

TRAVAUX PRATIQUES POUR LES SÉANCES

DU 19/01/2025 => 23/02/2025

A. ACTIVATION DES LOGS

1. DANS SRC>MAIN>RESOURCES, AJOUTER LE FICHIER DE CONFIGURATION LOGBACK-SPRING.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

  <property name="LOGS" value="./logs" />

  <appender name="Console"
    class="ch.qos.logback.core.ConsoleAppender">
    <layout class="ch.qos.logback.classic.PatternLayout">
      <Pattern>
        %black(%d{ISO8601}) %highlight(%-5level)
[%blue(%t)] %yellow(%C{1}): %msg%n%throwable
      </Pattern>
    </layout>
  </appender>

  <appender name="RollingFile"
    class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>${LOGS}/spring-boot-logger.log</file>
    <encoder
      class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
        <Pattern>%d %p %C{1} [%t] %m%n</Pattern>
      </encoder>

      <rollingPolicy
        class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
          <!-- rollover daily and when the file reaches 10
MegaBytes -->
          <fileNamePattern>${LOGS}/archived/spring-boot-logger-%d{yyyy-MM-dd}.%i.log
        </fileNamePattern>
        <timeBasedFileNamingAndTriggeringPolicy
          class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
          <maxFileSize>10MB</maxFileSize>
        </timeBasedFileNamingAndTriggeringPolicy>
      </rollingPolicy>
    </appender>
```

```

<!-- LOG everything at INFO level -->
<root level="info">
    <appender-ref ref="RollingFile" />
    <appender-ref ref="Console" />
</root>

<logger name="ma.gov.pfe" level="trace"
additivity="false">
    <appender-ref ref="RollingFile" />
    <appender-ref ref="Console" />
</logger>

</configuration>

```

2. DANS LE FICHIER APPLICATION.YAML, DÉFINIR LE NIVEAU DE LOG POUR LES PACKAGES DE VOTRE PROJET

```

logging:
  level:
    root: INFO
    ma.gov.pfe: DEBUG
    org.springframework: INFO

```

3. DANS LES CLASSES : CONTROLLERS, SERVICES AJOUTER L'ANNOTATION DE LOMBOK @Slf4J

```

@RequestMapping("/api/v1/users")
@RestController
@AllArgsConstructor
@Slf4j
public class UserController {
    ...
}

```

4. DANS TOUTES, LES MÉTHODES DU CONTROLLER ET DE SERVICE AJOUTER DES TRACES **DEBUG** POUR DÉBUT ET FIN DE MÉTHODE.

```

@PostMapping
public UserDto add(@RequestBody @Valid UserDto userDto) {
    log.debug(" start add user {} ",userDto);
    UserDto userDto1 = null;
    userDto1 = userService.add(userDto);
    log.debug(" end add user {} ",userDto1);
    return userDto1;
}

```

5. LANCER VOTRE PROJET ET VÉRIFIER LES LOGS DANS LA CONSOLE.

```

2025-02-22 13:48:46,127 WARN [main] o.s.b.a.o.j.JpaBaseConfiguration$JpaWebConfiguration: spring.jpa.open-in-view is enabled
2025-02-22 13:48:46,714 INFO [main] o.a.j.l.DirectJDKLog: Starting ProtocolHandler ["http-nio-9091"]
2025-02-22 13:48:46,744 INFO [main] o.s.b.w.e.t.TomcatWebServer: Tomcat started on port(s): 9091 (http) with context path ''
2025-02-22 13:48:46,762 INFO [main] o.s.b.StartupInfoLogger: Started MyApplication in 8.208 seconds (JVM running for 9.147)
2025-02-22 13:48:46,767 INFO [main] m.g.p.MyApplication: Access URLs:

-----
Url Swagger: — http://127.0.0.1:9091/swagger-ui/index.html
Url H2 Base: — http://127.0.0.1:9091/h2-console
-----

