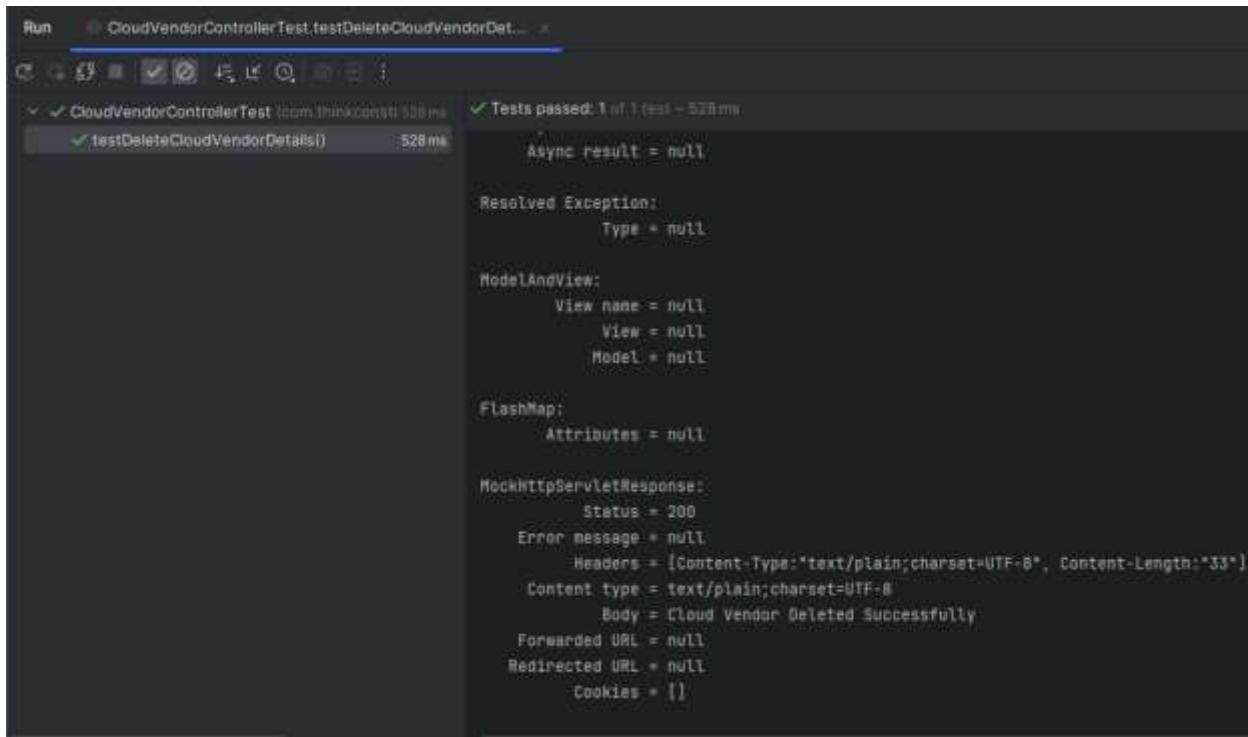
      Resolved Exception:
        Type = null

```

```

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 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());
    }
}
```

The screenshot shows a Java IDE interface with a test runner window. The title bar says "Run - CloudVendorControllerTest.testCreateCloudVendorDetails". Below it, a status bar indicates "Tests passed: 1 of 1 test - 663 ms". The main content area displays the following log output:

```
MockHttpServletRequest:
  HTTP Method = POST
  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 = {}

Handler:
  type = com.thinkconstructive.restdms.controller.controller.CloudVendorController
  Method = com.thinkconstructive.restdms.controller.controller.CloudVendorController#createCloudVendorDetails
```

```
✓ Tests passed: 1 of 1 test - 663 ms
```

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

```
✓ CloudVendorControllerTest [com.thinkandz:663 ms] ✓ Tests passed: 1 of 1 test - 663 ms
  ✓ testCreateCloudVendorDetails() 663 ms

  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 Created Successfully
    Forwarded URL = null
    Redirected URL = null
    Cookies = []
```

- 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());
}
```

```
Run CloudVendorControllerTest.testUpdateCloudVendorDetails -> Tests passed: 1 (0.1ms) - 304 ms
  ✓ testUpdateCloudVendorDetails() 304 ms

  MockHttpServletRequest:
    HTTP Method = PUT
    Request URI = /cloudvendor
    Parameters = {}
    Headers = [Content-Type="application/json; charset=UTF-8", Content-Length:"134"]
    Body = {
      "vendorId" : "1",
      "vendorName" : "Amazon",
      "vendorAddress" : "USA",
      "vendorPhoneNumbers" : "XXXX"
    }
    Session Attrs = {}

  Handler:
    Type = com.thinkconstructive.restdemo.controller.CloudVendorController
    Method = com.thinkconstructive.restdemo.controller.CloudVendorController#updateCloudVendorDetails

  Async:
    keySync started = false
    keySync result = null
```

✓ 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 [com.thinkconst:664 ms]
  ✓ testUpdateCloudVendorDetails() 664 ms
    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);

