WEEK 5 :  
Skill : Microservices with spring boot 3 and Spring Cloud

Hands-on : Creating Microservices for account and loan

// === 1. Eureka Server (Service Registry) ===

// File: EurekaServerApplication.java

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

// application.yml

server:

port: 8761

eureka:

client:

register-with-eureka: false

fetch-registry: false

server:

wait-time-in-ms-when-sync-empty: 0

---

// === 2. Config Server ===

// File: ConfigServerApplication.java

@SpringBootApplication

@EnableConfigServer

public class ConfigServerApplication {

public static void main(String[] args) {

SpringApplication.run(ConfigServerApplication.class, args);

}

}

// application.yml

server:

port: 8888

spring:

cloud:

config:

server:

git:

uri: https://github.com/your-user/your-config-repo

application:

name: config-server

---

// === 3. Account Service ===

// File: AccountServiceApplication.java

@SpringBootApplication

@EnableDiscoveryClient

public class AccountServiceApplication {

public static void main(String[] args) {

SpringApplication.run(AccountServiceApplication.class, args);

}

}

// application.yml

spring:

application:

name: account-service

datasource:

url: jdbc:mysql://localhost:3306/accounts

username: root

password: root

config:

import: optional:configserver:http://localhost:8888

server:

port: 8081

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

// AccountController.java

@RestController

@RequestMapping("/accounts")

public class AccountController {

@Autowired

private AccountRepository accountRepository;

@PostMapping

public Account create(@RequestBody Account account) {

return accountRepository.save(account);

}

@GetMapping("/{id}")

public ResponseEntity<Account> get(@PathVariable Long id) {

return accountRepository.findById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

}

// Account.java

@Entity

public class Account {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String type;

private Double balance;

// getters & setters

}

// AccountRepository.java

public interface AccountRepository extends JpaRepository<Account, Long> {}

---

// === 4. Loan Service ===

// File: LoanServiceApplication.java

@SpringBootApplication

@EnableDiscoveryClient

public class LoanServiceApplication {

public static void main(String[] args) {

SpringApplication.run(LoanServiceApplication.class, args);

}

}

// application.yml

spring:

application:

name: loan-service

datasource:

url: jdbc:mysql://localhost:3306/loans

username: root

password: root

config:

import: optional:configserver:http://localhost:8888

server:

port: 8082

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

// LoanController.java

@RestController

@RequestMapping("/loans")

public class LoanController {

@Autowired

private LoanRepository loanRepository;

@PostMapping

public Loan create(@RequestBody Loan loan) {

return loanRepository.save(loan);

}

@GetMapping("/{id}")

public ResponseEntity<Loan> get(@PathVariable Long id) {

return loanRepository.findById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

}

// Loan.java

@Entity

public class Loan {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String loanType;

private Double amount;

private Long customerId;

// getters & setters

}

// LoanRepository.java

public interface LoanRepository extends JpaRepository<Loan, Long> {}

**Step 1: Common Setup**

**application.yml (for both services)**

server:

port: 8081 # Change as per service

spring:

application:

name: account-service

**Step 2: Account Service Code (basic setup)**

**Account.java**

@Entity

public class Account {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String customerName;

private String accountType;

private Double balance;

// getters and setters

}

AccountRepository.java :  
@Repository

public interface AccountRepository extends JpaRepository<Account, Long> {

}

AccountController.java :  
@RestController

@RequestMapping("/accounts")

public class AccountController {

@Autowired

private AccountRepository accountRepository;

@PostMapping

public Account createAccount(@RequestBody Account account) {

return accountRepository.save(account);

}

@GetMapping("/{id}")

public Account getAccount(@PathVariable Long id) {

return accountRepository.findById(id).orElse(null);

}

}

**Step 3: Loan Service Code (basic setup)**

**Loan.java**

@Entity

public class Loan {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private Long accountId;

private Double amount;

private String loanType;

// getters and setters

}

LoanRepository.java :  
@Repository

public interface LoanRepository extends JpaRepository<Loan, Long> {

List<Loan> findByAccountId(Long accountId);

}

LoanController.java :  
@RestController

@RequestMapping("/loans")

public class LoanController {

@Autowired

private LoanRepository loanRepository;

@PostMapping

public Loan createLoan(@RequestBody Loan loan) {

return loanRepository.save(loan);

}

@GetMapping("/account/{accountId}")

public List<Loan> getLoansByAccount(@PathVariable Long accountId) {

return loanRepository.findByAccountId(accountId);

}

}