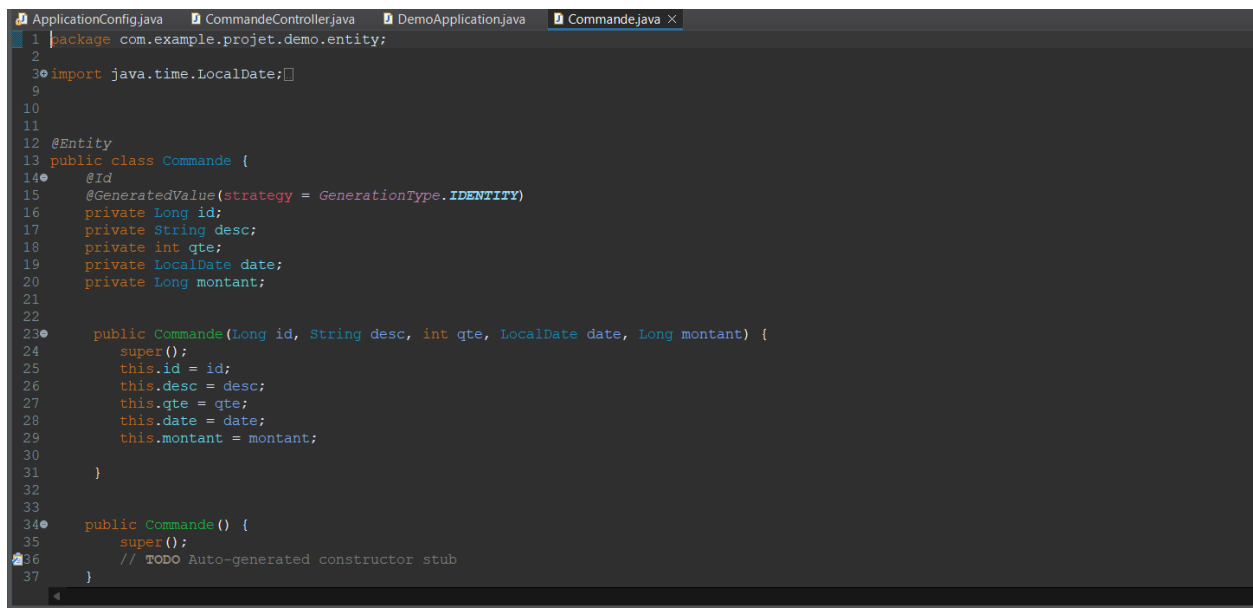

DEVOIR N°1 Module JEE : Développement Microservices avec Spring Cloud

Team de développement / binôme : Bouzekraoui Youssef & Drissi Morad

Etude de cas (1) :

Énoncé : Ajouter un « microservice-commandes » qui permet de réaliser les opérations CRUD sur une « COMMANDE » avec 0 ligne SQL

a. La version (1) de la table « COMMANDE » est composée » des colonnes suivantes [id, description, quantité, date, montant] :



```
1 package com.example.projet.demo.entity;
2
3 import java.time.LocalDate;
4
5
6
7
8
9
10
11
12 @Entity
13 public class Commande {
14     @Id
15     @GeneratedValue(strategy = GenerationType.IDENTITY)
16     private Long id;
17     private String desc;
18     private int qte;
19     private LocalDate date;
20     private Long montant;
21
22
23     public Commande(Long id, String desc, int qte, LocalDate date, Long montant) {
24         super();
25         this.id = id;
26         this.desc = desc;
27         this.qte = qte;
28         this.date = date;
29         this.montant = montant;
30     }
31
32
33
34     public Commande() {
35         super();
36         // TODO Auto-generated constructor stub
37     }
38 }
```

b. La configuration du « microservice-commandes » doit être gérée au niveau Spring

Cloud et github :

