Lesson 04 Demo 04

Registering Microservice with Eureka Server

Objective: To demonstrate how to register a microservice with Eureka Server using Spring Boot

Tool required: Eclipse IDE

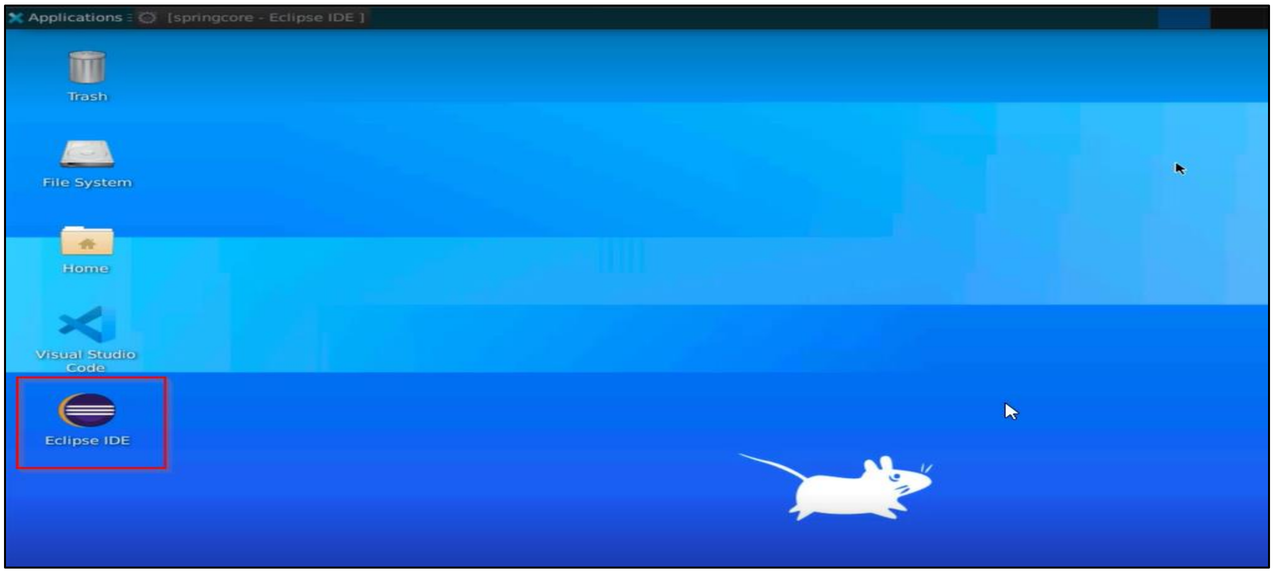
Prerequisites: None

Steps to be followed:

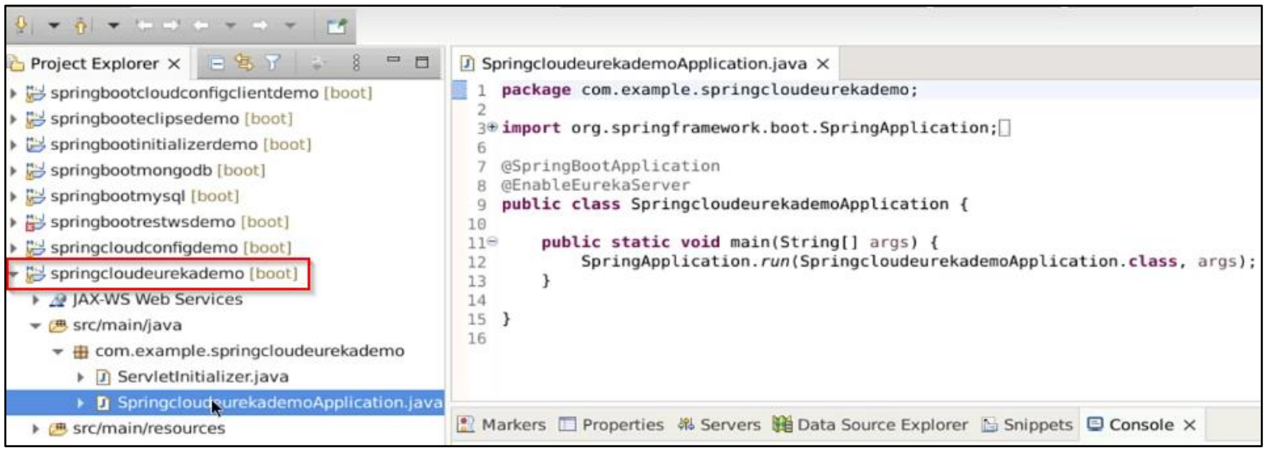
1. Configuring the Eureka server
2. Creating a Spring Starter project
3. Creating an AppController.java file
4. Testing the application on Eureka Server

Step 1: Configuring the Eureka server

1.1 Open **Eclipse IDE**



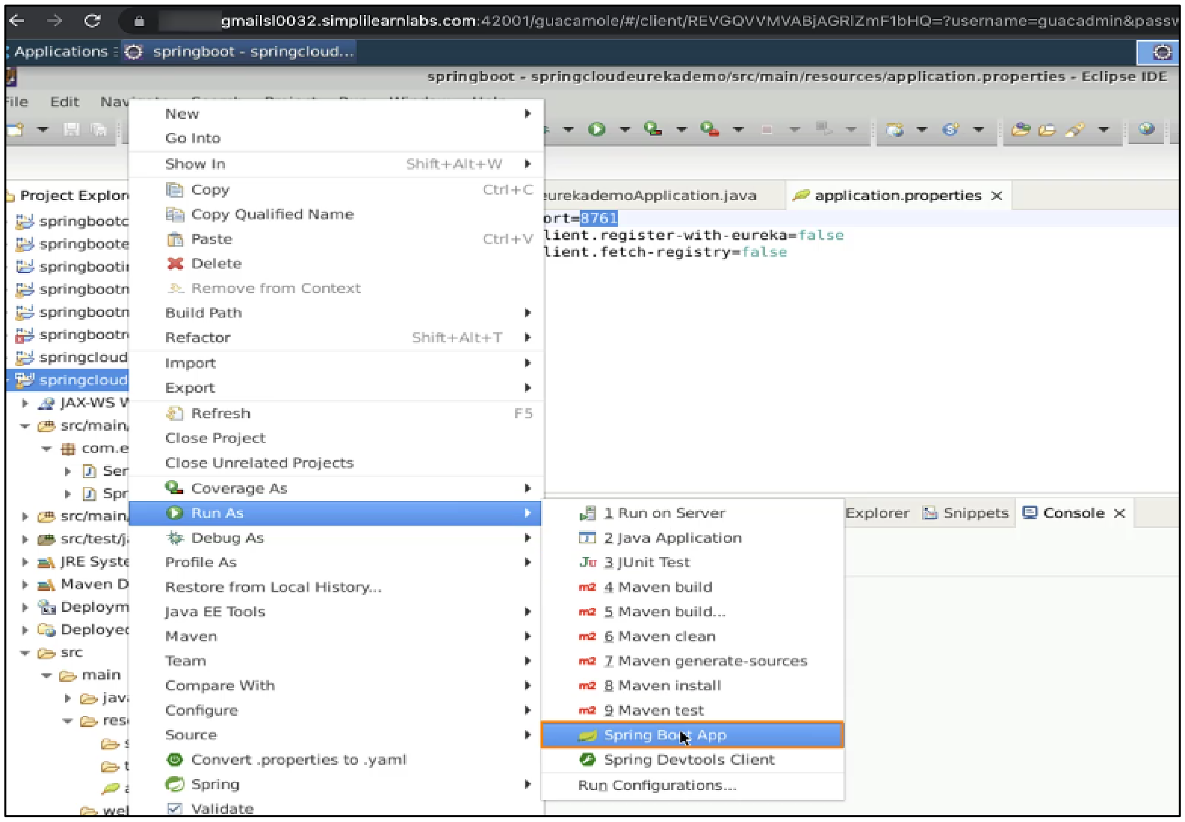
1.2 To configure the Eureka Server, navigate to the directory **src/main/java** in the **springcloudeurekademo** project


Project Explorer
springbootcloudconfigclientdemo [boot]
springbooteclipsedemo [boot]
springbootinitializerdemo [boot]
springbootmongodb [boot]
springbootmysql [boot]
springbootrestwsdemo [boot]
springcloudconfigdemo [boot]
springcloudeurekademo [boot]
JAX-WS Web Services
src/main/java
com.example.springcloudeurekademo
ServletInitializer.java
SpringcloudeurekademoApplication.java
src/main/resources
SpringcloudeurekademoApplication.java
Markers
Properties
Servers
Data Source Explorer
Snippets
Console

```
package com.example.springcloudeurekademo;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
@EnableEurekaServer
public class SpringcloudeurekademoApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringcloudeurekademoApplication.class, args);
    }

}
```

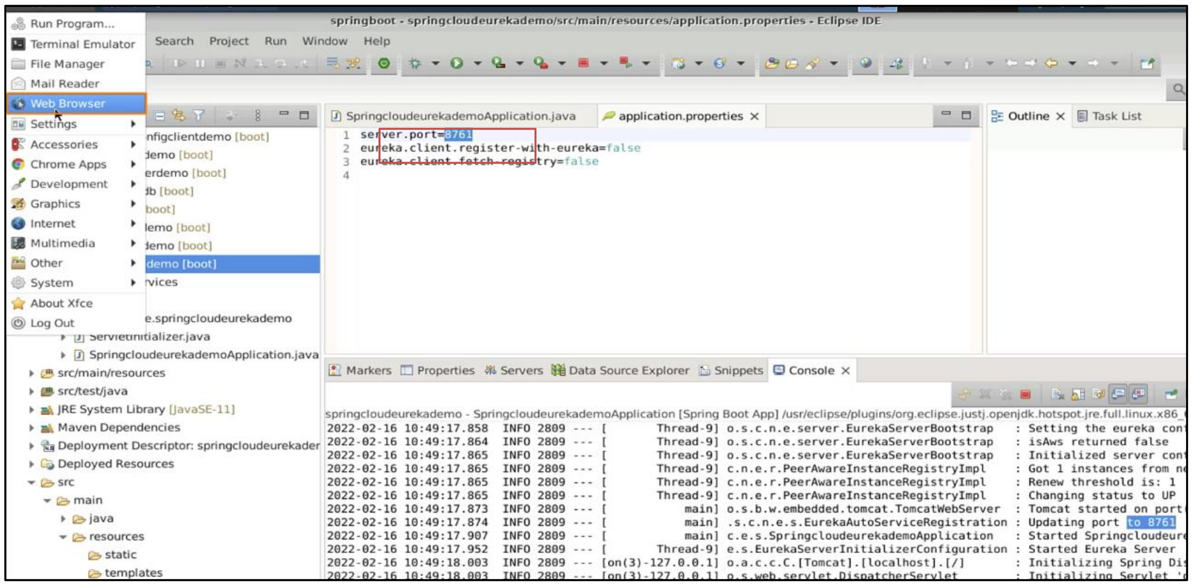
Note: Please refer to the previous demo on how to create the **springcloudeurekademo** project

1.3 Right-click on the **springcloudeurekademo** project and click **Run As > Spring Boot App**



```
ort=8761
lient.register-with-eureka=false
lient.fetch-registry=false
```

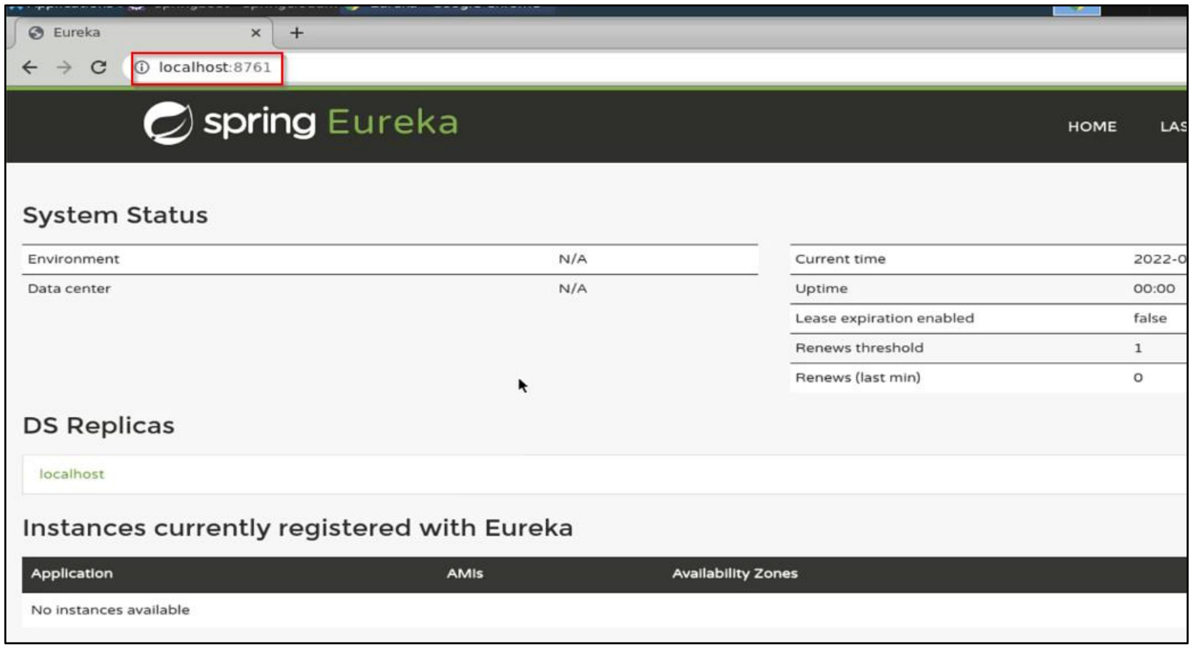
1.4 Once the server starts, open a web browser and type **localhost:8761** to access the **Eureka Server**



```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

