**Creating Microservices for account and loan**

**1. Account Microservice**

**AccountController.java:**

package com.cognizant.account.controller;

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

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

public Account getAccountDetails(@PathVariable String number) {

return new Account(number, "savings", 234343);

}

public static class Account {

private String number;

private String type;

private int balance;

public Account(String number, String type, int balance) {

this.number = number;

this.type = type;

this.balance = balance;

}

public String getNumber() {

return number;

}

public String getType() {

return type;

}

public int getBalance() {

return balance;

}

}

}

**AccountApplication.java (Main Class):**

package com.cognizant.account;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AccountApplication {

public static void main(String[] args) {

SpringApplication.run(AccountApplication.class, args);

}

}

### Run & Test:

Run from Eclipse or mvn spring-boot:run

Test: <http://localhost:8080/accounts/00987987973432>

**Output:**

json

CopyEdit

{

"number": "00987987973432",

"type": "savings",

"balance": 234343

}

**2. Loan Microservice**

### LoanController.java:

java

CopyEdit

package com.cognizant.loan.controller;

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

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

public Loan getLoanDetails(@PathVariable String number) {

return new Loan(number, "car", 400000, 3258, 18);

}

public static class Loan {

private String number;

private String type;

private int loan;

private int emi;

private int tenure;

public Loan(String number, String type, int 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 int getLoan() {

return loan;

}

public int getEmi() {

return emi;

}

public int getTenure() {

return tenure;

}

}

}

### application.properties:

properties

CopyEdit

server.port=8081

**LoanApplication.java (Main Class):**

java

CopyEdit

package com.cognizant.loan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LoanApplication {

public static void main(String[] args) {

SpringApplication.run(LoanApplication.class, args);

}

}

### Run & Test:

Run from Eclipse or mvn spring-boot:run

Test: <http://localhost:8081/loans/H00987987972342>

### Output:

json

CopyEdit

{

"number": "H00987987972342",

"type": "car",

"loan": 400000,

"emi": 3258,

"tenure": 18

}

**Create Eureka Discovery Server and register microservices**

**1. Eureka Discovery Server**

### pom.xml – Add Spring Cloud BOM (if not already present):

xml

CopyEdit

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>2022.0.4</version> <!-- or latest compatible -->

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

### Main Class: EurekaDiscoveryServerApplication.java

java

CopyEdit

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

properties

CopyEdit

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

### Run & View

Run via Eclipse or mvn spring-boot:run

Open: <http://localhost:8761>

### Output:

sql

CopyEdit

"Instances currently registered with Eureka" — Empty list