|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A green leaf logo with a white background  Description automatically generated | | | | |
|  |  |  | |  |
| Activité pratique N° 2  Architectures Micro-services avec Spring cloud | | |
| abderrahmane ettounani 🡪 iibdcc-3 | | 2023/2024 |

|  |
| --- |
|  |
| eNoNCE | |

A white board with black dots

Description automatically generated

|  |
| --- |
|  |

# 1 : MICRO-SERVICE Customer-service

@Entity

@Data @NoArgsConstructor @AllArgsConstructor @ToString

public class Customer {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    private String email;

}

@SpringBootApplication

public class CustomerApplication {

    public static void main(String[] args) {

        SpringApplication.run(CustomerApplication.class, args);

    }

    @Bean

    CommandLineRunner start(CustomerRepository customerRepository) {

        return args -> {

            customerRepository.save(new Customer(null, "Ettounani", "tounani@gmail.com"));

            customerRepository.save(new Customer(null, "youssef", "youssef@gmail.com"));

            customerRepository.save(new Customer(null, "Abderrahmane", "abdo@gmail.com"));

            customerRepository.findAll().forEach(System.out::println);

        };

    }

}

server.port=8081

spring.application.name=customer-service

spring.datasource.url=jdbc:h2:mem:customer-db

spring.cloud.discovery.enabled=true

@RepositoryRestController

public interface CustomerRepository extends JpaRepository<Customer, Long> {}

# 2 : MICRO-SERVICE Inventory-service

@Entity

@Data @NoArgsConstructor @AllArgsConstructor @ToString

public class Product {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    private double price;

    private double quantite;

}

@RepositoryRestController

public interface ProductRepository extends JpaRepository<Product, Long> {}

@SpringBootApplication

public class InventoryApplication {

    public static void main(String[] args) {

        SpringApplication.run(InventoryApplication.class, args);

    }

@Bean

    CommandLineRunner start(ProductRepository productRepository) {

        return args -> {

                productRepository.save( new Product(null,"Computer", 8000, 3));

                productRepository.save( new Product(null,"Printer", 2000, 2));

                productRepository.save( new Product(null,"Smartphone", 6000, 5));

            productRepository.findAll().forEach(System.out::println);

        };

    }

}

server.port=8082

spring.application.name=product-service

spring.datasource.url=jdbc:h2:mem:product-db

spring.cloud.discovery.enabled=true

# 3 : Gateway service

@SpringBootApplication

public class GatewayyApplication {

    public static void main(String[] args) {

        SpringApplication.run(GatewayyApplication.class, args);

    }

    //@Bean

    public RouteLocator myRoutes(RouteLocatorBuilder builder) {

        return builder.routes()

                .route(p -> p.path("/customers/\*\*").uri("lb://CUSTOMER-SERVICE"))

                .route(p -> p.path("/products/\*\*").uri("lb://PRODUCT-SERVICE"))

                .build();

    }

    @Bean

    public DiscoveryClientRouteDefinitionLocator definitionLocator(ReactiveCompositeDiscoveryClient discoveryClient, DiscoveryLocatorProperties properties ) {

        return new DiscoveryClientRouteDefinitionLocator(discoveryClient, properties);

    }

}

server.port=8888

spring.application.name=gateway-service

spring.cloud.discovery.enabled=true

spring:

  cloud:

    gateway:

      routes:

        - id: r1

          uri: http://localhost:8081/

          predicates:

            - Path=/customers/\*\*

        - id: r2

          uri: http://localhost:8082/

          predicates:

            - Path=/products/\*\*

# 4 : NetFlix Eureka Server

@SpringBootApplication

@EnableEurekaServer // Enable eureka server

public class DiscoveryApplication {

    public static void main(String[] args) {

        SpringApplication.run(DiscoveryApplication.class, args);

    }

}

server.port=8761

eurika.client.fetch-registry=false

eurika.client.register-with-eurika=false

# 5 : TEST

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# 6 : Billing-Service

@Entity

public class Bill {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private Date billingDate;

    @OneToMany(mappedBy = "bill")

    private Collection<ProductItem> productsItems;

    private Long customerID;

    @Transient

    private Customer customer;

}

@Entity

public class ProductItem {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private double quantity;

    private double price;

    private Long productId;

    @ManyToOne

    private Bill bill;

    @Transient

    private Product product;

}

@Data

public class Customer {

    private Long id;

    private String name;

    private String email;

}

@Data

public class Product {

    private Long id;

    private String name;

    private double price;

    private double quantity;

}

@FeignClient(name = "CUSTOMER-SERVICE")

public interface CustomerRestClient {

    @GetMapping(path = "/customers/{id}")

    public Customer findCustomerById(Long id);

}

@FeignClient(name = "PRODUCT-SERVICE")

public interface ProductItemRestClient {

    @GetMapping(path = "/products")

    PagedModel<Product> findAll(@RequestParam(value = "page") int page,@RequestParam(value="size") int size);

    @GetMapping(path = "/products/{id}")

    Product findProductById(@RequestParam(value = "id") Long id);

}

@RepositoryRestResource

public interface BillRepository  extends JpaRepository <Bill, Long>{

}

@RepositoryRestResource

public interface ProductRepository  extends JpaRepository <ProductItem, Long>{

    public Collection<ProductItem> findByBillId(Long id);

}

@SpringBootApplication

public class BellingApplication {

    public static void main(String[] args) {

        SpringApplication.run(BellingApplication.class, args);

    }

}