



Micro services avec Spring Cloud

30.11.2021

Mensour Ikram
5IIR-G11
EMSI

Pr. Youssfi Mohamed

Table de matières

Vue d'ensemble	2
Architecture	2
Diagramme de classes	3
Réalisation	3
I- Création du micro service Customer-Service:	4
1-Créer l'entité Customer:	5
2- Créer l'interface Custom Repository basée sur Spring Data:	5
3-Déployer l'API Restful du micro-service en utilisant Spring Data Rest :	6
II- Création du micro service Inventory-Service:	6
1-Créer l'entité Product :	7
2-Créer l'interface ProductRepository basée sur Spring Data :	7
3-Déployer l'API Restful du micro-service en utilisant Spring Data Rest :	8
III- Créer la Gateway service en utilisant Spring Cloud Gateway	8
1-Tester le Service proxy en utilisant une configuration Statique basée sur le fichier application.yml	8
2-Tester la Service proxy en utilisant une configuration Statique basée sur une configuration Java	9
IV- Créer l'annuaire Registry Service basé sur NetFlix Eureka Server	9
V- Créer le service Billing-Service en utilisant Open Feign pour communiquer avec les services Customer-service et Inventory-service	10

Vue d'ensemble

On souhaite réaliser une application basée sur deux services métiers: Service des clients, service d'inventaire et service de facturation. L'orchestration des services se fait via les techniques de Spring Cloud: Spring Cloud Gateway Service comme service proxy et Registry Eureka Service comme annuaire d'enregistrement et de découverte des services de l'architecture.

Architecture

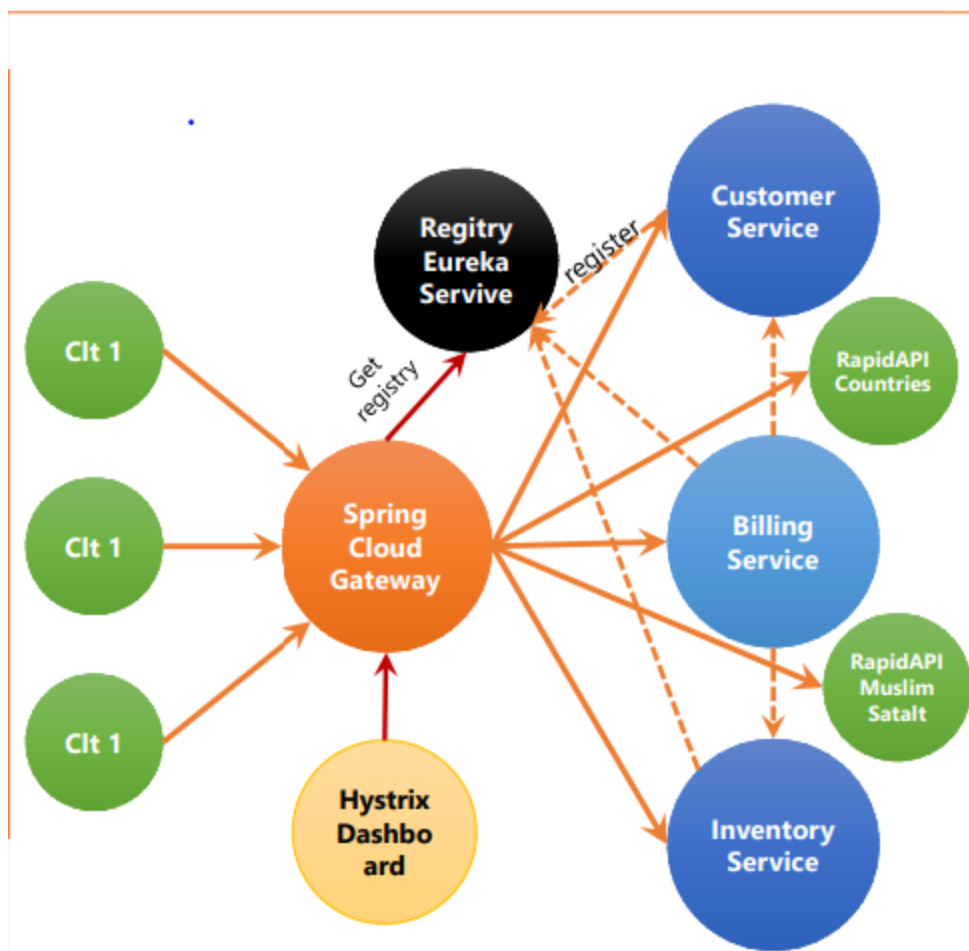
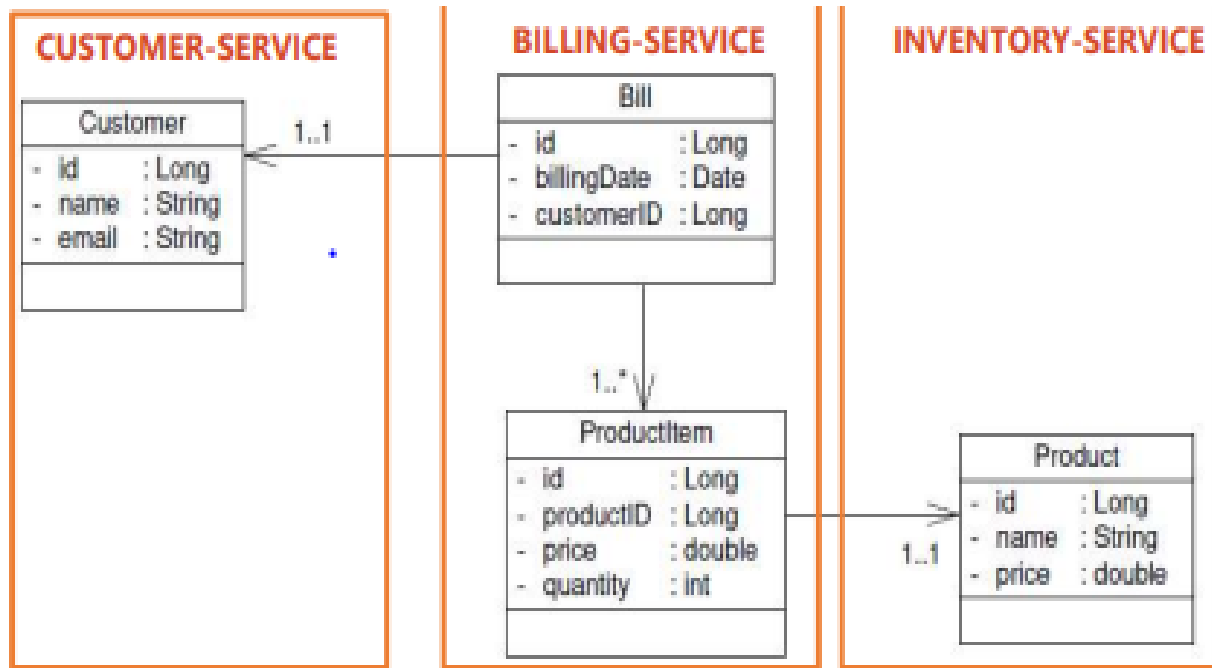


