

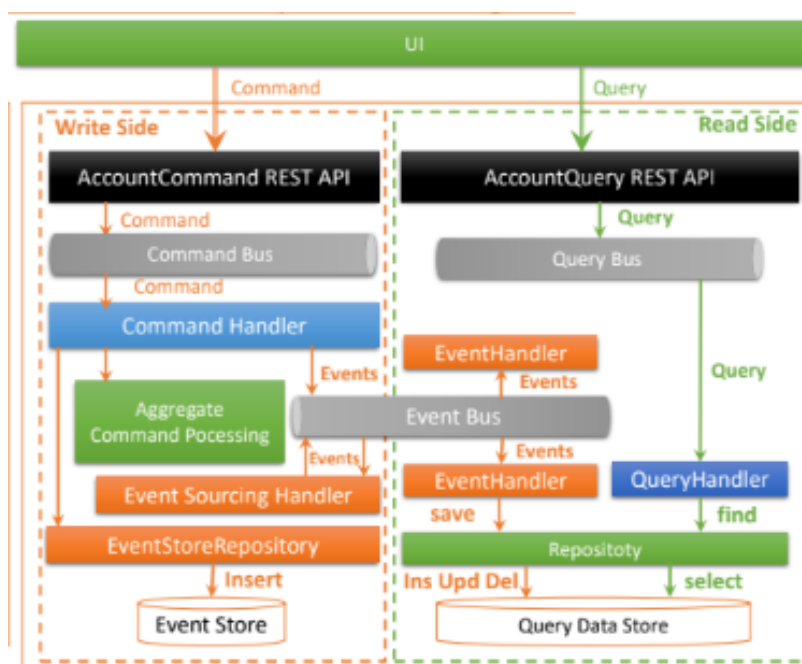
Département Mathématiques et Informatique

Cycle Ingénieur

« Ingénierie Informatique – Big Data et Cloud Computing »

COMPTE-RENDU:

Activité pratique N° 5 - Event Driven Architecture CQRS and Event Sourcing



Réalisé par: Asmaa ELASRI

Encadré par: Pr. Mohamed YOUSSEFI

Année Universitaire : 2022-2023

Travail à faire

Créer une application qui permet de gérer des comptes respectant les patterns CQRS et Event Sourcing avec les Framework AXON et Spring Boot.

PARTIE 1 : Ecriture

1. Creation des commandes (CommonApi)

- BaseCommand:

```
6 public class BaseCommand<T> {
7
8     1 usage
9     @TargetAggregateIdentifier
10    @Getter private T id;
11
12    3 usages
13    public BaseCommand(T id) {
14        this.id = id;
15    }
16 }
```

- CreateAccountCommand

```
5 public class CreateAccountCommand extends BaseCommand<String>{
6     1 usage
7     @Getter
8     private String currency ;
9     1 usage
10    @Getter private double initialSolde;
11    no usages
12    public CreateAccountCommand(String id, String currency, double initialSolde) {
13        super(id);
14        this.currency = currency;
15        this.initialSolde = initialSolde;
16    }
17 }
```

- CreditAccountCommand

```

5 public class CreditAccountCommand extends BaseCommand<String>{
    1 usage
6     @Getter
7     private String currency ;
    1 usage
8     @Getter private double amount;
    no usages
9     public CreditAccountCommand(String id, String currency, double amount) {
10         super(id);
11         this.currency = currency;
12         this.amount = amount;
13     }
14 }

```

- DebitAccountCommand

```

5 public class DebitAccountCommand extends BaseCommand<String>{
    1 usage
6     @Getter
7     private String currency ;
    1 usage
8     @Getter private double amount;
    no usages
9     public DebitAccountCommand(String id, String currency, double amount) {
10         super(id);
11         this.currency = currency;
12         this.amount = amount;
13     }
14 }

