

DEPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Activité Pratique N°6

Filière :

« Génie du Logiciel et des Systèmes Informatiques Distribués »

GLSID

Mise en œuvre d'un Micro-service

Réalisé par :

Najat ES-SAYYAD

Année Universitaire : 2022-2023

Introduction

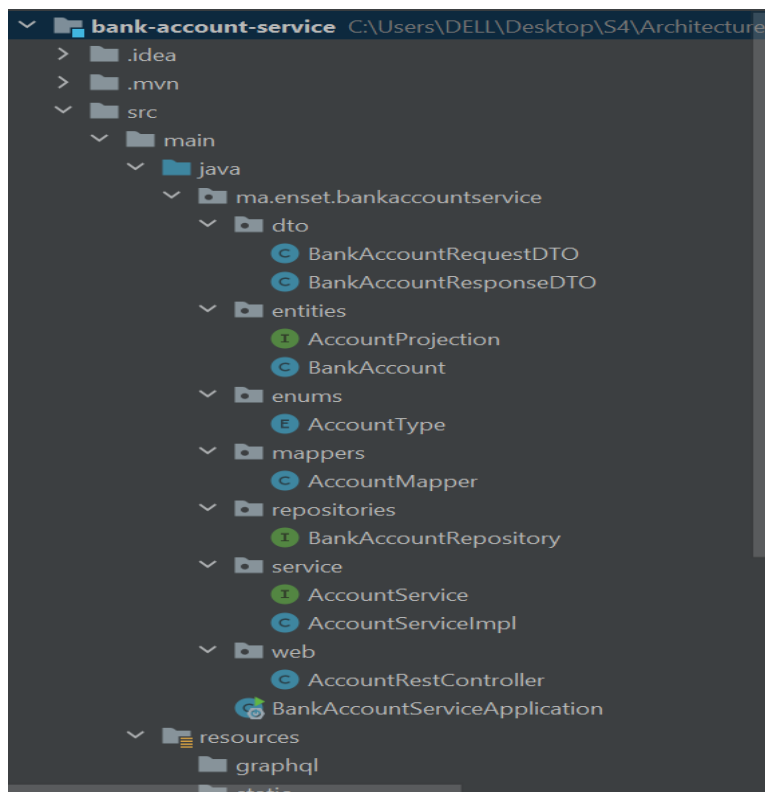
Dans ce travail pratique, nous allons développer un microservice de gestion de comptes bancaires en utilisant Spring Boot. Notre microservice utilise les dépendances Web, Spring Data JPA, H2 et Lombok. Nous avons également intégré Swagger pour générer la documentation de notre API RESTful.

Nous allons suivre un ensemble d'étapes pour créer ce microservice, notamment la création d'une entité JPA Compte, la création d'une interface CompteRepository basée sur Spring Data, la couche DAO de test, la création de méthodes CRUD pour notre entité Compte, et la création de la couche de service métier pour notre application.

