

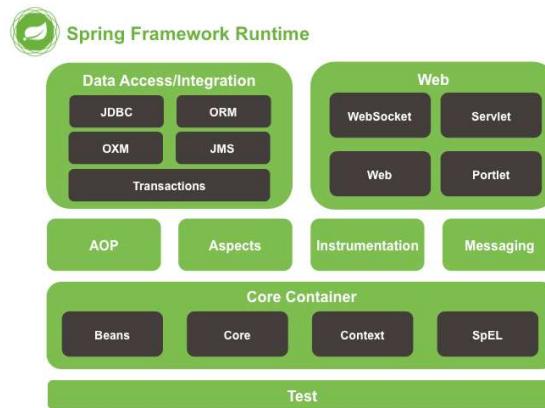
Spring 6

Spring Boot 3.2.1

Spring Framework

- **Spring Framework** est un framework mature, puissant et très flexible axé sur la création d'applications Web en Java.
- Il prend en charge la plupart des aspects de bas niveau de la construction de l'application
 - » concentrer réellement sur les fonctionnalités et la logique métier.
- Il est très activement maintenu et dispose d'une grande communauté de développeurs.
 - » Régulièrement mis à jour et aligné sur l'écosystème Java.

Spring Framework



Spring Framework

- Spring Framework se compose de fonctionnalités organisées en une vingtaine de modules.
- Ces modules sont regroupés en Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging et Test ...
 - **Core** : Fournit des fonctionnalités de base telles que DI (injection de dépendance), internationalisation, validation et AOP (programmation orientée aspect)
 - **Accès aux données** : prend en charge l'accès aux données via JTA (Java Transaction API), JPA (Java Persistence API) et JDBC (Java Database Connectivity)
 - **Web** : prend en charge à la fois l'API Servlet (Spring MVC) et l'API réactive (Spring WebFlux), et prend également en charge WebSockets, STOMP et WebClient
 - **Intégration** : Prend en charge l'intégration à Enterprise Java via JMS (Java Message Service), JMX (Java Management Extension) et RMI (Remote Method Invocation)
 - **Tests** : prise en charge étendue des tests unitaires et d'intégration via des objets fictifs (Mock), des montages de test, la gestion de contexte et la mise en cache

Module Core

- Module **Core** se compose des modules **spring-core**, **spring-beans**, **spring-context**, **spring-context-support** et **spring-expression** (Spring Expression Language).
- Les modules **spring-core** et **spring-beans** fournissent les éléments fondamentaux du framework, y compris les fonctionnalités *IoC* et *Dependency Injection*.
- **BeanFactory**, Il s'agit d'une implémentation sophistiquée du pattern Factory.
 - » vous permet de découpler la configuration et la spécification des dépendances de votre logique de programme.

Module Core

- Le module Context (**spring-context**) : c'est un moyen d'accéder aux objets d'une manière similaire à un registre JNDI (Annuaire).
- Le module **Context** hérite ses fonctionnalités du module **Beans**.
- Le module **Context** ajoute la prise en charge de l'internationalisation, la propagation d'événements, le chargement de ressources et la création transparente de contextes par un *conteneur Servlet*, par exemple.
- L' interface **ApplicationContext** est le point central du module Contexte.

Module Core

- Le module **spring-expression** fournit un langage d'expression puissant pour interroger et manipuler un graphe d'objets lors de l'exécution.
- Il s'agit d'une extension du langage d'expression unifié (unified EL) tel que spécifié dans la spécification JSP 2.1.
- Le langage prend en charge la définition et l'obtention des valeurs de propriété, l'attribution de propriété, l'invocation de méthode, l'accès au contenu des tableaux, des collections et des indexeurs, les opérateurs logiques et arithmétiques, les variables nommées et la récupération d'objets par nom à partir du conteneur IoC de Spring.
- Il prend également en charge la projection et la sélection de listes ainsi que les agrégations de listes courantes.

AOP

- Le module **spring-aop** fournit une implémentation de programmation orientée aspect conforme à Alliance AOP.
- Il permet de définir, par exemple, des intercepteurs de méthode et des points d'accrochage pour découpler proprement le code qui implémente des fonctionnalités qui doivent être séparées.
- Le module séparé **spring-aspects** fournit une intégration avec **AspectJ**.

Écosystème Spring

- Bien que la liste des projets de Spring soit longue et elle change constamment, il y en a quelques-uns qui méritent d'être mentionnés :
- **Boot** : nous fournit un ensemble de modèles très optionnels mais extensibles pour créer divers projets basés sur Spring en un rien de temps. Il est très facile de créer des applications Spring autonomes avec Tomcat intégré ou un conteneur similaire.
- **Cloud** : Fournit une assistance pour développer facilement certains des modèles de système distribué courants tels que service de discovery, le circuit breaker et le gateway API. Cela nous aide à réduire les efforts de déploiement de ces modèles passe-partout sur des plates-formes locales, distantes ou même gérées.

Écosystème Spring

- **Sécurité** : Fournit un mécanisme robuste pour développer l'authentification et l'autorisation pour les projets basés sur Spring de manière hautement personnalisable. Avec un support déclaratif minimal, nous obtenons une protection contre les attaques courantes telles que la fixation de session, le détournement de clics et la falsification de requêtes intersites.
- **Mobile** : Fournit des capacités pour détecter l'appareil et adapter le comportement de l'application en conséquence. De plus, prend en charge la gestion des vues adaptée aux appareils pour une expérience utilisateur optimale, la gestion des préférences de site et le changement de site.
- **Batch** : Fournit un cadre léger pour le développement d'applications de traitements par lots pour les systèmes d'entreprise tels que l'archivage de données. A un support intuitif pour la planification, le redémarrage, le saut, la collecte de métriques et la journalisation. En outre, prend en charge la mise à l'échelle pour les travaux à volume élevé grâce à l'optimisation et au partitionnement.

Spring Boot

26/12/2023

Mohammed Issam KABBAJ

11

Introduction à Spring Boot

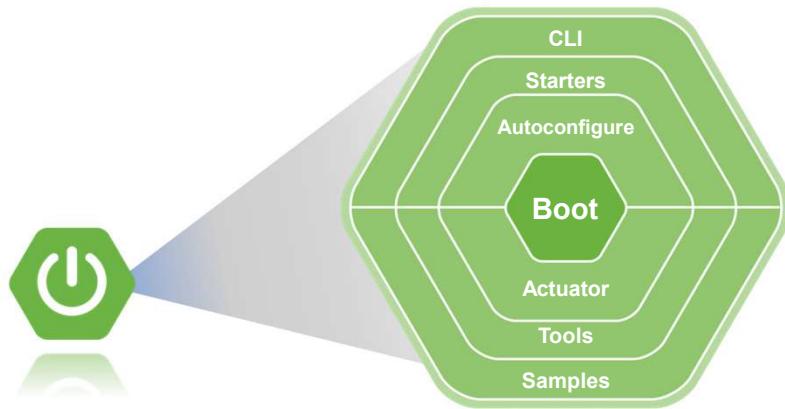
- Un point de focus unique
- Démarrage rapide avec Spring
- Ensemble des besoins non fonctionnelles une application
- Expose plusieurs fonctionnalités par défaut
- Issue rapide si vous voulez changer les valeurs par défaut
- Une façon pour Spring d'avoir une opinion

26/12/2023

Mohammed Issam KABBAJ

12

Spring Boot Modules



26/12/2023

Mohammed Issam KABBAJ

13

Spring Boot Modules

- Spring Boot - bibliothèque principale supportant les autres parties de Spring Boot
- Configuration automatique de démarrage de spring - une seule annotation `@EnableAutoConfiguration` crée un contexte de spring complet
- Spring Boot Starters - Un ensemble de descripteurs de dépendance pratiques que vous pouvez inclure dans votre application.
- Spring Boot CLI - compile et exécute un projet en tant qu'application Spring
- Spring Boot Actuator - fonctionnalités communes non fonctionnelles permettant de déployer une application instantanément et de la prendre en charge en production
- Spring Boot Tools - pour créer et exécuter des archives JAR et WAR autonomes
- Spring Boot Samples - un large éventail d'applications d'exemple

26/12/2023

Mohammed Issam KABBAJ

14

Idées fortes

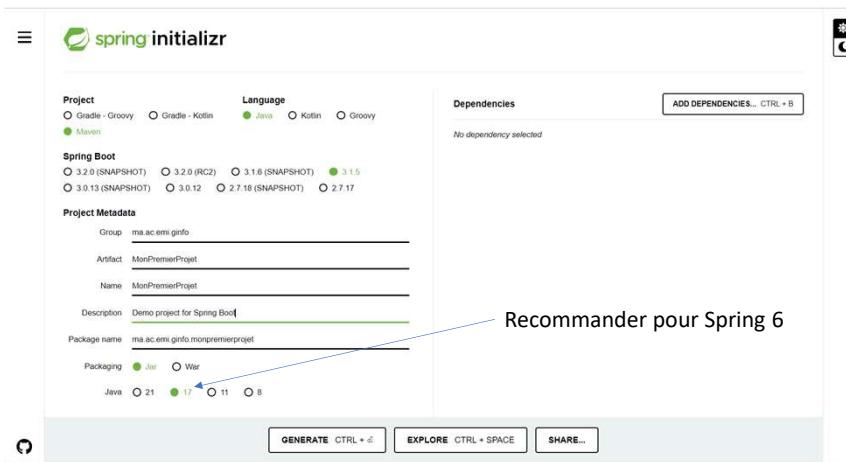
- Accélérer le développement d'applications Spring
- Convention over Configuration
- Pas de code généré
- Déploiement facilité
- Prêt pour la production

26/12/2023

Mohammed Issam KABBAJ

15

Création d'une application Spring



26/12/2023

Mohammed Issam KABBAJ

16

Projet Spring : POM.xml

```

11      <groupId>ma.ac.emi.ginfo</groupId>
12      <artifactId>MonPremierProjet</artifactId>
13      <version>0.0.1-SNAPSHOT</version>
14      <name>MonPremierProjet</name>
15      <description>Demo project for Spring Boot</description>
16      <properties>
17          <java.version>17</java.version>
18      </properties>

```

Projet Spring : configuration initiale

```

5      <parent>
6          <groupId>org.springframework.boot</groupId>
7          <artifactId>spring-boot-starter-parent</artifactId>
8          <version>3.1.5</version>
9          <relativePath/> 
10     </parent>

```

Gestion optimale des dépendances entre les versions des Frameworks nécessaires et utilisées au sein du projet

Projet Spring : ses dépendances

```

19      <dependencies>
20      @↑          <dependency>
21          <groupId>org.springframework.boot</groupId>
22          <artifactId>spring-boot-starter</artifactId>
23      </dependency> ↗ Pas de version spécifique
24          Tous est hérités du projet parent
25      @↑          <dependency>
26          <groupId>org.springframework.boot</groupId>
27          <artifactId>spring-boot-starter-test</artifactId>
28          <scope>test</scope>
29      </dependency>
30  </dependencies>

```

26/12/2023

Mohammed Issam KABBAJ

19

Dépendances 1 niveau

```

pom.xml (SpringBoot1)   spring-boot-starter-3.1.5.pom
43  <dependencies>
44      <dependency>
45          <groupId>org.springframework.boot</groupId>
46          <artifactId>spring-boot</artifactId>
47          <version>3.1.5</version>
48          <scope>compile</scope>
49      </dependency>
50      <dependency>
51          <groupId>org.springframework.boot</groupId>
52          <artifactId>spring-boot-autoconfigure</artifactId>
53          <version>3.1.5</version>
54          <scope>compile</scope>
55      </dependency>
56      <dependency>
57          <groupId>org.springframework.boot</groupId>
58          <artifactId>spring-boot-starter-logging</artifactId>
59          <version>3.1.5</version>
60          <scope>compile</scope>
61      </dependency>
62
63      <dependency>
64          <groupId>jakarta.annotation</groupId>
65          <artifactId>jakarta.annotation-api</artifactId>
66          <version>2.1.1</version>
67          <scope>compile</scope>
68      </dependency>
69      <dependency>
70          <groupId>org.springframework</groupId>
71          <artifactId>spring-core</artifactId>
72          <version>6.0.13</version>
73          <scope>compile</scope>
74      </dependency>
75      <dependency>
76          <groupId>org.yaml</groupId>
77          <artifactId>snakeyaml</artifactId>
78          <version>1.33</version>
79          <scope>compile</scope>
80      </dependency>
81  </dependencies>
82  </project>

```

Attention : Il faut assurer la compatibilité entre les sous dépendances de chaque dépendance vis-à-vis les autres dépendances



26/12/2023

Mohammed Issam KABBAJ

20

Exemple de dépendances 2 niveau

```

m pom.xml (SpringBoot1)  spring-boot-starter-3.1.5.pom  spring-boot-3.1.5.pom ×
35      <developerConnection>scm:git:ssh://git@github.com/spring-projects/sprin
36      <url>https://github.com/spring-projects/spring-boot</url>
37    </scm>
38    <issueManagement>
39      <system>GitHub</system>
40      <url>https://github.com/spring-projects/spring-boot/issues</url>
41    </issueManagement>
42    <dependencies>
43      <dependency>
44        <groupId>org.springframework</groupId>
45        <artifactId>spring-core</artifactId>
46        <version>6.0.13</version>
47        <scope>compile</scope>
48      </dependency>
49      <dependency>
50        <groupId>org.springframework</groupId>
51        <artifactId>spring-context</artifactId>
52        <version>6.0.13</version>
53        <scope>compile</scope>
54      </dependency>
55    </dependencies>
56  </project>
57

```

26/12/2023

Mohammed Issam KABBAJ

21

Projet Spring : Code Java

```

import ...
1 usage
@SpringBootApplication
public class MonPremierProjetApplication {

    public static void main(String[] args) {
        SpringApplication.run(MonPremierProjetApplication.class, args);
    }

}

```

26/12/2023

Mohammed Issam KABBAJ

22

Projet Spring : Première Exécution



```
Run: MonPremierProjetApplication ×
Console Actuator

.a.e.g.m.MonPremierProjetApplication : Starting MonPremierProjetApplication using Java 1.8.0
.a.e.g.m.MonPremierProjetApplication : No active profile set, falling back to 1 default prot
.a.e.g.m.MonPremierProjetApplication : Started MonPremierProjetApplication in 1.669 seconds
```

26/12/2023

Mohammed Issam KABBAJ

23

Ma première modification



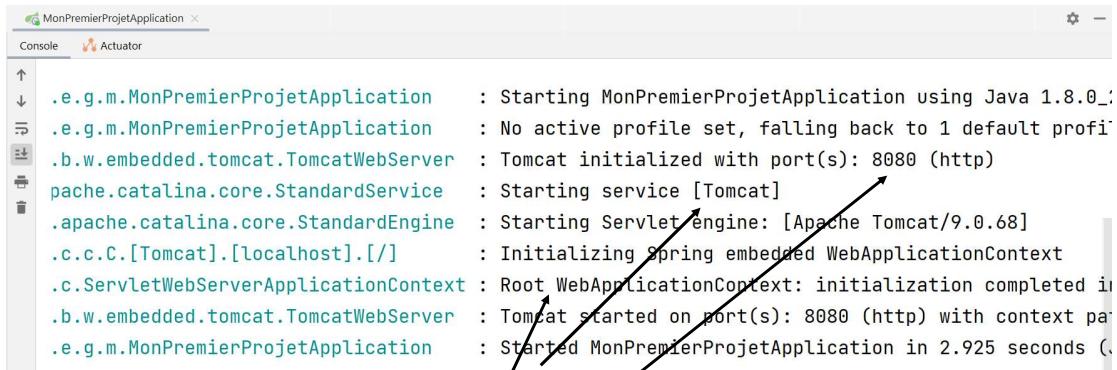
```
19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-web</artifactId>
23   </dependency>
24
25   <dependency>
26     <groupId>org.springframework.boot</groupId>
27     <artifactId>spring-boot-starter-test</artifactId>
28     <scope>test</scope>
29   </dependency>
30 </dependencies>
```

26/12/2023

Mohammed Issam KABBAJ

24

Nouvelle exécution



```

MonPremierProjetApplication
Console Actuator

.
↓ .e.g.m.MonPremierProjetApplication : Starting MonPremierProjetApplication using Java 1.8.0_252-b09-3015-1543-20230116-1105
.
↓ .e.g.m.MonPremierProjetApplication : No active profile set, falling back to 1 default profile
.
↓ .b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
.
↓ apache.catalina.core.StandardService : Starting service [Tomcat]
.
↓ apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.68]
.
↓ .c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
.
↓ .c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2.925 seconds
.
↓ .b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path /
.
↓ .e.g.m.MonPremierProjetApplication : Started MonPremierProjetApplication in 2.925 seconds (JVM running for 3.002)

```

26/12/2023 Mohammed Issam KABBAJ 25

Explication

- Nous utilisons la notation **@SpringBootApplication** sur la classe principale de l'application.
- Cette annotation transforme cette classe comme classe de configuration principale d'application.
- Cela équivaut à **@Configuration**, **@EnableAutoConfiguration** et **@ComponentScan** ensemble.