2025-02-22 13:49:01,078 INFO [http-nio-9091-exec-1] o.a.j.l.DirectJDKLog: Initializing Spring DispatcherServlet 'dispatcherS
2025-02-22 13:49:01,080 INFO [http-nio-9091-exec-1] o.s.w.s.FrameworkServlet: Initializing Servlet 'dispatcherServlet'
2025-02-22 13:49:01,082 INFO [http-nio-9091-exec-1] o.s.w.s.FrameworkServlet: Completed initialization in 2 ms
2025-02-22 13:49:01,276 DEBUG [http-nio-9091-exec-1] m.g.p.c.UserController: start add UserDto(id=0, nom=null, email=string,
Hibernate: select userentity0_.id as id1_0_0_, userentity0_.address as address2_0_0_, userentity0_.email as email3_0_0_, user
2025-02-22 13:49:01,533 ERROR [http-nio-9091-exec-1] m.g.p.e.GlobalExceptionHandler: apiError ApiError(code=C200, message=[Em
List of constraint violations:[

```

- ❖ AJOUTER DES LOGS DE TYPE **ERROR** POUR AFFICHER LES ERREURS EN CAS DE:
 - SI LES CHAMPS DE USERDTO SONT VIDES.
 - SI LE MOT DE PASSE SAISI NE CONTIENT AUCUN CARACTÈRE SPÉCIAL ET SI TAILLE EST INFÉRIEURE À 8.
- ❖ AJOUTER DES LOGS DE TYPE **INFO** POUR AFFICHER L'IDENTIFIANT DE L'UTILISATEUR AJOUTÉ DANS LA BASE.

B. ACTIVATION DES PROFILS SPRING

SUPPOSONS QU'ON A TROIS ENVIRONNEMENTS DE TRAVAIL : **DEV**, **QUALIF** ET **PRODUCTION**.

6. DANS SRC>MAIN>RESOURCES, CRÉER UN FICHIER DE CONFIGURATION PAR ENVIRONNEMENT:

Application-dev.yaml pour l'environnement de DEV:

```

server:
  port: 9080

logging:
  level:
    root: DEBUG
    ma.gov.pfe: DEBUG
    org.springframework: DEBUG

spring:
  datasource:
    driver-class-name: org.h2.Driver
    url: jdbc:h2:file:~/pfe6
    serverName: localhost
    username: sa
    password: password

```

Application-qua.yaml pour l'environnement de qualif:

```

server:
  port: 9070

```

```
logging:
  level:
    root: INFO
    ma.gov.pfe: DEBUG
    org.springframework: INFO

spring:
  datasource:
    driver-class-name: org.h2.Driver
    url: jdbc:h2:file:~/pfe5
    serverName: localhost
    username: sa
    password: password
```

Application-prod.yaml pour l'environnement de PROD:

```
server:
  port: 9050

logging:
  level:
    root: INFO
    ma.gov.pfe: INFO
    org.springframework: INFO

spring:
  datasource:
    driver-class-name: org.h2.Driver
    url: jdbc:h2:file:~/pfe6
    serverName: localhost
    username: sa
    password: password
```

SUR LE FICHIER APPLICATION.YAML METTRE LE PROFIL QUI SERA ACTIVÉ AU DÉMARRAGE.

```
spring:
  profiles:
    active: dev
```

DÉMARRER L'APPLICATION ET VÉRIFIER SUR LES LOGS LE PROFIL UTILISÉ .

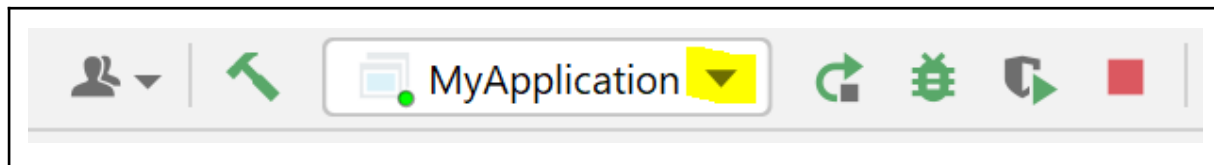
```