```
springcloudeurekademo - SpringcloudeurekademoApplication [Spring Boot App] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_
2022-02-16 10:49:17.858  INFO 2809 --- [       Thread-9] o.s.c.n.e.server.EurekaServerBootstrap   : Setting the eureka con
2022-02-16 10:49:17.864  INFO 2809 --- [       Thread-9] o.s.c.n.e.server.EurekaServerBootstrap   : isAws returned false
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] o.s.c.n.e.server.EurekaServerBootstrap   : Initialized server con
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl    : Got 1 instances from n
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl    : Renew threshold is: 1
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl    : Changing status to UP
2022-02-16 10:49:17.873  INFO 2809 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port
2022-02-16 10:49:17.874  INFO 2809 --- [           main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8761
2022-02-16 10:49:17.907  INFO 2809 --- [           main] c.e.s.SpringcloudeurekademoApplication   : Started Springcloudeur
2022-02-16 10:49:17.952  INFO 2809 --- [       Thread-9] e.s.EurekaServerInitializerConfiguration : Started Eureka Server
2022-02-16 10:49:18.003  INFO 2809 --- [on(3)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring Di
2022-02-16 10:49:18.003  INFO 2809 --- [on(3)-127.0.0.1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet '
```

1.5 You should be able to access the **Spring Eureka server** by typing **localhost:8761** in the browser



System Status

Environment	N/A
Data center	N/A

Current time	2022-0
Uptime	00:00
Lease expiration enabled	false
Renews threshold	1
Renews (last min)	0

DS Replicas

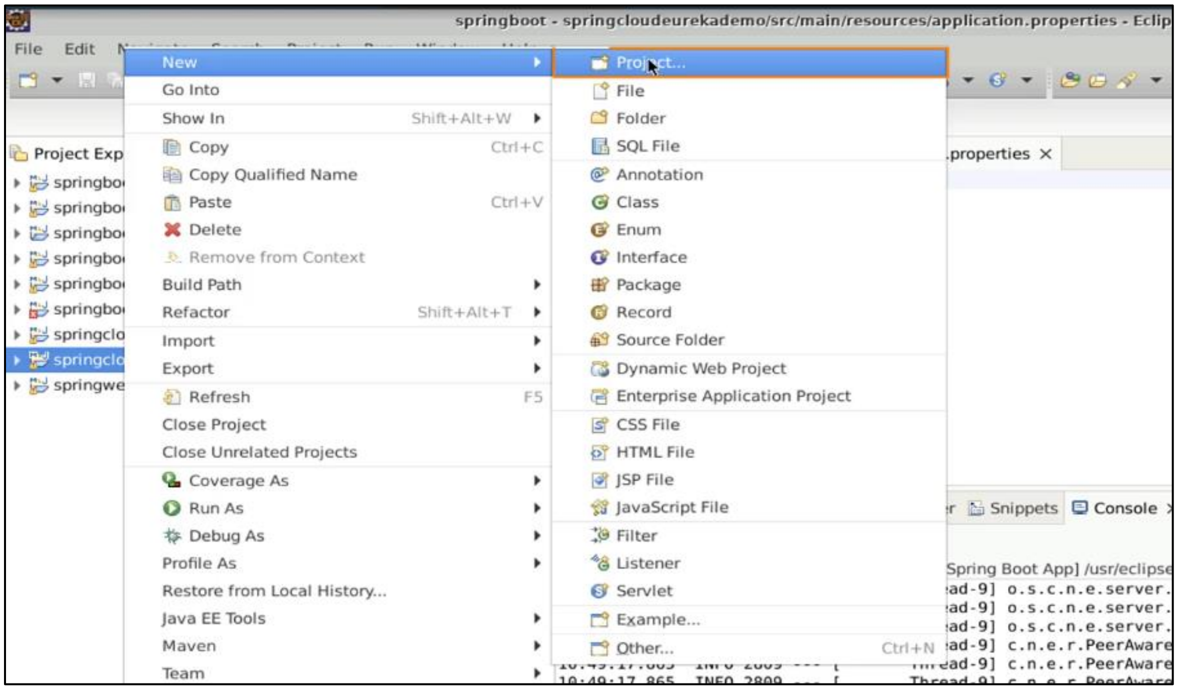
localhost

Instances currently registered with Eureka

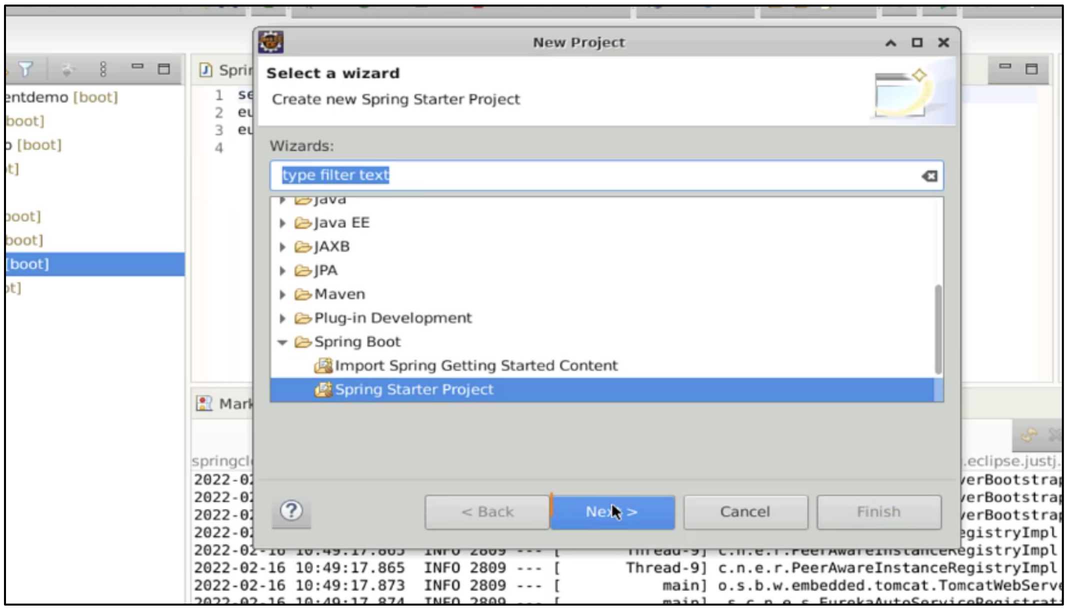
Application	AMIs	Availability Zones
No instances available		

Step 2: Creating a Spring Starter project

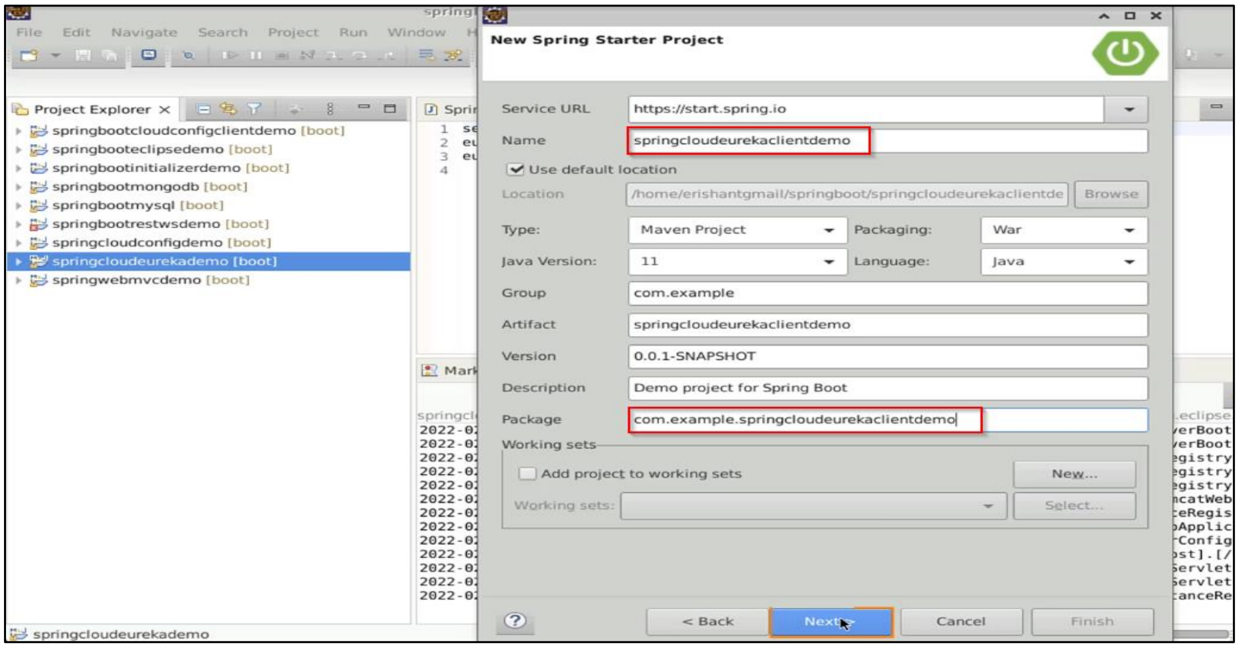
2.1 Access the **Springcloudeurekademo** project and select **New > Project**



2.2 Choose **Spring Starter Project** and click **Next**



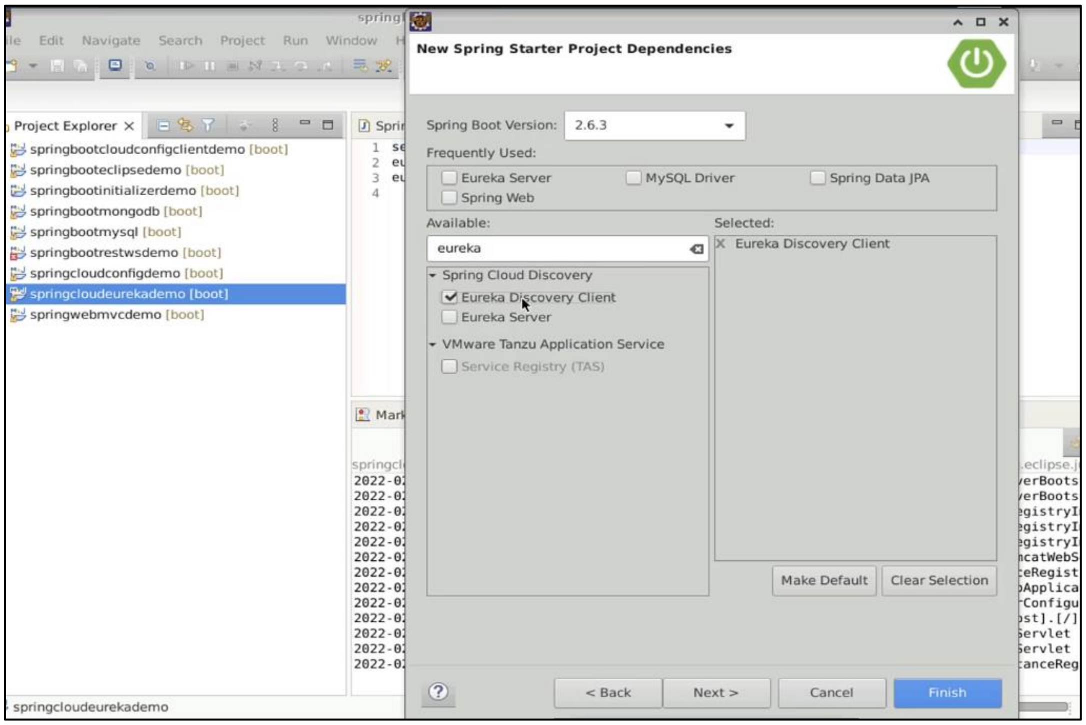
2.3 Provide a name for the project, such as **springcloudeurekaclientdemo**, and set the package name as **example.springcloudeurekaclientdemo**. Now, click **Next**


New Spring Starter Project

Service URL: https://start.spring.io
Name: springcloudeurekaclientdemo
☑ Use default location
Location: /home/erishantgmail/springboot/springcloudeurekaclientde
Type: Maven Project Packaging: War
Java Version: 11 Language: Java
Group: com.example
Artifact: springcloudeurekaclientdemo
Version: 0.0.1-SNAPSHOT
Description: Demo project for Spring Boot
Package: com.example.springcloudeurekaclientdemo
Working sets
☐ Add project to working sets
Working sets:

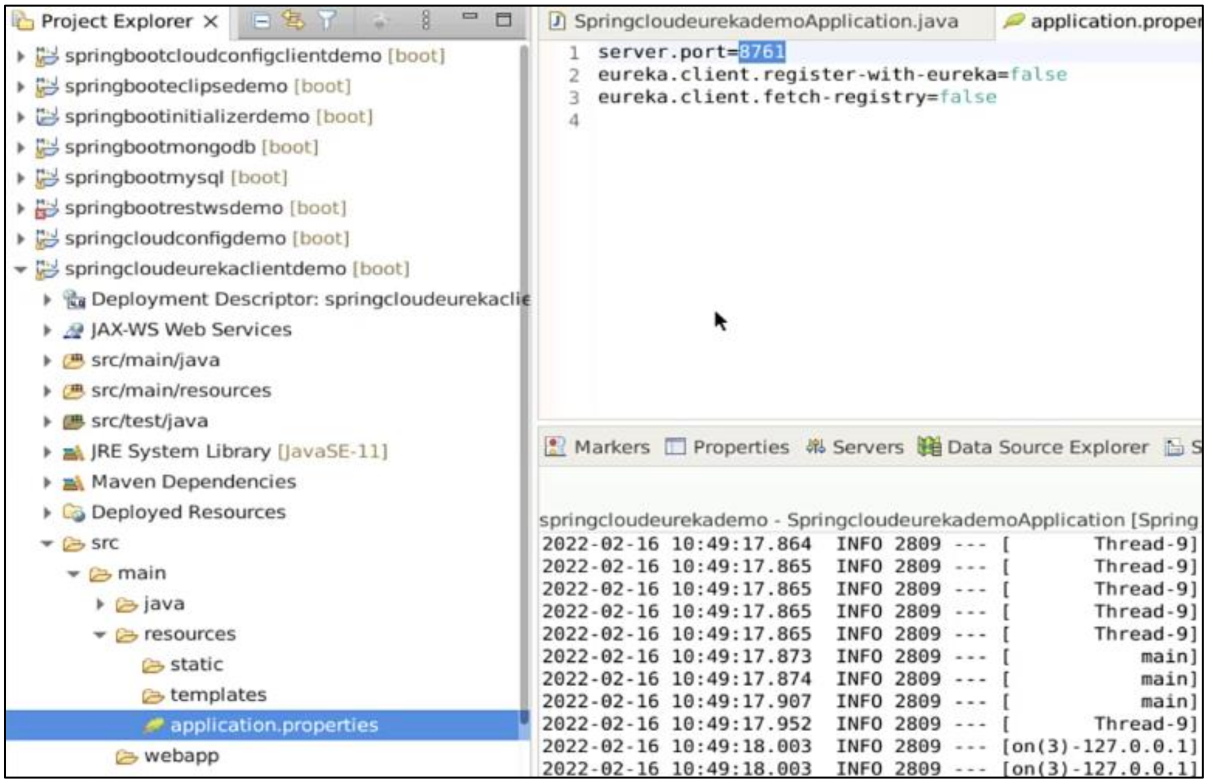
Project Explorer
springbootcloudconfigclientdemo [boot]
springbooteclipsedemo [boot]
springbootinitializerdemo [boot]
springbootmongodb [boot]
springbootmysql [boot]
springbootrestwsdemo [boot]
springcloudconfigdemo [boot]
springcloudeurekademo [boot]
springwebmvcdemo [boot]

2.4 Select **Eureka Discovery Client** as the dependency and click **Next > Finish**


New Spring Starter Project Dependencies

Spring Boot Version: 2.6.3
Frequently Used:
☐ Eureka Server ☐ MySQL Driver ☐ Spring Data JPA
☐ Spring Web
Available: eureka
Spring Cloud Discovery
☑ Eureka Discovery Client
☐ Eureka Server
VMware Tanzu Application Service
☐ Service Registry (TAS)
Selected:
Eureka Discovery Client

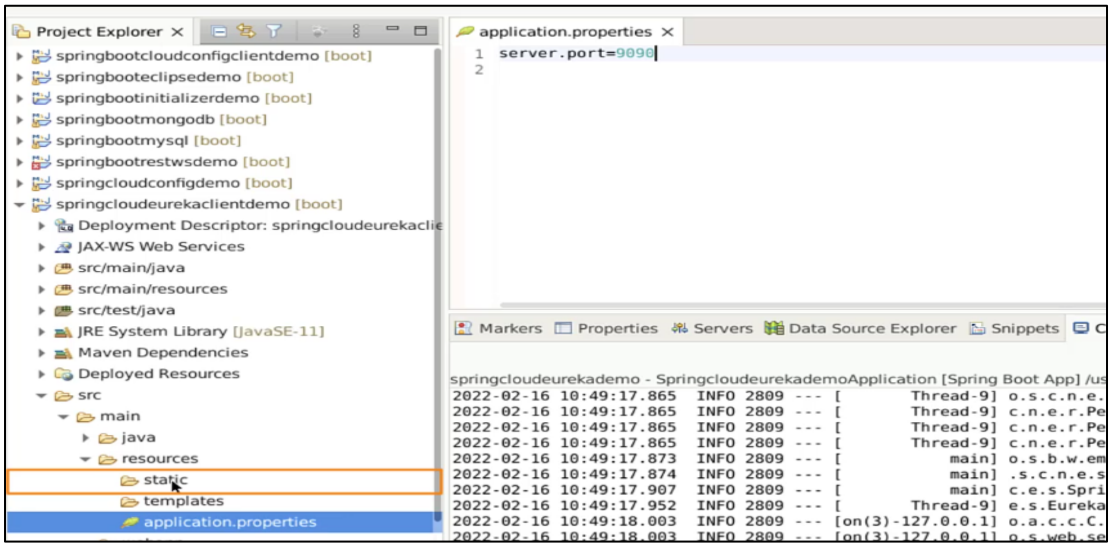
2.5 Navigate to the **springcloudeurekaclientdemo** project and open the file **application.properties**


Project Explorer
springbootcloudconfigclientdemo [boot]
springbooteclipsedemo [boot]
springbootinitializerdemo [boot]
springbootmongodb [boot]
springbootmysql [boot]
springbootrestwsdemo [boot]
springcloudconfigdemo [boot]
springcloudeurekaclientdemo [boot]
Deployment Descriptor: springcloudeurekaclie
JAX-WS Web Services
src/main/java
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
Deployed Resources
src
main
java
resources
static
templates
application.properties
webapp
SpringcloudeurekademoApplication.java
application.proper
Markers
Properties
Servers
Data Source Explorer

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