@SpringBootApplication

- **@Configuration** : définit la classe comme source de définitions de bean pour le contexte de l'application.
- **@EnableAutoConfiguration** : indique à Spring Boot de commencer à ajouter des beans en fonction des paramètres de chemin de classe, d'autres beans et de divers paramètres de propriété.
 - Par exemple, si spring-web se trouve sur le POM.XML, cette annotation marque l'application en tant qu'application Web et active des comportements clés, tels que la configuration d'un DispatcherServlet et le démarrage de Tomcat.
- **@ComponentScan** : indique à Spring de rechercher d'autres composants, configurations et services dans le package.

Auto-configuration

- L'annotation **@SpringBootApplication** déclenche la configuration automatique de l'infrastructure Spring
- Au démarrage de l'application, Spring Boot :
 - Scanne toutes classes de @Configuration
 - Classes de configuration spécifiques à l'application
 - Classes de Spring Boot suffixées par AutoConfiguration
 - 3rd party starters
 - Utilise les JAR présents dans le classpath pour prendre des décisions

Comment Changer la configuration

- Plusieurs options sont possible :
- POM.XML
- Application.properties
- Annotation Java
- Fichiers XML de configuration
- ...

26/12/2023

Mohammed Issam KABBAJ

29

Configuration via POM.xml



The screenshot shows an IDE interface with two main panes. On the left, the code editor displays the `pom.xml` file for a project named "MonPremierProjet". The code is as follows:

```

19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter</artifactId>
23   </dependency>
24
25   <dependency>
26     <groupId>org.springframework.boot</groupId>
27     <artifactId>spring-boot-starter-test</artifactId>
28     <scope>test</scope>
29   </dependency>
30 </dependencies>
31

```

A yellow highlight covers the first dependency block (lines 21-23). On the right, the "Run" tab shows the output of the application's startup. It includes the command used to start the application, the profile set (or lack thereof), and the duration it took to start.

26/12/2023

Mohammed Issam KABBAJ

30

Configuration via POM.xml

```

19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-web</artifactId>
23   </dependency>
24
25   <dependency>
26     <groupId>org.springframework.boot</groupId>
27     <artifactId>spring-boot-starter-test</artifactId>
28     <scope>test</scope>
29   </dependency>
30 </dependencies>

```

Changement d'une seule dépendance du projet à impliquer le démarrage de Apache Tomcat et sa configuration sur le port HTTP 8080

Configuration via application.properties

```

mierProjetApplication : Starting MonPremierProjetApplication using Java 1.8.0_211 on kabbaj
mierProjetApplication : No active profile set, falling back to 1 default profile: "default"
.tomcat.TomcatWebServer : Tomcat initialized with port(s): 9999 (http)
a.core.StandardService : Starting service [Tomcat]
ina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.68]
t].[localhost].[/] : Initializing Spring embedded WebApplicationContext
.Server ApplicationContext : Root WebApplicationContext: initialization completed in 1642 ms
.tomcat.TomcatWebServer : Tomcat started on port(s): 9999 (http) with context path ''
mierProjetApplication : Started MonPremierProjetApplication in 3.242 seconds (JVM running f

```

Configuration via Annotations

The screenshot shows an IDE interface with two panes. On the left, the code editor displays `MonPremierProjetApplication.java` containing a main method and a CommandLineRunner bean named `traitementParallele`. The code prints "Je suis un traitement parallele". On the right, the terminal window shows the application's log output, which includes the printed message "Je suis un traitement parallele". A black arrow points from the code in the editor to the corresponding log entry in the terminal.

```

1 usage
8 @SpringBootApplication
9 public class MonPremierProjetApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(MonPremierProjetApplication.class, args);
13     }
14
15     @Bean
16     public CommandLineRunner traitementParallele(){
17         return args -> {
18             System.out.println("Je suis un traitement parallele");
19         };
20     }
21 }

```

26/12/2023

Mohammed Issam KABBAJ

33

Attention

Certaines modifications nécessite d'autres éléments pour fonctionner correctement.

The screenshot shows an IDE interface with two panes. On the left, the XML file `pom.xml` lists several dependencies, including Spring Boot Starter Data JPA and MySQL Connector Java. On the right, the terminal window shows an error message indicating that the application failed to start because it failed to configure a DataSource. The error message specifies that the 'url' attribute is not specified and no embedded datasource can be found.

```

20
21 <dependencies>
22     <dependency>
23         <groupId>org.springframework.boot</groupId>
24         <artifactId>spring-boot-starter-data-jpa</artifactId>
25     </dependency>
26     <dependency>
27         <groupId>mysql</groupId>
28         <artifactId>mysql-connector-java</artifactId>
29         <version>8.0.23</version>
30     </dependency>
31     <dependency>
32         <groupId>org.springframework.boot</groupId>
33         <artifactId>spring-boot-starter-test</artifactId>
34         <scope>test</scope>
35     </dependency>
36 </dependencies>

```

26/12/2023

Mohammed Issam KABBAJ

34

Configurer ou pas ?

- Ajouter dans le fichier **application.properties** la ligne suivante :

logging.level.org.springframework=debug

- puis exécuter

The screenshot shows two side-by-side Spring Boot application consoles. Both are titled "MonPremierProjectApplication". The left console, labeled "Configurer", displays a "CONDITIONS EVALUATION REPORT" with sections for "Positive matches" and "AopAutoConfiguration matched". The right console, labeled "Non Configurer", displays a "Negative matches" section with entries for "ActiveMQAutoConfiguration", "AopAutoConfiguration.AspectAutoProxyingConfiguration", and "ArtemisAutoConfiguration", all indicating "Did not match" due to missing required classes.

```

MonPremierProjectApplication
Run MonPremierProjectApplication
Console Actuator G M S D E

=====
CONDITIONS EVALUATION REPORT
=====

Positive matches:
-----
AopAutoConfiguration matched:
- @ConditionalOnProperty (spring.aop.auto=true) matched (OnPropertyCondition)

AopAutoConfiguration.ClassProxyingConfiguration matched:
- @ConditionalOnMissingClass did not find unwanted class 'org.aspectj.weaver.Advice' (OnClassCondition)
- @ConditionalOnProperty (spring.aop.proxy-target-class=true) matched (OnPropertyCondition)

=====
Configurer

Run MonPremierProjectApplication
Console Actuator G M S D E

=====
Negative matches:
-----
ActiveMQAutoConfiguration:
Did not match:
- @ConditionalOnClass did not find required class 'jakarta.jms.ConnectionFactory' (OnClassCondition)

AopAutoConfiguration.AspectAutoProxyingConfiguration:
Did not match:
- @ConditionalOnClass did not find required class 'org.aspectj.weaver.Advice' (OnClassCondition)

ArtemisAutoConfiguration:
Did not match:
- @ConditionalOnClass did not find required class 'jakarta.jms.ConnectionFactory' (OnClassCondition)

Non Configurer

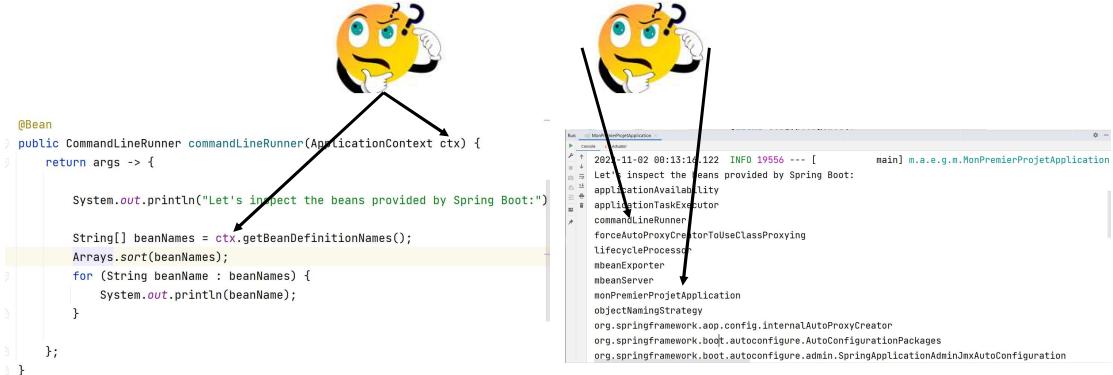
```

- Qu'est ce qui est configuré avec spring boot web ?

Spring Boot Starter

- Spring Boot** propose une variété de projets de démarrage :
 - spring-boot-starter-web** pour créer des applications Web avec API REST
 - spring-webmvc, spring-web, spring-boot-starter-tomcat, spring-boot-starter-json
 - spring-boot-starter-test** pour tests unitaires
 - spring-boot-starter-data-jpa** pour communiquer avec la base de données à l'aide de JPA
 - spring-boot-starter-jdbc** pour communiquer avec la base de données à l'aide de JDBC
 - spring-boot-starter-security** pour sécuriser votre site Web ou votre API REST

Injection des dépendances (ID)

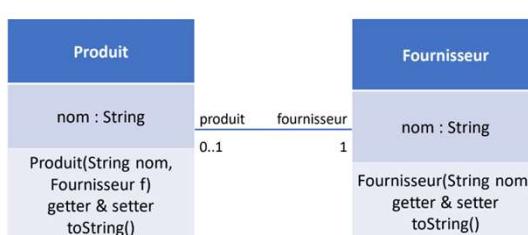


26/12/2023

Mohammed Issam KABBAJ

37

Origine du problème : Dépendance forte



Main.java Fournisseur.java Produit.java

```

10 usages new *
1 public class Fournisseur {
2     3 usages
3     private String name;
4     3 usages
5     private Produit produit;
6
7     2 usages new *
8     public Fournisseur(String name) { setName(name); }
9
10    no usages new *
11    public Fournisseur(String name, Produit produit) {
12        this(name);
13        setProduit(produit);
14    }

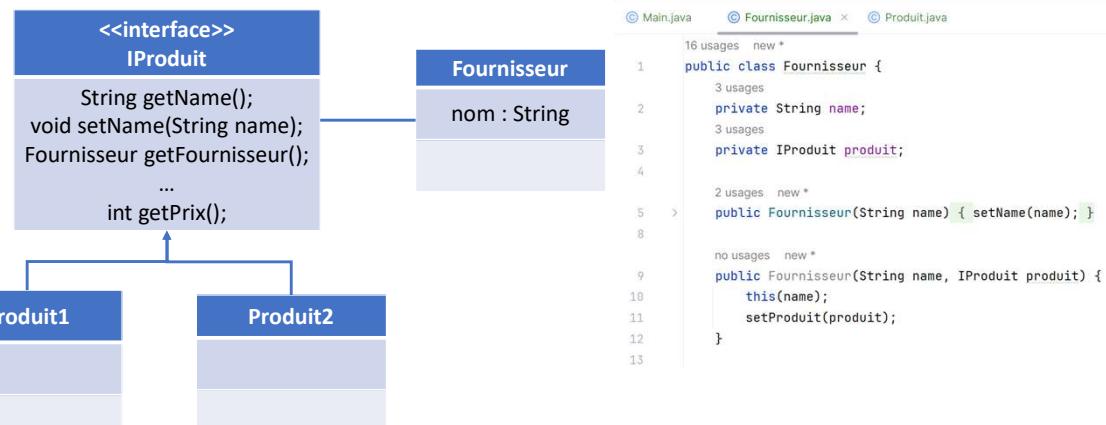
```

26/12/2023

Mohammed Issam KABBAJ

38

Première réflexion



26/12/2023

Mohammed Issam KABBAJ

39

Aller plus loin dans cette réflexion

The left code snippet (**Main.java**) shows the implementation of the **IProduit** interface:

```

Main.java Fournisseur.java * IPProduit.java * IFournisseur.java *
1 10 usages 2 implementations new *
2 @ 1 public interface IPProduit {
3     2 implementations new *
4     1 String getName();
5
6     2 usages 2 implementations new *
7     void setName(String name);
8
9     no usages 2 implementations new *
10    IFournisseur getFournisseur();
11
12    2 usages 2 implementations new *
13    void setFournisseur(IFournisseur fournisseur);
14
15    2 usages 2 implementations new *
16    int getPrix();
17
18    }
  
```

The right code snippet (**Fournisseur.java**) shows the implementation of the **IFournisseur** interface:

```

Main.java * Fournisseur.java * IPProduit.java * IFournisseur.java *
12 usages 1 implementation new *
1 @ 1 public interface IFournisseur {
2     1 usage 1 implementation new *
3     void setName(String name);
4
5     1 implementation new *
6     String getName();
7
8     no usages 1 implementation new *
9     IPProduit getProduit();
10
11    3 usages 1 implementation new *
12    void setProduit(IPProduit produit);
13
14    }
  
```

26/12/2023

Mohammed Issam KABBAJ

40

Dépendance vis-à-vis de new

```

    ① Main.java x ② Fournisseur.java ③ IProduit.java ④ IFournisseur.java ⑤ Produit.java
new *
3 ▷ public class Main {
    new *
4 ▷     public static void main(String[] args) {
            //attention encore une dependence vers la Classe Founisseur
            IFournisseur hp = new Fournisseur( name: "HP");
            //attention encore une dependence vers la Classe Produit
            IProduit x360 = new Produit( name: "Elitebook x360", hp);
o

```

The code editor shows the Main.java file with several annotations:

- Annotation 1: A note above the first line of the class definition.
- Annotation 2: A note above the first line of the main method.
- Annotation 3: A note above the line where `IFournisseur hp = new Fournisseur(...);` is written.
- Annotation 4: A note above the line where `IProduit x360 = new Produit(...);` is written.
- Annotation 5: A note below the line where `IProduit x360 = new Produit(...);` is written.