Diagramme de classes

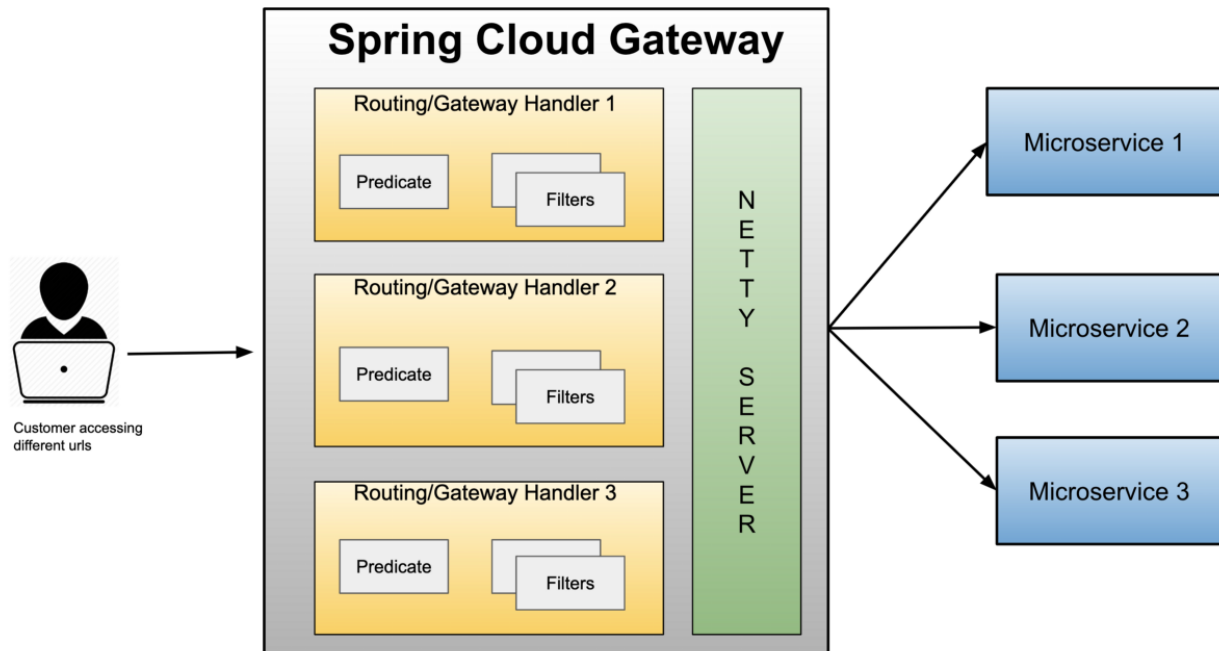


Réalisation

Spring Cloud Gateway:

Spring Cloud Gateway fournit une bibliothèque pour créer des passerelles API sur Spring et Java. Il fournit un moyen flexible d'acheminer les demandes en fonction d'un certain nombre de critères, et se concentre sur des préoccupations transversales telles que la sécurité, la résilience et la surveillance.

Une passerelle API peut aider à simplifier la communication entre un client et un service, que ce soit entre le navigateur Web d'un utilisateur et un serveur distant, ou entre une application frontale et les services principaux sur lesquels elle repose.



I- Création du micro service Customer-Service:

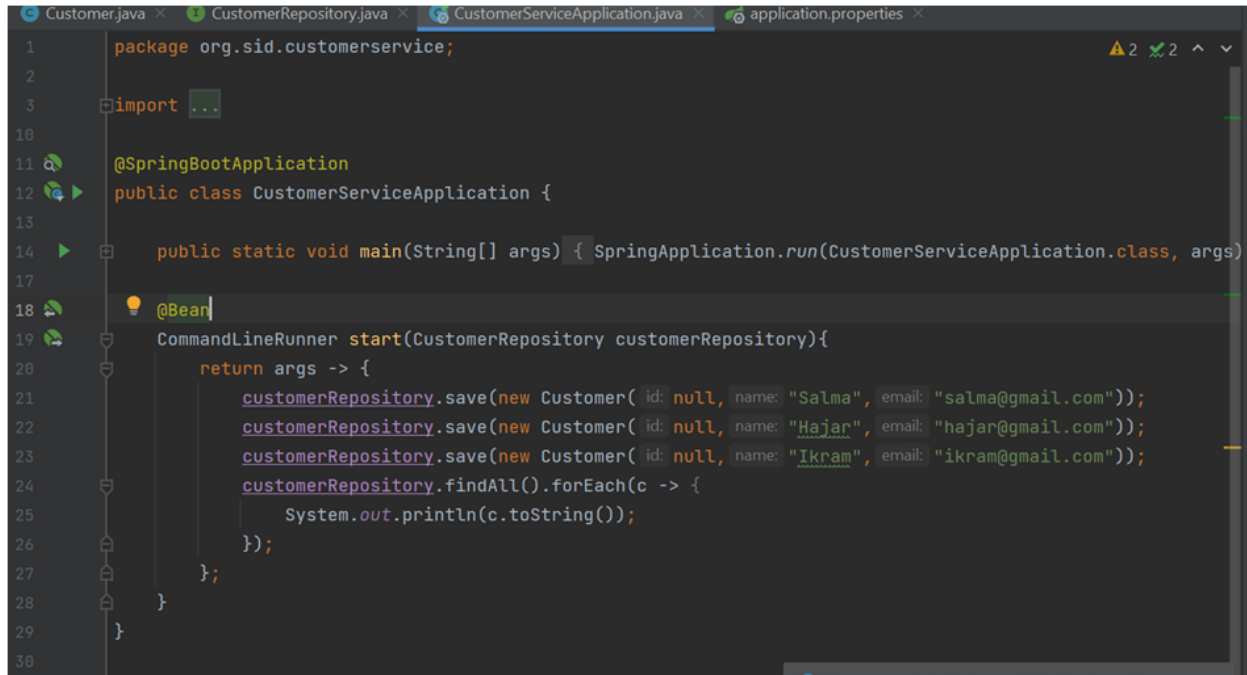
1-Créer l'entité Customer:

```
Customer.java x CustomerRepository.java x CustomerServiceApplication.java x application.properties x
1 package org.sid.customerservice.entities;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6 import lombok.ToString;
7
8 import javax.persistence.Entity;
9 import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12
13 @Entity
14 @Data @NoArgsConstructor @AllArgsConstructor @ToString
15 public class Customer {
16     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private Long id;
18     private String name;
19     private String email;
20 }
21
```

2- Créer l'interface Custom Repository basée sur Spring Data:

```
Customer.java x CustomerRepository.java x CustomerServiceApplication.java x application.properties x
1 package org.sid.customerservice.repository;
2
3 import org.sid.customerservice.entities.Customer;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.data.rest.core.annotation.RepositoryRestResource;
6
7 @RepositoryRestResource
8 public interface CustomerRepository extends JpaRepository<Customer, Long> {
9 }
10
```

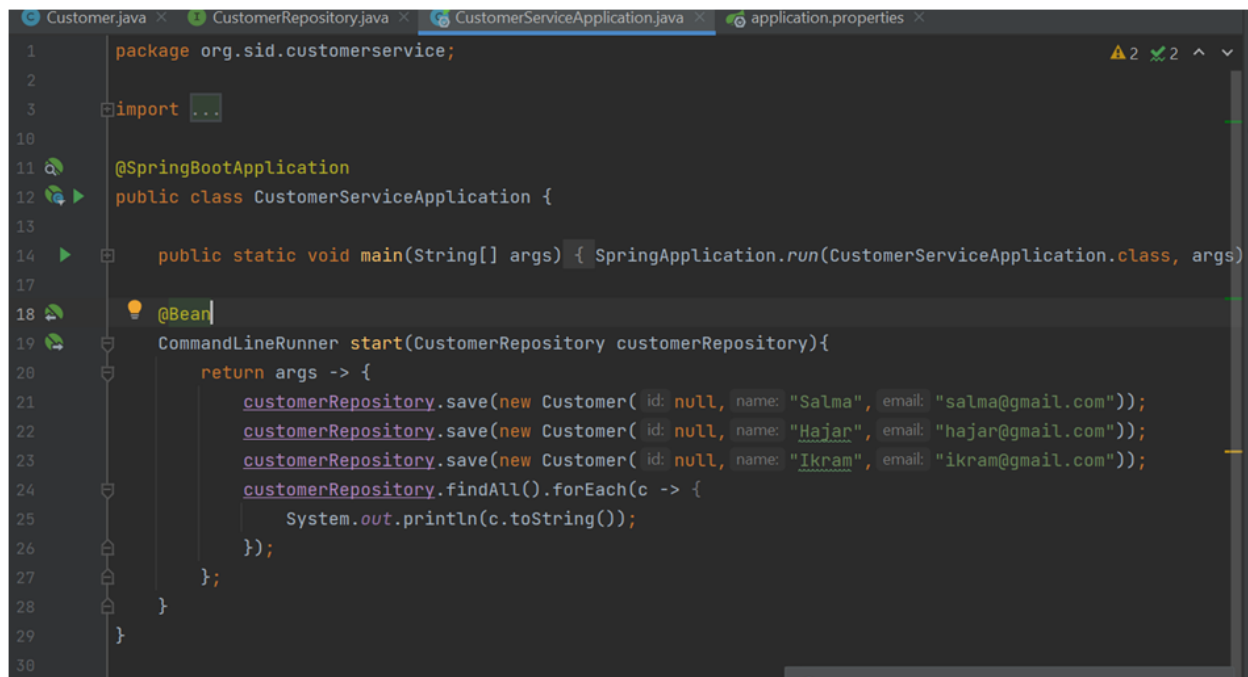
3-Déployer l'API Restful du micro-service en utilisant Spring Data Rest :



```
1 package org.sid.customerservice;
2
3 import ...
4
5 @SpringBootApplication
6 public class CustomerServiceApplication {
7
8     public static void main(String[] args) { SpringApplication.run(CustomerServiceApplication.class, args)
9
10
11     @Bean
12     CommandLineRunner start(CustomerRepository customerRepository){
13
14         return args -> {
15
16             customerRepository.save(new Customer( id: null, name: "Salma", email: "salma@gmail.com"));
17             customerRepository.save(new Customer( id: null, name: "Hajar", email: "hajar@gmail.com"));
18             customerRepository.save(new Customer( id: null, name: "Ikram", email: "ikram@gmail.com"));
19             customerRepository.findAll().forEach(c -> {
20                 System.out.println(c.toString());
21             });
22         };
23     }
24 }
25
26 }
```

II- Création du micro service Inventory-Service:

1-Créer l'entité Product :



```
1 package org.sid.customerservice;
2
3 import ...
4
5
6
7
8
9
10
11 @SpringBootApplication
12 public class CustomerServiceApplication {
13
14     public static void main(String[] args) { SpringApplication.run(CustomerServiceApplication.class, args)
15
16
17
18     @Bean
19     CommandLineRunner start(CustomerRepository customerRepository){
20         return args -> {
21             customerRepository.save(new Customer( id: null, name: "Salma", email: "salma@gmail.com"));
22             customerRepository.save(new Customer( id: null, name: "Hajar", email: "hajar@gmail.com"));
23             customerRepository.save(new Customer( id: null, name: "Ikram", email: "ikram@gmail.com"));
24             customerRepository.findAll().forEach(c -> {
25                 System.out.println(c.toString());
26             });
27         };
28     }
29 }
30
```

2-Créer l'interface ProductRepository basée sur Spring Data :



```
1 package org.sid.inventoryservice.repository;
2
3 import ...
4
5
6
7 @RepositoryRestResource
8 public interface ProductRepository extends JpaRepository<Product, Long> {
9 }
10
```


3-Déployer l'API Restful du micro-service en utilisant Spring Data Rest :




```
1 package org.sid.inventoryservice.repository;
2
3 import ...
4
5
6
7 @RepositoryRestResource
8 public interface ProductRepository extends JpaRepository<Product, Long> {
9 }
10 |
```

III- Créer la Gateway service en utilisant Spring Cloud Gateway

1-Tester le Service proxy en utilisant une configuration Statique basée sur le fichier application.yml

2-Tester la Service proxy en utilisant une configuration Statique basée sur une configuration Java