```
springcloudeurekademo - SpringcloudeurekademoApplication [Spring
2022-02-16 10:49:17.864  INFO 2809 --- [       Thread-9]
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9]
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9]
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9]
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9]
2022-02-16 10:49:17.873  INFO 2809 --- [           main]
2022-02-16 10:49:17.874  INFO 2809 --- [           main]
2022-02-16 10:49:17.907  INFO 2809 --- [           main]
2022-02-16 10:49:17.952  INFO 2809 --- [       Thread-9]
2022-02-16 10:49:18.003  INFO 2809 --- [on(3)-127.0.0.1]
2022-02-16 10:49:18.003  INFO 2809 --- [on(3)-127.0.0.1]
```

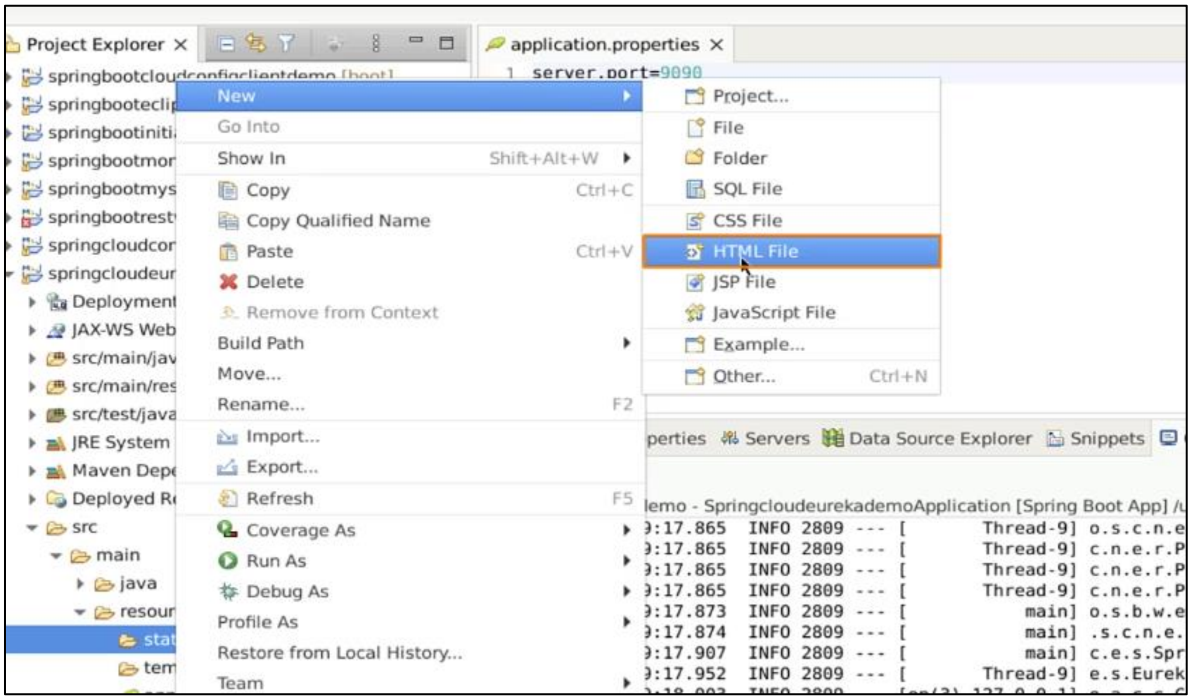
2.6 Add the property **server.port=9090** in the **application.properties** file



```
server.port=9090
```

```
springcloudeurekademo - SpringcloudeurekademoApplication [Spring Boot App] /us
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] o.s.c.n.e.
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.Pe
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.Pe
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.Pe
2022-02-16 10:49:17.873  INFO 2809 --- [           main] o.s.b.w.em
2022-02-16 10:49:17.874  INFO 2809 --- [           main] .s.c.n.e.s
2022-02-16 10:49:17.907  INFO 2809 --- [           main] c.e.s.Spri
2022-02-16 10:49:17.952  INFO 2809 --- [       Thread-9] e.s.Eureka
2022-02-16 10:49:18.003  INFO 2809 --- [on(3)-127.0.0.1] o.a.c.c.C.
2022-02-16 10:49:18.003  INFO 2809 --- [on(3)-127.0.0.1] o.s.web.se
```

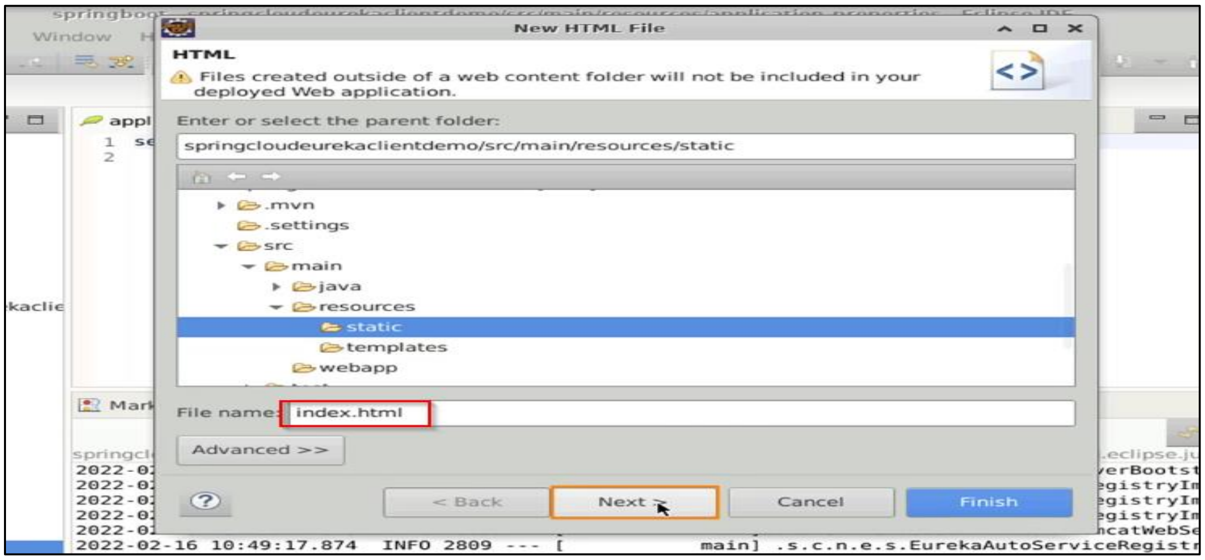
2.7 Navigate to a **static** resource and click **New > HTML File** to create an index page



```
server.port=9090
```

```
lemo - SpringcloudeurekademoApplication [Spring Boot App] /u
):17.865  INFO 2809 --- [        Thread-9] o.s.c.n.e
):17.865  INFO 2809 --- [        Thread-9] c.n.e.r.P
):17.865  INFO 2809 --- [        Thread-9] c.n.e.r.P
):17.865  INFO 2809 --- [        Thread-9] c.n.e.r.P
):17.873  INFO 2809 --- [            main] o.s.b.w.e
):17.874  INFO 2809 --- [            main] .s.c.n.e.
):17.907  INFO 2809 --- [            main] c.e.s.Spr
):17.952  INFO 2809 --- [        Thread-9] e.s.Eurek
```

2.8 Name the file **index.html** and click **Next > Finish**



Files created outside of a web content folder will not be included in your deployed Web application.

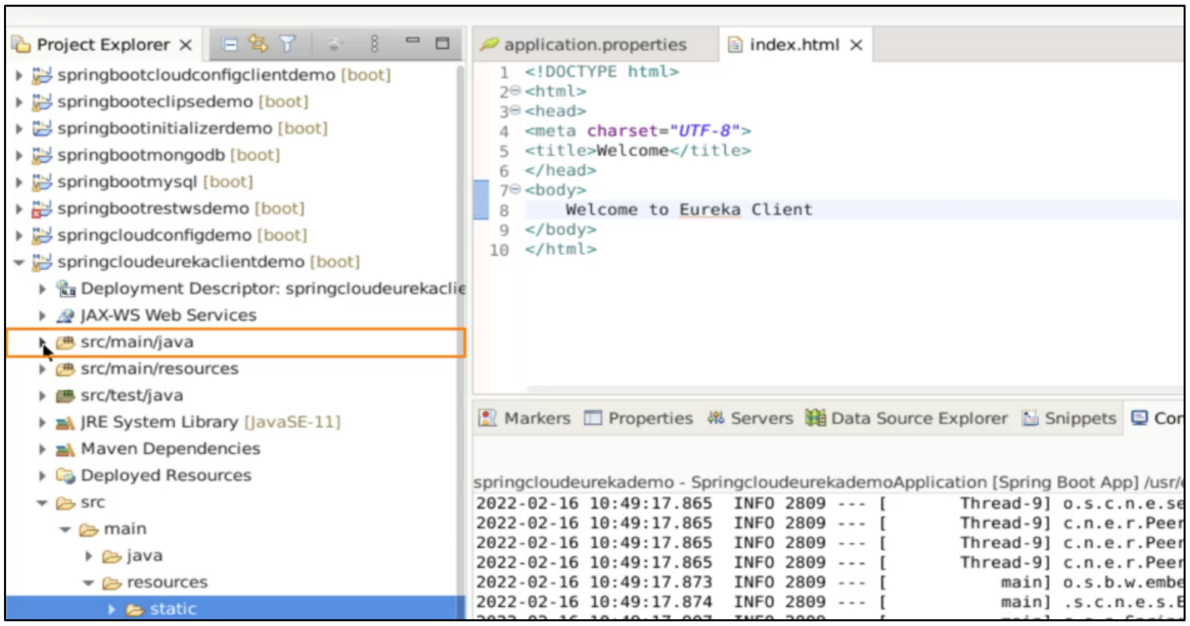
Enter or select the parent folder:

springcloudeurekaclientdemo/src/main/resources/static

File name: index.html

```
2022-02-16 10:49:17.874  INFO 2809 --- [           main] .s.c.n.e.s.EurekaAutoServiceRegistr
```

2.9 Inside the **<title>** tag, enter **Welcome**. Inside the **<body>** tag, enter **Welcome to Eureka Client**

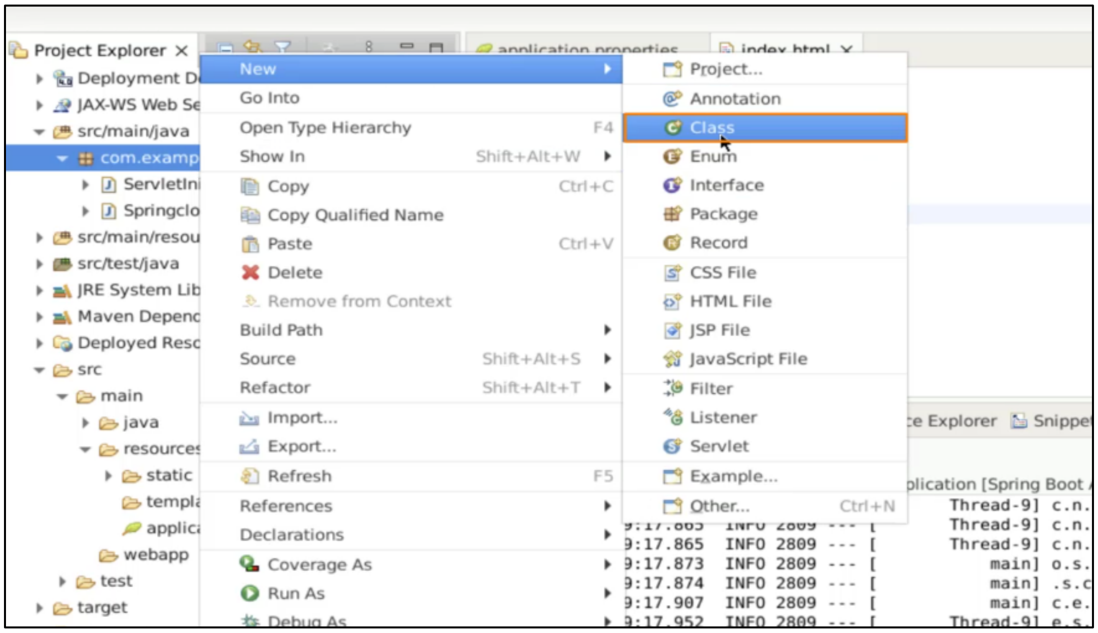


```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Welcome</title>
</head>
<body>
    Welcome to Eureka Client
</body>
</html>
```

```
springcloudeurekademo - SpringcloudeurekademoApplication [Spring Boot App] /usr/
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] o.s.c.n.e.se
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.Peer
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.Peer
2022-02-16 10:49:17.865  INFO 2809 --- [       Thread-9] c.n.e.r.Peer
2022-02-16 10:49:17.873  INFO 2809 --- [           main] o.s.b.w.embe
2022-02-16 10:49:17.874  INFO 2809 --- [           main] .s.c.n.e.s.E
```

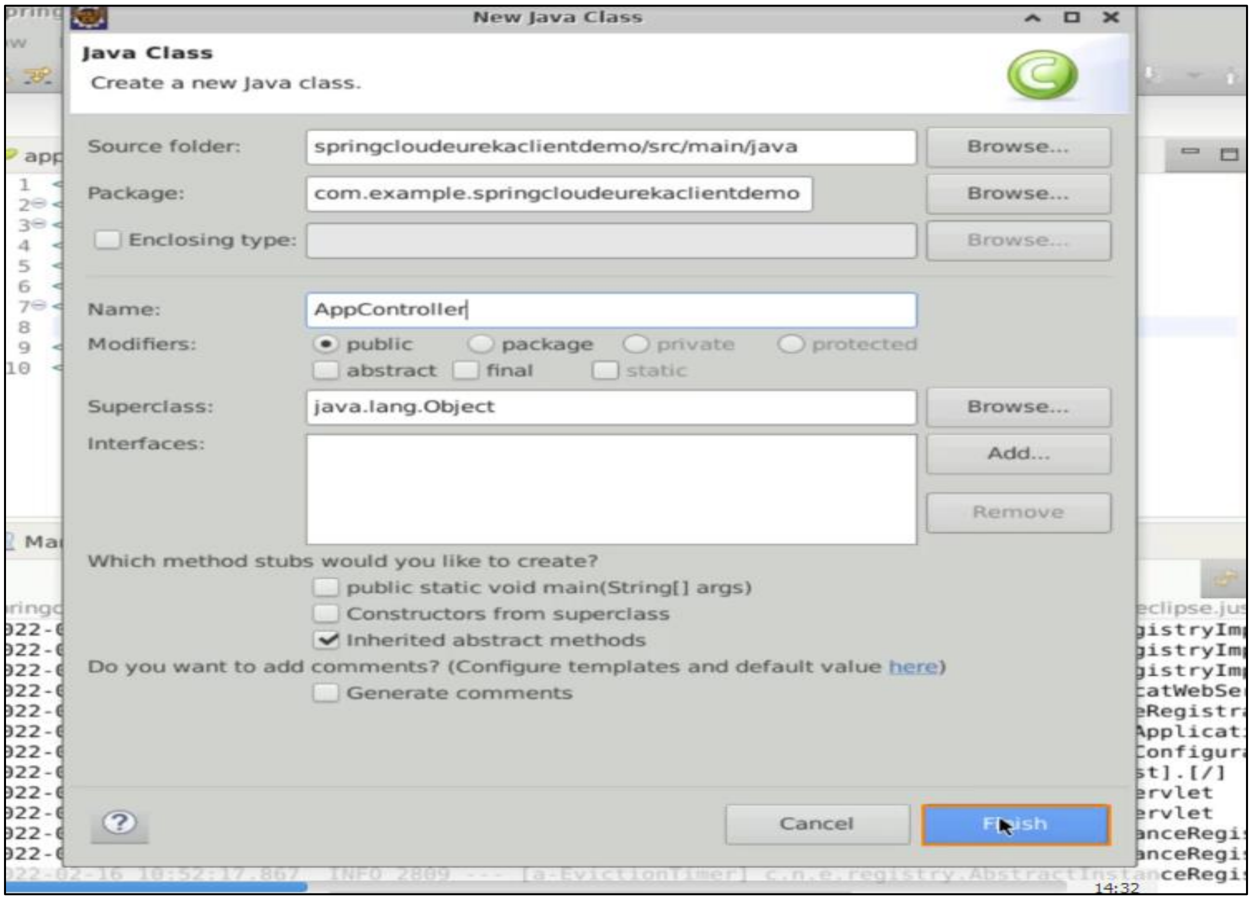
Step 3: Creating an AppController.java file

3.1 Navigate to **com.example.springcloudeurekaclientdemo** and right-click **New > Class** to create the **AppController**



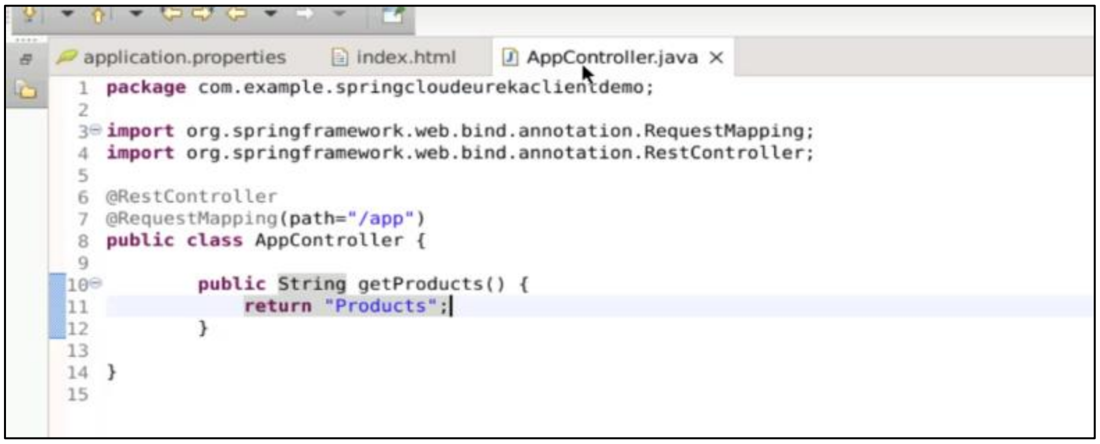
```
plication [Spring Boot
                                   Thread-9] c.n.
                                   Thread-9] c.n.
9:17.865  INFO 2809 --- [          Thread-9] c.n.
9:17.873  INFO 2809 --- [              main] o.s.
9:17.874  INFO 2809 --- [              main] .s.c
9:17.907  INFO 2809 --- [              main] c.e.
9:17.952  INFO 2809 --- [          Thread-9] e.s.
```

3.2 Name the class **AppController** and click **Finish**



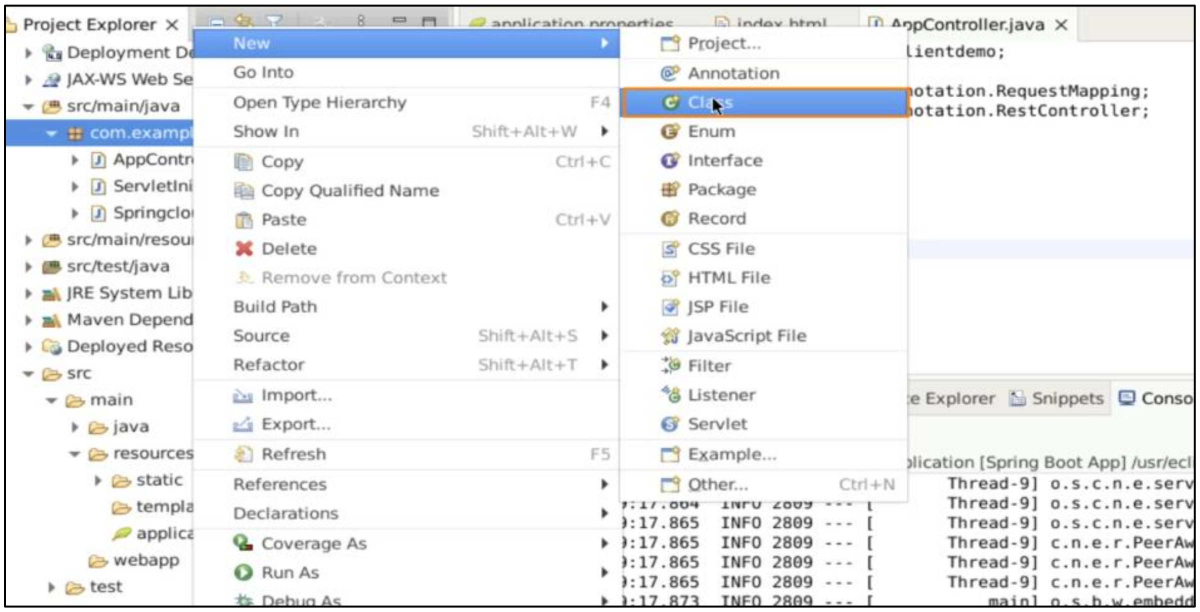
Source folder: springcloudeurekaclientdemo/src/main/java
Package: com.example.springcloudeurekaclientdemo
Enclosing type:
Name: AppController
Modifiers: [✓ public] package private protected
abstract final static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
[✓ Inherited abstract methods]
Do you want to add comments? (Configure templates and default value here)
Generate comments

3.3 Use the **@RequestMapping** annotation with the path **/app** to create a microservice in the **AppController.java** file

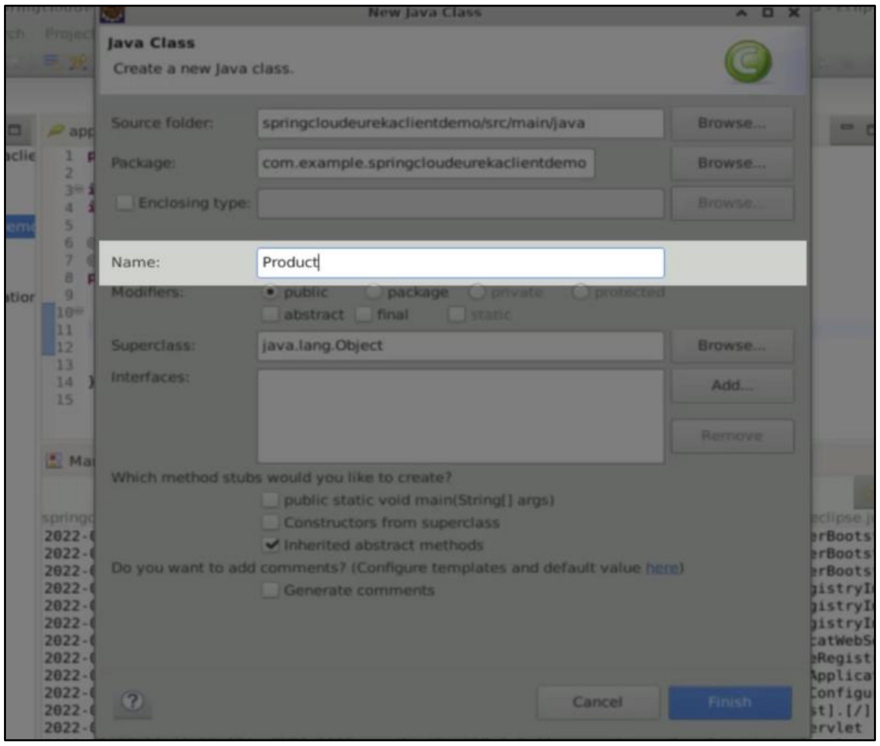