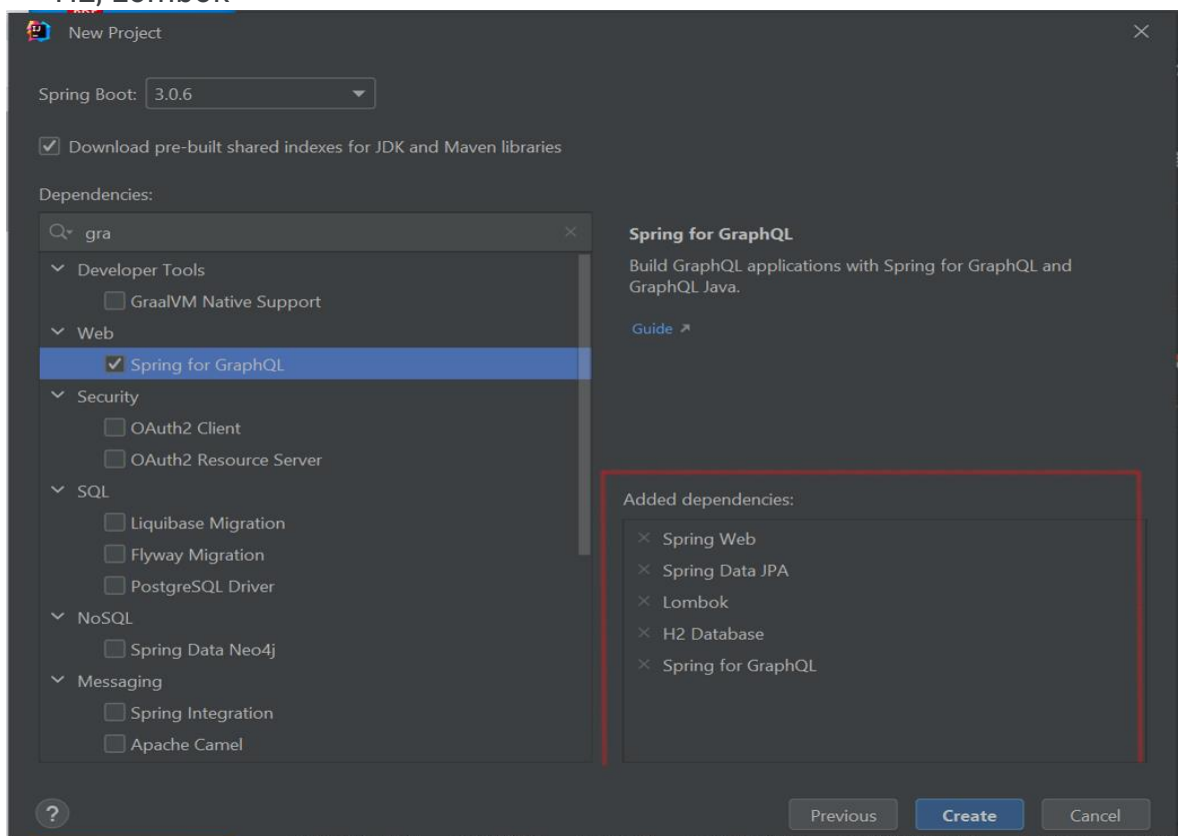
Nous allons également utiliser Spring Data Rest pour exposer une API RESTful en utilisant des projections, ce qui facilite l'accès aux données. Nous allons créer des DTOs et des Mappers pour faciliter la manipulation des données.

Enfin, nous allons tester notre microservice en utilisant un client REST comme Postman pour nous assurer que tout fonctionnait correctement. La documentation Swagger générée nous va permettre de visualiser toutes les méthodes de notre API RESTful.

Structure du projet :



1. Créer un projet Spring Boot avec les dépendances Web, Spring Data JPA, H2, Lombok



2. Créer l'entité JPA BankAccount

L'entité JPA BankAccount ayant les attributs :

- id de type String
- createdAt de type Date
- balance de type Double
- currency de type String
- type de type enum AccountType

```
15 @Entity
16 @Data @NoArgsConstructor @AllArgsConstructor
17 @Builder
18 public class BankAccount {
19     @Id
20     private String id ;
21     private Date createdAt;
22     private Double balance;
23     private String currency;
24     @Enumerated(EnumType.STRING)
25     private AccountType type ;
26 }
```

AccountType

```
1 package ma.enset.bankaccountservice.enums;
2
3 public enum AccountType {
4     CURRENT_ACCOUNT, SAVING_ACCOUNT
5 }
```

3. Créer l'interface BankAccountRepository basée sur Spring Data

```
6 public interface BankAccountRepository extends JpaRepository<BankAccount, String> {
7 }
8
```

Configurer l'unité de persistance dans le fichier application.properties

```
BankAccount.java × BankAccountRepository.java × BankAccountServiceApplication.java × application.properties ×
1 spring.datasource.url=jdbc:h2:mem:account-db
2 spring.h2.console.enabled=true
3 server.port=8081
```