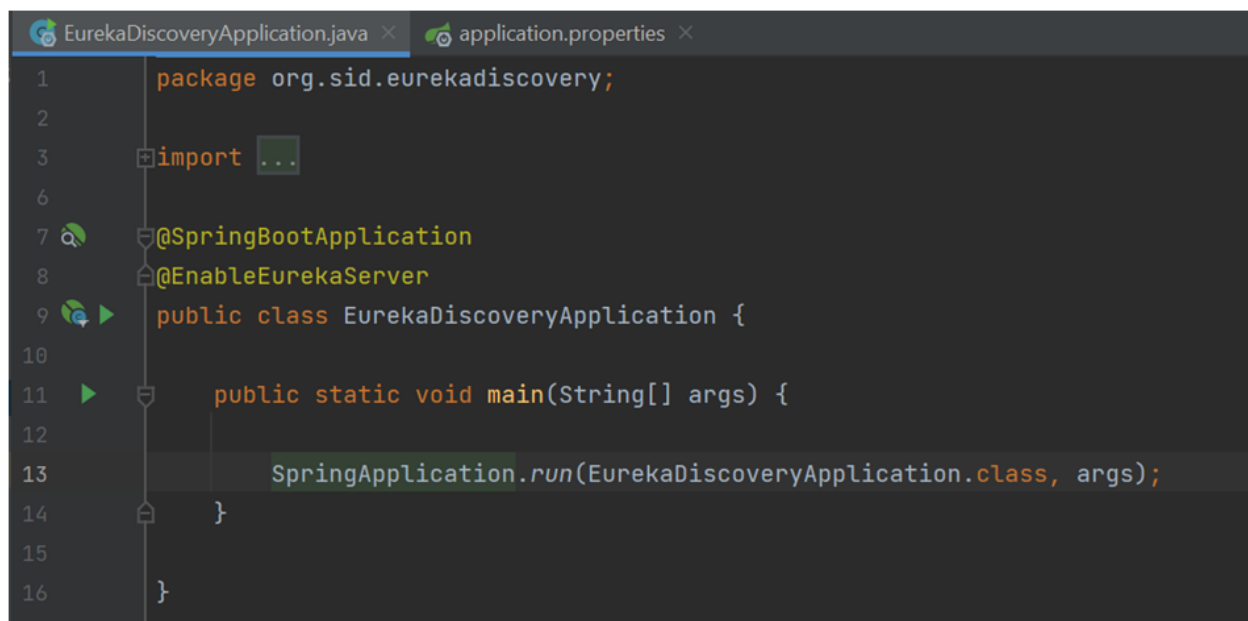


```

1 package org.sid.gateway;
2
3 import ...
4
11
12 @SpringBootApplication
13 public class GatewayApplication {
14
15     public static void main(String[] args) { SpringApplication.run(GatewayApplication.class, args); }
16
17     // @Bean
18     @RouteLocator gatewayRoutes(RouteLocatorBuilder builder)
19     {
20         return builder.routes()
21             .route(r->r.path( ...patterns: "/customers/**").uri("lb://CUSTOMER-SERVICE"))
22             .build();
23     }
24
25     @Bean
26     DiscoveryClientRouteDefinitionLocator definitionLocator(ReactiveDiscoveryClient rdc, DiscoveryLocatorProperties dlp){
27         return new DiscoveryClientRouteDefinitionLocator(rdc, dlp);
28     }
29
30 }

```

IV- Créer l'annuaire Registry Service basé sur NetFlix Eureka Server



```

1 package org.sid.eurekadiscovery;
2
3 import ...
4
6
7 @SpringBootApplication
8 @EnableEurekaServer
9 public class EurekaDiscoveryApplication {
10
11     public static void main(String[] args) {
12
13         SpringApplication.run(EurekaDiscoveryApplication.class, args);
14     }
15
16 }

```

EurekaDiscoveryApplication.java

V- Créer le service Billing-Service en utilisant Open Feign pour communiquer avec les services Customer-service et Inventory-service



```
application.properties x BillingServiceApplication.java x Product.java x Customer.java x BillReposito
package org.sid.bilingservice;

import ...

@SpringBootApplication
public class BillingServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(BillingServiceApplication.class, args);

    }
}
```

- BillingApplicationService.java

```

on.properties x BillingRestController.java x BillingServiceApplication.java x Product.java x Customer.java x BillRepository.java x ProductItemRepository.java
14
15 @RestController
16 public class BillingRestController {
17     private BillRepository billRepository;
18     private ProductItemRepository productItemRepository;
19     private CustomerRestClient customerRestClient;
20     private ProductItemRestClient productItemRestClient;
21     public BillingRestController(BillRepository billRepository, ProductItemRepository productItemRepository, CustomerRestClient customerRestClient, ProductItemRestClient productItemRestClient) {
22         this.billRepository = billRepository;
23         this.productItemRepository = productItemRepository;
24         this.customerRestClient = customerRestClient;
25         this.productItemRestClient = productItemRestClient;
26     }
27     @GetMapping(path = "/fullBill/{id}")
28     public Bill getBill(@PathVariable(name="id") Long id){
29         Bill bill = billRepository.findById(id).get();
30         Customer customer=customerRestClient.getCustomerById(bill.getCustomerID());
31         bill.setCustomer(customer);
32         bill.getProductItems().forEach(p1->{
33             Product product=productItemRestClient.getProductById(p1.getProductID());
34             p1.setProductName(product.getName());
35         });
36         return bill;
37     }