```

- Controller CreateAccountCommand (Commands)

```

7 @AllArgsConstructor @NoArgsConstructor
8 @Data
9 public class CreateAccountRequestDTO {
    no usages
10     private String currency ;
    no usages
11     private double initialBalance;
12 }

```

```

21 @RestController
22 @RequestMapping(path = "/commands/account")
23 @AllArgsConstructor
24 public class AccountCommandController {
25     1 usage
26     private CommandGateway commandGateway;
27     no usages
28     @PostMapping(path = "/create")
29     @
30     public CompletableFuture<String> createAccount(CreateAccountRequestDTO request){
31         CompletableFuture<String>commandResponse= commandGateway.send(new CreateAccountCommand(
32             UUID.randomUUID().toString(),
33             request.getCurrency(),
34             request.getInitialBalance()
35         ));
36     }
37 }

```

```

35 @ExceptionHandler(Exception.class)
36 @
37 public ResponseEntity<String> exceptionHandler(Exception exception){
38     ResponseEntity<String>entity= new ResponseEntity<>(
39         exception.getMessage(),
40         HttpStatus.INTERNAL_SERVER_ERROR);
41     return entity;
42 }

```

```

1 spring.application.name=compte-service
2 spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:${MYSQL_PORT:3306}/bank?createDatabaseIfNotExist=true
3 spring.datasource.username=${MYSQL_USER:root}
4 spring.datasource.password=${MYSQL_PASSWORD:}
5 spring.jpa.hibernate.ddl-auto=update
6 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
7 server.port=8082

```

- Base de données (PhpMyAdmin)

Table	Action	Lignes	Type	Interclassement	Taille	Perte
<input type="checkbox"/> association_value_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48,0 kio	-
<input type="checkbox"/> domain_event_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48,0 kio	-
<input type="checkbox"/> hibernate_sequence	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> saga_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	-
<input type="checkbox"/> snapshot_event_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	32,0 kio	-
<input type="checkbox"/> token_entry	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	-
6 tables	Somme	1	InnoDB	utf8mb4_general_ci	176,0 kio	0 o

☐ Tout cocher
 Avec la sélection :

2. Creation des événements (CommonApi)

- BaseEvent

```

5 public abstract class BaseEvent <T>{
6     1 usage
7     @Getter
8     private T id ;
9     1 usage
10    public BaseEvent(T id) { this.id = id; }
11 }
12
13

```

- AccountCreatedEvent

```

6 public class AccountCreatedEvent extends BaseEvent<String>{
7
8     1 usage
9     @Getter
10    private String currency ;
11
12    1 usage
13    @Getter
14    private double initialBalance ;
15
16    1 usage
17    public AccountCreatedEvent(String id, String currency, double initialBalance) {
18        super(id);
19        this.currency = currency;
20        this.initialBalance = initialBalance;
21    }

```

- Creation de l'agrégat (AccountAggregate)

```

14 public class AccountAggregate {
15     1 usage
16     @AggregateIdentifier
17     private String accountId;
18     1 usage
19     private String currency;
20     1 usage
21     private double balance ;
22     1 usage
23     private AccountStatus status;
24
25     no usages
26     public AccountAggregate(){
27         //Required by AXON
28     }

```

- CommandHandler

```

23 @CommandHandler
24 public AccountAggregate(CreateAccountCommand createAccountCommand) {
25     if(createAccountCommand.getInitialBalance()<0) throw new RuntimeException("Impossible...");
26     AggregateLifecycle.apply(new AccountCreatedEvent(
27         createAccountCommand.getId(),
28         createAccountCommand.getCurrency(),
29         createAccountCommand.getInitialBalance()
30     ));
31 }

```

- EventSourcingHandler

```

32         @EventSourcingHandler
33         @ public void on(AccountCreatedEvent event){
34             this.accountId = event.getId();
35             this.balance = event.getInitialBalance() ;
36             this.status = AccountStatus.CREATED;
37             this.currency = event.getCurrency() ;
38         }
39     }
40

```

- Test (Postman)

The screenshot shows the Postman interface for a POST request to `http://localhost:8082/commands/account/create`. The 'Body' tab is selected, and the request body is a JSON object: `{ "initialBalance": 1200, "currency": "MAD" }`. The response is displayed in the 'Test Results' section, showing a 200 OK status with the response body `c1c70a93-3a25-4f05-90f3-2ea9c1da2ac5`.

- EventStore

The screenshot shows a file explorer window displaying a file named `domain_event_entry-payload.html`. The file's content is a long alphanumeric string: `2e655574-5d4b-4478-a181-747877d1f0c80.0`.

- EventStore (Controller)

```

44     @GetMapping("/eventStore/{accountId}")
45     public Stream eventStore(@PathVariable String accountId){
46         return eventStore.readEvents(accountId).asStream() ;
47     }

```

```
localhost:8082/commands/account/eventStore/666187e6-c6ce-4a76-abd4-b1395ce76d6e

[
  {
    "type": "AccountAggregate",
    "aggregateIdentifier": "666187e6-c6ce-4a76-abd4-b1395ce76d6e",
    "sequenceNumber": 0,
    "identifier": "4d61f38c-cfce-42e1-a1da-dab12ae46a69",
    "timestamp": "2022-12-25T16:39:57.930Z",
    "payload": {
      "id": "666187e6-c6ce-4a76-abd4-b1395ce76d6e",
      "currency": null,
      "initialBalance": 0
    },
    "metaData": {
      "traceId": "fd8b2e6a-2080-4431-aafc-5bd8d9102ff2",
      "correlationId": "fd8b2e6a-2080-4431-aafc-5bd8d9102ff2"
    },
    "payloadType": "ma.enset.comptecqrse.commonapi.events.AccountCreatedEvent"
  }
]
```

- AccountActivatedEvent & EventSourcingHandler

```
6 public class AccountActivatedEvent extends BaseEvent<String> {
7     @Getter
8     private AccountStatus status;
9
10    public AccountActivatedEvent(String id, AccountStatus status) {
11        super(id);
12        this.status = status;
13    }
14 }
15
```



```

33     @EventSourcingHandler
34     public void on(AccountCreatedEvent event){
35         this.accountId = event.getId();
36         this.balance = event.getInitialBalance() ;
37         this.status = AccountStatus.CREATED;
38         this.currency = event.getCurrency() ;
39         AggregateLifecycle.apply(new AccountActivatedEvent(
40             event.getId(),
41             AccountStatus.ACTIVATED
42         ));
43     }
44
45     no usages
46     @EventSourcingHandler
47     public void on(AccountActivatedEvent event) {
48         this.status = event.getStatus();
49     }

```

- Test (Postman)

The screenshot shows a Postman interface for a POST request to `http://localhost:8082/commands/account/create`. The request body is a JSON object with `initialBalance` and `currency` fields. The response is a long alphanumeric string.

Tab	Content
POST	http://localhost:8082/commands/account/create
Params	
Authorization	
Headers (9)	
Body	<pre> 1 { 2 "initialBalance": 20000, 3 "currency": "MAD" 4 } </pre>
Pre-request Script	
Tests	
Settings	
Body	<pre> 1 02e43466-2145-4628-8a27-ad0d92332e10 </pre>

```
localhost:8082/commands/account/eventStore/02e43466-2145-4628-8a27-ad0d92332e10

[
  {
    "type": "AccountAggregate",
    "aggregateIdentifier": "02e43466-2145-4628-8a27-ad0d92332e10",
    "sequenceNumber": 0,
    "identifier": "73ca4262-05b0-4b8c-beb7-8c00185d1ec7",
    "timestamp": "2022-12-25T17:30:23.945Z",
    "payload": {
      "id": "02e43466-2145-4628-8a27-ad0d92332e10",
      "currency": null,
      "initialBalance": 0
    },
    "payloadType": "ma.enset.comptecqrse.commonapi.events AccountCreatedEvent",
    "metaData": {
      "traceId": "51a363e8-e4f1-4796-8f78-749020d5b6c4",
      "correlationId": "51a363e8-e4f1-4796-8f78-749020d5b6c4"
    }
  },
  {
    "type": "AccountAggregate",
    "aggregateIdentifier": "02e43466-2145-4628-8a27-ad0d92332e10",
    "sequenceNumber": 1,
    "identifier": "1e45fe4e-27b2-4519-868d-39946f24c785",
    "timestamp": "2022-12-25T17:30:23.958Z",
    "payload": {
      "id": "02e43466-2145-4628-8a27-ad0d92332e10",
      "status": "ACTIVATED"
    },
    "payloadType": "ma.enset.comptecqrse.commonapi.events AccountActivatedEvent",
    "metaData": {
      "traceId": "51a363e8-e4f1-4796-8f78-749020d5b6c4",

```

- AccountCrediteddEvent

```
3 usages
5 public class AccountCreditedEvent extends BaseEvent<String> {
6     1 usage
7     @Getter private double amount;
8     1 usage
9     @Getter private String currency;
10
11     1 usage
12     public AccountCreditedEvent(String id, double amount, String currency) {
13         super(id);
14         this.amount = amount;
15         this.currency = currency;
16     }

```

- EventSourcingHandler

```
53 @CommandHandler
54 @ public void handle(CreditAccountCommand command) {
55     if (command.getAmount() < 0) {
56         throw new AmountNegativeException("Amount should not be negative");
57     }
58
59     AggregateLifecycle.apply(new AccountCreditedEvent(
60         command.getId(),
61         command.getAmount(),
62         command.getCurrency()
63     ));
64 }
no usages
65 @EventSourcingHandler
66 @ public void on(AccountCreditedEvent event) {
67     this.balance += event.getAmount();

```

- **CreditAccount (Controller)**

```

36 @PutMapping(path = "/credit")
37 public CompletableFuture<String> crediteAccount(@RequestBody CreditAccountRequestDTO request){
38     CompletableFuture<String> commandResponse= commandGateway.send(new CreditAccountCommand(
39         request.getAccountId(),
40         request.getCurrency(),
41         request.getAmount()
42     ));
43     return commandResponse;
44 }

```

- **Test (Postman)**

The image shows a Postman interface for a PUT request to `http://localhost:8082/commands/account/credit`. The request body is a JSON object with the following fields:

```

{
  "accountId": "5c19aeb4-5074-426f-972c-bbc7a12df13a",
  "amount": -120,
  "currency": "MAD"
}

```

Below the request, the 'Test Results' section shows a single test failure:

```

1 Amount should not be negative

```

```

{
  "type": "AccountAggregate",
  "aggregateIdentifier": "5c19aeb4-5074-426f-972c-bbc7a12df13a",
  "sequenceNumber": 3,
  "identifier": "07c8f5df-500c-471c-bc76-b240d25dd18a",
  "timestamp": "2022-12-25T18:21:00.587Z",
  "payloadType": "ma.enset.comptecqrses.commonapi.events.AccountCreditedEvent",
  "metaData": {
    "traceId": "4a8e5190-2b18-4c43-b676-fcdac401d4c7",
    "correlationId": "4a8e5190-2b18-4c43-b676-fcdac401d4c7"
  },
  "payload": {
    "id": "5c19aeb4-5074-426f-972c-bbc7a12df13a",
    "amount": 122000,
    "currency": "MAD"
  }
}

```

- **AccountDebitedEvent**

```

5      public class AccountDebitedEvent extends BaseEvent<String> {
6          1 usage
7          @Getter private double amount;
8          1 usage
9          @Getter private String currency;
10
11         1 usage
12         public AccountDebitedEvent(String id, double amount, String currency) {
13             super(id);
14             this.amount = amount;
15             this.currency = currency;
16         }