```java
package com.example.springcloudeurekaclientdemo;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(path="/app")
public class AppController {

        public String getProducts() {
            return "Products";
        }

}

```

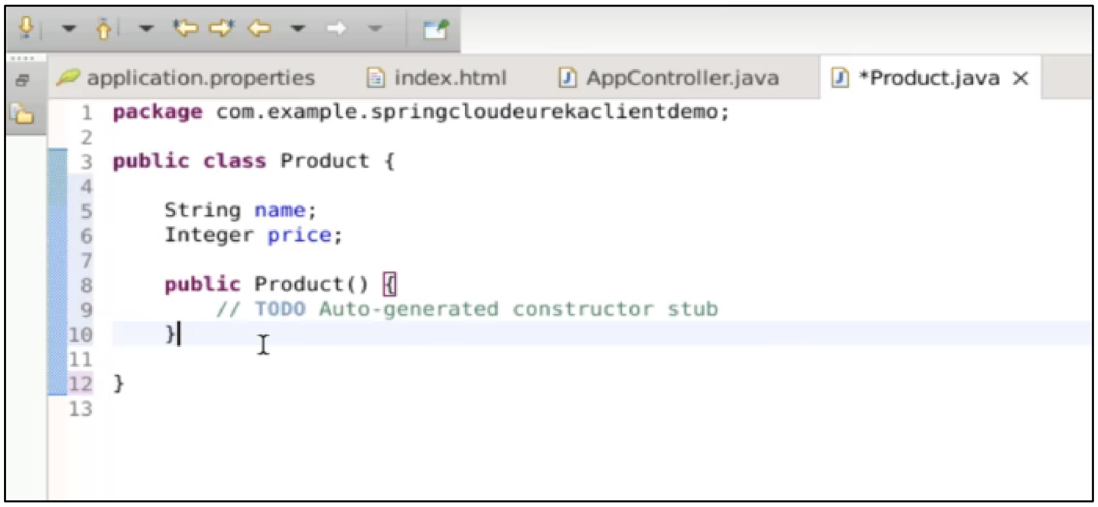
3.4 Navigate to the current project and right-click and select **New > Clas**s to create a new product class



3.5 Name it as **Product** and click **Finish**

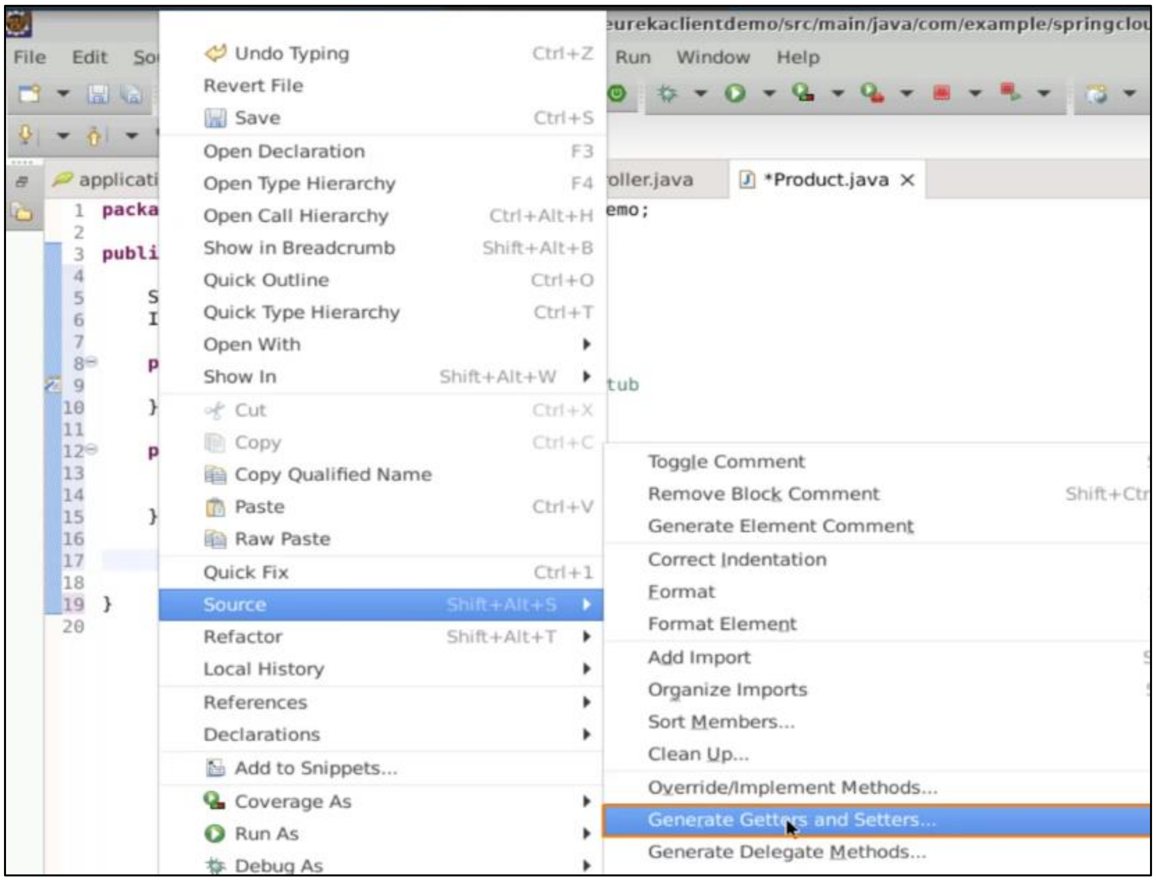

New Java Class
Java Class
Create a new Java class.
Source folder: springcloudeurekaclientdemo/src/main/java
Package: com.example.springcloudeurekaclientdemo
Enclosing type:
Name: Product
Modifiers: public package private protected
abstract final static
Superclass: java.lang.Object
Interfaces:
Browse...
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Cancel
Finish

3.6 Create two String variables, **name** and **price,** and create a default constructor for the class

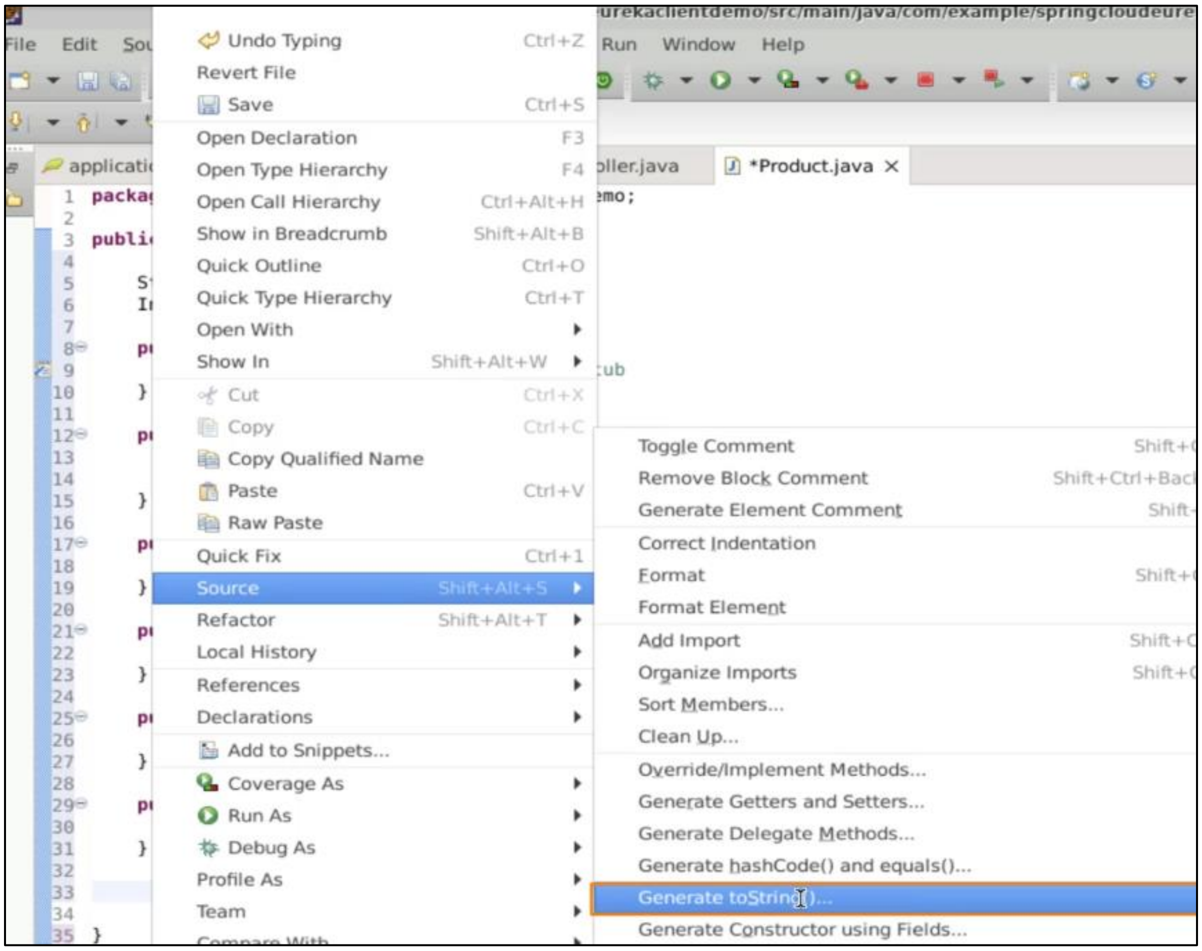


```
package com.example.springcloudeurekaclientdemo;

public class Product {

    String name;
    Integer price;

    public Product() {
        // TODO Auto-generated constructor stub
    }

}
```

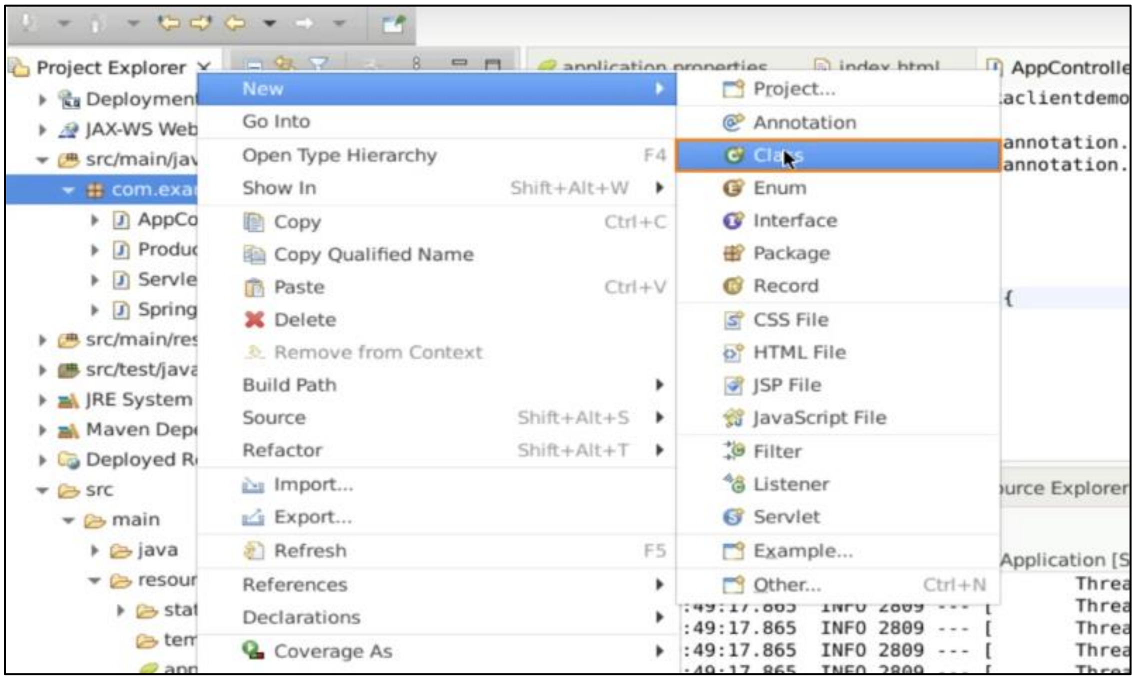
3.7 Right-click on the **Product.java** file and select **Source > Generate Getters and Setters**



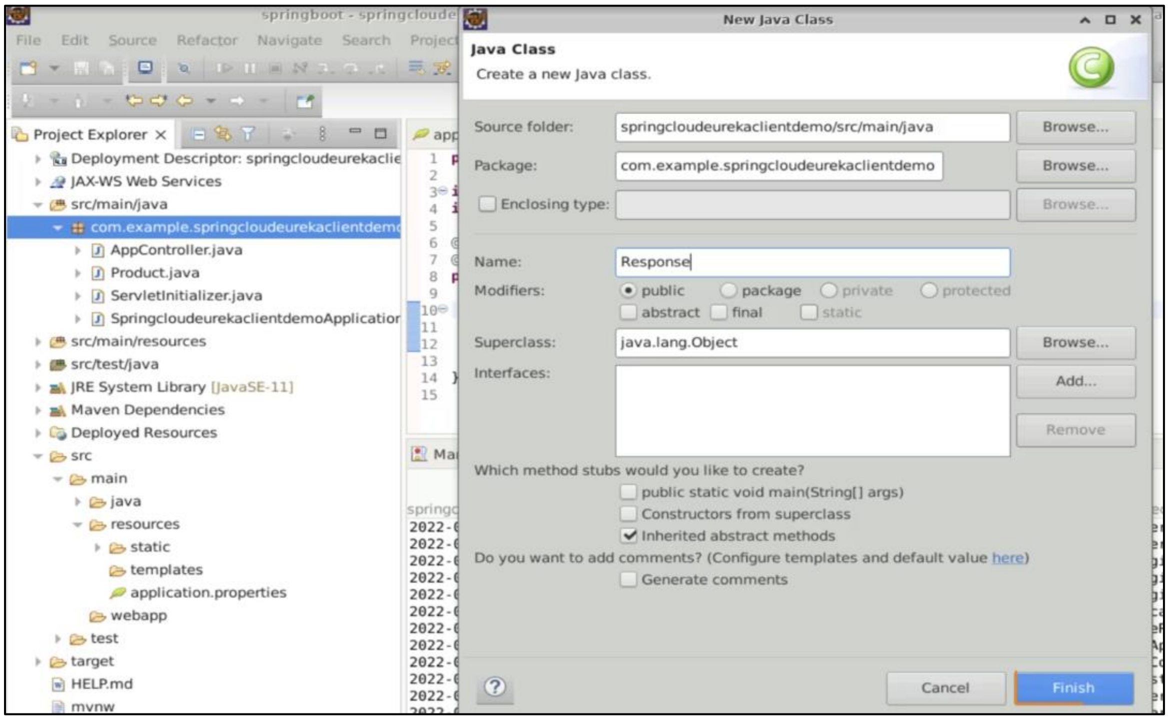
3.8 Click on **Source > Generate toString()**



3.9 Navigate to **com.example.springcloudeurekaclientdemo** and right-click and select **New > Class** to create a Response class

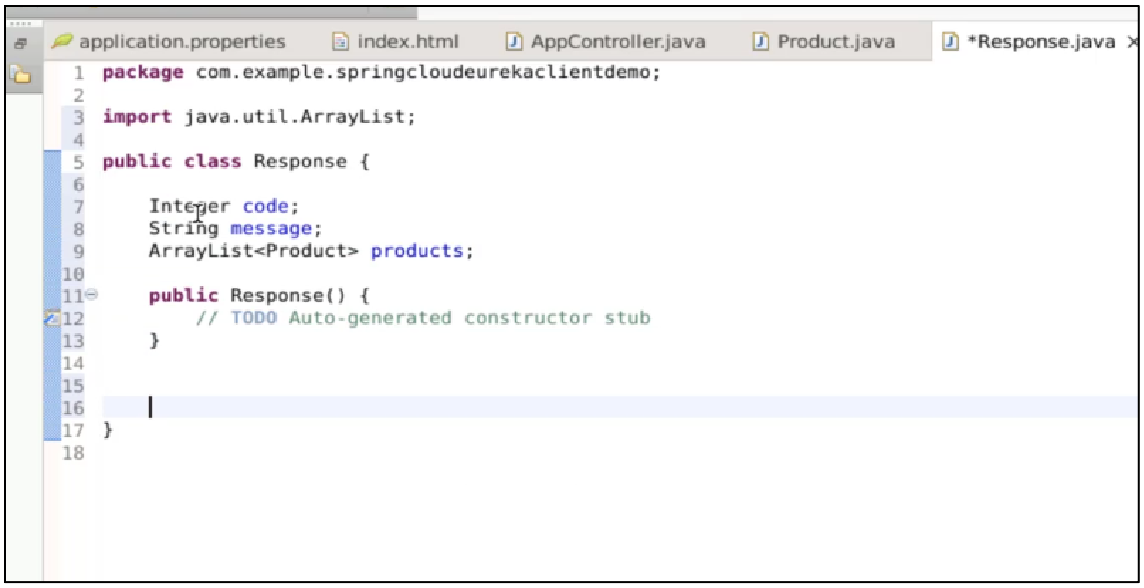


3.10 Name the class **Response** and click **Finish**



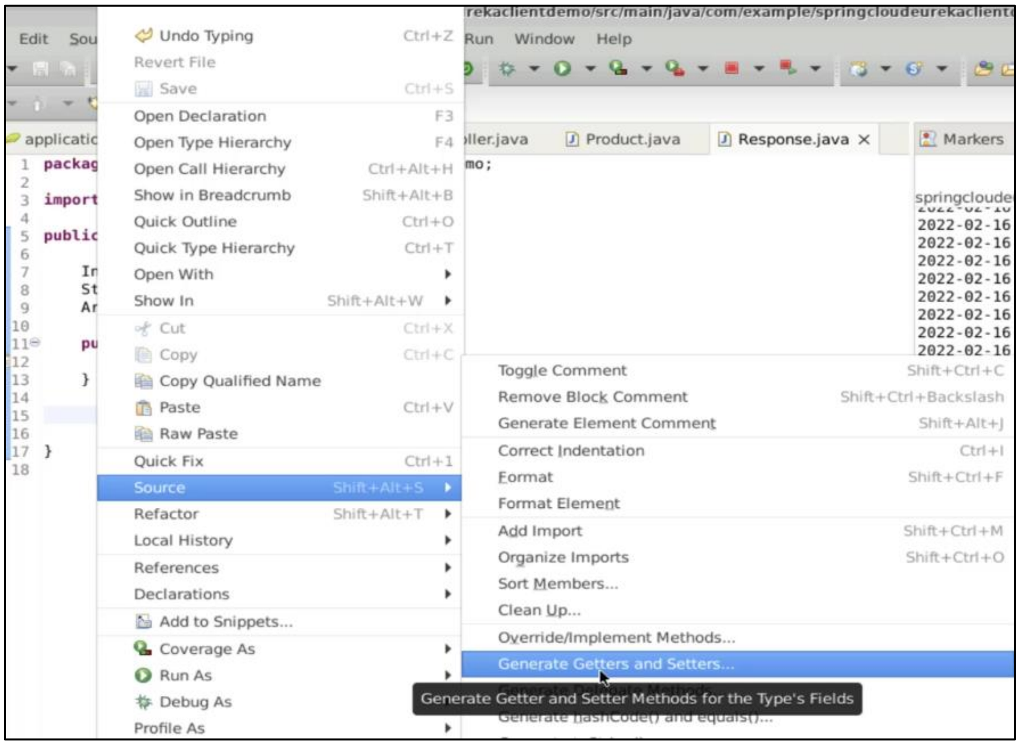
Source folder: springcloudeurekaclientdemo/src/main/java
Package: com.example.springcloudeurekaclientdemo
Enclosing type:
Name: Response
Modifiers: [✓ public] package private protected; abstract final static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
[✓ Inherited abstract methods]
Do you want to add comments? (Configure templates and default value here)
Generate comments

3.11 Create two variables, **code** and **message,** in the Response class and an **ArrayList<Product>** to store the products. Create constructors for the Response class

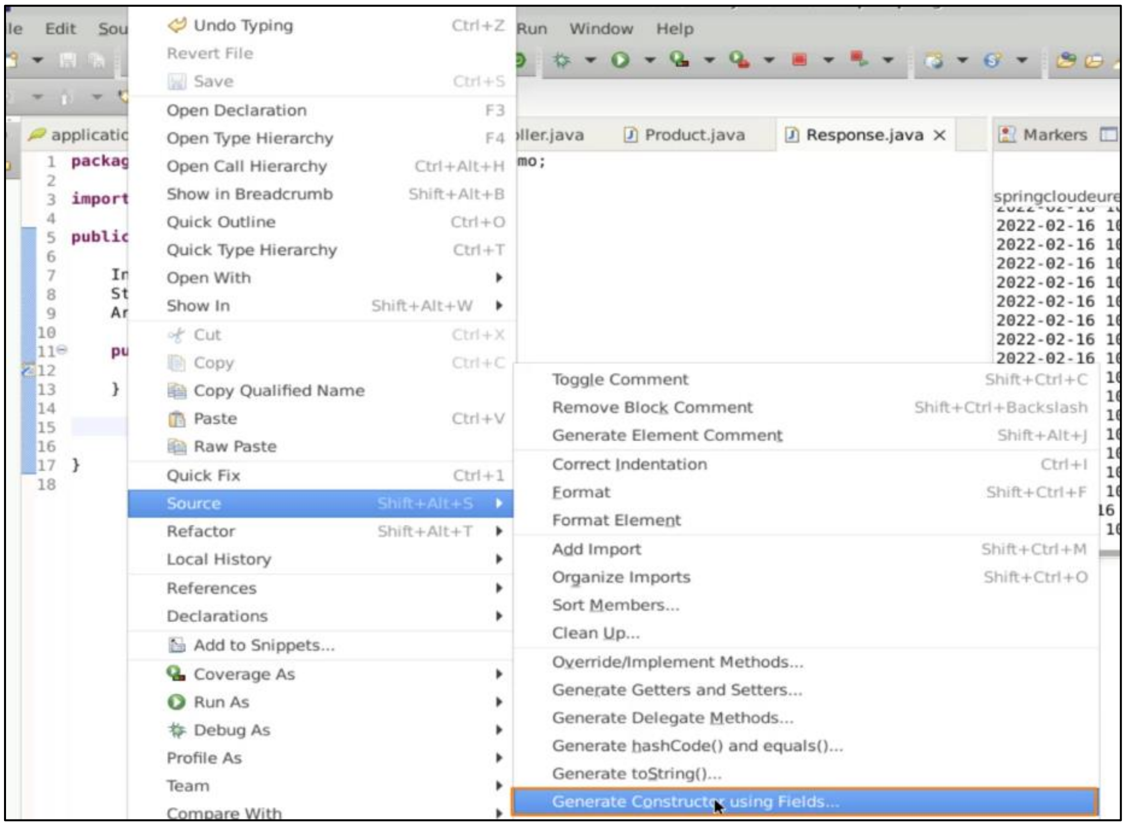