```

BillingRestController.java

```

application.properties x ProductItemRepository.java x BillingRestController.java x BillingServiceApplication.java x
package org.sid.bilingservice.repository;

import org.sid.bilingservice.entities.ProductItem;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import java.util.Collection;

@RepositoryRestResource
public interface ProductItemRepository extends JpaRepository<ProductItem,Long> {
    public Collection<ProductItem> findByBillId(Long id);
}

```

ProductItemRepository.java

```
application.properties x ProductItemRepository.java x BillingRestController.java x BillingServiceApplication.java x
1 package org.sid.bilingservice.repository;
2
3 import org.sid.bilingservice.entities.Bill;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.data.rest.core.annotation.RepositoryRestResource;
6
7 @RepositoryRestResource
8 public interface BillRepository extends JpaRepository<Bill, Long> {
9 }
10
```

BillRepository

```
application.properties x Customer.java x ProductItemRepository.java x
1 package org.sid.bilingservice.model;
2
3 import lombok.Data;
4
5 @Data
6 public class Customer {
7     private Long id;
8     private String name;
9     private String email;
10 }
11
```

Customer.java

```
emRestClient.java × application.properties × Product.java × BillingRe

package org.sid.bilingservice.model;

import lombok.Data;

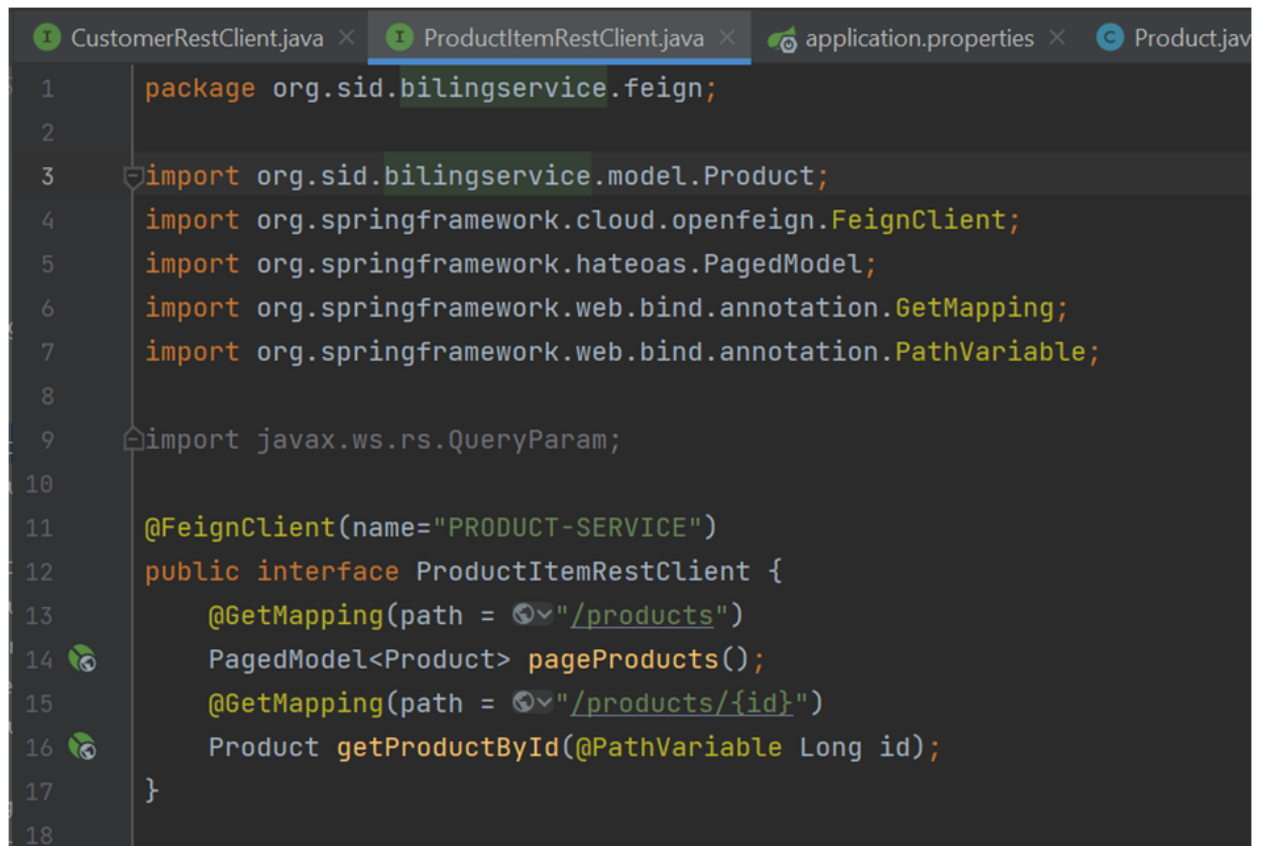
@Data
public class Product {
    private Long id;
    private String name;
    private double price;
    private double quantity;
}
```