26/12/2023

Mohammed Issam KABBAJ

41

Première solution

```

try {
    Scanner scanner = new Scanner(System.in);
    Scanner scanner = new Scanner(new FileReader( fileName: "Config.txt"));

    classProduitName = scanner.next();
    classProduit2Name = scanner.next();
    classFournisseurName = scanner.next();
    methodeSetNameName = scanner.next();
    methodeSetFournisseurName = scanner.next();

    // Chargement dynamique des classes
    Class produitClass = Class.forName(classProduitName);
    // Creation d'instance
    // IProduit x360 = new Produit()
    x360 = (IProduit) produitClass.newInstance();
    // Invocation d'une methode d'objet
    Method methodeSetName2 = produitClass.getMethod(methodeSetNameName, new Class[]{String.class});
    // x360.setName("Elitebook x360");
    methodeSetName2.invoke(x360, ...args: "Elitebook x360");
}

```

The code editor shows two files:

- `Main.java` (active tab): Contains Java code that reads configuration from `Config.txt` and creates instances of `IProduit` and `Produit` using reflection.
- `Config.txt`: Contains configuration data.

26/12/2023

Mohammed Issam KABBAJ

42

Solution Spring : IOC + ID

1 - Déclarer

```
<bean id="hp" class="Fournisseur">
    <property name="name" value="HP" />
</bean>
<bean id="x3602" class="Produit3">
    <property name="name" value="Elitebook x3602" />
    <property name="fournisseur" ref="hp" />
</bean>
<bean id="x360" class="Produit3">
    <constructor-arg name="name" value="Elitebook x360" />
    <constructor-arg name="fournisseur" ref="hp" />
</bean>
```

26/12/2023

Mohammed Issam KABBAJ

43

Mieux Encore : se concerter sur le métier

The screenshot shows an IDE interface with two panes. The left pane displays the `Main.java` file, which contains a main method that prints "Hello" and then enters a loop to handle withdrawals from a bank account. The right pane shows the command line output of running the application, which includes a password prompt, a denial of access, and the message "Process finished with exit code 0". Below the command line, a stack trace is visible, indicating a `java.lang.RuntimeException` was thrown due to a limit being exceeded.

```
Main.java x Logging.aj ExceptionHandlerAspect.aj Depacemv
Main.java
public class Main {
    public static void main(String[] args) {
        traitement();
    }

    System.out.println();
    System.out.println("-----");
    System.out.println();

    try {
        Compte compte = new Compte();
        Scanner clavier = new Scanner(System.in);
        System.out.print("Code :");
        int code = clavier.nextInt();
        compte.setCode(code);
        while (true) {
            System.out.print("Montant à verser :");
            double mt1 = clavier.nextDouble();
            compte.verser(mt1);

            System.out.println(compte.toString());
            System.out.print("main a Retirer :");
            double mt2 = clavier.nextDouble();
            compte.retirer(mt2);
            System.out.println(compte.toString());
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

public static void traitement() {
    Compte cpt1 = new Compte();
    Compte cpt2 = new Compte();
    Compte cpt3 = new Compte();
    Compte cpt4 = new Compte();
    cpt1.setCode(1);
    cpt1.verser(1000);
    cpt2.setCode(2);
    cpt2.setCompte(cpt1);
    cpt2.verser(9000);
    System.out.println(cpt1.toString());
    System.out.println(cpt2.toString());
}
```

"C:\Program Files\Java\jdk-17\bin\java.exe"
Login:admin
Pass:admin
Accès refusé
Pas de sécurité dans votre code métier
Process finished with exit code 0

```
INFO: ****  
nov. 13, 2023 4:46:00 PM ma.emi.aspect.Compte.setCode  
INFO: Après void ma.emi.aspect.Compte.setCode(int)  
nov. 13, 2023 4:46:00 PM ma.emi.aspect.Logging ejc$after$ma.emi.aspectLogging$2$0d14ed2  
INFO: Durée d'exécution: 1 ms  
Exception in thread "main" java.lang.RuntimeException Create breakpoint: Nombre d'instances dépassées  
at Main.int$aroundBody1$advise$1$main$22  
at Main.traitement$1$main$20  
at Main.main$aroundBody2$advise$1$main$27  
at Main.main$aroundBody2$advise$1$main$25
```

- Contrôle ajustable et paramétrable
- Prendre en charge de besoin Non Fonctionnels
 - Performance
 - Tracabilité

26/12/2023

Mohammed Issam KABBAJ

44

Programmation par Aspect

- Comment ?

```

Main.java
new *
public aspect Logging {
    ...
}

Logging.aj
new *
public aspect Logging {
    pointcut avant():execution(* ..Compte.*(..));
    pointcut apres():execution(* ..Compte.*(..));

    before():avant() {
        t1 = System.currentTimeMillis();
        logger.info( msg: "*****");
        logger.info( msg: "Avant " + thisJoinPoint.getSignature());
    }

    after():apres() {
        t2 = System.currentTimeMillis();
        logger.info( msg: "*****");
        logger.info( msg: "Apres " + thisJoinPoint.getSignature());
        logger.info( msg: "Duree d'execution=" + (t2 - t1) + " ms");
    }
}

ExceptionHandlerAspect.aj
package ma.emi.aspect;
new *
public aspect |exceptionHandlerAspect {
    pointcut anyMethod():call(* *.*(..));
    after()throwing(Exception e): anyMethod(){
        System.out.println(thisJoinPoint.getSignature());
        System.out.println(e.getClass());
        System.out.println(e.getMessage());
    }
}

public aspect NBObjet {
    private List<Compte> listComptes = new ArrayList<>();
    pointcut instantiation():call(Compte.new(..));
    Compte around(): instantiation() {
        System.out.println("*****");
        if (listComptes.size() < 1) {
            Compte p = (Compte) proceed();
            System.out.println("Nouvelles instanciation");
            p.setCode(listComptes.size() + 1);
            listComptes.add(p);
            System.out.println("Initialisation de l'instance");
        }
        System.out.println("*****");
        return p;
    } else {
        throw new RuntimeException("Nombre d'instances dépassées");
    }
}

```

26/12/2023

Mohammed Issam KABBAJ

45

Programmation par Aspect

Ainsi

```

new *
public aspect SecurityAspect {
    private String login;
    private String pass;
    private Scanner clavier;

    pointcut security():execution(* *.main(..));
}

void around() : security() {
    if (login == null) {
        clavier = new Scanner(System.in);
        System.out.print("Login:");
        String l = clavier.next();
        System.out.print("Pass:");
        String p = clavier.next();
        if (l.equals("root") && p.equals("root")) {
            login = l;
            pass = p;
            proceed();
        } else
            System.out.println("Accès refusé");
    }
}

```

Ou encore mieux, ainsi

```

@Configuration
@EnableWebSecurity
public class SecurityConfiguration {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        // @formatter:off
        http
            .authorizeHttpRequests((authorize) -> authorize
                .anyRequest().authenticated()
            )
            .httpBasic(withDefaults())
            .formLogin(withDefaults());
        // @formatter:on
        return http.build();
    }
}

@Builder
public UserDetailsService userDetailsService() {
    UserDetails user = User.withDefaultPasswordEncoder()
        .username("user")
        .password("password")
        .roles("USER")
        .build();
    return new InMemoryUserDetailsManager(user);
}

```

26/12/2023

Mohammed Issam KABBAJ

46

Profiles

26/12/2023

Mohammed Issam KABBAJ

47

Environnements de déploiement

- Plusieurs environnements pour déployer une application
 - dev, QA, stage, prod, ...
- Configuration spécifique pour chaque environnement
 - Niveau de log différents (trace, debug, info, warning, error, off)
 - Bases de données différentes
 - web services différentes
 - ...
- ➔ la solution est proposée avec les profiles

26/12/2023

Mohammed Issam KABBAJ

48

Profiles (intro)

- Créer vos profiles

```
application-dev.properties ✘ application-prod.properties  
1 logging.level.org.springframework=trace
```

```
application-dev.properties ✘ application-prod.properties ✘  
1 logging.level.org.springframework=info
```

- Activer le profile souhaité

```
application.properties ✘ application-dev.properties ✘ application-prod.properties  
1 logging.level.org.springframework=debug  
2 spring.profiles.active=prod
```

Spring Data - JPA

Mapping Objet Relationnel

Objet

- Diagramme de classe (DC)
- Classe
- Attribut
- Objet

Relationnel

- Modèle conceptuel de données (MCD)
- Table
- Champ
- Enregistrement

Mapping Objet Relationnel : Divergence

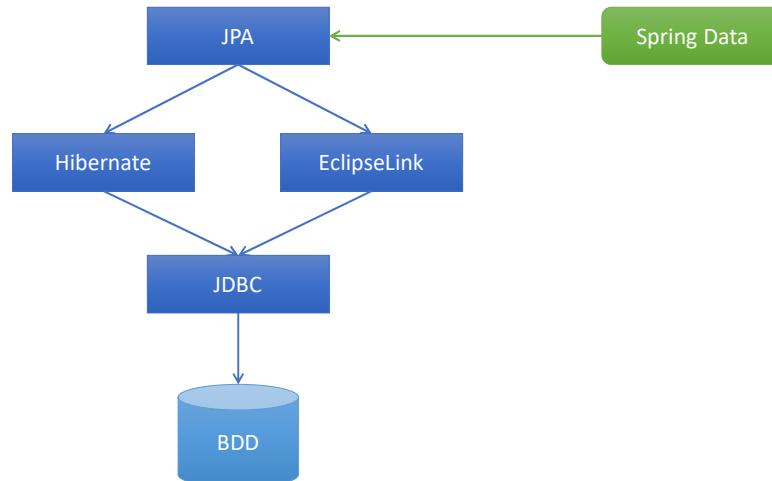
Objet

- DC pour la communication
 - Diagramme dynamique
- Relation bidirectionnelle
 - ManyToMany, ManyToOne, ...
- Id implicite
 - Variable de classe mais bon
- Contrainte de domaine implicite
- Héritage
- Contrainte d'entité non déclarée

Relationnel

- Schéma pour réduire la redondance
 - Stockage seulement
- Relation unidirectionnelle
 - PK → FK
- Clé primaire (PK) explicite
 - Il peut être auto incrémental
- Contrainte de domaine explicite
- Héritage non supporté
- Contrainte d'entité

Spring Data JPA



26/12/2023

Mohammed Issam KABBAJ

53

Changement du POM

The screenshot shows the Eclipse IDE interface. On the left, the 'pom.xml' file is displayed with several dependency blocks highlighted in yellow. The code snippet below shows the relevant parts of the XML:

```

19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-data-jpa</artifactId>
23   </dependency>
24   <dependency>
25     <groupId>com.h2database</groupId>
26     <artifactId>h2</artifactId>
27   </dependency>
28   <dependency>
29     <groupId>org.springframework.boot</groupId>
30     <artifactId>spring-boot-starter-test</artifactId>
31     <scope>test</scope>
32   </dependency>
33 </dependencies>
  
```

On the right, the terminal window shows the application's logs:

```

2022-11-02 08:27:04.752 INFO 8352 --- [           main] m.a.e.g.m.MonPremierProjectApplication : Starting MonPremierProjectApplication on DESKTOP-1QH9D9A with PID 8352 (C:\Users\Issam\IdeaProjects\MonPremierProject\target\classes started by DESKTOP-1QH9D9A in C:\Users\Issam\IdeaProjects\MonPremierProject)
2022-11-02 08:27:04.757 INFO 8352 --- [           main] m.a.e.g.m.MonPremierProjectApplication : The following profiles are active: dev
2022-11-02 08:27:05.632 INFO 8352 --- [           main] m.a.e.g.m.MonPremierProjectApplication :
  
```

26/12/2023

Mohammed Issam KABBAJ

54

Configuration h2

```
application.properties
1 spring.datasource.url=jdbc:h2:mem:testdb;
2 spring.datasource.driverClassName=org.h2.Driver
3 spring.datasource.username=sa
4 spring.datasource.password=
5 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
6 spring.jpa.show-sql=true
7 spring.h2.console.enabled=true
```

26/12/2023

Mohammed Issam KABBAJ

55

Configuration MYSQL

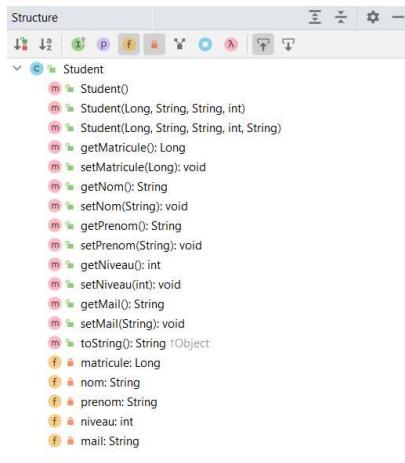
```
application.properties
1 spring.datasource.url=jdbc:mysql://localhost:3306/myjee
2 spring.datasource.username=root
3 spring.datasource.password=toor
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5
6 spring.jpa.hibernate.ddl-auto=create-drop
7 spring.jpa.show-sql=true
```

26/12/2023

Mohammed Issam KABBAJ

56

JPA : Object/Relational Mapping



Minimum pour mapper une classe avec une table dans une base de données relationnelle

```

package ma.ac.emi.ginfo.monpremierprojet.models;

import javax.persistence.*;
import java.util.List;

@Entity
public class Student {
    @Id
    private Long matricule;
    private String nom;
    private String prenom;
}

```

26/12/2023

Mohammed Issam KABBAJ

57

JPA

```

@javax.persistence.Entity
@Table(name = "etudiants")
public class Student implements Comparable<Student> {

    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    private String prenom;
    private String mail;
    @Id
    private Long matricule;
}

```

Query 1 students - Table

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
matricule	BIGINT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
name	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
mail	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
prenom	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- Nom de la table
- Nom du colonne
- Not null
- Les valeurs doubles non autorisées
- Taille du chaine de caractère

26/12/2023

Mohammed Issam KABBAJ

58

Clé primaire Auto incrémentale

```

6      4 usages
7  @Entity
8  public class Note {
9
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private Long id;
13     4 usages
14     private int note;
15
16     Note n = new Note(15);
17     System.out.println(n);
18     n = noteRepository.save(n);
19     System.out.println(n);
20
2023-11-15T11:16:34.139+01:00  INFO 25484 ---  

Note{id=null, note=15}
Hibernate: insert into note (note) values (?)  

Note{id=1, note=15}

```

Query 1 etudiants - Table note - Table

Table Name: note Schema: myjee
Charset/Collation: utf8ml utf8ml Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
note	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
id	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				

Id auto incrémentale →

1. Pas d'id dans le constructeur
2. La récupération de l'objet après avoir persisté dans la base de données pour récupérer son id



26/12/2023

Mohammed Issam KABBAJ

59

Clé primaire composée

```

@Embeddable
public class StudentId implements Serializable {
    4 usages
    private String nom;
    4 usages
    private String prenom;

    @Entity
    @Table(name = "etudiants")
    public class Student implements Comparable<Student>{

        6 usages
        @EmbeddedId
        private StudentId identifiant;
    }
}

```

etudiants - Table

Table Name: etudiants Schema: myjee
Charset/Collation: utf8ml utf8ml Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
matricule	BIGINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
mail	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
nom	VARCHAR(255)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
prenom	VARCHAR(255)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				

26/12/2023

Mohammed Issam KABBAJ

60

Clé primaire autres options

The screenshot shows the MySQL Workbench interface. On the left, the code editor displays a Java entity class named `Student` with annotations for @Entity, @Table, @Id, and @GeneratedValue. The database browser on the right shows two tables: `etudiants` and `id_student`. The `etudiants` table has columns `matricule`, `name`, `mail`, and `prenom`. The `id_student` table has a single column `next_val`. Below the browser, a query result grid shows the value 10 for `next_val`.

