Cognizant - DN 4.0 I Deep Skilling

Week-5

Creating Microservices for account and loan

Account Service

@Entity

public class Account {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String accountNumber;

private String accountType;

private double balance;

}

Repository

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

Service

@Service

public class AccountService {

@Autowired

private AccountRepository repo;

public Account createAccount(Account account) {

return repo.save(account);

}

public Optional<Account> getAccount(Long id) {

return repo.findById(id);

}

}

Controller

@RestController

@RequestMapping("/api/accounts")

public class AccountController {

@Autowired

private AccountService service;

@PostMapping

public ResponseEntity<Account> create(@RequestBody Account account) {

return new ResponseEntity<>(service.createAccount(account), HttpStatus.CREATED);

}

@GetMapping("/{id}")

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

return service.getAccount(id)

.map(ResponseEntity::ok)

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

}

}

Loan Service

@Entity

public class Loan {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String loanNumber;

private double loanAmount;

private Long accountId; // Reference to account

}

Repository

public interface LoanRepository extends JpaRepository<Loan, Long> {

List<Loan> findByAccountId(Long accountId);

}

Service

@Service

public class LoanService {

@Autowired

private LoanRepository repo;

public Loan createLoan(Loan loan) {

return repo.save(loan);

}

public List<Loan> getLoansByAccountId(Long accountId) {

return repo.findByAccountId(accountId);

}

}

Controller

@RestController

@RequestMapping("/api/loans")

public class LoanController {

@Autowired

private LoanService service;

@PostMapping

public ResponseEntity<Loan> create(@RequestBody Loan loan) {

return new ResponseEntity<>(service.createLoan(loan), HttpStatus.CREATED);

}

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

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

return ResponseEntity.ok(service.getLoansByAccountId(accountId));

}

}

Create Eureka Discovery Server and register microservices

<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>2022.0.4</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

server:

port: 8761

spring:

application:

name: eureka-server

eureka:

client:

register-with-eureka: false

fetch-registry: false

<dependency>

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

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

</dependency>

@SpringBootApplication

@EnableEurekaClient

public class AccountServiceApplication {

public static void main(String[] args) {

SpringApplication.run(AccountServiceApplication.class, args);

}

}

server:

port: 8081

spring:

application:

name: account-service

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka>

server:

port: 8082

spring:

application:

name: loan-service

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka>