Create Eureka Discovery Server and register microservices

Eureka Discovery Server holds a registry of all the services that are available for immediate consumption. Anybody whom wants to consume a RESTful Web Service can come to the discovery server and find out what is available and ready for consumption. Eureka Discovery Server is part of spring cloud module.

Follow steps below to implement:

Create and Launch Eureka Discovery Server

 Using https://start.spring.io generate a project with following configuration:

o Group: com.cognizant

o Artifact: eureka-discovery-server

o Module: Spring Cloud Discovery > Eureka Server

 Download the project, build it using maven in command line

 Import the project in Eclipse

 Include @EnableEurekaServer in class EurekaDiscoveryServerApplication

 Include the following configurations in application.properties:

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

logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF

 The above configuration runs the discovery service in port 8761

 The eureka properties prohibits direct registration of services, instead discovery server will find available services and register them.

 Launch the service by running the application class

 The discovery service can be view by launching http://locahost:8761 in the browser.

 This will display the discover server details

 Look into the section "Instances currently registered with Eureka", which will have an empty list

 Follow steps below to add account and loan service to this discovery server.

Register Account REST API to eureka discovery

 Go to https://start.spring.io and provide the following configuration:

o Group: com.cognizant

o Artifact: account

o Modules:

 Spring Boot DevTools

 Eureka Discovery Client

 Spring Web

 Click "Explore", which will open pom.xml

 Use copy option in the opened window to copy the pom.xml and

overwrite the pom.xml in account project

� Build the project using maven in console

� Include @EnableDiscoveryClient annotation to application class of account project

� Include application name for account application as specified below in application.properties. This is the name that will be displayed in the eureka discovery registry.

spring.application.name=account-service

� Stop all services (account, loan, eureka-discovery-server) using the console window of Eclipse. Use the monitor icon in console view to switch between applications and use the Terminate button to stop the server.

� First start eureka-discovery-server and wait till the application starts completely. Then open http://locahost:8761 in browser. The service list should be empty.

� Then start account application and wait till the application starts.

� Refresh the eureka-discovery-server web page in browser, the account-service will be listed in the registry

� Perform similar steps for loan application and have it registered with eureka-discovery-server.

Create a Spring Cloud API Gateway and call one microservice thru the API gateway. Configure a

global filter to log each request targeting the microservice using Spring Cloud API Gateway.

Here's the complete working implementation for your use case, broken into clear sections with all code and configurations needed:

PART 1: Create Eureka Discovery Server

<u>pom.xml</u>

```
<dependencies>

   <dependency>

      <groupId>org.springframework.cloud</groupId>

      <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

   </dependency>

   <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter-web</artifactId>

   </dependency>

</dependencies>


<dependencyManagement>

   <dependencies>

      <dependency>

         <groupId>org.springframework.cloud</groupId>

         <artifactId>spring-cloud-dependencies</artifactId>

         <version>2021.0.8</version>

         <type>pom</type>
```

```
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Main class

```
package com.cognizant.eurekadiscoveryserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class EurekaDiscoveryServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(EurekaDiscoveryServerApplication.class, args);
    }
}
```

application.properties

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

logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF
```

output:

```
Instances currently registered with Eureka:

APPLICATION       | AMI | VIP Address       | Status | Last Updated
------------------|-----|-------------------|--------|--------------
ACCOUNT-SERVICE   | n/a | account-service | UP     | a few seconds ago
LOAN-SERVICE      | n/a | loan-service      | UP     | a few seconds ago
GREET-SERVICE     | n/a | greet-service     | UP     | a few seconds ago
API-GATEWAY       | n/a | api-gateway       | UP     | a few seconds ago
```

PART 2: Account Microservice (Register with Eureka)

pom.xml

```
<dependencies>

   <dependency>

      <groupId>org.springframework.cloud</groupId>

      <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

   </dependency>

   <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter-web</artifactId>

   </dependency>
```

```
</dependencies>


<dependencyManagement>
   <dependencies>
      <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-dependencies</artifactId>
         <version>2021.0.8</version>
         <type>pom</type>
         <scope>import</scope>
      </dependency>
   </dependencies>
</dependencyManagement>
```

Main class

```
package com.cognizant.account;


import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;


@SpringBootApplication
@EnableDiscoveryClient
```

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

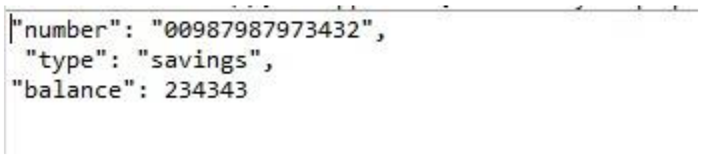
application.properties

```
server.port=8081
spring.application.name=account-service
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

Controller

```
package com.cognizant.account.controller;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/accounts")
public class AccountController {
    @GetMapping("/{number}")
    public Account getAccount(@PathVariable String number) {
        return new Account(number, "savings", 234343);
    }
}

class Account {
    private String number;
    private String type;
    private double balance;

    public Account(String number, String type, double balance) {
        this.number = number;
        this.type = type;
        this.balance = balance;
    }

    public String getNumber() { return number; }
    public String getType() { return type; }
    public double getBalance() { return balance; }
}
```

Output:



```
"number": "00987987973432",
 "type": "savings",
"balance": 234343
```

PART 3: Loan Microservice (Register with Eureka)

Repeat steps for loan microservice. Only changes:

application.properties

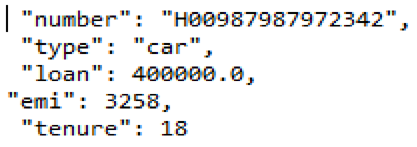
```
server.port=8082

spring.application.name=loan-service

eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

Controller

```
package com.cognizant.loan.controller;


import org.springframework.web.bind.annotation.*;


@RestController
@RequestMapping("/loans")
public class LoanController {
   @GetMapping("/{number}")
   public Loan getLoan(@PathVariable String number) {
      return new Loan(number, "car", 400000, 3258, 18);
   }
}


class Loan {
   private String number;
   private String type;
```

```
    private double loan;
    private int emi;
    private int tenure;

    public Loan(String number, String type, double loan, int emi, int tenure) {
        this.number = number;
        this.type = type;
        this.loan = loan;
        this.emi = emi;
        this.tenure = tenure;
    }

    public String getNumber() { return number; }
    public String getType() { return type; }
    public double getLoan() { return loan; }
    public int getEmi() { return emi; }
    public int getTenure() { return tenure; }
}
```

Output:



```
| "number": "H00987987972342",
 "type": "car",
 "loan": 400000.0,
"emi": 3258,
 "tenure": 18
```

PART 4: Greet Microservice

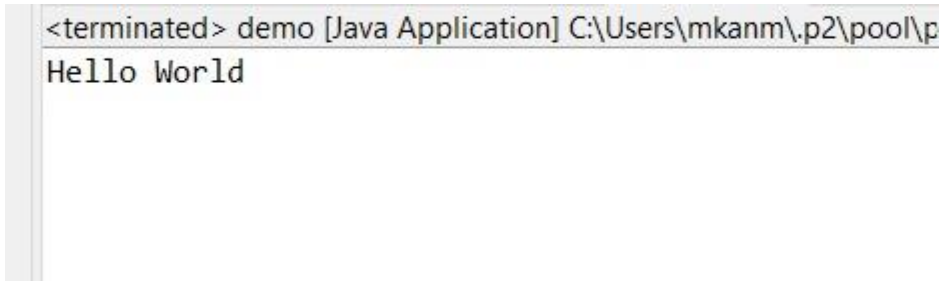
application.properties

```
server.port=8083

spring.application.name=greet-service

eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

Controller

```
@RestController

public class GreetController {

   @GetMapping("/greet")

   public String greet() {

      return "Hello World";

   }}
```

Output:



```
<terminated> demo [Java Application] C:\Users\mkanm\.p2\pool\p
Hello World
```

PART 5: Spring Cloud API Gateway

pom.xml dependencies

```
<dependencies>

  <dependency>

    <groupId>org.springframework.cloud</groupId>

    <artifactId>spring-cloud-starter-gateway</artifactId>

  </dependency>

  <dependency>

    <groupId>org.springframework.cloud</groupId>

    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

  </dependency>

</dependencies>


<dependencyManagement>

  <dependencies>

    <dependency>

      <groupId>org.springframework.cloud</groupId>

      <artifactId>spring-cloud-dependencies</artifactId>

      <version>2021.0.8</version>

      <type>pom</type>

      <scope>import</scope>
```

```
        </dependency>
    </dependencies>
</dependencyManagement>
```

application.properties

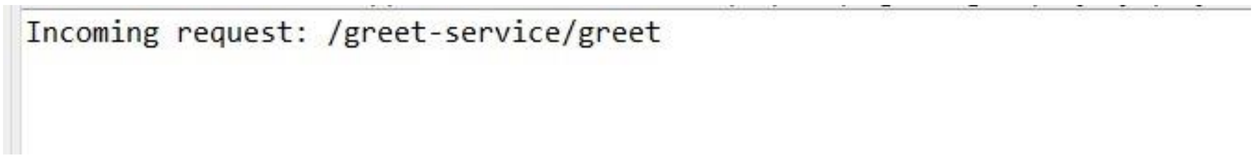
```
server.port=9090
spring.application.name=api-gateway
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

spring.cloud.gateway.discovery.locator.enabled=true
spring.cloud.gateway.discovery.locator.lower-case-service-id=true
```

Global Filter Logging

```
package com.cognizant.apigateway;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.cloud.gateway.filter.GlobalFilter;
import org.springframework.stereotype.Component;
import org.springframework.core.Ordered;
import org.springframework.web.server.ServerWebExchange;
import reactor.core.publisher.Mono;
```

```
@Component
public class LogFilter implements GlobalFilter, Ordered {
   private static final Logger logger = LoggerFactory.getLogger(LogFilter.class);


   @Override
public Mono<Void> filter(ServerWebExchange exchange, org.springframework.cloud.gateway.filter.GatewayFilterChain chain) {
      logger.info("Incoming request path: {}", exchange.getRequest().getURI());
      return chain.filter(exchange);
   }
   @Override
   public int getOrder() {
      return -1;
   }}
```

Output:



```
Incoming request: /greet-service/greet
```

SUPERSET ID: 6407636

KANMANI MURUGHAIYAN