Product.java

```
CustomerRestClient.java × ProductItemRestClient.java × application.properties × Product.java ×

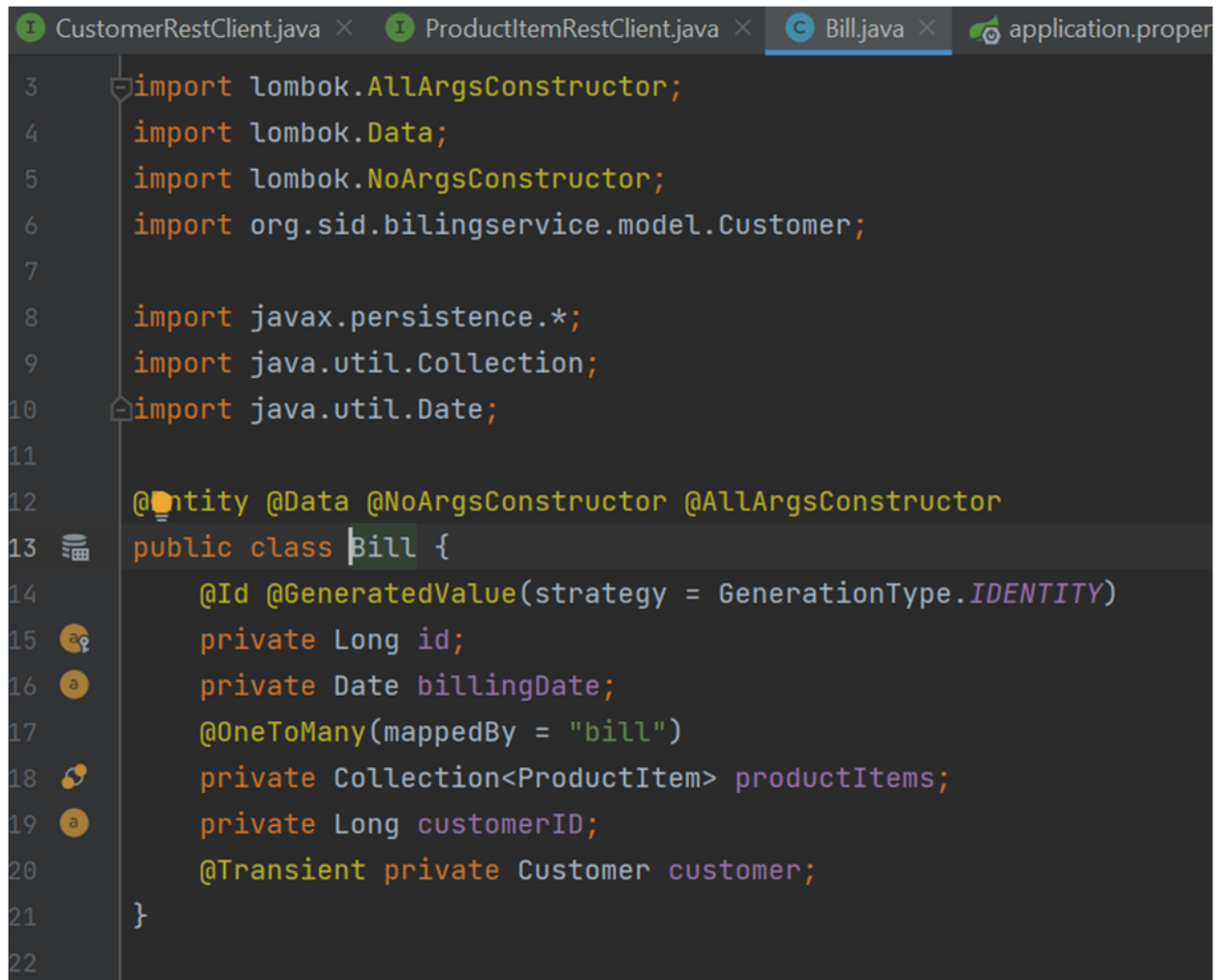
1 package org.sid.bilingservice.feign;
2
3 import org.sid.bilingservice.model.Customer;
4 import org.springframework.cloud.openfeign.FeignClient;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.PathVariable;
7
8 @FeignClient(name = "CUSTOMER-SERVICE")
9 public interface CustomerRestClient {
10     @GetMapping(path="/customers/{id}")
11     public Customer getCustomerById(@PathVariable(name="id") Long id);
12 }
```