26/12/2023

Mohammed Issam KABBAJ

61

Valeur par défaut

The screenshot shows the MySQL Workbench interface. On the left, the code editor displays a Java entity class named `Note` with annotations for @Entity and @Id. The database browser on the right shows a table named `note` with columns `note` and `id`. The `note` column has a data type of INT and a default value of '0'. The `id` column is defined as a BIGINT primary key.

26/12/2023

Mohammed Issam KABBAJ

62

Données non persistantes

```

<!-- @Entity
@Table(name = "etudiants")
public class Student implements Comparable<Student> {

    @Id
    private Long matricule;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    4 usages
    @Transient
    private String mail;
}
  
```

The screenshot shows the MySQL Workbench interface. On the left, the code editor displays the Java code for the `Student` entity. On the right, the 'etudiants - Table' configuration window is open, showing the table name as 'etudiants', schema as 'myjee', charset as 'utf8ml', and engine as 'InnoDB'. The table structure includes columns: matricule (BIGINT, PK, NN, UQ), name (VARCHAR(50)), and prenom (VARCHAR(255)). Below this, a query editor window contains the SQL command `SELECT * FROM myjee.etudiants;`. The results grid shows one row of data: matricule 1234, name Kabbaj, and prenom Mohammed Issam.

26/12/2023

Mohammed Issam KABBAJ

63

Persistante des énumérations sans annotation

```

9
10
11
12
13
14
15
16
17
18
19
20
  
```

```

    @Entity
    @Table(name = "etudiants")
    public class Student implements Comparable<Student> {

        @Id
        private Long matricule;
        4 usages
        @Column(name = "name", nullable = false, length = 50, unique = true)
        private String nom;
        4 usages
        private String prenom;
        4 usages
        @Transient
        private String mail;
        1 usage
        private Genre genre;

        4 usages
        public enum Genre {
            no usages
            Femme,
            1 usage
            Homme
        }
    }
  
```

The screenshot shows the MySQL Workbench interface. On the left, the code editor displays the Java code for the `Student` entity, including the addition of a `Genre` enum. On the right, the 'etudiants - Table' configuration window is open, showing the table name as 'etudiants', schema as 'myjee', charset as 'utf8ml', and engine as 'InnoDB'. The table structure includes columns: genre (TINYINT), matricule (BIGINT, PK, NN, UQ), name (VARCHAR(50)), and prenom (VARCHAR(255)). Below this, a query editor window contains the SQL command `SELECT * FROM myjee.etudiants;`. The results grid shows one row of data: genre 1, matricule 1234, name Kabbaj, and prenom Mohammed Issam.

26/12/2023

Mohammed Issam KABBAJ

64

Persistante des énumérations avec annotation

The screenshot shows the MySQL Workbench interface. On the left, there is a code editor displaying Java code for a Student entity. The code includes annotations like @Entity, @Table(name = "etudiants"), @Id, @Column(name = "name", nullable = false, length = 50, unique = true), and @Enumerated(EnumType.STRING). A tooltip for the @Enumerated annotation is visible, stating "Create a new stored procedure in the active schema in the connected server". On the right, there is a table editor for the 'etudiants' table. The table has four columns: matricule (BIGINT, PK), name (VARCHAR(50)), genre (ENUM('Femme', 'Homme')), and prenom (VARCHAR(255)). Below the table editor, a result grid displays data for the 'etudiants' table.

```

    10  @Entity
    11  @Table(name = "etudiants")
    12  public class Student implements Comparable<Student> {
    13
    14      @Id
    15      private Long matricule;
    16      @Column(name = "name", nullable = false, length = 50, unique = true)
    17      private String nom;
    18      private String prenom;
    19      @Transient
    20      private String mail;
    21      @Usage
    22      @Enumerated(EnumType.STRING)
    23      private Genre genre;
    24
    25      4 usages
    26      public enum Genre {
    27          no usages
    28          Femme,
    29          1 usage
    30          Homme
    31      }
  
```

26/12/2023

Mohammed Issam KABBAJ

65

OneToOne Unidirectionnelle (1/2)

The screenshot shows the MySQL Workbench interface. On the left, there is a code editor displaying Java code for a Student entity. The code includes annotations like @Entity, @Table(name = "etudiants"), @Id, @Column(name = "name", nullable = false, length = 50, unique = true), and @Transient. On the right, there is a table editor for the 'etudiants' table. The table has four columns: matricule (BIGINT, PK), name (VARCHAR(50)), genre (ENUM('Femme', 'Ho...'), and prenom (VARCHAR(255)). Below the table editor, a result grid displays data for the 'etudiants' table.

```

    10  @Entity
    11  @Table(name = "etudiants")
    12  public class Student implements Comparable<Student> {
    13
    14      @Id
    15      private Long matricule;
    16      @Column(name = "name", nullable = false, length = 50, unique = true)
    17      private String nom;
    18      private String prenom;
    19      @Transient
    20      private String mail;
    21      @Usage
    22      private Genre genre;
  
```

26/12/2023

Mohammed Issam KABBAJ

66

OneToOne Unidirectionnelle (2/2)

```

5 usages
@Entity
public class Note {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(columnDefinition = "integer default 0")
    private int note;

    4 usages
    @OneToOne
    private Student student;

    Foreign Key Name Reference
    FKtbrro72vvvh0waqv8iq... "myjee".*
    Column note
    id
    student_matricule
    Foreign Key Options
    On Update: RESTRICT
    On Delete: RESTRICT
    Skip in SQL generation
  
```

Table Name: note Schema: myjee
 Charset/Collation: utf8mbi utf8mbi Engine: InnoDB
 Column Name Datatype PK NN UQ B UN ZF AI
 note INT
 id BIGINT
 student_matricule BIGINT

Comments:
 Column Name Data Type: BIGINT
 Charset/Collation: Default: NULL
 Storage: Virtual Stored
 Primary Key Not Null Unique
 Binary Unsigned Zero Fill
 Auto Increment Generated

26/12/2023

Mohammed Issam KABBAJ

67

OneToOne Bidirectionnelle Erreur (1/2)

```

@Entity
@Table(name = "etudiants")
public class Student implements Comparable<Student> {

    @Id
    private Long matricule;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    4 usages
    @Transient
    private String mail;
    1 usage
    @Enumerated(EnumType.STRING)
    private Genre genre;
    2 usages
    @OneToOne
    private Note note;
  
```

```

@Entity
public class Note {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    5 usages
    @Column(columnDefinition = "integer default 0")
    private Long note;

    4 usages
    @OneToOne
    private Student student;
  
```

26/12/2023

Mohammed Issam KABBAJ

68

OneToOne Bidirectionnelle Erreur (2/2)

26/12/2023

Mohammed Issam KABBAJ

69

OneToOne Bidirectionnelle (1/2)

```

@Entity
@Table(name = "etudiants")
public class Student implements Comparable<Student> {

    @Id
    private Long matricule;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    4 usages
    @Transient
    private String mail;
    1 usage
    @Enumerated(EnumType.STRING)
    private Genre genre;
    2 usages
    @OneToOne(mappedBy = "student")
    private Note note;
}

@Entity
public class Note {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    5 usages
    @Column(columnDefinition = "integer default 0")
    private Long note;
    5 usages
    @OneToOne()
    @JoinColumn(name = "matricule") //au lieu de student_matricule
    private Student student;
}

```

26/12/2023

Mohammed Issam KABBAJ

70

OneToOne Bidirectionnelle (2/2)

etudiants - Table

Table Name: etudiants Schema: myjee

Charset/Collation: utf8ml utf8ml Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
matricule	BIGINT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
name	VARCHAR(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
genre	ENUM('Femme', 'Homme')	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
prenom	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Column Name: Data Type:

Charset/Collation: Default:

Comments: Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

etudiants - Table

Table Name: note Schema: myjee

Charset/Collation: utf8ml utf8ml Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
id	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
matricule	BIGINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
note	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Foreign Key Name: FK96gvx73bh9bv33wkp... Reference: myjee.id Column: id Foreign Key Options: On Update: RESTRICT On Delete: RESTRICT Skip in SQL generation

Columns Indexes Foreign Keys Triggers Partitioning Options

26/12/2023

Mohammed Issam KABBAJ

71

OneToOne Bidirectionnelle partage PK (1/2)

```

@Entity
@Table(name = "etudiants")
public class Student implements Comparable<Student> {

    @Id
    private Long matricule;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    4 usages
    @Transient
    private String mail;
    1 usage
    @Enumerated(EnumType.STRING)
    private Genre genre;

    2 usages
    @OneToOne(mappedBy = "student")
    Note note;
}

```

```

@Entity
public class Note {

    @Id
    @GeneratedValue(generator="gen")
    @GenericGenerator(name="gen", strategy="foreign",
        parameters=@Parameter(name="property", value="student"))
    private Long id;
    5 usages
    @Column(columnDefinition = "integer default 0")
    private long note;

    5 usages
    @OneToOne
    @PrimaryKeyJoinColumn
    private Student student;
}

```

26/12/2023

Mohammed Issam KABBAJ

72

OneToOne Bidirectionnelle partage PK (2/2)

etudiants - Table note - Table

Table Name: etudiants Schema: myjee
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
matricule	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL					
name	VARCHAR(50)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL					
genre	ENUM('Femme', 'Homme')	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
prenom	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Name: id Datatype: BIGINT PK NN UQ B UN ZF AI G Default/Expression
note INT

Column Name: note Data Type:

Foreign Key Name: Referenced Table: Column: Referenced Column: Foreign Key Options:
On Update: On Delete:
Auto Increment Generated

2023-11-15T19:06:59.741+01:00 DEBUG 17140 --- [main] o.s.orm.jpa.JpaTransactionManager : Closing JPA EntityManager
Student{nom='Kabbaj', prenom='Mohammed Issam', mail='kabbaj@emi.ac.ma', matricule=1234}
Note{id=1234, note=15, student=Student{nom='Kabbaj', prenom='Mohammed Issam', mail='kabbaj@emi.ac.ma', matricule=1234}}

26/12/2023

Mohammed Issam KABBAJ

73

Entity en deux Tables

```
@Entity
@Table(name = "etudiants")
@SecondaryTable(name="notes", pkJoinColumns = @PrimaryKeyJoinColumn(name = "matricule") )
public class Student implements Comparable<Student> {

    @Id
    private Long matricule;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    1 usage
    @Enumerated(EnumType.STRING)
    private Genre genre;
    2 usages
    @Column(name = "note", table = "notes")
    private long note;
```

notes - Table notes

Table Name: notes Schema: myjee
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
matricule	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL					
note	BIGINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column Name: matricule Data Type: BIGINT
note BIGINT

Foreign Key Name: Referenced Table: Column: Referenced Column: Foreign Key Options:
On Update: RESTRICT
On Delete: RESTRICT
Auto Increment Generated

notes - Table notes

Result Grid | Filter Rows: |

matricule	note
1234	15
NULL	NULL

26/12/2023

Mohammed Issam KABBAJ

74

ManyToOne Unidirectionnelle

```
@Entity
@Table(name = "etudiants")
public class Student implements Comparable<Student> {

    @Id
    private Long matricule;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    4 usages
    @Transient
    private String mail;
    1 usage
    @Enumerated(EnumType.STRING)
    private Genre genre;
```

Table Name: Students
Schema: myjee
Engine: InnoDB
Comments:
Column Name Data Type PK NN UQ B UN ZF AI G Default/Expression
id BIGINT ✓
note INT
student_matricule BIGINT

```
@Entity
public class Note {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    4 usages
    @Column(columnDefinition = "integer default 0")
    private long note;
    4 usages
    @ManyToOne
    @JoinColumn(name = "student_matricule", nullable = false)
    private Student student;
```

note-Table
Comments:
Column Name Data Type PK NN UQ B UN ZF AI G Default/Expression
id BIGINT ✓
note INT
student_matricule BIGINT
Foreign Key Name Referenced Table Column Referenced Column Foreign Key Options
FKfbmo7zvvvhwaquqisq... "myjee`.`etudiants` id id On Update: RESTRICT
student_matricule matricule note On Delete: RESTRICT
Unsigned Zero Fill
Auto Increment Generated

26/12/2023

Mohammed Issam KABBAJ

75

OneToMany Unidirectionnelle

26/12/2023

Mohammed Issam KABBAJ

76

ManyToOne bidirectionnelle

The screenshot shows the MySQL Workbench interface with two main parts:

- Entity Student:**

```
@Entity
@Table(name = "etudiants")
public class Student implements Comparable<Student> {

    @Id
    private Long matricule;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    4 usages
    @Transient
    private String mail;
    1 usage
    @Enumerated(EnumType.STRING)
    private Genre genre;
    6 usages
    @OneToMany(mappedBy = "student", fetch = FetchType.EAGER)
    private List<Note> notes = new ArrayList<>();
```
- Entity Note:**

```
@Entity
public class Note {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    4 usages
    @Column(columnDefinition = "integer default 0")
    private long note;
    4 usages
    @ManyToOne
    @JoinColumn(name = "student_matricule", nullable = false)
    private Student student;
```

Below the entities, there are two tables:

- Table etudiants:** Shows columns: matricule (BIGINT), nom (VARCHAR(50)), prenom (VARCHAR(255)).
- Table note:** Shows columns: id (BIGINT), note (INT).

At the bottom left is the date: 26/12/2023. At the bottom right is the name: Mohammed Issam KABBAJ.

77

ManyToMany (1/2)

The screenshot shows the MySQL Workbench interface with two main parts:

- Entity Student:**

```
@Entity
@Table(name = "etudiants")
public class Student implements Comparable<Student> {

    @Id
    private Long matricule;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    4 usages
    @Transient
    private String mail;
    1 usage
    @Enumerated(EnumType.STRING)
    private Genre genre;
    6 usages
    @ManyToMany(fetch = FetchType.EAGER)
    private List<Note> notes = new ArrayList<>();
```
- Entity Note:**

```
@Entity
public class Note {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    4 usages
    @Column(columnDefinition = "integer default 0")
    private long note;
    4 usages
    @JoinTable(
        name = "student_note",
        joinColumns = @JoinColumn(name = "note"),
        inverseJoinColumns = @JoinColumn(name = "matricule"))
    private List<Student> students = new ArrayList<>();
```

At the bottom left is the date: 26/12/2023. At the bottom right is the name: Mohammed Issam KABBAJ.

78

ManyToMany (2/2)

The screenshot shows the MySQL Workbench interface with three tables being created:

- students - Table**: Contains columns id (PK), name, and matricule.
- note - Table**: Contains columns id (PK) and note.
- student_note - Table**: Contains columns student_id (FK to students.id) and note_id (FK to note.id).

Foreign key constraints are defined in the student_note table:

- FK from student_id to students.id
- FK from note_id to note.id

26/12/2023

Mohammed Issam KABBAJ

79

Classe d'association(1/2)

```

@Entity
public class StudentNote {
    @EmbeddedId
    private StudentNotePK id;

    @ManyToOne
    @JoinColumn(name = "student_id")
    private Student student;

    @ManyToOne
    @JoinColumn(name = "note_id")
    private Note note;

    private String matiere;
}

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

    @OneToOne(mappedBy = "note")
    private Student student;

    private String matiere;
}

```

26/12/2023

Mohammed Issam KABBAJ

80

Classe d'association(1/2)

The screenshot shows the MySQL Workbench interface with three tables:

- student_note - Table**: Contains columns note_id (PK), student_id, and matiere.
- students - Table**: Contains columns matricule, name, genre, and prenom.
- note - Table**: Contains column id.

Foreign key relationships are defined:

- student_note.note_id references students.id
- student_note.student_id references note.id
- students.matricule references note.id

26/12/2023

Mohammed Issam KABBAJ

81

Héritage par jointure des tables

```

@Inheritance(strategy = InheritanceType.JOINED)
public class Personne {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private String prenom;
    @Enumerated(EnumType.STRING)
    private Genre genre;
}

@Entity
public class Professor extends Personne {
    private Long numeroSum;
    private String matiere;
}

@Entity
public class Student extends Personne implements Comparable<Student> {
    private Long matricule;
    @Transient
    private String mail;
    @OneToMany(mappedBy = "student", fetch = FetchType.EAGER)
    private List<StudentNote> notes = new ArrayList<>();
}

```

26/12/2023

Mohammed Issam KABBAJ

82

Héritage par jointure des tables

personne - Table

Table Name: personne Schema: myjee Charset/Collation: utf8mb4_0900_ai_ Engine: InnoDB

Comments:

Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression

id	BIGINT	PK	NN	UQ	B	UN	ZF	AI	G	
name	VARCHAR(50)									NULL
genre	ENUM('Femme', 'Homme')									NULL
prenom	VARCHAR(255)									NULL

etudiants - Table

Table Name: etudiants Schema: myjee Charset/Collation: utf8mb4_0900_ai_ Engine: InnoDB

Comments:

Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression

id	BIGINT	PK	NN	UQ	B	UN	ZF	AI	G	
matricule	BIGINT									NULL

Foreign Key Name: FK6wmfogf1fujha5ktv4... Referenced Table: personne Column: id Referenced Column: id Foreign Key Options: On Update: RESTRICT On Delete: RESTRICT

Comments: Storage: Virtual Primary Key Unsigned Generated

On Null: Not Null Unique Binary Unsigned Zero Fill Auto Increment Generated

Columns Indexes Foreign Keys Triggers Partitioning Options

Result Grid | Filter Rows: | Result Grid | Filter Rows:

	id	numero_sum	matiere
1	Kabbaj	Homme	Mohammed Issam
2	Anwar	Homme	Adil
*	HULL	HULL	HULL

	id	matricule
1	1234	HULL
*	HULL	HULL

26/12/2023

Mohammed Issam KABBAJ

83

Héritage par une Table

personne - Table

Table Name: personne Schema: myjee Charset/Collation: utf8mb4_0900_ai_ Engine: InnoDB

Comments:

Column Name Datatype PK NN UQ B UN ZF AI G Default/Expression

type	VARCHAR(3)									
id	BIGINT	PK	NN	UQ	B	UN	ZF	AI	G	
matricule	BIGINT									NULL
numero_sum	BIGINT									NULL
name	VARCHAR(50)									NULL
genre	ENUM('Femme', 'Homme')									NULL
matiere	VARCHAR(255)									NULL
prenom	VARCHAR(255)									NULL

personne - Table

Limit to 1000 rows

1 • | SELECT * FROM myjee.personne;

Result Grid | Filter Rows: | Edit: | Export/Import: |

type	id	matricule	numero_sum	name	genre	matiere	prenom
Stu	1	1234	HULL	Kabbaj	Homme	HULL	Mohammed Issam
Pro	2	HULL	2345	Anwar	Homme	FrontEnd	Adil
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL

Entity
@DiscriminatorValue("Pro")
public class Professor extends Personne {
 4 usages
 private Long numeroSum;
 4 usages
 private String matiere;

 @Entity
 @Inheritance(strategy = InheritanceType.SINGLE_TABLE)
 @DiscriminatorColumn(name="type", discriminatorType = DiscriminatorType.STRING, length = 3)
 @DiscriminatorValue("Per")
 public class Personne {
 0 usages
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Long id;
 4 usages
 @Column(name = "name", nullable = false, length = 50, unique = true)
 private String name;
 4 usages
 private String prenom;
 1 usage
 @Enumerated(EnumType.STRING)
 private Genre genre;
 9 usages
 @Transient
 private Long matricule;
 4 usages
 @Transient
 private String mail;
 7 usages
 @OneToMany(mappedBy = "student", fetch = FetchType.EAGER)
 private List<StudentNote> notes = new ArrayList<>();
 }

26/12/2023

Mohammed Issam KABBAJ

84

Héritage par plusieurs tables (1/2)

```
4 usages 2 inheritors
@Entity
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
public class Personne {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO) 
    private Long id;
    4 usages
    @Column(name = "name", nullable = false, length = 50, unique = true)
    private String nom;
    4 usages
    private String prenom;
    1 usage
    @Enumerated(EnumType.STRING)
    private Genre genre;
```

Clé primaire ne doit pas être auto générer par IDENTITY

```
✓ @Entity
@Table(name = "etudiants")
public class Student extends Personne implements Comparable<Student> {
    9 usages
    private Long matricule;
    4 usages
    @Transient
    private String mail;
    7 usages
    @OneToOne(mappedBy = "student", fetch = FetchType.EAGER)
    private List<StudentNote> notes = new ArrayList<>();

@Entity
public class Professor extends Personne {
    4 usages
    private Long numeroSum;
    4 usages
    private String matiere;
```

26/12/2023

Mohammed Issam KABBAJ

85

Héritage par plusieurs tables (2/2)

Table Name	Schema	Comments
personne	myjee	
professor	myjee	
etudiants	myjee	

Table Name	Schema	Comments
personne	myjee	
professor	myjee	
etudiants	myjee	

Table Name	Schema	Comments
personne	myjee	
professor	myjee	
etudiants	myjee	

26/12/2023

Mohammed Issam KABBAJ

86

Spring Data

26/12/2023

Mohammed Issam KABBAJ

87

Spring Data - Introduction

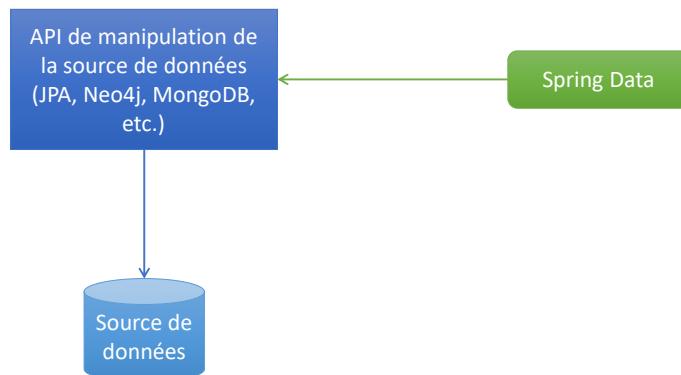
- Module Spring
- Son but :
 - Faciliter l'écriture des couches d'accès aux données.
 - Tenter d'offrir une abstraction commune pour l'accès aux données quelques soient les sources de données sous-jacentes, tout en prenant en compte les spécificités de celles-ci
- Sources de données : JPA, Neo4j, MongoDB, GemFire, Hadoop, ElasticSearch, REST, Redis, Couchbase, ...

26/12/2023

Mohammed Issam KABBAJ

88

Spring Data



26/12/2023

Mohammed Issam KABBAJ

89

Spring Data



26/12/2023

Mohammed Issam KABBAJ

90

Spring Data JPA

- Principe de base
- Pour écrire le DAO pour un type d'entités, il faut étendre certaines interfaces fournies par Spring Data :

```
public interface PersonneDao extends CrudRepository<Personne, Long> {
```

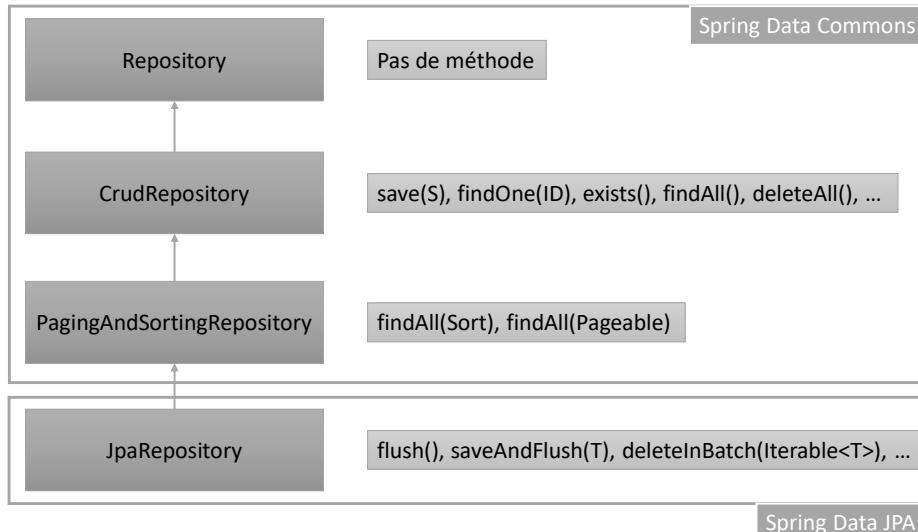
Nom de l'entité qu'on manipule

Type de l'identifiant de l'entité

Spring Data JPA

- Les différentes interfaces :
 - **Repository** : vous ne ferez pas grand-chose avec, hormis les méthodes que vous ajouterez
 - **CrudRepository** : vous aurez des fonctionnalités CRUD de base
 - **PagingAndSortingRepository** : vous aurez en plus des méthodes pour la pagination et le tri
 - **JpaRepository** : vous aurez en plus des méthodes propres à JPA

Spring Data JPA



26/12/2023

Mohammed Issam KABBAJ

93

Spring Data JPA

- Vous pouvez aussi écrire une requête juste avec le nom de la méthode
- Exemple :

```

public interface RequetesPersonnaliseesDao extends CrudRepository<Personne, Long> {
    public Personne findByNom(String nom);
    public Personne findByNomOrPrenom(String nom, String prenom);
    public Personne findByVelo(Velo velo);
    public Personne findByVeloAndNom(Velo velo, String nom);
    List<Person> findDistinctPeopleByLastnameOrFirstname(String lastname, String firstname);
    List<Person> findByLastnameAndFirstnameAllIgnoreCase(String lastname, String firstname);
    List<Person> findByLastnameOrderByFirstnameAsc(String lastname);
    ...
}
  
```

26/12/2023

Mohammed Issam KABBAJ

94

Spring Data JPA

- Il existe une série de mots-clés pour écrire sa requête
- (cf. annexe de la documentation de référence)

Logique	Mot-clé Spring Data
GREATER_THAN	GreaterThan, IsGreaterThan
IN	In, IsIn
IS	Is, Equals, (or no keyword)
IS_NOT_NULL	NotNull, IsNotNull
IS_NULL	Null, IsNull
LESS_THAN	LessThan, IsLessThan
LIKE	Like, IsLike
NOT	Not, IsNot
NOT_IN	NotIn, IsNotIn
NOT_LIKE	NotLike, IsNotLike
REGEX	Regex, MatchesRegex, Matches
...	...

Spring Data JPA

On peut utiliser tout un tas d'autres mots clés dans le nom des méthodes pour construire nos requêtes :

- And
- Or
- GreaterThan
- LessThan
- Like
- IsNull
- OrderBy
- ...

Spring Data JPA

- Spring va créer une requête à partir des propriétés et des mots-clés mentionnés dans le nom de la méthode
- Si on fait une recherche à partir d'une « sous-propriété », on donne à Spring le chemin vers celle-ci.
- Exemple : Si `Person` a une propriété `Address` qui a une propriété `ZipCode`, on peut faire :

```
List<Person> findByAddressZipCode(ZipCode zipCode);
```

Spring Data JPA

- S'il y a ambiguïté dans les propriétés, on peut mettre un « `_` ». Exemple :

Si la classe Person a un attribut `addressZip` et un autre `address` (de type `Address` qui contient `ZipCode`) :

```
List<Person> findByAddress_ZipCode(ZipCode zipCode);
```

Spring Data JPA

- Certains types de Spring sont automatiquement reconnus. Du coup, on peut ajouter des paramètres de pagination et de tri :

```
Page<User> findByLastname(String lastname, Pageable pageable);
List<User> findByLastname(String lastname, Sort sort);
List<User> findByLastname(String lastname, Pageable pageable);
```

Spring Data JPA

- Exemple :

```
public interface RequetesPersonnaliseesDao extends CrudRepository<Personne, Long> {
    public List<Personne> findByNomStartingWith(String nom, Sort ordreTri);
}
public void testRecuperationParNomEtTri() {
    String baseNom = "aaa";
    // sauvegarde des personnes avec pour nom : "baseNom"+i, avec i={0, 1, 2, 3}
    final List<Personne> listePersonnes =
        this.requetesPersonnaliseesDao.findByNomStartingWith(baseNom,new Sort(Direction.DESC, "nom"));
    for (Personne personne : listePersonnes)
        { System.out.println(personne.getNom()); }
}
```

Affiche : aaa3, aaa2, aaa1, aaa0

Spring Data JPA

- Requêtes nommées :

- On peut les mettre dans le META-INF/orm.xml ou en annotations dans l'entité
- Exemple :

```
@Entity
@NamedQuery(name = "User.findByEmailAddress",
query = "select u from User u where u.emailAddress = ?1")
public class User { ... }
```

- Et dans le repositories, on ne fait que déclarer la méthode :

```
public interface UserRepository
extends JpaRepository<User, Long> {
    User findByEmailAddress(String emailAddress);
}
```

Spring Data JPA

- Vous pouvez aussi ajouter l'annotation @Query si vos noms de méthodes sont beaucoup trop longues

- Exemple :

```
@Query("from Personne p where p.nom = ?1 and p.prenom = ?2")
public Personne maRequêteAvecQueryDeRechercheParNomEtPrenom(String nom, String prenom);
```

- Ca marche aussi avec les requêtes de modification, pour lesquelles il faut l'annotation @Modifying :

```
@Query("update Personne p set p.nom = :nom where p.id = :id")
@Modifying
public int metaJourNom(@Param("nom") String nom, @Param("id") Long id);
```