```java
package com.example.springcloudeurekaclientdemo;

import java.util.ArrayList;

public class Response {

    Integer code;
    String message;
    ArrayList<Product> products;

    public Response() {
        // TODO Auto-generated constructor stub
    }


    
}
```

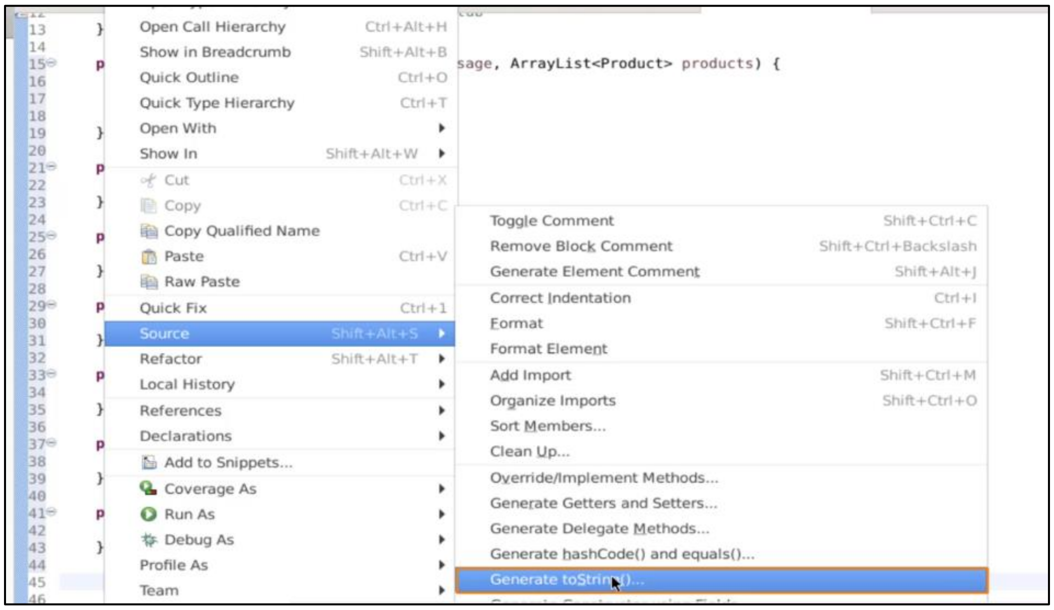
3.12 Right-click on the project and select **Source > Generate Getters and Setters**



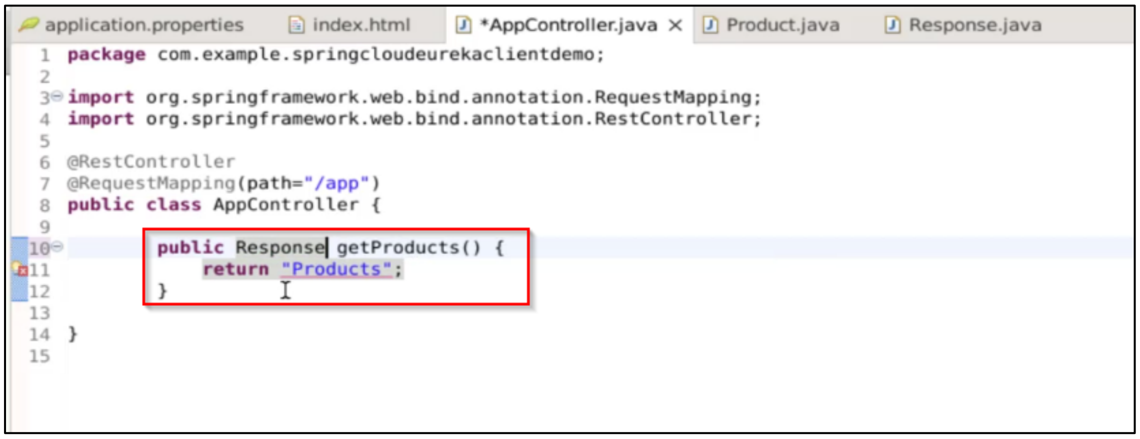
3.13 Right-click on the project and select **Source > Generate Constructor using fields**



3.14 Right-click on the project and select **Source > Generate toString()**


Open Call Hierarchy Ctrl+Alt+H
Show in Breadcrumb Shift+Alt+B
Quick Outline Ctrl+O
Quick Type Hierarchy Ctrl+T
Open With
Show In Shift+Alt+W
Cut Ctrl+X
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Raw Paste
Quick Fix Ctrl+1
Source Shift+Alt+S
Refactor Shift+Alt+T
Local History
References
Declarations
Add to Snippets...
Coverage As
Run As
Debug As
Profile As
Team
Toggle Comment Shift+Ctrl+C
Remove Block Comment Shift+Ctrl+Backslash
Generate Element Comment Shift+Alt+J
Correct Indentation Ctrl+I
Format Shift+Ctrl+F
Format Element
Add Import Shift+Ctrl+M
Organize Imports Shift+Ctrl+O
Sort Members...
Clean Up...
Override/Implement Methods...
Generate Getters and Setters...
Generate Delegate Methods...
Generate hashCode() and equals()...
Generate toString()...

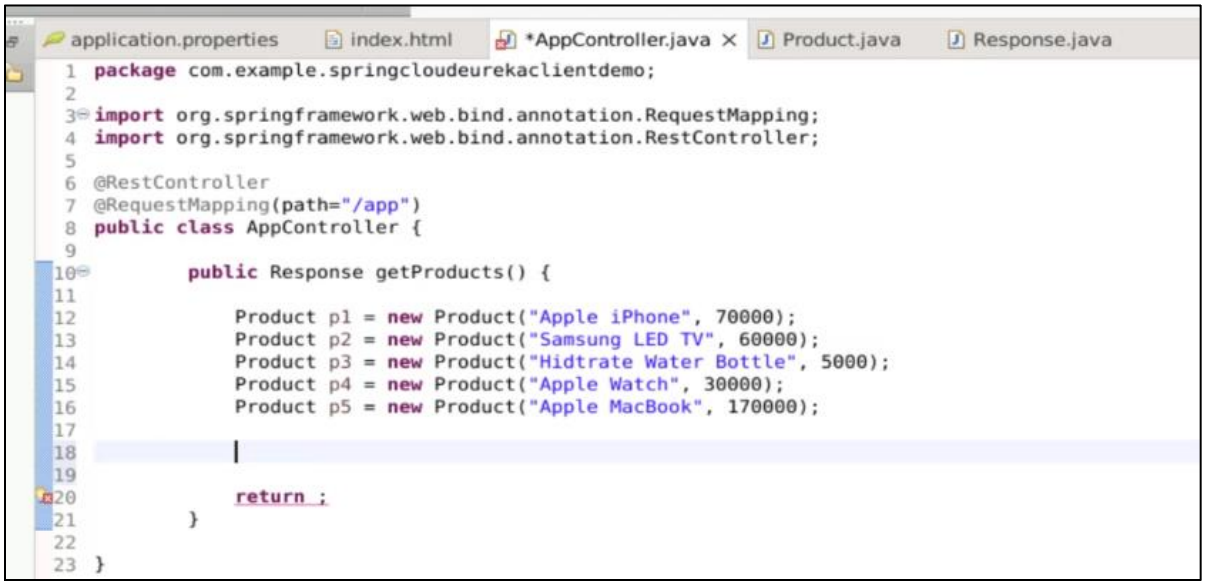
3.15 Change the return type of **getProducts()** in **AppController.java** from String to Response, as **getProducts()** should return a Response object



application.properties | index.html | *AppController.java | Product.java | Response.java

```java
package com.example.springcloudeurekaclientdemo;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(path="/app")
public class AppController {

        public Response getProducts() {
            return "Products";
        }

}
```

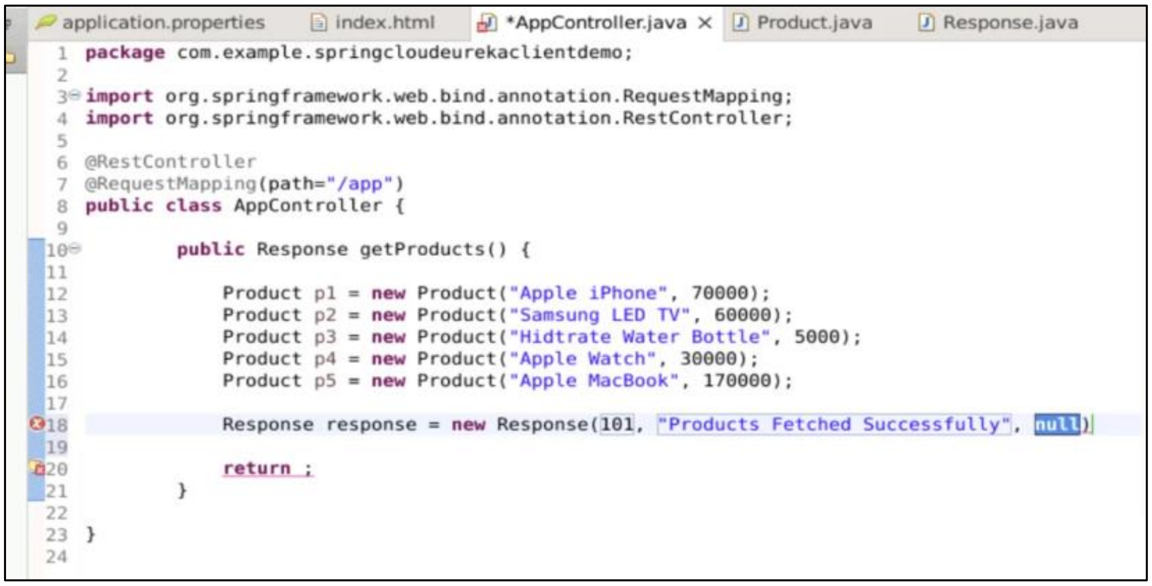
3.16 Create Product objects as **p1, p2, p3, p4, and p5** in the **getProducts** object



application.properties | index.html | *AppController.java | Product.java | Response.java

```java
package com.example.springcloudeurekaclientdemo;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(path="/app")
public class AppController {

        public Response getProducts() {

            Product p1 = new Product("Apple iPhone", 70000);
            Product p2 = new Product("Samsung LED TV", 60000);
            Product p3 = new Product("Hidtrate Water Bottle", 5000);
            Product p4 = new Product("Apple Watch", 30000);
            Product p5 = new Product("Apple MacBook", 170000);



            return ;
        }

}
```

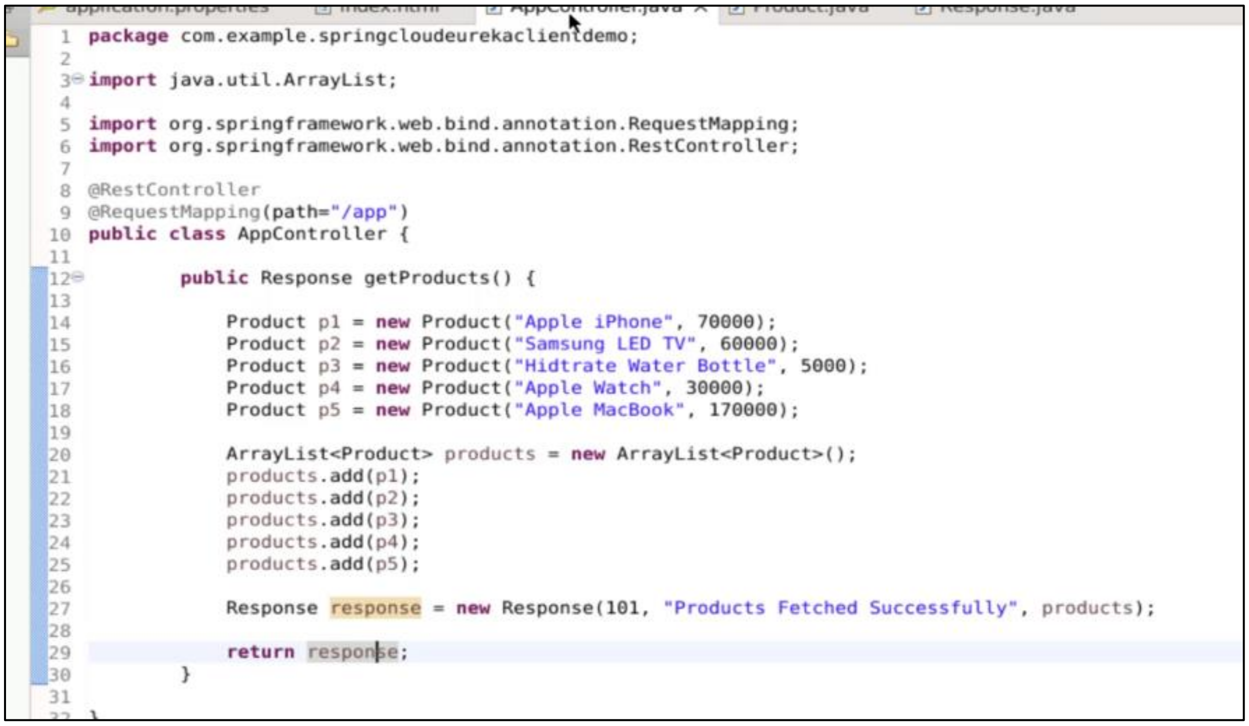
3.17 In the application, create a **Response object,** retrieve the product data, and use the **ArrayList<Product>** to store the data. Now, return the Response object



application.properties | index.html | *AppController.java | Product.java | Response.java

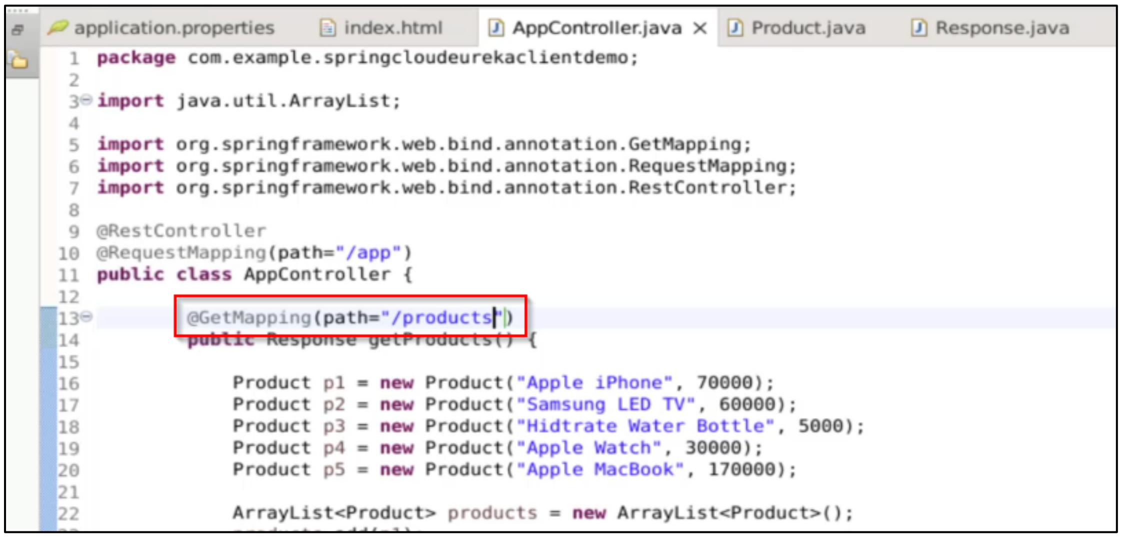
```java
package com.example.springcloudeurekaclientdemo;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(path="/app")
public class AppController {

        public Response getProducts() {

            Product p1 = new Product("Apple iPhone", 70000);
            Product p2 = new Product("Samsung LED TV", 60000);
            Product p3 = new Product("Hidtrate Water Bottle", 5000);
            Product p4 = new Product("Apple Watch", 30000);
            Product p5 = new Product("Apple MacBook", 170000);

            Response response = new Response(101, "Products Fetched Successfully", null)

            return ;
        }

}
```

3.18 Create an **ArrayList object** to store the product in one list and use **add** method to store the data and return the **response** object



```java
package com.example.springcloudeurekaclientdemo;

import java.util.ArrayList;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(path="/app")
public class AppController {

        public Response getProducts() {

            Product p1 = new Product("Apple iPhone", 70000);
            Product p2 = new Product("Samsung LED TV", 60000);
            Product p3 = new Product("Hidtrate Water Bottle", 5000);
            Product p4 = new Product("Apple Watch", 30000);
            Product p5 = new Product("Apple MacBook", 170000);

            ArrayList<Product> products = new ArrayList<Product>();
            products.add(p1);
            products.add(p2);
            products.add(p3);
            products.add(p4);
            products.add(p5);

            Response response = new Response(101, "Products Fetched Successfully", products);

            return response;
        }

}
```

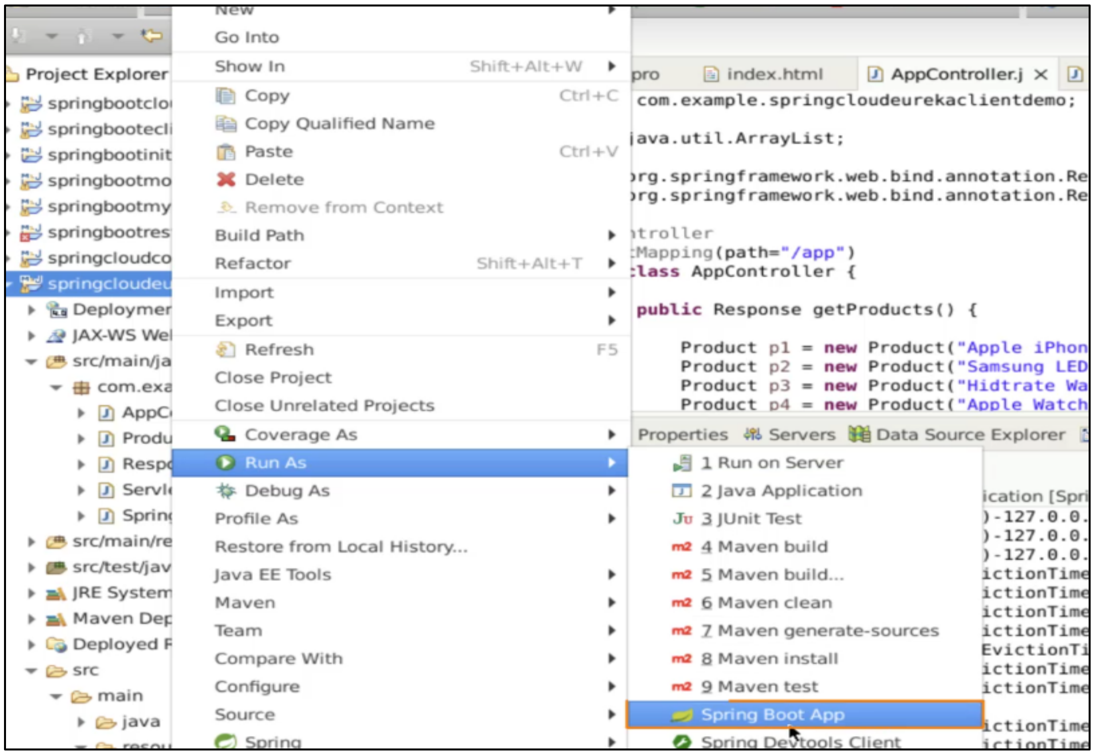
3.19 Add the **@GetMapping** annotation with the path **/products in AppController.java** to connect to the Eureka server



```
package com.example.springcloudeurekaclientdemo;

import java.util.ArrayList;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(path="/app")
public class AppController {

        @GetMapping(path="/products")
        public Response getProducts() {

            Product p1 = new Product("Apple iPhone", 70000);
            Product p2 = new Product("Samsung LED TV", 60000);
            Product p3 = new Product("Hidtrate Water Bottle", 5000);
            Product p4 = new Product("Apple Watch", 30000);
            Product p5 = new Product("Apple MacBook", 170000);

            ArrayList<Product> products = new ArrayList<Product>();
```

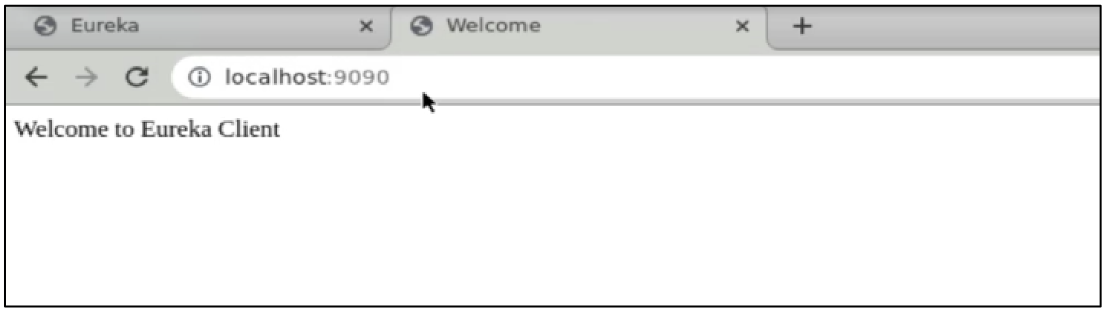
Step 4: Testing the application on Eureka Server

4.1 Navigate to the **springcloudeurekaclientdemo** project and right-click and select **Run As > Spring Boot App**



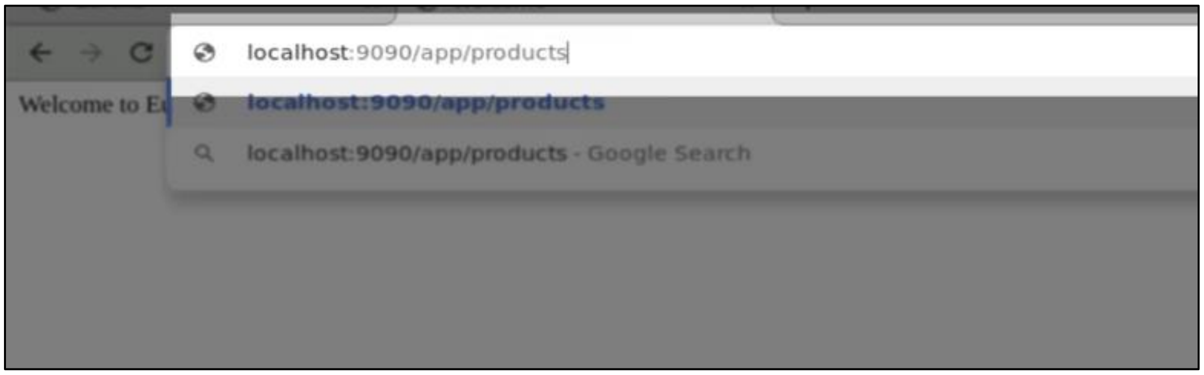
```
com.example.springcloudeurekaclientdemo;

java.util.ArrayList;

org.springframework.web.bind.annotation.Re
org.springframework.web.bind.annotation.Re

troller
Mapping(path="/app")
class AppController {

public Response getProducts() {

    Product p1 = new Product("Apple iPhon
    Product p2 = new Product("Samsung LED
    Product p3 = new Product("Hidtrate Wa
    Product p4 = new Product("Apple Watch
```

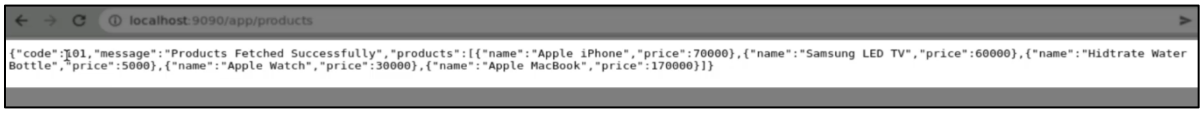
4.2 Open the **web browser** and enter **localhost:9090** to access the application