CustomerRestClient.java



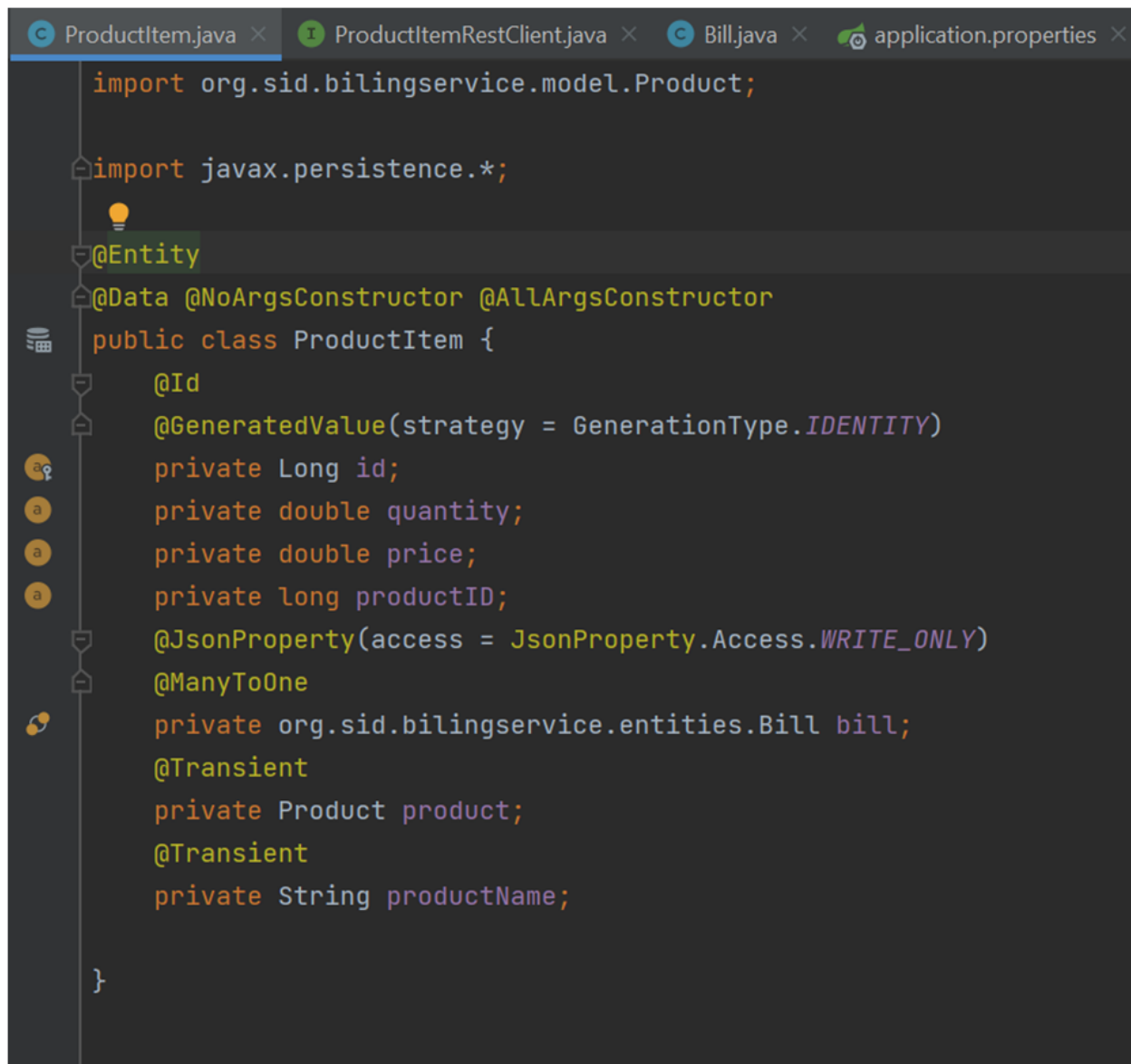
```
1 package org.sid.bilingservice.feign;
2
3 import org.sid.bilingservice.model.Product;
4 import org.springframework.cloud.openfeign.FeignClient;
5 import org.springframework.hateoas.PagedModel;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.PathVariable;
8
9 import javax.ws.rs.QueryParam;
10
11 @FeignClient(name="PRODUCT-SERVICE")
12 public interface ProductItemRestClient {
13     @GetMapping(path = "/products")
14     PagedModel<Product> pageProducts();
15     @GetMapping(path = "/products/{id}")
16     Product getProductById(@PathVariable Long id);
17 }
18
```

ProductItemRestClient.java



```
CustomerRestClient.java × ProductItemRestClient.java × Bill.java × application.properties
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6  import org.sid.bilingservice.model.Customer;
7
8  import javax.persistence.*;
9  import java.util.Collection;
10 import java.util.Date;
11
12 @Entity @Data @NoArgsConstructor @AllArgsConstructor
13 public class Bill {
14     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16     private Date billingDate;
17     @OneToMany(mappedBy = "bill")
18     private Collection<ProductItem> productItems;
19     private Long customerID;
20     @Transient private Customer customer;
21 }
22
```

Bill.java



```
ProductItem.java × ProductItemRestClient.java × Bill.java × application.properties ×
import org.sid.bilingservice.model.Product;

import javax.persistence.*;

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class ProductItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private double quantity;
    private double price;
    private long productID;
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    @ManyToOne
    private org.sid.bilingservice.entities.Bill bill;
    @Transient
    private Product product;
    @Transient
    private String productName;
}
```

ProductItem.java