- On peut nommer les paramètres avec @Param

Spring Data JPA

- On peut mettre plusieurs arguments que Spring comprendra en fonction de leur déclaration dans la méthode

```
@Query("from Personne p where p.nom = :nom and p.prenom = :prenom")
public Personne maRequêteAvecQueryDeRechercheParNomEtPrenom(String nom, String prenom);
```

- @Query prend l'ascendant sur les requêtes nommées

Spring Data JPA

- Que ce soit pour @Query ou @NamedQuery, on peut mettre du code SQL natif (respectivement avec l'attribut nativeQuery et @NamedNativeQuery)

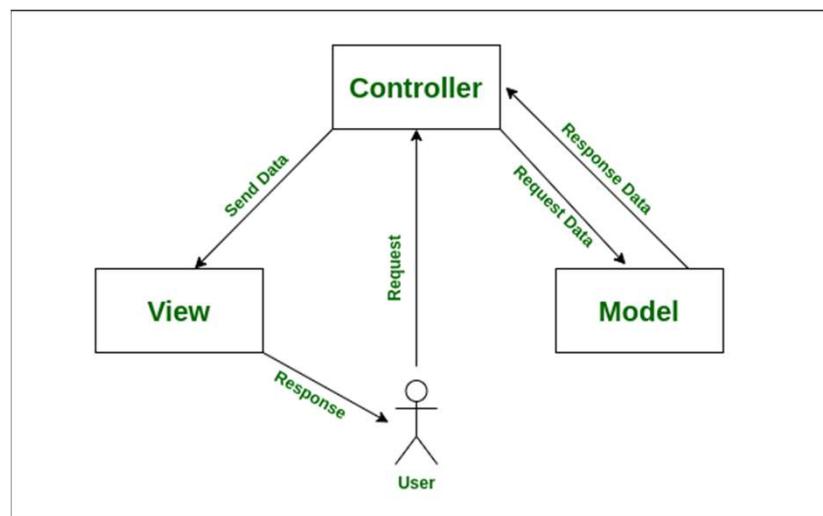
Spring Web

26/12/2023

Mohammed Issam KABBAJ

105

MVC

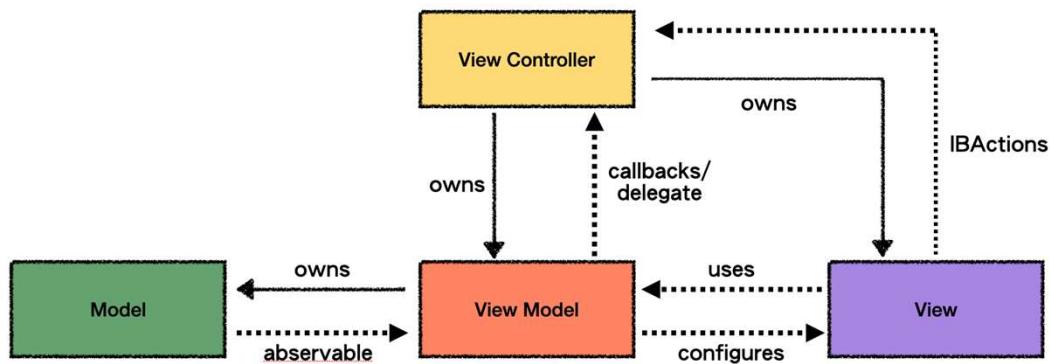


27/12/2023

Mohammed Issam KABBAJ

106

MVVM

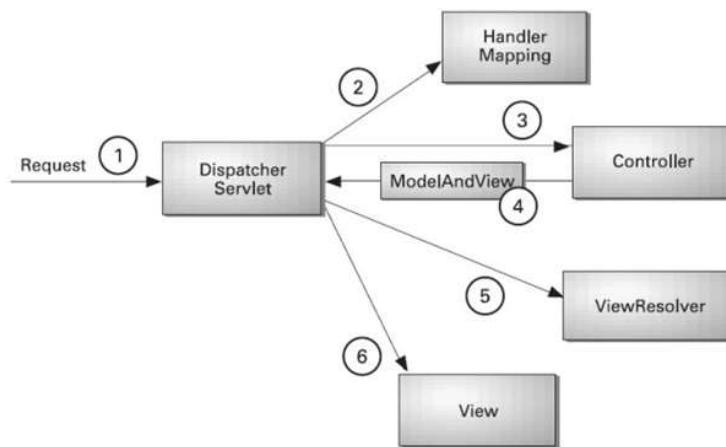


27/12/2023

Mohammed Issam KABBAJ

107

Spring MVC



27/12/2023

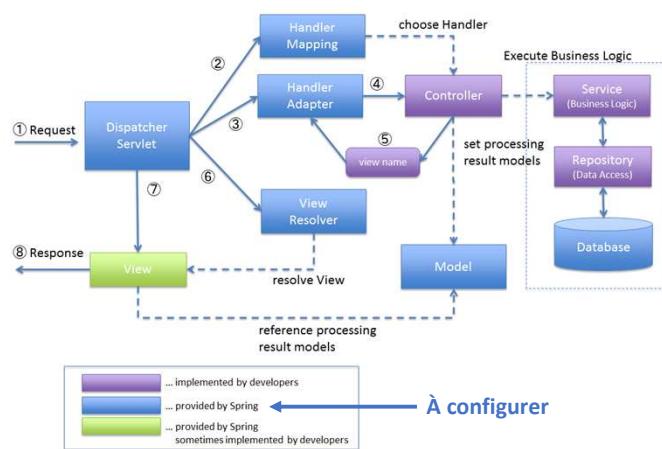
Mohammed Issam KABBAJ

108

Workflow Spring MVC

1. La demande sera reçue par le contrôleur frontal, c'est-à-dire DispatcherServlet.
2. DispatcherServlet transmet cette demande à HandlerMapping.
 1. HandlerMapping trouvera le contrôleur approprié pour la requête;
 2. Le HandlerMapping envoie les détails du contrôleur au DispatcherServlet.
3. Le DispatcherServlet appelle le contrôleur identifié par le HandlerMapping.
 1. Le contrôleur traitera la requête en appelant la méthode appropriée et en préparant les données.
 2. Il peut appeler une certaine logique métier ou extraire directement les données de la base de données.
4. Le contrôleur enverra ModelAndView (données du modèle et nom de la vue) à DispatcherServlet.
5. Une fois que DispatcherServlet reçoit l'objet ModelAndView, il le transmet à ViewResolver pour qu'il trouve la vue appropriée.
 1. ViewResolver identifiera la vue et la renverra à DispatcherServlet.
6. Le DispatcherServlet appellera la vue appropriée identifiée par ViewResolver.
 1. La vue crée une réponse sous forme de HTML et l'envoie au DispatcherServlet.
7. Le DispatcherServlet envoie la réponse au navigateur. Le navigateur rendra le code HTML et l'affichera à l'utilisateur final.

Spring MVC



Database as API

- Modifier le POM en ajoutant la dépendance suivante :

```
37 <dependency>
38     <groupId>org.springframework.boot</groupId>
39     <artifactId>spring-boot-starter-data-rest</artifactId>
40 </dependency>
```

- Modifier vos interfaces Repository en ajoutant l'annotation **RestController**

```
6 @RepositoryRestResource(collectionResourceRel = "students", path = "students")
7 public interface StudentRepository extends JpaRepository<Student, Long> {
8 }
```

27/12/2023

Mohammed Issam KABBAJ

111

GET

The screenshot shows a Java IDE interface with several tabs at the top: pom.xml (RepositoryRestController), application.properties, Student.java, StudentRepository.java, and rest-api_1.http. The rest-api_1.tab is active.

The rest-api_1.tab contains a test configuration:

```
1 > GET http://localhost:8080/students
2 Accept: application/json
3
4 <> 2023-12-27T060859.200.json
5
6 ###
```

Below the test configuration, there is a Services section and an expanded view of the test results:

Services

- HTTP Request
- GET rest-api_1 | #1 Status: 200 (267 ms)
- Docker

HTTP/1.1 200

(Headers) ...Content-Type: application/json...

```
{
  "_embedded": {
    "students": []
  },
  "_links": {
    "self": {
      "href": "http://localhost:8080/students?page=0&size=20"
    },
    "profile": {
      "href": "http://localhost:8080/profile/students"
    }
}
```

27/12/2023

Mohammed Issam KABBAJ

112

GET

```
D:\Dev\Spring\gs-accessing-data-rest\RepositoryRestController>curl http://localhost:8080/students
{
  "_embedded" : {
    "students" : [ ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/students?page=0&size=20"
    },
    "profile" : {
      "href" : "http://localhost:8080/profile/students"
    }
  },
  "page" : {
    "size" : 20,
    "totalElements" : 0,
    "totalPages" : 0,
  }
}
```

27/12/2023

Mohammed Issam KABBAJ

113

POST

The screenshot shows a developer tool interface with several tabs at the top: pom.xml (RepositoryRestController), application.properties, Student.java, StudentRepository.java, and rest-api_1.http. The rest-api_1.http tab is active.

In the main area, there is a code editor with the following POST request:

```
1 > POST http://localhost:8080/students
2 Content-Type: application/json
3
4 {
5   "matricule": 1234,
6   "nom": "Kabbaj",
7   "prenom": "Mohammed Issam"
8 }
```

Below the code editor, there is a "Services" section with a tree view. Under "HTTP Request", there are two entries: "POST rest-api_1 | #1 Status: 201 (110 ms)" and "GET rest-api_1 | #2 Status: 200 (19 ms)". The second entry is highlighted. To the right of the tree view, there is a preview of the response:

HTTP/1.1 200
(Headers) ...Content-Type: application/json...

```
{
  "nom": "Kabbaj",
  "prenom": "Mohammed Issam",
  "_links": {
    "self": {
      "href": "http://localhost:8080/students/1234"
    }
  }
}
```

27/12/2023

Mohammed Issam KABBAJ

114

POST

```
Terminal Local Command Prompt + 
D:\Dev\Spring\gs-accessing-data-rest\RepositoryRestController>curl -i -H "Content-Type:application/json" -d "{\"matricule\": 2345, \"nom\": \"ANWAR\", \"prenom\": \"Adil\"}" http://localhost:8080/students
HTTP/1.1 201
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Location: http://localhost:8080/students/2345
Content-Type: application/hal+json
Transfer-Encoding: chunked
Date: Wed, 27 Dec 2023 05:19:10 GMT

{
  "nom" : "ANWAR",
  "prenom" : "Adil",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/students/2345"
    },
    "student" : {
      "href" : "http://localhost:8080/students/2345"
    }
}
```

27/12/2023

Mohammed Issam KABBAJ

115

PUT

```
D:\Dev\Spring\gs-accessing-data-rest\RepositoryRestController>curl -X PUT -H "Content-Type:application/json" -d "{\"nom\": \"KABBAJ\", \"prenom\": \"Adil\"}" http://localhost:8080/students/1234
{
  "nom" : "KABBAJ",
  "prenom" : "Adil",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/students/1234"
    },
    "student" : {
      "href" : "http://localhost:8080/students/1234"
    }
}
```

27/12/2023

Mohammed Issam KABBAJ

116

PATCH

```
D:\Dev\Spring\gs-accessing-data-rest\RepositoryRestController>curl -X PATCH -H "Content-Type:application/json" -d "{\"prenom\": \"Mohammed Issam\"}" http://localhost:8080/students/2345
{
  "nom" : "ANWAR",
  "prenom" : "Mohammed Issam",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/students/2345"
    },
    "student" : {
      "href" : "http://localhost:8080/students/2345"
    }
  }
}
```

27/12/2023

Mohammed Issam KABBAJ

117

Database as API - DELETE

```
D:\Dev\Spring\gs-accessing-data-rest\RepositoryRestController>curl -X DELETE http://localhost:8080/students/2345
{
  "nom" : "ANWAR",
  "prenom" : "Mohammed Issam",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/students/2345"
    },
    "student" : {
      "href" : "http://localhost:8080/students/2345"
    }
  }
}
D:\Dev\Spring\gs-accessing-data-rest\RepositoryRestController>curl http://localhost:8080/students/2345
```

27/12/2023

Mohammed Issam KABBAJ

118

Spring MVC : Dependence

```

19 <dependencies> Edit Starters...
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-web</artifactId>
23   </dependency>
24
25   <dependency>
26     <groupId>org.springframework.boot</groupId>
27     <artifactId>spring-boot-starter-test</artifactId>
28     <scope>test</scope>
29   </dependency>
30 </dependencies>

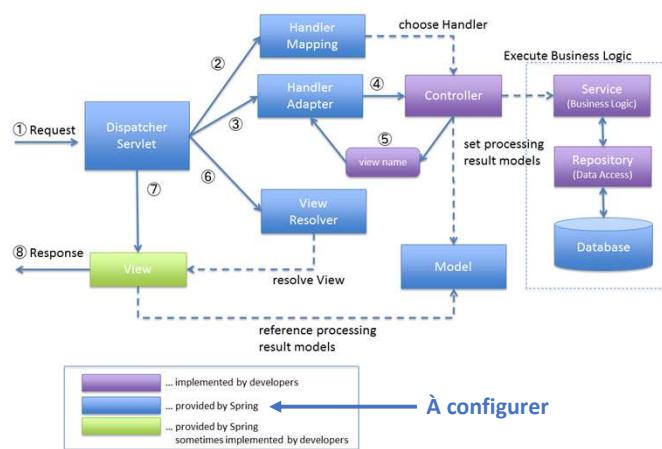
```

27/12/2023

Mohammed Issam KABBAJ

119

Spring MVC



27/12/2023

Mohammed Issam KABBAJ

120

Controller

```

5  @Controller
6  public class Controleur {
7
8      no usages
9      public String Coucou(){
10         return "Hello";
11     }

```

Son HandlerMapping ?

27/12/2023

Mohammed Issam KABBAJ

121

HandlerMapping

```

8  @Controller
9  public class Controleur {
10
11  @RequestMapping(path = "/coco", method = RequestMethod.GET)
12  public String Coucou(){
13      return "Hello";
14  }
15 }

```

ViewResolver ?