4. Tester la couche DAO

```
BankAccount.java × BankAccountRepository.java × BankAccountServiceApplication.java × application.properties × AccountType.java ×
14 @SpringBootApplication
15 public class BankAccountServiceApplication {
16
17     no usages
18     public static void main(String[] args) {
19         SpringApplication.run(BankAccountServiceApplication.class, args);
20     }
21
22     no usages
23     @Bean
24     CommandLineRunner start(BankAccountRepository bankAccountRepository){
25         return args -> {
26             for (int i = 0; i < 10; i++) {
27                 BankAccount bankAccount=BankAccount.builder()
28                     .id(UUID.randomUUID().toString())
29                     .type(Math.random()>0.5? AccountType.CURRENT_ACCOUNT:AccountType.SAVING_ACCOUNT)
30                     .balance(10000+Math.random()*90000)
31                     .createdAt(new Date())
32                     .currency("MAD")
33                     .build();
34                 bankAccountRepository.save(bankAccount);
35             }
36         };
37     }
38 }
```

La base de données sous H2 :

localhost:8081/h2-console/login.do?sessionId=35e126f69d4f26d2c7eac57a7110059c

Auto commit: ☒ Max rows: 1000 Auto complete: Off Auto select: On

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM BANK_ACCOUNT BANK_ACCOUNT

jdbc:h2:mem:account-db

- BANK_ACCOUNT
 - ID
 - CHARACTER VARYING(36)
 - BALANCE
 - DOUBLE(15,2)
 - CREATED_AT
 - TIMESTAMP(6)
 - CURRENCY
 - CHARACTER VARYING(3)
 - TYPE
 - CHARACTER VARYING(20)
 - Indexes
 - PRIMARY_KEY_1
 - INFORMATION_SCHEMA
 - Users
 - H2 2.1.214 (2022-06-13)

SELECT * FROM BANK_ACCOUNT BANK_ACCOUNT;

ID	BALANCE	CREATED_AT	CURRENCY	TYPE
b3054499-420d-474f-a27f-0fa5bf296a13	92851.92837056537	2023-05-04 16:44:07.316	MAD	SAVING_ACCOUNT
79e7e9a2-454c-47ab-82b6-3401059b4c23	19965.146047180104	2023-05-04 16:44:07.459	MAD	SAVING_ACCOUNT
e506d9e2-40d5-4bf6-8579-c6b03d7c83ec	52972.54260158131	2023-05-04 16:44:07.461	MAD	SAVING_ACCOUNT
8c77dc73-13ef-4e97-9b79-368983dd00a7	91321.54553661846	2023-05-04 16:44:07.464	MAD	SAVING_ACCOUNT
27e4b17c-02ec-4eaa-be79-046f5e4fa125	27228.58681082109	2023-05-04 16:44:07.467	MAD	SAVING_ACCOUNT
6770b30e-3345-4f63-8fd4-3f2d4b7b6dd6	29193.377991405345	2023-05-04 16:44:07.469	MAD	CURRENT_ACCOUNT
57526873-4361-49ee-b593-380994751e41	78693.93584729808	2023-05-04 16:44:07.472	MAD	CURRENT_ACCOUNT
b8d3ab1d-f355-41a2-84d2-fc3feb87a93a	16704.529539387233	2023-05-04 16:44:07.475	MAD	SAVING_ACCOUNT
8dd5f68b-d1a5-485d-bbca-9f86c3018932	26995.476610145437	2023-05-04 16:44:07.477	MAD	SAVING_ACCOUNT
aa1058b4-720a-4ba4-8816-b282d609a941	33393.555735309594	2023-05-04 16:44:07.48	MAD	CURRENT_ACCOUNT

(10 rows, 6 ms)

Edit

5. Créer le Web service Restfull qui permet de gérer des comptes

```
BankAccount.java x BankAccountRepository.java x BankAccountServiceApplication.java x AccountRestController.java x application.properties x AccountType.java x
no usages
11 @RestController
12 public class AccountRestController {
13
14     7 usages
15     private BankAccountRepository bankAccountRepository;
16
17     no usages
18     public AccountRestController(BankAccountRepository bankAccountRepository){
19         this.bankAccountRepository=bankAccountRepository;
20     }
21
22     no usages
23     @GetMapping("/bankAccounts")
24     public List<BankAccount> bankAccounts(){
25         return bankAccountRepository.findAll();
26     }
27
28     no usages
29     @GetMapping("/{id}")
30     public BankAccount bankAccount(@PathVariable String id){
31         return bankAccountRepository.findById(id)
32             .orElseThrow(()->new RuntimeException(String.format("Account %s not found",id)));
33     }
34
35     @PostMapping("/bankAccounts")
36     public BankAccount save(@RequestBody BankAccount bankAccount){
37         if(bankAccount.getId()==null) bankAccount.setId(UUID.randomUUID().toString());
38         return bankAccountRepository.save(bankAccount);
39     }
40
41     no usages
42     @PutMapping("/{id}")
43     public BankAccount update(@PathVariable String id,@RequestBody BankAccount bankAccount){
44         BankAccount account=bankAccountRepository.findById(id).orElseThrow();
45         if(bankAccount.getBalance()!=null) account.setBalance(bankAccount.getBalance());
46         if(bankAccount.getCreatedAt()!=null) account.setCreatedAt(new Date());
47         if(bankAccount.getType()!=null) account.setType(bankAccount.getType());
48         if(bankAccount.getCurrency()!=null) account.setCurrency(bankAccount.getCurrency());
49         return bankAccountRepository.save(account);
50     }
51
52     no usages
53     @DeleteMapping("/{id}")
54     public void deleteAccount(@PathVariable String id){
55         bankAccountRepository.deleteById(id);
56     }
57 }
```

Résultat :

```
localhost:8081/bankAccounts
[
  {
    "id": "e6604ad1-d0d6-4dcf-899d-ea26a76472f0",
    "createdAt": "2023-05-04T16:05:18.340+00:00",
    "balance": 23695.235654023145,
    "currency": "MAD",
    "type": "SAVING_ACCOUNT"
  },
  {
    "id": "dc83271d-194f-4e7a-adce-892e8563fcfd",
    "createdAt": "2023-05-04T16:05:18.700+00:00",
    "balance": 16433.68033859411,
    "currency": "MAD",
    "type": "SAVING_ACCOUNT"
  },
  {
    "id": "004c210a-9293-4499-9778-80bce370bcc7",
    "createdAt": "2023-05-04T16:05:18.706+00:00",
    "balance": 16873.159710990913,
    "currency": "MAD",
    "type": "CURRENT_ACCOUNT"
  },
  {
    "id": "fd25dbb6-d4c1-4852-b7a2-a2bb21c1d976",
    "createdAt": "2023-05-04T16:05:18.717+00:00",
    "balance": 73013.99995982955,
    "currency": "MAD",
    "type": "SAVING_ACCOUNT"
  },
  {
    "id": "7173be79-013d-43a7-8088-5248691d6b94",
    "createdAt": "2023-05-04T16:05:18.721+00:00",
    "balance": 16873.159710990913,
    "currency": "MAD",
    "type": "CURRENT_ACCOUNT"
  }
]
```

localhost:8081/bankAccounts/e6604ad1-d0d6-4dcf-899d-ea26a76472f0

```
{
  "id": "e6604ad1-d0d6-4dcf-899d-ea26a76472f0",
  "createdAt": "2023-05-04T16:05:18.340+00:00",
  "balance": 23695.235654023145,
  "currency": "MAD",
  "type": "SAVING_ACCOUNT"
}
```

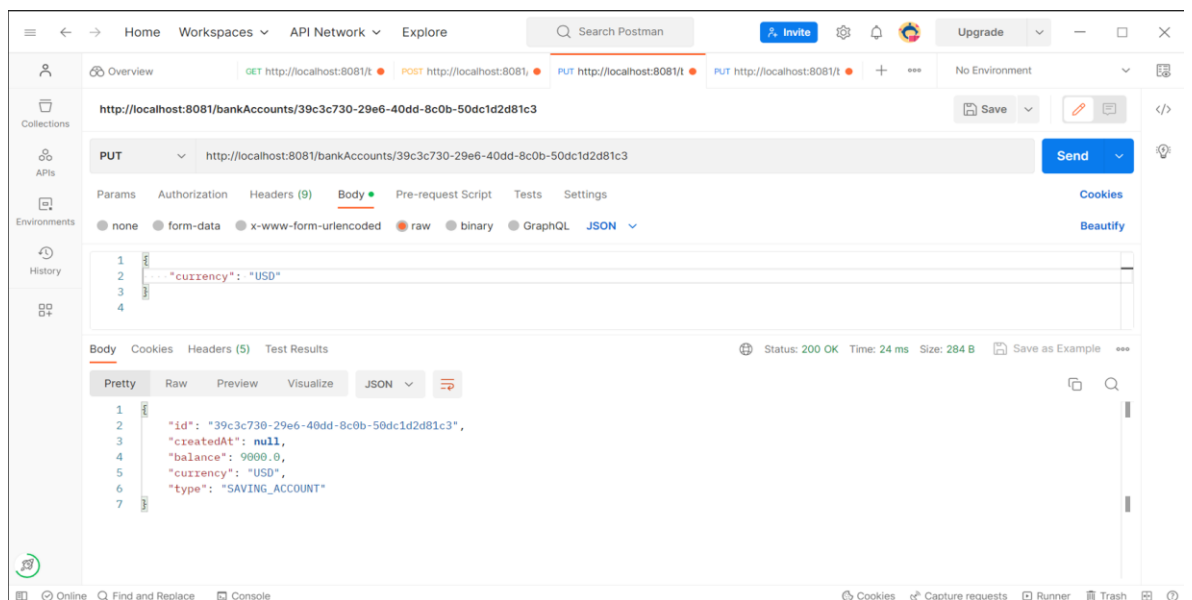
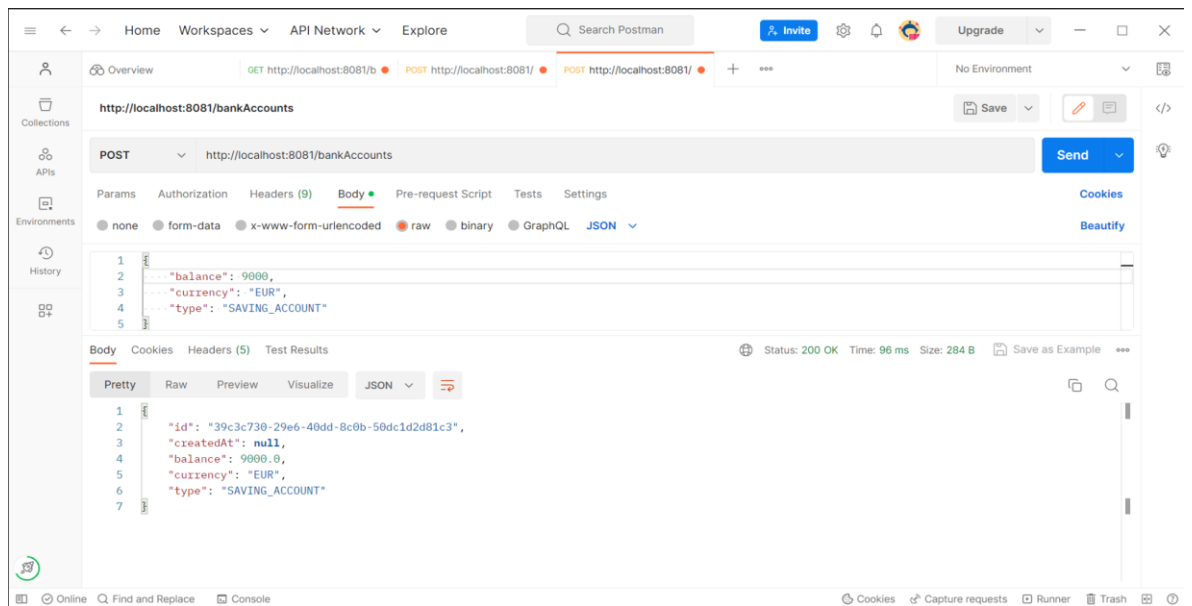
6. Tester le web micro-service en utilisant un client REST comme Postman

The screenshot shows the Postman interface with a GET request to `http://localhost:8081/bankAccounts`. The response status is 200 OK, and the body is a JSON array of 3 bank accounts. The response is displayed in the 'Pretty' format.

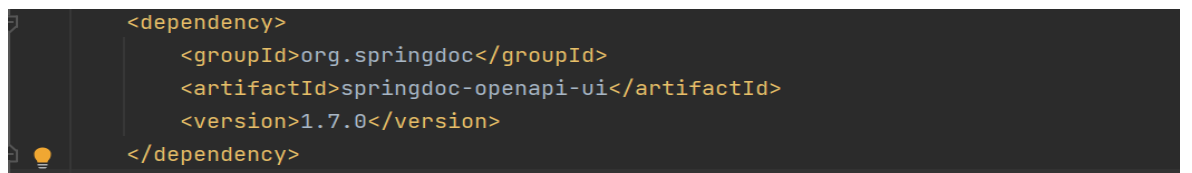
```
1 {
2   {
3     "id": "2e389907-33dc-4fab-a4f2-b6b758dc48a1",
4     "createdAt": "2023-05-04T16:29:22.916+00:00",
5     "balance": 46369.263133720306,
6     "currency": "MAD",
7     "type": "CURRENT_ACCOUNT"
8   },
9   {
10    "id": "a405347f-16ae-481a-8e64-3ae637fe13f7",
11    "createdAt": "2023-05-04T16:29:23.051+00:00",
12    "balance": 63316.59157238687,
13    "currency": "MAD",
14    "type": "CURRENT_ACCOUNT"
15  },
16  {
17    "id": "d257e8ac-56b4-42cb-a90a-d5839dc147ea",
18    "createdAt": "2023-05-04T16:29:23.054+00:00",
19    "balance": 99441.87175070489,
```

The screenshot shows the Postman interface with a GET request to `http://localhost:8081/bankAccounts/2e389907-33dc-4fab-a4f2-b6b758dc48a1`. The response status is 200 OK, and the body is a JSON object representing a single bank account. The response is displayed in the 'Pretty' format.

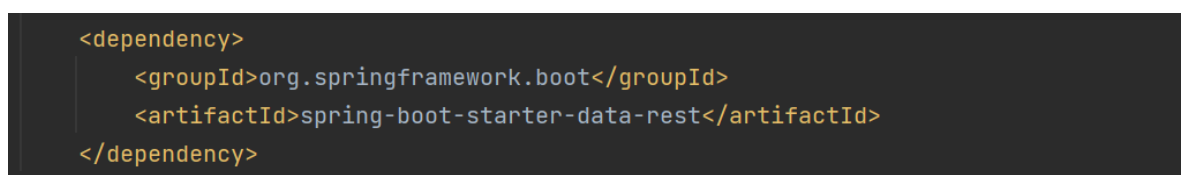
```
1 {
2   "id": "2e389907-33dc-4fab-a4f2-b6b758dc48a1",
3   "createdAt": "2023-05-04T16:29:22.916+00:00",
4   "balance": 46369.263133720306,
5   "currency": "MAD",
6   "type": "CURRENT_ACCOUNT"
7 }
```



7. Générer et tester le documentation Swagger de des API Rest du Web service



8. Exposer une API Restful en utilisant Spring Data Rest en exploitant des projections




```
← → ↻ localhost:8081/bankAccounts?page=0&size=2

{
  "_embedded": {
    "bankAccounts": [ {
      "createdAt": "2023-05-04T19:10:34.732+00:00",
      "balance": 62072.66064966165,
      "currency": "MAD",
      "type": "CURRENT_ACCOUNT",
      "_links": {
        "self": {
          "href": "http://localhost:8081/bankAccounts/04384f98-24b3-44f8-b641-284a0d9697d"
        },
        "bankAccount": {
          "href": "http://localhost:8081/bankAccounts/04384f98-24b3-44f8-b641-284a0d9697d"
        }
      }
    }, {
      "createdAt": "2023-05-04T19:10:34.901+00:00",
      "balance": 85025.60710791558,
      "currency": "MAD",
      "type": "CURRENT_ACCOUNT",
      "_links": {
        "self": {
          "href": "http://localhost:8081/bankAccounts/d108bf17-b697-431d-9c87-c89affcf1b44"
        },
        "bankAccount": {
          "href": "http://localhost:8081/bankAccounts/d108bf17-b697-431d-9c87-c89affcf1b44"
        }
      }
    }
  ],
  "_links": {
    "first": {
      "href": "http://localhost:8081/bankAccounts?page=0&size=2"
    },
    "self": {
      "href": "http://localhost:8081/bankAccounts?page=0&size=2"
    },
    "next": {
      "href": "http://localhost:8081/bankAccounts?page=1&size=2"
    },
    "last": {
      "href": "http://localhost:8081/bankAccounts?page=4&size=2"
    },
    "profile": {
      "href": "http://localhost:8081/profile/bankAccounts"
    }
  },
  "page": {
    "size": 2,
    "totalElements": 10,
    "totalPages": 5,
    "number": 0
  }
}
```

```
itory.java × AccountService.java × AccountServiceImpl.java × BankAccountResponseDTO.java × AccountProjection.java ×
1 package ma.enset.bankaccountservice.entities;
2
3 import ma.enset.bankaccountservice.enums.AccountType;
4 import org.springframework.data.rest.core.config.Projection;
5
6 no usages
7 @Projection(types= BankAccount.class, name="p1")
8 public interface AccountProjection {
9     no usages
10     public String getId();
11     no usages
12     public AccountType getType();
13 }
14
```

```
← → ↻ localhost:8081/bankAccounts?projection=p1

{
  "_embedded": {
    "bankAccounts": [ {
      "id": "4418d63e-2871-44e7-a690-0dab846c324c",
      "type": "SAVING_ACCOUNT",
      "_links": {
        "self": {
          "href": "http://localhost:8081/bankAccounts/4418d63e-2871-44e7-a690-0dab846c324c"
        },
        "bankAccount": {
          "href": "http://localhost:8081/bankAccounts/4418d63e-2871-44e7-a690-0dab846c324c{?projection}",
          "templated": true
        }
      }
    }, {
      "id": "c0246191-4aef-486d-83c2-d8d34e47decc",
      "type": "SAVING_ACCOUNT",
      "_links": {
        "self": {
          "href": "http://localhost:8081/bankAccounts/c0246191-4aef-486d-83c2-d8d34e47decc"
        },
        "bankAccount": {
          "href": "http://localhost:8081/bankAccounts/c0246191-4aef-486d-83c2-d8d34e47decc{?projection}",
          "templated": true
        }
      }
    }, {
      "id": "d83b743d-10be-449c-adf9-6111244f4f1a",
      "type": "CURRENT_ACCOUNT",
      "_links": {
        "self": {
          "href": "http://localhost:8081/bankAccounts/d83b743d-10be-449c-adf9-6111244f4f1a"
        },
        "bankAccount": {
          "href": "http://localhost:8081/bankAccounts/d83b743d-10be-449c-adf9-6111244f4f1a{?projection}",
          "templated": true
        }
      }
    }
  ],
  "_links": {
    "first": {
      "href": "http://localhost:8081/bankAccounts?projection=p1"
    },
    "self": {
      "href": "http://localhost:8081/bankAccounts?projection=p1"
    },
    "next": {
      "href": "http://localhost:8081/bankAccounts?projection=p1"
    },
    "last": {
      "href": "http://localhost:8081/bankAccounts?projection=p1"
    },
    "profile": {
      "href": "http://localhost:8081/profile/bankAccounts"
    }
  },
  "page": {
    "size": 3,
    "totalElements": 10,
    "totalPages": 4,
    "number": 0
  }
}
```

```

1 package ma.enset.bankaccounts.service.repositories;
2
3 import ma.enset.bankaccounts.service.entities.BankAccount;
4 import ma.enset.bankaccounts.service.enums.AccountType;
5 import org.springframework.data.jpa.repository.JpaRepository;
6 import org.springframework.data.repository.query.Param;
7 import org.springframework.data.rest.core.annotation.RepositoryRestResource;
8 import org.springframework.data.rest.core.annotation.RestResource;
9
10 import java.util.List;
11
12
13 @RepositoryRestResource
14 public interface BankAccountRepository extends JpaRepository<BankAccount, String> {
15     @RestResource(path="/byType")
16     List<BankAccount> findByType(@Param("t") AccountType type);
17 }
18