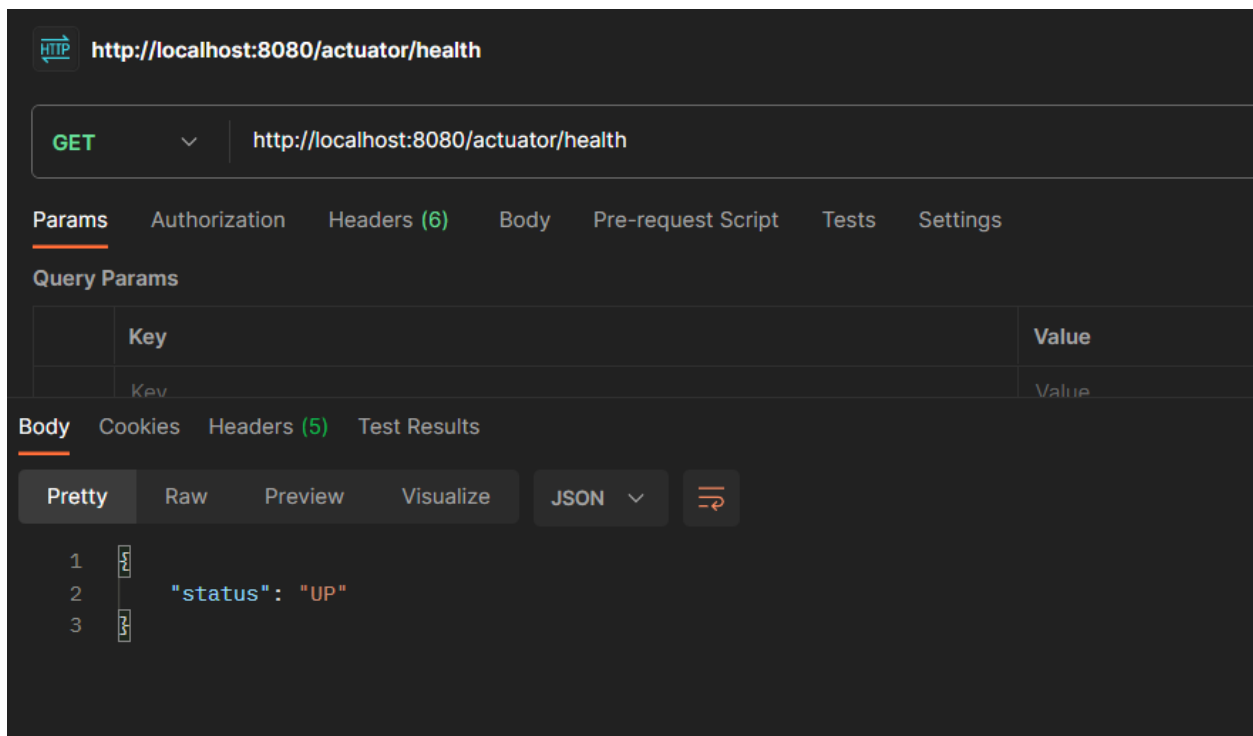
```
application.yml x
1 spring:
2   cloud:
3     config:
4       server:
5         git:
6           uri: https://github.com/DrissiMourad22/microservice.git
7           username: DrissiMourad22
8           token: ghp_HgruvQeYop640MHPYQXFcIOzMxig4x4LO1hx
9 mes-config-ms:
10  commandes-last: 10
11
12
```

c. La configuration du « microservice-commandes » contient une propriété personnalisée « mes-config-ms.commandes-last » qui permet d’afficher les dernières commandes reçues. Dans notre cas : « mes-config-ms.commandes-last = 10 » permet d’afficher les commandes reçues les 10 derniers jours.

En se basant sur le service Actuator de spring, modifier cette propriété à 20 et réaliser un chargement à chaud pour que le « microservice-commandes » affiche les commandes reçues les 20 derniers jours :

```
ApplicationConfig.java x
1 package com.example.projet.demo.config;
2
3 import org.springframework.beans.factory.annotation.Value;
4
5
6
7 @Configuration
8 public class ApplicationConfig {
9
10  @Value("${mes-config-ms.commandes-last:10}")
11  private int comdLast;
12
13  public int getCommandesLast() {
14      return comdLast;
15  }
16 }
17
```

d. En se basant sur le service Actuator de spring, Implémenter la supervision la bonne santé du « microservice-commandes » : le statut à afficher « UP »



e. Personnaliser la supervision de la bonne santé du « microservice-commandes » :
dans notre cas, un « microservice-commandes » est en bonne santé lorsqu'il y'a des
commandes dans la table « COMMANDE », dans ce cas, le statut est « UP » sinon le
statut à afficher est « DOWN »

```
4  @GetMapping("/health")
5  public ResponseEntity<String> checkHealth() {
6      if (commandeRepository.count() > 0) {
7          return ResponseEntity.ok("Up");
8      } else {
9          return ResponseEntity.status(HttpStatus.SERVICE_UNAVAILABLE).body("Down");
10     }
11 }
12
13 }
```

- Liste des 100 Commandes Générées Aléatoirement :

HTTP <http://localhost:8080/cmd>

GET <http://localhost:8080/cmd>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1  [  
2    {  
3      "id": 1,  
4      "desc": "Description 47",  
5      "qte": 8,  
6      "date": "2023-12-14",  
7      "montant": 214  
8    },  
9    {  
10     "id": 2,  
11     "desc": "Description 37",  
12     "qte": 10,  
13     "date": "2023-12-15",  
14     "montant": 272  
15   },  
16   {  
17     "id": 3,  
18     "desc": "Description 36",  
19     "qte": 1,  
20     "date": "2023-12-09",  
21     "montant": 257  
22   },  
23 ]
```

-Affichage d'une Commande avec le détail par son ID (exemple: 4 et 48)

The screenshot shows a REST client interface with the URL `http://localhost:8080/4` and a `GET` method. The response body is displayed in JSON format:

```
1 {
2   "id": 4,
3   "desc": "Description 17",
4   "qte": 3,
5   "date": "2023-12-20",
6   "montant": 297
7 }
```

The screenshot shows a REST client interface with the URL `http://localhost:8080/48` and a `GET` method. The response body is displayed in JSON format:

```
1 {
2   "id": 48,
3   "desc": "Description 14",
4   "qte": 10,
5   "date": "2024-01-02",
6   "montant": 435
7 }
```