27/12/2023

Mohammed Issam KABBAJ

122

ViewResolver

Controller.java × pom.xml (Controller)

```

9  public class Controller {
10
11     @RequestMapping(value = "/coco", method = RequestMethod.GET)
12     @ResponseBody
13     > public String coucou() { return "Hello"; }
14
15 }
```

Terminal Local + ▾

```
D:\Dev\Spring\Controller>curl http://localhost:8080/coco
Hello
D:\Dev\Spring\Controller>
```

27/12/2023

Mohammed Issam KABBAJ

123

Méthode

Controller.java × pom.xml (Controller)

```

8  @Controller
9  public class Controller {
10
11     @RequestMapping(value = "/coco", method = RequestMethod.GET)
12     @ResponseBody
13     > public String coucou1() { return "Hello"; }
14     @RequestMapping(value = "/coco", method = RequestMethod.POST)
15     @ResponseBody
16     > public String coucou2() { return "Bonjour"; }
17
18 }
```

Terminal Local + ▾

```
D:\Dev\Spring\Controller>curl http://localhost:8080/coco
Hello
D:\Dev\Spring\Controller>curl -X POST http://localhost:8080/coco
Bonjour
D:\Dev\Spring\Controller>
```

27/12/2023

Mohammed Issam KABBAJ

124

RestController

The screenshot shows a Java IDE interface. At the top, there are tabs for 'Controller.java' and 'pom.xml (Controller)'. Below the tabs is the code editor containing the following Java code:

```

8
9  @RestController
10 public class Controleur {
11
12      @RequestMapping(value = "/coco", method = RequestMethod.GET)
13      >     public String coucou1() { return "Hello"; }
14      @RequestMapping(value = "/coco", method = RequestMethod.POST)
15      >     public String coucou2() { return "Bonjour"; }
16
17 }

```

Below the code editor is a terminal window titled 'Terminal' with the following output:

```

D:\Dev\Spring\Controller>curl -X POST http://localhost:8080/coco
Bonjour
D:\Dev\Spring\Controller>curl http://localhost:8080/coco
Hello
D:\Dev\Spring\Controller>

```

27/12/2023

Mohammed Issam KABBAJ

125

@XXXMapping

The screenshot shows a Java IDE interface. At the top, there are tabs for 'Controller.java' and 'pom.xml (Controller)'. Below the tabs is the code editor containing the following Java code:

```

6
7  @RestController
8
9      @GetMapping(value = "/coco")
10     >    public String coucou1() { return "Hello"; }
11      @PostMapping(value = "/coco")
12     >    public String coucou2() { return "Bonjour"; }
13
14 }

```

Below the code editor is a terminal window titled 'Terminal' with the following output:

```

D:\Dev\Spring\Controller>curl http://localhost:8080/coco
Hello
D:\Dev\Spring\Controller>curl -X POST http://localhost:8080/coco
Bonjour
D:\Dev\Spring\Controller>

```

27/12/2023

Mohammed Issam KABBAJ

126

RequestMapping

The screenshot shows a Java code editor with a file named `Controller.java` open. The code defines a `@RestController` class named `Controlleur` with two methods: `coucou1()` and `coucou2()`. The `coucou1()` method is annotated with `@GetMapping(value = "/coco")`, and the `coucou2()` method is annotated with `@PostMapping(value = "/coco")`. Below the code editor is a terminal window showing curl commands being run against a local host at port 8080. The first command, `curl http://localhost:8080/coco`, results in a 404 error. The second command, `curl http://localhost:8080/api/v1/coco`, returns "Hello". The third command, `curl -X POST http://localhost:8080/api/v1/coco`, returns "Bonjour".

```

Controller.java × pom.xml (Controller)
6   @RestController
7   @RequestMapping(value = "/api/v1")
8   public class Controlleur {
9
10      @GetMapping(value = "/coco")
11     >     public String coucou1() { return "Hello"; }
12     @PostMapping(value = "/coco")
13     >     public String coucou2() { return "Bonjour"; }
14
15
16
17
18 }

Terminal Local × + ▾

D:\Dev\Spring\Controller>curl http://localhost:8080/coco
{"timestamp":"2023-12-27T09:48:33.233+00:00","status":404,"error":"Not Found","path":"/coco"}
D:\Dev\Spring\Controller>curl http://localhost:8080/api/v1/coco
Hello
D:\Dev\Spring\Controller>curl -X POST http://localhost:8080/api/v1/coco
Bonjour
D:\Dev\Spring\Controller>

```

27/12/2023

Mohammed Issam KABBAJ

127

Liste des étudiants

The screenshot shows a Java code editor with a file named `StudentController.java` open. The code defines a `@RestController` class named `StudentController` with one method `studentList()`. This method is annotated with `@GetMapping(value = "/")` and returns a list of students from a `StudentRepository`. Below the code editor is a terminal window showing curl commands being run against a local host at port 8080. The command `curl http://localhost:8080/api/v1/students/` returns an empty JSON array [].

```

StudentController.java × pom.xml (RestController)
11   @RequestMapping(value = "/api/v1/students")
12   public class StudentController {
13
14     11 usages
15     StudentRepository studentRepository;
16
17     >     StudentController(StudentRepository studentRepository) { this.studentRepository = studentRepository; }
18
19
20     @GetMapping(value = "/")
21     >     public List<Student> studentList(){
22       return studentRepository.findAll();
23     }
24
25

Terminal Local × Command Prompt × + ▾

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/
[]
D:\Dev\Spring\RestController>

```

27/12/2023

Mohammed Issam KABBAJ

128

RequestBody

The screenshot shows an IDE interface with two tabs: 'StudentController.java' and 'pom.xml (RestController)'. The code in 'StudentController.java' contains a POST mapping method:

```

24     @PostMapping("/")
25     > public Student studentList(@RequestBody Student s) { return studentRepository.save(s); }
26
27
28
  
```

Below the code is a terminal window showing the output of a curl command:

```

D:\Dev\Spring\RestController>curl -i -H "Content-Type:application/json" -d "{\"matricule\": 1234, \"nom\": \"KABBAJ\", \"prenom\": \"Mohammed Issam\"}" http://localhost:8080/api/v1/students/
HTTP/1.1 200
Content-Type: application/json
Transfer-Encoding: chunked
Date: Wed, 27 Dec 2023 08:46:36 GMT

{"matricule":1234,"nom":"KABBAJ","prenom":"Mohammed Issam"}
D:\Dev\Spring\RestController>
  
```

27/12/2023

Mohammed Issam KABBAJ

129

PathVariable

The screenshot shows an IDE interface with two tabs: 'StudentController.java' and 'pom.xml (RestController)'. The code in 'StudentController.java' contains a GET mapping method with a path variable:

```

44
45     @GetMapping("/{id}")
46     > public Student student(@PathVariable Long id) { return studentRepository.findById(id).get(); }
47
  
```

Below the code is a terminal window showing the output of a curl command:

```

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/1234
{"matricule":1234,"nom":"KABBAJ","prenom":"Mohammed Issam"}
D:\Dev\Spring\RestController>
  
```

27/12/2023

Mohammed Issam KABBAJ

130

PathVariable

The screenshot shows a Java IDE interface with two tabs: 'StudentController.java' and 'pom.xml (RestController)'. The code in 'StudentController.java' is as follows:

```

50     @GetMapping("/{nom}/{prenom}")
51     public List<Student> student(@PathVariable String nom, @PathVariable String prenom){
52         return studentRepository.findByNomAndPrenom(nom, prenom);
53     }

```

Below the code editor is a terminal window titled 'Command Prompt' with the following history:

```

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/ANWAR/Adil
[{"matricule":2345,"nom":"ANWAR","prenom":"Adil"}]
D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/KABBAJ/Mohammed%20Issam
[{"matricule":1234,"nom":"KABBAJ","prenom":"Mohammed Issam"}]
D:\Dev\Spring\RestController>

```

27/12/2023

Mohammed Issam KABBAJ

131

RequestParam

The screenshot shows a Java IDE interface with two tabs: 'StudentController.java' and 'pom.xml (RestController)'. The code in 'StudentController.java' is as follows:

```

54
55     @GetMapping("/search")
56     public List<Student> student(@RequestParam String nom) { return studentRepository.findByNom(nom); }
57
58
59

```

Below the code editor is a terminal window titled 'Command Prompt' with the following history:

```

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/search
{"timestamp":"2023-12-27T08:49:59.607+00:00","status":400,"error":"Bad Request","path":"/api/v1/students/search"}
D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/search?nom=KABBAJ
[{"matricule":1234,"nom":"KABBAJ","prenom":"Mohammed Issam"}]
D:\Dev\Spring\RestController>

```

27/12/2023

Mohammed Issam KABBAJ

132

Put

The screenshot shows a Java IDE interface with two tabs: 'StudentController.java' and 'pom.xml (RestController)'. The code in 'StudentController.java' is as follows:

```

32     @PutMapping("/{matricule}")
33     public void studentList(@RequestBody Student s, @PathVariable Long matricule){
34         studentRepository.findById(matricule)
35             .map(student -> {
36                 student.setNom(s.getNom());
37                 student.setPrenom(s.getPrenom());
38                 return studentRepository.save(student);
39             })
40             .orElseGet(() -> studentRepository.save(s));
41     }

```

Below the code editor is a terminal window titled 'Command Prompt' with the following history:

```

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/
[{"matricule":1234,"nom":"KABBAJ","prenom":"Mohammed Issam"}]
D:\Dev\Spring\RestController>curl -X PUT http://localhost:8080/api/v1/students/1234 -H "Content-type:application/json" -d "{\"nom\": \"KABBAJ\", \"prenom\": \"Adil\"}"
D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/
[{"matricule":1234,"nom":"KABBAJ","prenom":"Adil"}]
D:\Dev\Spring\RestController>

```

27/12/2023

Mohammed Issam KABBAJ

133

Delete

The screenshot shows a Java IDE interface with two tabs: 'StudentController.java' and 'pom.xml (RestController)'. The code in 'StudentController.java' is as follows:

```

42
43     @DeleteMapping("/{id}")
44     public void delete(@PathVariable Long id) { studentRepository.deleteById(id); }
45

```

Below the code editor is a terminal window titled 'Command Prompt' with the following history:

```

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/
[{"matricule":1234,"nom":"KABBAJ","prenom":"Mohammed Issam"}]
D:\Dev\Spring\RestController>curl -X DELETE http://localhost:8080/api/v1/students/1234

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/
[]
D:\Dev\Spring\RestController>

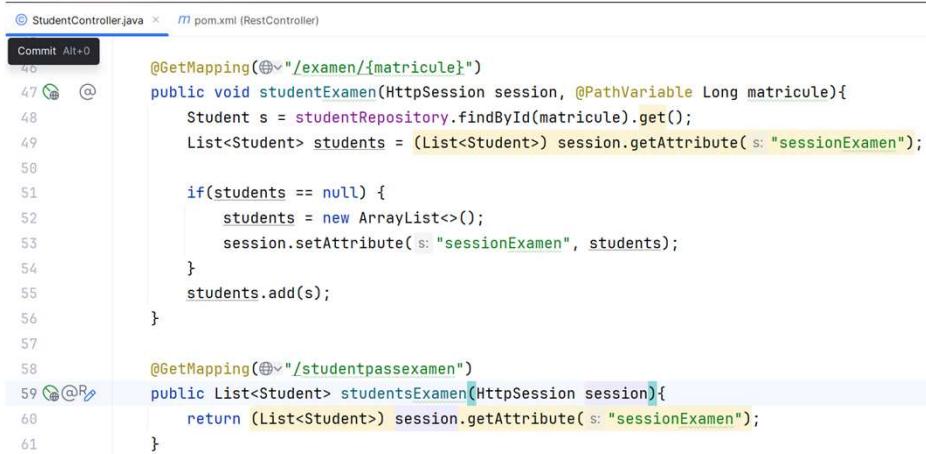
```

27/12/2023

Mohammed Issam KABBAJ

134

HttpSession



```

  ① StudentController.java × pom.xml (RestController)
Commit Alt+0
  46
  47 @GetMapping("/examen/{matricule}")
  48   public void studentExamen(HttpServletRequest session, @PathVariable Long matricule){
  49     Student s = studentRepository.findById(matricule).get();
  50     List<Student> students = (List<Student>) session.getAttribute("sessionExamen");
  51
  52     if(students == null) {
  53       students = new ArrayList<>();
  54       session.setAttribute("sessionExamen", students);
  55     }
  56     students.add(s);
  57   }
  58
  59 @GetMapping("/studentpassexamen")
  60   public List<Student> studentsExamen(HttpServletRequest session){
  61     return (List<Student>) session.getAttribute("sessionExamen");
  62   }

```

27/12/2023

Mohammed Issam KABBAJ

135

Session Test

27/12/2023

Mohammed Issam KABBAJ

136

Redirection

StudentController.java

```

75 @GetMapping(value = "/SiteEMI")
76     public ModelAndView emi(){
77         return new ModelAndView(viewName: "redirect:/api/v1/students/search?nom=KABBAJ");
78     }

```

New Tab

- http://localhost:8080/api/v1/students/SiteEMI
- http://localhost:8080/api/v1/students/SiteEMI**
- http://localhost:8080/api/v1/students/SiteEMI - Google Search

localhost:8080/api/v1/students/

```
[{"matricule":1234,"nom":"KABBAJ","prenom":"Mohammed Issam"}]
```

Mohammed Issam KABBAJ

27/12/2023 137

Redirection

Services

HTTP Request

- POST generated-requests | #1
- GET generated-requests | #2
- GET generated-requests | #3
- GET generated-requests | #4 Status: 200 (91 ms)**

Docker

GET http://localhost:8080/api/v1/students/SiteEMI

Show Request

Redirections: Disable

-> http://localhost:8080/api/v1/students/search?nom=KABBAJ

HTTP/1.1 200

(Headers) Content-Type: application/json...

```
[
  {
    "matricule": 1234,
    "nom": "KABBAJ",
    "prenom": "Mohammed Issam"
  }
]
```

Mohammed Issam KABBAJ

27/12/2023 138

Profiles : propriétés de configuration

- Créer une classe

```

no usages
6 @ConfigurationProperties(prefix = "service")
7 @Component
8 class ServiceConfiguration {
9     private String url;
10    private String username;
11    private String password;
12
13    public String getUrl() {
14        return url;
15    }
16
17    public void setUrl(String url) {
18        this.url = url;
19    }
20
21    public String getUsername() {
22
23    }

```

26/12/2023

Mohammed Issam KABBAJ

139

Profiles : propriétés de configuration

- Définir les valeurs dans application.properties

application.properties	application-dev.properties	application-prod.properties
logging.level.org.springframework=debug	logging.level.org.springframework=trace	logging.level.org.springframework=info
spring.profiles.active=prod	monservice.url=http://localhost:9999/	monservice.url=http://service.emi.ac.ma/
monservice.username=defaultuser	monservice.username=devuser	monservice.username=produser
monservice.password=servicepassword	monservice.password=mysecretpassword	monservice.password=complexepassword

26/12/2023

Mohammed Issam KABBAJ

140

Profiles : propriétés de configuration

- Utiliser les valeurs

```
2023-11-14T01:31:19.011+01:00 TRACE 202 http://localhost:9999/
devuser
mysecretpassword
2023-11-14T01:31:19.016+01:00 TRACE 202 http://service.emi.ac.ma/
produser
complexepassword
```

```
MonPremierProjetApplication.java
1 package ma.ac.emi.ginfo.monpremierprojet;
2
3 > import ...
4
5 @SpringBootApplication
6 public class MonPremierProjetApplication {
7
8     @Autowired
9     private ServiceConfiguration configuration;
10
11     public static void main(String[] args) { SpringApplication.run(MonPremierProjetApplication.class, args); }
12
13     @Bean
14     public CommandLineRunner useService(){
15         return args->{
16             System.out.println(configuration.getUrl());
17             System.out.println(configuration.getUsername());
18             System.out.println(configuration.getPassword());
19         };
20     }
21
22 }
23
24
25
26
27 }
```