2025-02-22 16:16:45,414 DEBUG [background-preinit] o.j.l.LoggerProviders: Logging Provider: org.jboss.logging.Log
2025-02-22 16:16:45,418 INFO [background-preinit] o.h.v.i.u.Version: HV000001: Hibernate Validator 6.2.5.Final
2025-02-22 16:16:45,427 DEBUG [background-preinit] o.h.v.i.x.c.ValidationXmlParser: Trying to load META-INF/val
2025-02-22 16:16:45,429 DEBUG [background-preinit] o.h.v.i.x.c.ResourceLoaderHelper: Trying to load META-INF/val
2025-02-22 16:16:45,429 INFO [main] o.s.b.StartupInfoLogger: Starting MyApplication using Java 1.8.0_333 on MA-
2025-02-22 16:16:45,429 DEBUG [background-preinit] o.h.v.i.x.c.ResourceLoaderHelper: Trying to load META-INF/val
2025-02-22 16:16:45,430 DEBUG [background-preinit] o.h.v.i.x.c.ValidationXmlParser: No META-INF/validation.xml f
2025-02-22 16:16:45,430 DEBUG [main] o.s.b.StartupInfoLogger: Running with Spring Boot v2.7.18, Spring v5.3.31
2025-02-22 16:16:45,431 INFO [main] o.s.b.SpringApplication: The following 1 profile is active: "dev"
2025-02-22 16:16:45,432 DEBUG [main] o.s.b.SpringApplication: Loading source class ma.gov.pfe.MyApplication

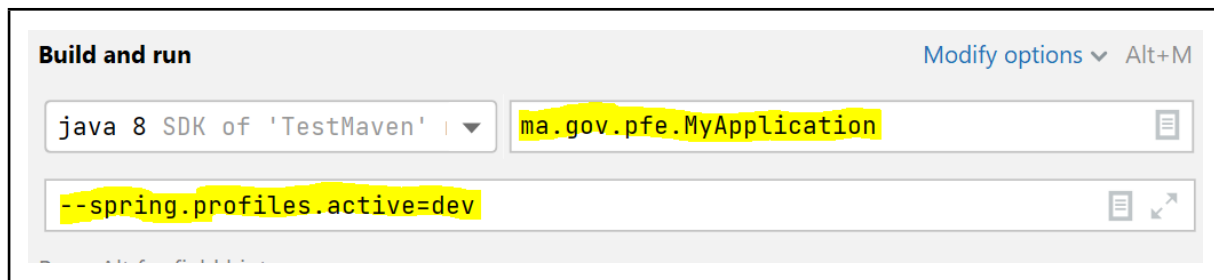
```

IL EST AUSSI POSSIBLE D'ACTIVER UN PROFIL PAR COMMANDE DANS LES ARGUMENTS DU PROGRAMME.

CLIQUER SUR LA FLÈCHE DANS L'OUTIL DE LANCEMENT DE L'APPLICATION. PUIS EDIT CONFIGURATION



SUR PROGRAMME ARGUMENTS METTRE : --SPRING.PROFILES.ACTIVE=DEV



C. CRÉATION DES COMPOSANTS DES PROFILS SPRING

SUPPOSONS QUE DANS NOTRE APPLICATION ON VA ENVOYER DES FICHIERS PAR CFT ([CROSS FILE TRANSFER — WIKIPÉDIA](#)) ET QUE CETTE SOLUTION EST DISPONIBLE EN QUALIF ET PROD ET N'EST PAS DISPONIBLE EN DEV.

1. CRÉER UNE INTERFACE DANS LA COUCHE DES SERVICES POUR L'ENVOI DU FICHIER.

```

PACKAGE MA.GOV.PFE.SERVICES;

PUBLIC INTERFACE ICFTService {

    VOID SENDFILEPGP (STRING PATHFILE);
}

```

2. CRÉER UNE CLASSE D'IMPLÉMENTATION À UTILISER EN QUAL ET PROD POUR UN ENVOI REAL DU FICHIER EN CFT (LA SOLUTION CFT EXISTE SUR CES DEUX ENV)

```

@Service
@Profile({"QUA", "PROD"})
@Slf4j
public class CFTServiceImplReal implements ICFTService {

    public CFTServiceImplReal() {
        log.debug("CONSTRUCTOR CFTServiceImplReal");
    }

    @Override
    public void sendFilePgp(String pathFile) {
        log.debug("REAL SEND FILE BY CFT");
    }
}

```

3. CRÉER UNE CLASSE D'IMPLÉMENTATION À UTILISER EN DEV POUR SIMULER UN ENVOI CFT (MOCK CFT)

```

@Service
@ConditionalOnMissingBean(CFTServiceImplReal.class)
@Slf4j
public class CFTServiceImplMock implements ICFTService {

    public CFTServiceImplMock() {
        log.debug("CONSTRUCTOR CFTServiceImplMock");
    }