```

- **CommandHandler & EventSourcingHandler**

```

73      @CommandHandler
74      public void handle(DebitAccountCommand command) {
75          if (command.getAmount() < 0) {
76              throw new AmountNegativeException("Amount should not be negative");
77          }
78          if(this.balance<command.getAmount()) throw new BalanceNotSufficientException("Balance not sufficient Exception =>" + balance);
79
80          AggregateLifecycle.apply(new AccountDebitedEvent(
81              command.getId(),
82              command.getAmount(),
83              command.getCurrency()
84          ));
85      }
86      no usages
87      @EventSourcingHandler
88      public void on(AccountDebitedEvent event) {
89          this.balance -= event.getAmount();
90      }

```

- **DebitAccount (Controller)**

```

45      @PostMapping(path = "/debit")
46      public CompletableFuture<String> debitAccount(@RequestBody DebitAccountRequestDTO request) {
47          CompletableFuture<String> commandResponse = commandGateway.send(new DebitAccountCommand(
48              request.getAccountId(),
49              request.getCurrency(),
50              request.getAmount()));
51          return commandResponse;
52      }

```

- **Test (Postman)**

PUT http://localhost:8082/commands/account/debit

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

```

1      {
2          "accountId": "5c19aeb4-5074-426f-972c-bbc7a12df13a",
3          "amount": 1200000000,
4          "currency": "MAD"
5      }

```

Body Cookies Headers (4) Test Results Status: 500

Pretty Raw Preview Visualize Text

```

1      Balance not sufficient Exception =>133420.6

```

PUT http://localhost:8082/commands/account/debit

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

```
1 ...
2 ... "accountId": "5c19aeb4-5074-426f-972c-bbc7a12df13a",
3 ... "amount": -1200,
4 ... "currency": "MAD"
5 ...
```

Body Cookies Headers (4) Test Results Status: 500

Pretty Raw Preview Visualize Text

1 Amount should not be negative

```
{
  "type": "AccountAggregate",
  "aggregateIdentifier": "5c19aeb4-5074-426f-972c-bbc7a12df13a",
  "sequenceNumber": 4,
  "identifier": "3216d3d9-ee8a-4908-b3bc-644644326794",
  "timestamp": "2022-12-25T18:42:28.763Z",
  "metaData": {
    "traceId": "40746deb-314a-42a9-b05b-0f28d5bc5707",
    "correlationId": "40746deb-314a-42a9-b05b-0f28d5bc5707"
  },
  "payloadType": "ma.enset.comptecqrses.commonapi.events.AccountDebitedEvent",
  "payload": {
    "id": "5c19aeb4-5074-426f-972c-bbc7a12df13a",
    "amount": 12000,
    "currency": "MAD"
  }
}
```

PARTIE 2 : Lecture

- Entity Account

```

14  @AllArgsConstructor @NoArgsConstructor
15  public class Account {
16      no usages
17      @Id
18      private String id;
19      no usages
20      private double balance ;
21      no usages
22      @Enumerated(EnumType.STRING)
23      private AccountStatus status;
24      no usages
25      private String currency ;
26      no usages
27      @OneToMany(mappedBy = "account")
28      private Collection<Operation> operations;

```

- Entity Operation

```

12  public class Operation {
13
14      no usages
15      @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
16      private Long id;
17      no usages
18      private Date date;
19      no usages
20      private double amount;
21      no usages
22      @Enumerated(EnumType.STRING)
23      private OperationType type ;
24      no usages
25      @ManyToOne
26      private Account account;
27  }

```

- Repository AccountRepository

```

7  public interface AccountRepository extends JpaRepository<Account, String> {
8
9  }

```

- Repository OperationRepository

```

7  public interface OperationRepository extends JpaRepository<Operation, String> {
8
9  }