26/12/2023

Mohammed Issam KABBAJ

141

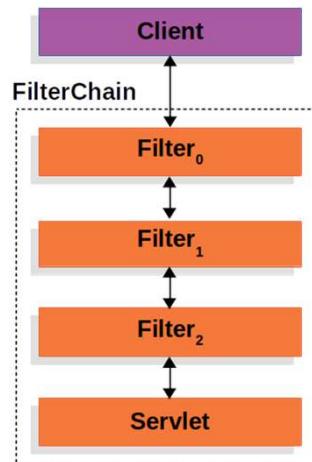
Spring Sécurité

26/12/2023

Mohammed Issam KABBAJ

142

Filtres

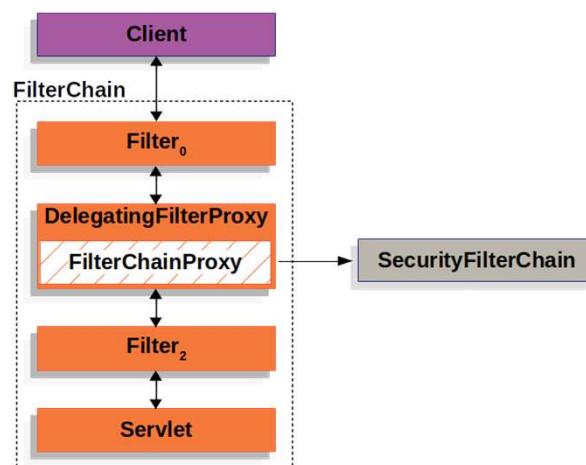


27/12/2023

Mohammed Issam KABBAJ

143

Sécurité autant que filtre

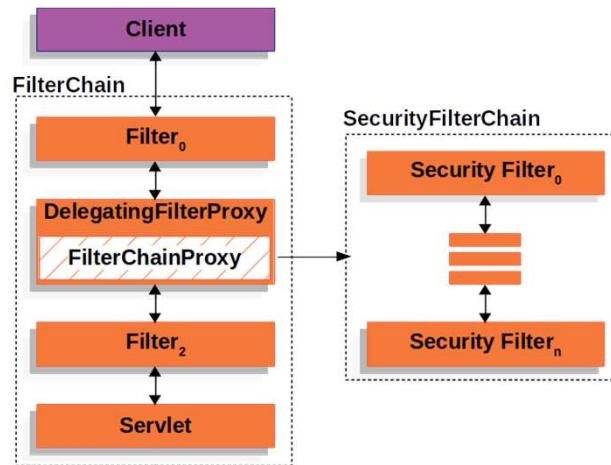


27/12/2023

Mohammed Issam KABBAJ

144

SecurityFilterChain

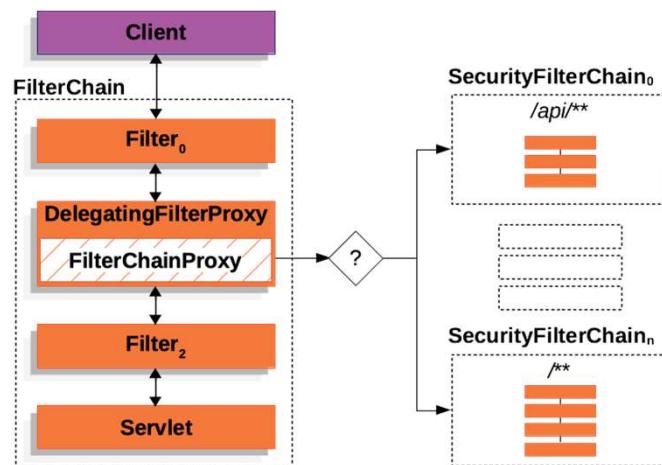


27/12/2023

Mohammed Issam KABBAJ

145

Multiple SecurityFilterChain

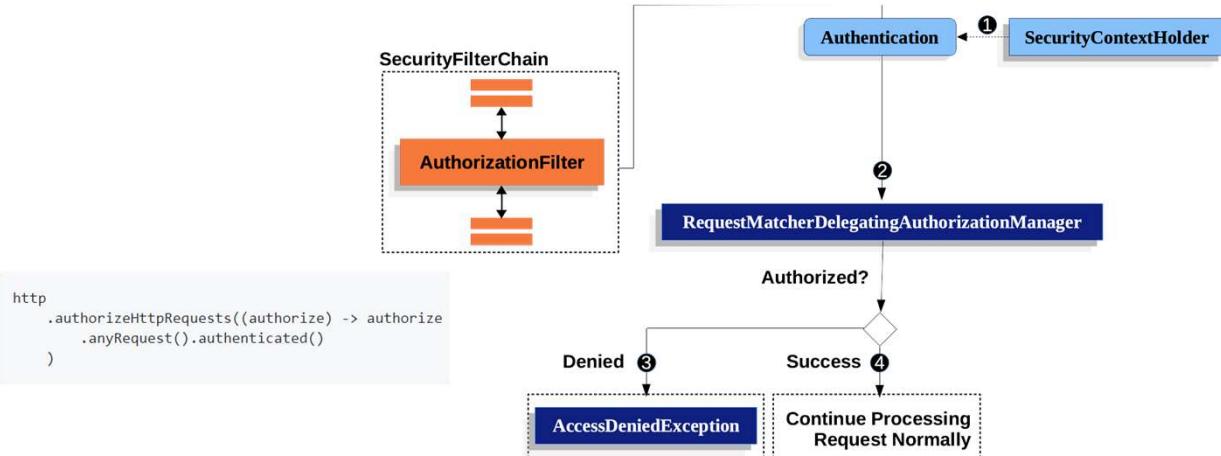


27/12/2023

Mohammed Issam KABBAJ

146

Sécurité basé sur les requêtes



27/12/2023

Mohammed Issam KABBAJ

147

Example

```

@Bean
SecurityFilterChain web(HttpSecurity http) throws Exception {
    http
        // ...
        .authorizeHttpRequests(authorize -> authorize
            .dispatcherTypeMatchers(FORWARD, ERROR).permitAll() ①
            .requestMatchers("/static/**", "/signup", "/about").permitAll() ②
            .requestMatchers("/admin/**").hasRole("ADMIN") ③
            .requestMatchers("/db/**").access(allOf(hasAuthority('db'), hasRole('ADMIN'))) ④
            .anyRequest().denyAll() ⑤
        );
        return http.build();
} ⑥

```

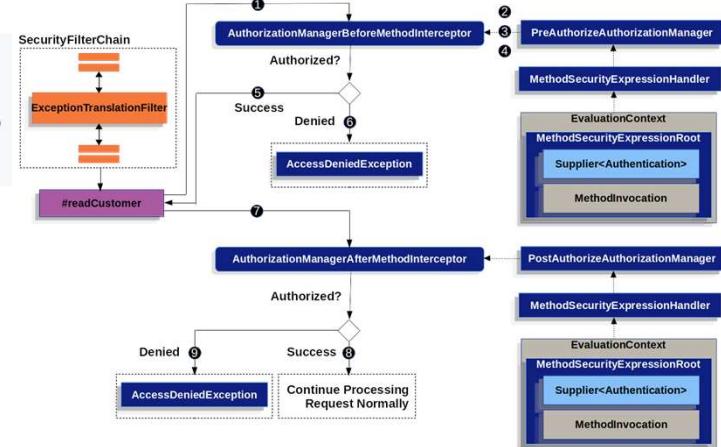
27/12/2023

Mohammed Issam KABBAJ

148

Sécurité basé sur les méthodes

```
@Service
public class MyCustomerService {
    @PreAuthorize("hasAuthority('permission:read')")
    @PostAuthorize("returnObject.owner == authentication.name")
    public Customer readCustomer(String id) { ... }
}
```



27/12/2023

Mohammed Issam KABBAJ

149

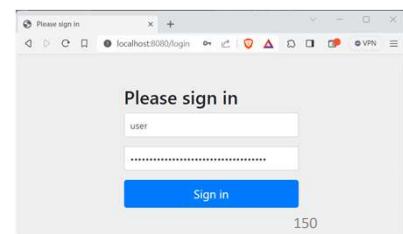
Spring security

- ajouter la dépendance suivante au pom de notre application.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

- Spring Boot nous fournit donc un mot de passe par défaut.**

```
:
:: Spring Boot ::      (v2.0.4.RELEASE)
2018-09-10 22:44:09.059  INFO 645 --- [           main] .s.s.UserDetailsServiceAutoConfiguration :
Using generated security password: c657aeef6-758a-409d-ac02-814ff4df55be
```



26/12/2023

Mohammed Issam KABBAJ

150

Spring security

- Dans Spring Boot, en ajoutant simplement le démarreur de sécurité, la configuration automatique configure un certain nombre de fonctionnalités de sécurité par défaut telles que :
 - Créez un seul utilisateur avec le nom d'utilisateur « user » et le mot de passe généré automatiquement.
 - Configure automatiquement l'authentification de base pour l'accès à l'API.
 - Configure un formulaire de connexion par défaut pour les applications Web.
 - La protection CSRF (Cross-Site Request Forgery) est activée par défaut pour empêcher les attaques CSRF.
 - Permet la surveillance et le suivi des activités liées à la sécurité dans l'application.
 - Configure la gestion de session, le délai d'expiration de session et la protection de fixation de session.
 - Les mots de passe sont automatiquement cryptés à l'aide de bcrypt par défaut.
 - Et plus.

WebSecurityConfig

```
14 @Configuration  
15 @EnableWebSecurity  
16  public class WebSecurityConfig {
```

Authorization

```

@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

    http
        .authorizeHttpRequests((authorize) -> authorize
            .anyRequest().authenticated()
        )
        .httpBasic(withDefaults())
        .formLogin(withDefaults());
    return http.build();
}

```

27/12/2023

Mohammed Issam KABBAJ

153

Authentification

```

@Bean
public UserDetailsService users() {
    UserDetails user = User.builder()
        .username("user")
        .password("{bcrypt}$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0Fx0/BTk76kLW")
        .roles("USER")
        .build();
    UserDetails admin = User.builder()
        .username("admin")
        .password("{bcrypt}$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0Fx0/BTk76kLW")
        .roles("USER", "ADMIN")
        .build();
    return new InMemoryUserDetailsManager(user, admin);
}

```

27/12/2023

Mohammed Issam KABBAJ

154

Authentification

```

@Bean
public InMemoryUserDetailsManager userDetailsService() {
    UserDetails user = User.withDefaultPasswordEncoder()
        .username("user")
        .password("password")
        .roles("USER")
        .build();
    return new InMemoryUserDetailsManager(user);
}

```

27/12/2023

Mohammed Issam KABBAJ

155

Login page

```

login.html ✘
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
3     xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
4     <head>
5         <title>Spring Security Example </title>
6     </head>
7     <body>
8         <div th:if="${param.error}">
9             Invalid username and password.
10        </div>
11        <div th:if="${param.Logout}">
12            You have been logged out.
13        </div>
14        <form th:action="@{/Login}" method="post">
15            <div><label> User Name : <input type="text" name="username"/> </label></div>
16            <div><label> Password: <input type="password" name="password"/> </label></div>
17            <div><input type="submit" value="Sign In"/></div>
18        </form>
19    </body>
20 </html>

```

Endpoint security

26/12/2023

Mohammed Issam KABBAJ

156

Utilisateur Principal



```

1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
3     xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
4     <head>
5         <title>Hello World!</title>
6     </head>
7     <body>
8         <h1 th:inline="text">Hello [[${#httpServletRequest.remoteUser}]]!</h1>
9         <form th:action="@{/logout}" method="post">
10            <input type="submit" value="Sign Out"/>
11        </form>
12    </body>
13 </html>

```

Utilisateur principal

Endpoint security

26/12/2023

Mohammed Issam KABBAJ

157

Test de Get

```

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/
[]

D:\Dev\Spring\RestController>curl http://localhost:8080/api/v1/students/ -u user:password
[]

```

26/12/2023

Mohammed Issam KABBAJ

158

Test de POST

```
D:\Dev\Spring\RestController>curl -i -H "Content-Type:application/json" -d "{\"matricule\": 1234, \"nom\": \"KABBAJ\", \"prenom\": \"Mohammed Issam\"}" http://localhost:8080/api/v1/students/
HTTP/1.1 401
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Set-Cookie: JSESSIONID=1D23FEAA0481859324E60C59F4836A6A; Path=/; HttpOnly
X-Content-Type-Options: nosniff
X-XSS-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
WWW-Authenticate: Basic realm="Realm"
Content-Length: 0
Date: Wed, 27 Dec 2023 10:38:43 GMT
```

27/12/2023

Mohammed Issam KABBAJ

159

Création Base de données

- create table users(
 - username varchar_ignorecase(50) not null primary key,
 - password varchar_ignorecase(500) not null,
 - enabled boolean not null
 -);

- create table authorities (
 - username varchar_ignorecase(50) not null,
 - authority varchar_ignorecase(50) not null,
 - constraint fkAuthorities_users foreign key(username) references users(username)
 -);
- create unique index ix_auth_username on authorities (username,authority);

27/12/2023

Mohammed Issam KABBAJ

160

Déclaration de la DataSource

```
@Bean
DataSource dataSource() {
    return new EmbeddedDatabaseBuilder()
        .setType(EmbeddedDatabaseType.H2)
        .addScript(JdbcDaoImpl.DEFAULT_USER_SCHEMA_DDL_LOCATION)
        .build();
}
```

Définition de l'utilisateur

```
@Bean
UserDetailsManager users(DataSource dataSource) {
    UserDetails user = User.builder()
        .username("user")
        .password("{bcrypt}$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0Fx0/BTk76kLW")
        .roles("USER")
        .build();
    UserDetails admin = User.builder()
        .username("admin")
        .password("{bcrypt}$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0Fx0/BTk76kLW")
        .roles("USER", "ADMIN")
        .build();
    JdbcUserDetailsManager users = new JdbcUserDetailsManager(dataSource);
    users.createUser(user);
    users.createUser(admin);
    return users;
}
```

Sécurité à travers Spring DATA

```
@Bean  
public SecurityEvaluationContextExtension securityEvaluationContextExtension() {  
    return new SecurityEvaluationContextExtension();  
}
```

Actuator

Actuator

- Spring Boot Actuator nous permet de surveiller notre application, d'obtenir des informations sur le projet, de recueillir des métriques ou simplement de vérifier l'état de notre application/base de données.
- Fournit un certain nombre de points de terminaison :
 - beans : Liste complète des beans de Spring dans votre application
 - health : Informations sur l'état de l'application
 - metrics : Métriques d'application
 - mappings : Détails sur les mappages de requêtes
- Seul health est activé par défaut
- Pour activer les autres, il faut ajouter à application.properties

`management.endpoints.web.exposure.include=*`

or

`management.endpoints.web.exposure.include=health,metrics`

Actuator

- localhost:8080/actuator
- <http://localhost:8080/actuator/health>
- <http://localhost:8080/actuator/beans>
- <http://localhost:8080/actuator/configprops>
- <http://localhost:8080/actuator/env>
- <http://localhost:8080/actuator/metrics>
- localhost:8080/actuator/metrics/http.server.requests
 - A quoi correspondent les métriques affichées ?

Actuator

- Comment connaitre la version de l'application en cours d'exécution ?
- Terminaison Info vous fournie des informations importantes sur le contexte de votre application, comme il vous permet de les enrichir avec des informations personnalisées.
- Modifier le fichier de application.properties

management.info.env.enabled=true

management.info.os.enabled=true

info.app.version=1.0-Beta

- Consulter le terminaison Info
 - <http://localhost:8080/actuator/info>