    @Override
    public void sendFilePgp(String pathFile) {
        log.debug("SIMULATION SEND FILE BY CFT");
    }
}

```

4. LANCER L'APPLICATION AVEC LE PROFIL QUAL OU PORD ET VÉRIFIER DANS LES LOGS LE BEAN ICFT SERVICE CRÉÉ : VOUS DEVRIEZ AVOIR **CONSTRUCTOR CFTServiceImplReal**
5. LANCER L'APPLICATION AVEC LE PROFIL DEV ET VÉRIFIER DANS LES LOGS LE BEAN ICFT SERVICE CRÉÉ : VOUS DEVRIEZ AVOIR **CONSTRUCTOR CFTServiceImplMock**.

D. GESTION DES EXCEPTIONS

MAINTENANT ON VA SUPPOSER QUE

- LE MAIL DE L'UTILISATEUR DOIT ÊTRE UNIQUE AU NIVEAU DE LA BASE,
 - LE MAIL NE PEUT PAS ÊTRE NULL ET DOIT AVOIR UN FORMAT DE MAIL VALIDE.
 - LE NOM NE PEUT PAS ÊTRE NULL ET SA TAILLE EST ENTRE 2 ET 30.
 - L'ADRESSE NE PEUT PAS ÊTRE NULL ET SA TAILLE MAX EST 255.
 -
1. CHANGER L'ENTITY UserEntity EN AJOUTANT LES ANNOTATIONS DE VALIDATION JAVAX OU JAKARTA SELON LA VERSION JAVA QUE VOUS UTILISEZ.

```

import javax.validation.constraints.Email;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

@Entity
@Table(name = "T_USERS")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class UserEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull(message = "Name cannot be null")
    @Size(min = 2, max = 30, message = "Name must be between 2
and 100 characters")
    private String name;

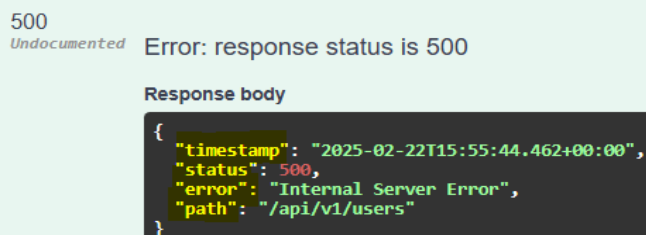
    @NotNull(message = "Address cannot be null")
    @Size(max = 255, message = "Address can be a maximum of 255
characters")
    private String address;

    @NotNull(message = "Email cannot be null")
    @Email(message = "Email should be valid")
    @Column(unique = true, nullable = false)
    private String email;

    private String password;
}

```

2. LANCER L'APPLICATION ET AJOUTER UN UTILISATEUR SANS NOM OU AVEC UN NOM QUI DÉPASSE LA TAILLE MAXIMALE DU NOM. ESSAYER AVEC DES ERREURS SUR LES AUTRES CHAMPS. PAR DÉFAUT VOUS AUREZ



500
Undocumented Error: response status is 500

Response body

```

{
  "timestamp": "2025-02-22T15:55:44.462+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "path": "/api/v1/users"
}

```

VÉRIFIER AUSSI SUR LES LOGS DE LA CONSOLE.

```

2025-02-22 16:55:44,451 ERROR [http-nio-9091-exec-3] o.a.j.l.DirectJDKLog: Servlet.service() for
List of constraint violations:[
— ConstraintViolationImpl{interpolatedMessage='Address cannot be null', propertyPath=address,
— ConstraintViolationImpl{interpolatedMessage='Email should be valid', propertyPath=email, roc
— ConstraintViolationImpl{interpolatedMessage='Name cannot be null', propertyPath=name, rootBe
]] with root cause
javax.validation.ConstraintViolationException: Validation failed for classes [ma.gov.pfe.entitie
List of constraint violations:[
— ConstraintViolationImpl{interpolatedMessage='Address cannot be null', propertyPath=address,
— ConstraintViolationImpl{interpolatedMessage='Email should be valid', propertyPath=email, roc
— ConstraintViolationImpl{interpolatedMessage='Name cannot be null', propertyPath=name, rootBe
]

```

NOTER LE TYPE DE CETTE EXCEPTION : `ConstraintViolationException`

NOUS SOUHAITONS PERSONNALISER L'ERREUR HTTP 500 (ERREUR SERVER) AFFICHER À L'UTILISATEUR ET LA RENDRE ERREUR 400 (BAD REQUEST).

3. DANS LE PACKAGE `DTO`, CRÉER LA CLASSE `ApiError` DÉFINIE COMME SUIVANT:

```

@Data
@Builder
public class ApiError {
    private final String code;
    private final List<String> message;
    private final LocalDate timestamp;
}

```

4. CRÉER UN PACKAGE "EXCEPTIONS" POUR LA GESTION DE TOUTES LES EXCEPTIONS DE VOTRE PROJET.

5. DANS CE PACKAGE, CRÉER LA CLASSE `GlobalExceptionHandler` SUIVANTE:

```

@ControllerAdvice
@Slf4j
public class GlobalExceptionHandler {
    @ExceptionHandler({ConstraintViolationException.class})
    public ResponseEntity<ApiError>
handleValidationException(ConstraintViolationException e) {
        List<String> errorMessages = new ArrayList<>();
        for (ConstraintViolation<?> violation :
e.getConstraintViolations()) {
            errorMessages.add(violation.getMessage());
        }
        ApiError apiError = ApiError
            .builder()
            .code("C200")
            .message(errorMessages)
            .timestamp(LocalDate.now());
    }
}

```



```

        .build();
        log.error("apiError {}", apiError);
        return new ResponseEntity<>(apiError,
        HttpStatus.BAD_REQUEST);
    }
}

```

6. LANCER VOTRE APPLICATION ET ESSAYER D'AJOUTER UN UTILISATEUR NON VALIDE. VOUS AUREZ ERREUR 400 AVEC LES MESSAGES D'ERREUR.

400
Undocumented Error: response status is 400

Response body

```

{
  "code": "C200",
  "message": [
    "Email should be valid",
    "Name cannot be null",
    "Address cannot be null"
  ],
  "timestamp": "2025-02-22"
}

```

7. REMARQUER QUE L'EXCEPTION EST CORRECTEMENT GÉRÉ PAR LE CONTROLLER ADVICE

```

2025-02-22 17:15:01,298 INFO [http-nio-9091-exec-1] o.a.j.l.DirectJDKLog: Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-02-22 17:15:01,299 INFO [http-nio-9091-exec-1] o.s.w.s.FrameworkServlet: Initializing Servlet 'dispatcherServlet'
2025-02-22 17:15:01,300 INFO [http-nio-9091-exec-1] o.s.w.s.FrameworkServlet: Completed initialization in 1 ms
2025-02-22 17:15:01,401 DEBUG [http-nio-9091-exec-1] m.g.p.c.UserController: start add UserDto(id=0, nom=null, email=string, pas
Hibernate: select userentity0_.id as id1_0_0_, userentity0_.address as address2_0_0_, userentity0_.email as email3_0_0_, userenti
2025-02-22 17:15:01,514 ERROR [http-nio-9091-exec-1] m.g.p.e.GlobalExceptionHandler: apiError ApiError(code=C200, message=[Email

```

REMARQUER AUSSI QUE L'APPEL EST ARRIVÉ JUSQU'À LA BASE DE DONNÉE:

```

HIBERNATE: SELECT USERENTITY0_.ID AS ID1_0_0_, USERENTITY0_.ADDRESS AS ADDRESS2_0_0_,
USERENTITY0_.EMAIL AS EMAIL3_0_0_, USERENTITY0_.NAME AS NAME4_0_0_, USERENTITY0_.PASSWORD AS
PASSWORD5_0_0_ FROM T_USERS USERENTITY0_ WHERE USERENTITY0_.ID=?

```

8. AJOUTEZ AUX MESSAGES DE VALIDATION LA CHAÎNE "ENTITY" POUR S'ASSURER QUE LES ERREURS SONT DÉTECTÉES AU NIVEAU REPOSITORIES DE VOTRE PROJET (LA COUCHE BASSE).

9.