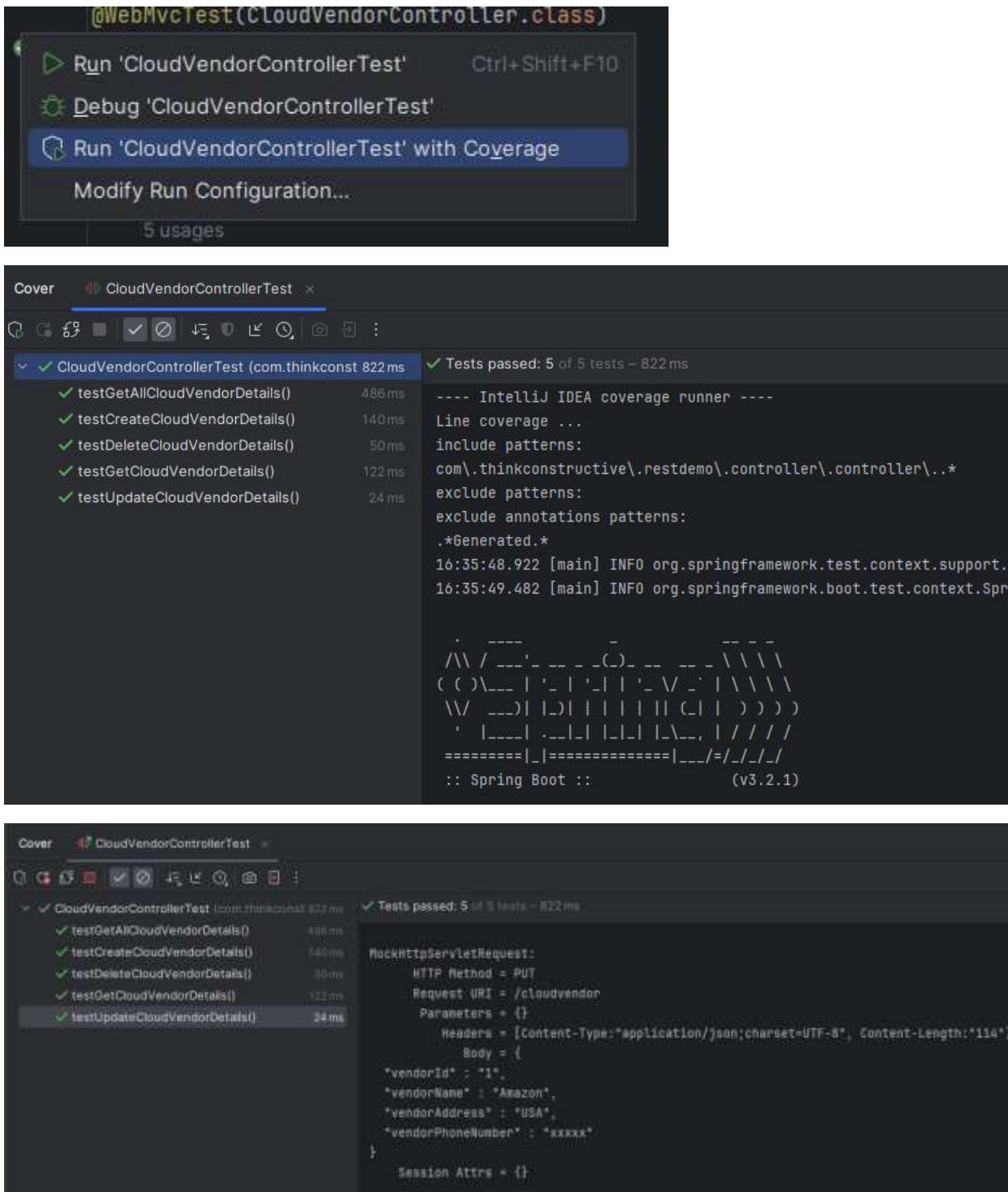
    when(cloudVendorService.updateCloudVendor(cloudVendorOne))
        .thenReturn("Success");
    this.mockMvc.perform(put("/cloudvendor")
        .contentType(MediaType.APPLICATION_JSON)
        .content(requestJson))
        .andDo(print()).andExpect(status().isOk());
}

}

@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(CloudVendorController) 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;
no usage
public CloudVendorController(CloudVendorService cloudVendorService) {
    this.cloudVendorService = cloudVendorService;
}

// Read Specific Cloud Vendor Details from DB
no usage
@GetMapping("{vendorId}")
public ResponseEntity<Object> getCloudVendorDetails(@PathVariable("vendorId")
{
    return ResponseHandler.responseBuilder("Requested Vendor Details are")
        .HttpStatus.OK,cloudVendorService.getCloudVendor(vendorId));
}

// Read All Cloud Vendor Details from DB
no usage
@GetMapping()
public List<CloudVendor> getAllCloudVendorDetails()
{
    return cloudVendorService.getAllCloudVendors();
}
```

```
CloudVendorController.java
35 }
36 }
37
38     no usages
39     @PostMapping
40     public String createCloudVendorDetails(@RequestBody CloudVendor cloudVendor)
41     {
42         cloudVendorService.createCloudVendor(cloudVendor);
43         return "Cloud Vendor Created Successfully";
44     }
45     no usages
46     @PutMapping
47     public String updateCloudVendorDetails(@RequestBody CloudVendor cloudVendor)
48     {
49         cloudVendorService.updateCloudVendor(cloudVendor);
50         return "Cloud Vendor Updated Successfully";
51     }
52     no usages
53     @DeleteMapping("{vendorId}")
54     public String deleteCloudVendorDetails(@PathVariable("vendorId") String vendorId)
55     {
56         cloudVendorService.deleteCloudVendor(vendorId);
57         return "Cloud Vendor Deleted Successfully";
58     }
59 }
```

CloudVendorControllerTest.java

5 1 ▲ ▼

- 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.