```
Welcome to Eureka Client
```

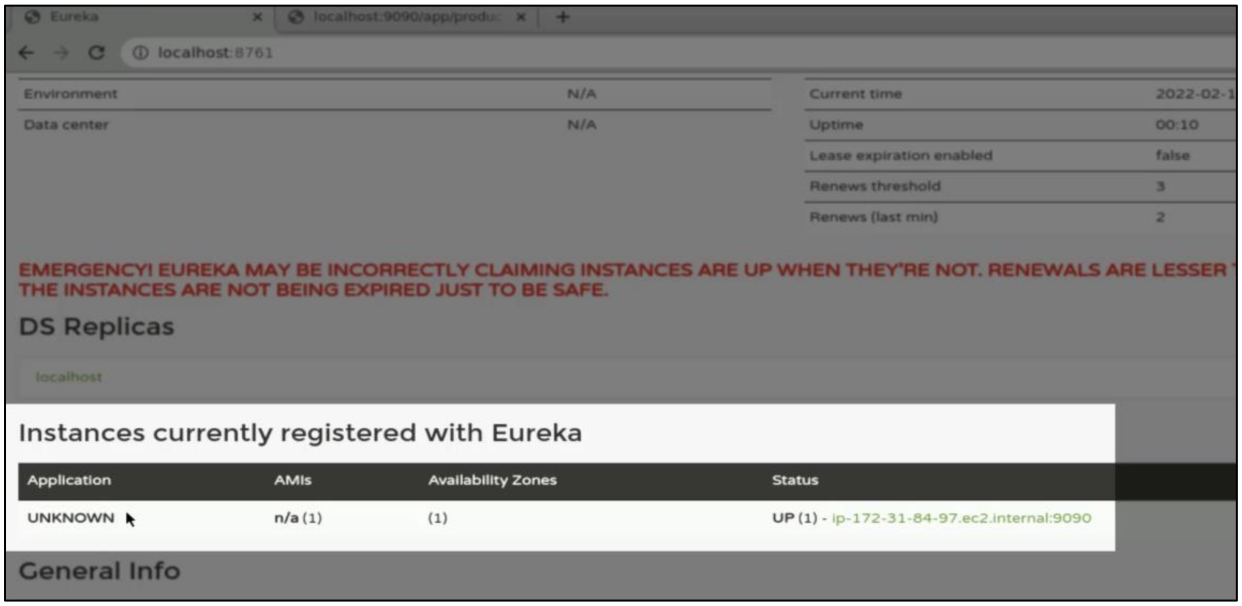
4.3 To fetch data from the **AppController.java file**, enter **localhost:9090/app/products** in the browser. The response will be in JSON format.





```
{"code":101,"message":"Products Fetched Successfully","products":[{"name":"Apple iPhone","price":70000},{"name":"Samsung LED TV","price":60000},{"name":"Hidtrate Water
Bottle","price":5000},{"name":"Apple Watch","price":30000},{"name":"Apple MacBook","price":170000}]}
```

4.4 Refresh the **Eureka Server**. You will see the service registered as **UNKNOWN** in the application database



Environment	N/A
Data center	N/A

Current time	2022-02-1
Uptime	00:10
Lease expiration enabled	false
Renews threshold	3
Renews (last min)	2

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

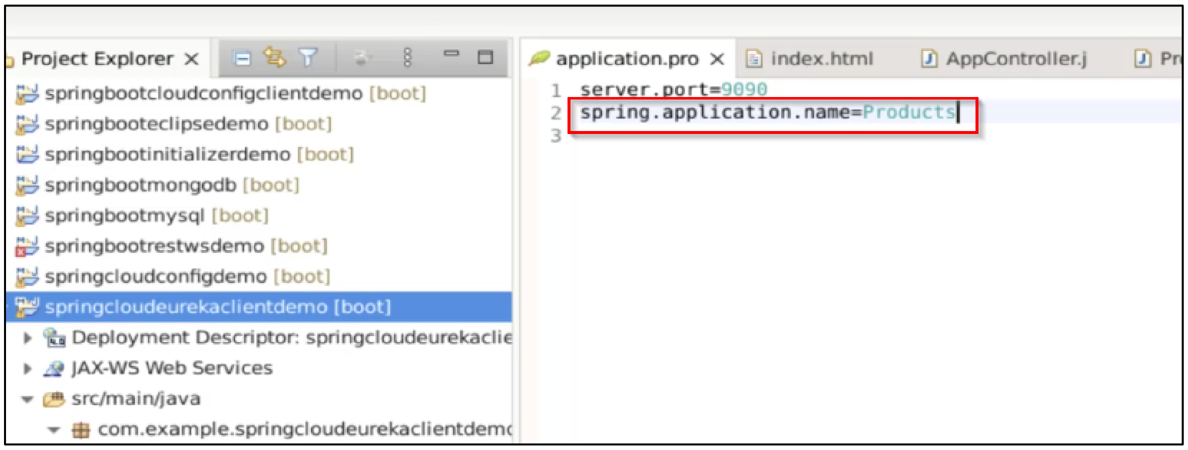
localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
UNKNOWN	n/a (1)	(1)	UP (1) - ip-172-31-84-97.ec2.internal:9090

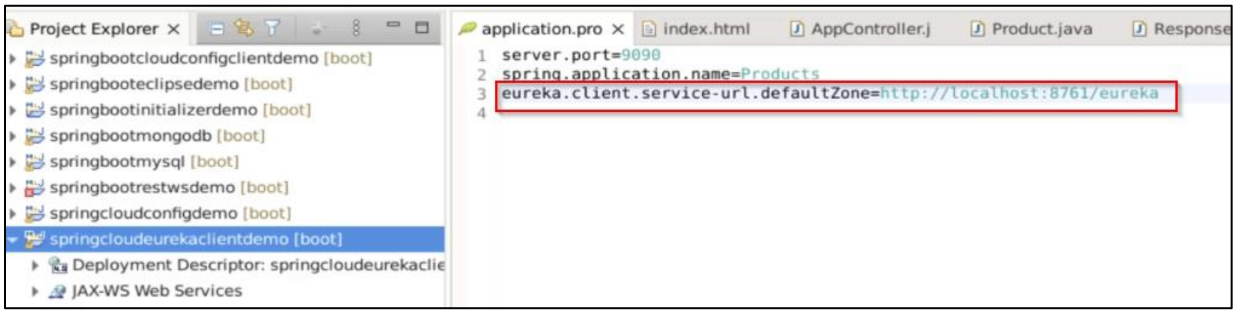
General Info

4.5 To configure the application name, add **spring.application.name=Products** in the **application.properties** file



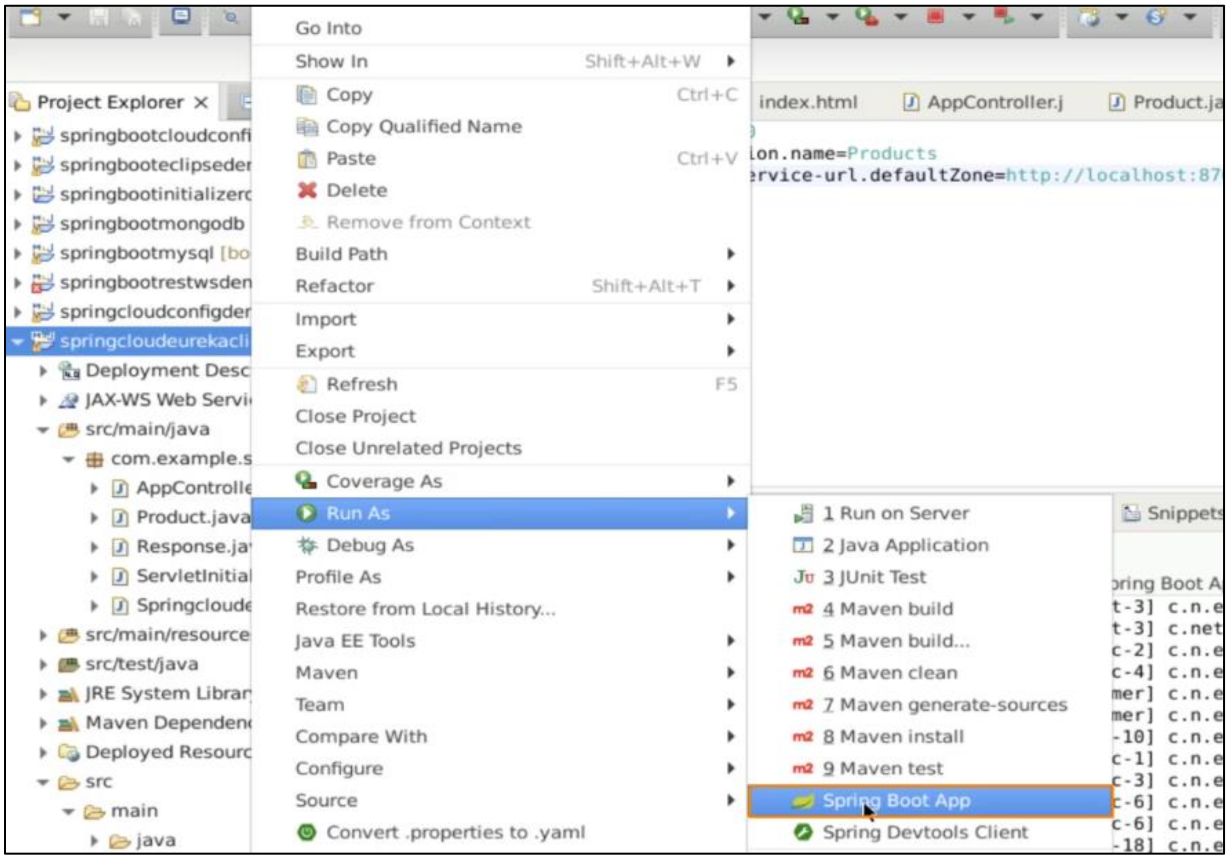
```
server.port=9090
spring.application.name=Products
```

4.6 Set the Eureka service URL as http://localhost:8761/eureka

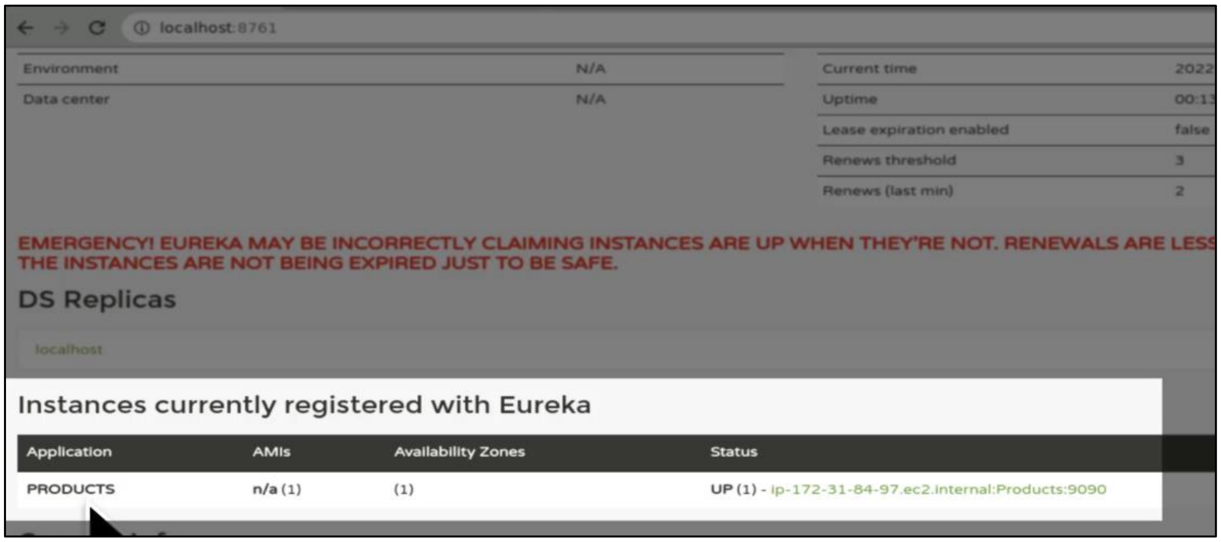

Project Explorer
springbootcloudconfigclientdemo [boot]
springbooteclipsedemo [boot]
springbootinitializerdemo [boot]
springbootmongodb [boot]
springbootmysql [boot]
springbootrestwsdemo [boot]
springcloudconfigdemo [boot]
springcloudeurekaclientdemo [boot]
Deployment Descriptor: springcloudeurekaclie
JAX-WS Web Services
application.pro
index.html
AppController.j
Product.java
Response

```
server.port=9090
spring.application.name=Products
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

4.7 Run the **springcloudeurekaclientdemo** project by right-clicking and selecting **Run As > Spring Boot App**



4.8 Refresh the browser and go to **localhost:8761**. The application will now be listed as **PRODUCTS** in the Eureka Server's application database.



localhost:8761

Environment	N/A
Data center	N/A

Current time	2022
Uptime	00:1
Lease expiration enabled	false
Renews threshold	3
Renews (last min)	2

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESS
THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
PRODUCTS	n/a (1)	(1)	UP (1) - ip-172-31-84-97.ec2.internal:Products:9090

The demo provides a step-by-step guide on how to configure the Eureka Server, create a Spring Starter Project, implement an **AppController.java** file, and test the application on the Eureka Server.