```

public class UserEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull(message = "Entity Name cannot be null")
    @Size(min = 2, max = 30, message = "Entity Name must be
between 2 and 30 characters")
    private String name;

    @NotNull(message = "Entity Address cannot be null")

```

```

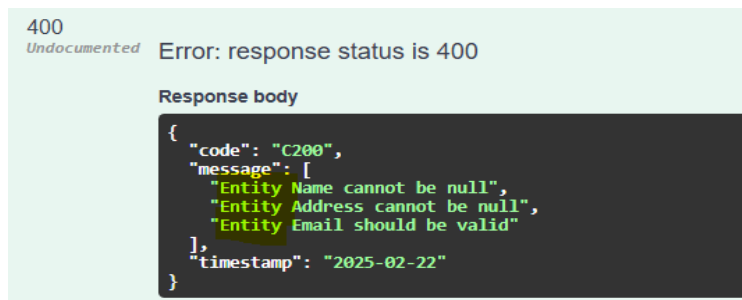
    @Size(max = 255, message = "Entity Address can be a maximum of 255 characters")
    private String address;

    @NotNull(message = "Entity Email cannot be null")
    @Email(message = "Entity Email should be valid")
    @Column(unique = true, nullable = false)
    private String email;

    private String password;

```

VÉRIFIER SI VOUS AVEZ L'ERREUR 400 AVEC CES MESSAGES:



REMARQUER AUSSI QUE L'APPEL EST ARRIVÉ JUSQU'À LA BASE DE DONNÉE:

```

HIBERNATE: SELECT USERENTITY0_.ID AS ID1_0_0_, USERENTITY0_.ADDRESS AS ADDRESS2_0_0_,
USERENTITY0_.EMAIL AS EMAIL3_0_0_, USERENTITY0_.NAME AS NAME4_0_0_, USERENTITY0_.PASSWORD AS
PASSWORD5_0_0_ FROM T_USERS USERENTITY0_ WHERE USERENTITY0_.ID=?

```

OR LE CONTROLLER DOIT D'ABORD VALIDER LE DTO AVANT DE L'ENVOYER À LA BASE. POUR CETTE RAISON NOUS ALLONS AUSSI AJOUTER LES ANNOTATIONS DE VALIDATIONS SUR LES DTOS DE NOTRE PROJET:

10. AJOUTER LES ANNOTATIONS JAVAX.VALIDATION SUR LE USERDTO

```

@Data
@AllArgsConstructor
@NoArgsConstructor
public class UserDto {
    private Long id;

    @NotNull(message = "Dto Name cannot be null")
    @Size(min = 2, max = 30, message = "Dto Name must be between 2 and 30 characters")
    private String name;

    @NotNull(message = "Dto Address cannot be null")
    @Size(max = 255, message = "Dto Address can be a maximum of 255 characters")
    private String address;

```

```

@NotNull(message = "Dto Email cannot be null")
>Email(message = "Dto Email should be valid")
>Column(unique = true, nullable = false)
private String email;

private String password;
}

```

11. AJOUTER L'ANNOTATION `@Valid` À LA MÉTHODE D'AJOUT ET DE MODIFICATION AU NIVEAU DU CONTROLLER :

```

>PostMapping
public UserDto add(@RequestBody @Valid UserDto userDto) {
    log.debug(" start add {} ",userDto);
    UserDto userDto1 = null;
    userDto1 = userService.add(userDto);
    log.debug(" end add {} ",userDto1);
    return userDto1;
}

```

12. LANCER L'APPLICATION ET AJOUTER UN UTILISATEUR NON VALIDE
REMARQUER L'ERREUR 400 ET DANS LES LOGS LE TYPE D'EXCEPTION :
`METHODARGUMENTNOTVALIDEXCEPTION`

13. AJOUTER UNE MÉTHODE POUR GÉRER CETTE EXCEPTION DANS LE CONTROLLER ADVICE

```

>ExceptionHandler(MethodArgumentNotValidException.class)
public ResponseEntity<ApiError>
handleMethodArgumentNotValidException(MethodArgumentNotValidExce
ption e) {
    List<String> errors =
e.getBindingResult().getAllErrors().stream()
        .map(error -> {
            String fieldName = ((FieldError)
error).getField();
            String message = error.getDefaultMessage();
            return fieldName + ": " + message;
        })
        .collect(Collectors.toList());
    ApiError apiError = ApiError
        .builder()
        .code("C200")
        .message(errors)

```

```

        .timestamp(LocalDate.now())
        .build();
    log.error("apiError {}", apiError);
    return new ResponseEntity<>(apiError,
HttpStatus.BAD_REQUEST);
}

```

14. LANCER ET AJOUTER UN UTILISATEUR NON VALIDE.

VÉRIFIER LA RÉPONSE

400
Undocumented Error: response status is 400

Response body