```

- AccountServiceHandler (AccountCreatedEvent)

```

22      @EventHandler
23      @ public void on(AccountCreatedEvent event) {
24          log.info("*****");
25          log.info("AccountCreatedEvent received");
26          Account account = new Account();
27          account.setId(event.getId());
28          account.setBalance(event.getInitialBalance());
29          account.setCurrency(event.getCurrency());
30          account.setStatus(event.getStatus());
31          accountRepository.save(account);
32      }

```

```

2022-12-25 20:30:51.368 INFO 11516 --- [very.service]-0] m.e.c.c.q.service.AccountServiceHandler : AccountCreatedEvent received
2022-12-25 20:30:51.395 INFO 11516 --- [very.service]-0] m.e.c.c.q.service.AccountServiceHandler : *****
2022-12-25 20:30:51.396 INFO 11516 --- [very.service]-0] m.e.c.c.q.service.AccountServiceHandler : AccountCreatedEvent received
2022-12-25 20:30:51.411 INFO 11516 --- [very.service]-0] m.e.c.c.q.service.AccountServiceHandler : *****
2022-12-25 20:30:51.411 INFO 11516 --- [very.service]-0] m.e.c.c.q.service.AccountServiceHandler : AccountCreatedEvent received

```

- **AccountServiceHandler (AccountActivatedEvent, AccountDebitedEvent, AccountCreditedEvent)**

```

40      @EventHandler
41      @ public void on(AccountActivatedEvent event) {
42          log.info("*****");
43          log.info("AccountActivatedEvent received");
44          Account account = accountRepository.findById(event.getId()).get();
45          account.setStatus(event.getStatus());
46          accountRepository.save(account);
47      }

```

```

50      @EventHandler
51      @ public void on(AccountDebitedEvent event) {
52          log.info("*****");
53          log.info("AccountDebitedEvent received");
54          Account account = accountRepository.findById(event.getId()).get();
55          Operation operation=new Operation();
56          operation.setAmount(event.getAmount());
57          operation.setDate(new Date());
58          operation.setType(OperationType.DEBIT);
59          operation.setAccount(account);
60          operationRepository.save(operation);
61          account.setBalance(account.getBalance() - event.getAmount());
62          accountRepository.save(account);
63      }

```

```

66     @EventHandler
67     public void on(AccountCreditedEvent event) {
68         log.info("*****");
69         log.info("AccountCreditedEvent received");
70         Account account = accountRepository.findById(event.getId()).get();
71         Operation operation = new Operation();
72         operation.setAmount(event.getAmount());
73         operation.setDate(new Date());
74         operation.setType(OperationType.CREDIT);
75         operation.setAccount(account);
76         operationRepository.save(operation);
77         account.setBalance(account.getBalance() + event.getAmount());
78         accountRepository.save(account);
79     }

```

- Contrôleur pour la lecture (AllAccounts) et Contrôleur pour la lecture (getAccountById)

```

21 @RequestMapping(path = "/query/accounts")
22 public class AccountQueryController {
23     private QueryGateway queryGateway;
24
25     @GetMapping(path = "/allAccounts")
26     public List<Account> accountList() {
27         return queryGateway.query(new GetAllAccountsQuery(), ResponseTypes.multipleInstancesOf(Account.class)).join();
28     }
29
30     @GetMapping(path = "/getAccount/{id}")
31     public Account getAccount(@PathVariable String id) {
32         return queryGateway.query(new GetAccountQuery(id), ResponseTypes.instanceOf(Account.class)).join();
33     }
34

```

```

81     @QueryHandler
82     public List<Account> on(GetAllAccountsQuery query) {
83         return accountRepository.findAll();
84     }
85
86     @QueryHandler
87     public Account on(GetAccountQuery query) {
88         return accountRepository.findById(query.getId()).get();
89     }
90

```