```

```

localhost:8081/bankAccounts/search/byType?t=CURRENT_ACCOUNT

{
  "_embedded": {
    "bankAccounts": [ {
      "createdAt": "2023-05-04T19:46:22.587+00:00",
      "balance": 11514.18978821456,
      "currency": "MAD",
      "type": "CURRENT_ACCOUNT",
      "_links": {
        "self": {
          "href": "http://localhost:8081/bankAccounts/ebc1ea29-d36d-462d-b830-eb47743e4344"
        },
        "bankAccount": {
          "href": "http://localhost:8081/bankAccounts/ebc1ea29-d36d-462d-b830-eb47743e4344{?projection}",
          "templated": true
        }
      }
    }, {
      "createdAt": "2023-05-04T19:46:22.745+00:00",
      "balance": 33282.52427581932,
      "currency": "MAD",
      "type": "CURRENT_ACCOUNT",
      "_links": {
        "self": {
          "href": "http://localhost:8081/bankAccounts/8ba4397c-360d-4811-9843-289ca4e9336c"
        },
        "bankAccount": {
          "href": "http://localhost:8081/bankAccounts/8ba4397c-360d-4811-9843-289ca4e9336c{?projection}",
          "templated": true
        }
      }
    }, {
      "createdAt": "2023-05-04T19:46:22.752+00:00",
      "balance": 53850.42389399113,
      "currency": "MAD",
      "type": "CURRENT_ACCOUNT",
      "_links": {
        "self": {
          "href": "http://localhost:8081/bankAccounts/0535fe31-ca0d-4d24-a4ef-428635734a25"
        }
      }
    }
  ]
}

```

9. Créer les DTOs et Mappers

```

8 @Component
9 public class AccountMapper {
10     public BankAccountResponseDTO fromBankAccount(BankAccount bankAccount){
11         BankAccountResponseDTO bankAccountResponseDTO=new BankAccountResponseDTO();
12
13         BeanUtils.copyProperties(bankAccount,bankAccountResponseDTO);
14         return bankAccountResponseDTO;
15     }
16 }
17

```

```

13  @Data @NoArgsConstructor @AllArgsConstructor @Builder
14  public class BankAccountRequestDTO {
15
16      no usages
17      private Double balance;
18      no usages
19      private String currency;
20      no usages
21      private AccountType type ;
22  }

```

```

13  @Data @NoArgsConstructor @AllArgsConstructor @Builder
14  public class BankAccountResponseDTO {
15      no usages
16      private String id ;
17      no usages
18      private Date createdAt;
19      no usages
20      private Double balance;
21      no usages
22      private String currency;
23      no usages
24      private AccountType type ;
25  }

```

10. Créer la couche Service (métier) et du micro service

```

6  public interface AccountService {
7      1 usage 1 implementation
8      BankAccountResponseDTO addAccount(BankAccountRequestDTO bankAccountDTO);
9  }

```

```

14  @Service
15  @Transactional
16  public class AccountServiceImpl implements AccountService {
17      2 usages
18      private BankAccountRepository bankAccountRepository;
19      2 usages
20      private AccountMapper accountMapper;
21      no usages
22      public AccountServiceImpl(BankAccountRepository bankAccountRepository, AccountMapper accountMapper) {
23          this.bankAccountRepository = bankAccountRepository;
24          this.accountMapper = accountMapper;
25      }
26
27      1 usage
28      @Override
29      public BankAccountResponseDTO addAccount(BankAccountRequestDTO bankAccountDTO) {
30          BankAccount bankAccount = BankAccount.builder()
31              .id(UUID.randomUUID().toString())
32              .createdAt(new Date())
33              .balance(bankAccountDTO.getBalance())
34              .type(bankAccountDTO.getType())
35              .currency(bankAccountDTO.getCurrency())
36              .build();
37          BankAccount saveBankAccount = bankAccountRepository.save(bankAccount);
38          BankAccountResponseDTO bankAccountResponseDTO = accountMapper.fromBankAccount(saveBankAccount);
39          return bankAccountResponseDTO;
40      }
41  }

```