```

{
  "code": "C200",
  "message": [
    "name: Dto Name cannot be null",
    "address: Dto Address cannot be null",
    "email: Dto Email should be valid"
  ],
  "timestamp": "2025-02-22"
}

```

VÉRIFIER QU'ON APPELLE PLUS L'ENTITÉ ET QUE LES EXCEPTIONS DE CE GENRE SONT GÉRÉ AU NIVEAU SUPÉRIEUR (PROCHE COUCHE CONTROLLERS).

Url Swagger: — <http://127.0.0.1:9091/swagger-ui/index.html>

Url H2 Base: — <http://127.0.0.1:9091/h2-console>

```

2025-02-22 17:48:07,208 INFO [http-nio-9091-exec-1] o.a.j.l.DirectJDKLog: Initializing Spring DispatcherServlet '
2025-02-22 17:48:07,208 INFO [http-nio-9091-exec-1] o.s.w.s.FrameworkServlet: Initializing Servlet 'dispatcherSer
2025-02-22 17:48:07,210 INFO [http-nio-9091-exec-1] o.s.w.s.FrameworkServlet: Completed initialization in 1 ms
2025-02-22 17:48:07,392 ERROR [http-nio-9091-exec-1] m.g.p.e.GlobalExceptionHandler: apiError ApiError(code=C200,

```

AUCUNE TRACE SUR L'ACCÈS À LA BASE DE DONNÉE 😊 😊

E. PERSONALISATION DES EXCEPTIONS

SUPPOSONS QU'ON SOUHAITE CRÉER NOTRE PROPRE ANNOTATION POUR LA VALIDATION SELON UN CONTEXTE FONCTIONNEL BIEN DÉFINI.

EXEMPLE : CRÉER UNE ANNOTATION POUR VALIDER LES EMAILS.

15. CRÉER L'ANNOTATION @PASSWORD SUIVANTE :

```

package ma.gov.pfe.validation;

import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

@Retention(RetentionPolicy.RUNTIME)
@Constraint(validatedBy = PasswordValidator.class)
public @interface Password {

```

```

                                String          message()          default
"{javax.validation.constraints.NotValid.password}";

    Class<?>[] groups() default {};

    Class<? extends Payload>[] payload() default {};
}

```

16. CRÉER LA CLASSE DE VALIDATION PASSWORD VALIDATOR

```

package ma.gov.pfe.validation;

import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

@Retention(RetentionPolicy.RUNTIME)
@Constraint(validatedBy = PasswordValidator.class)
public @interface Password {

    String          message()          default
"{javax.validation.constraints.NotValid.password}";

    Class<?>[] groups() default {};

    Class<? extends Payload>[] payload() default {};
}

```

17. AJOUTER UNE MÉTHODE POUR GÉRER CETTE EXCEPTION DANS LE CONTROLLER ADVICE

```

public class PasswordValidator implements
ConstraintValidator<Password, String> {

    @Override
    public void initialize(Password constraintAnnotation) {
    }

    @Override
    public boolean isValid(String password,
ConstraintValidatorContext context) {
        if (password == null) {
            return false;
        }
        boolean hasUpperCase = false;

```

```

        boolean hasLowerCase = false;
        boolean hasDigit = false;
        boolean hasSpecialChar = false;
        for (char c : password.toCharArray()) {
            if (Character.isUpperCase(c)) hasUpperCase = true;
            if (Character.isLowerCase(c)) hasLowerCase = true;
            if (Character.isDigit(c)) hasDigit = true;
            if (!Character.isLetterOrDigit(c)) hasSpecialChar =
true;
        }
        return hasUpperCase && hasLowerCase && hasDigit &&
hasSpecialChar && password.length() >= 8;
    }
}

```

18. UTILISER L'ANNOTATION SUR USERDTO AU NIVEAU DE L'ATTRIBUT PASSWORD.

```

@Password
private String password;

```

19. LANCER L'APPLICATION ET AJOUTER UN UTILISATEUR NON VALIDE. VÉRIFIER LE MESSAGE DE L'ERREUR 400.

Response body

```

{
  "code": "C200",
  "message": [
    "password: {javax.validation.constraints.NotValid.password}",
    "email: Dto Email should be valid",
    "address: Dto Address cannot be null",
    "name: Dto Name cannot be null"
  ],
  "timestamp": "2025-02-22"
}

```