FORMATION

- Management 2024

Bienvenue sur cette formation React

- Clément Hamon
- Développeur depuis 5 ans
- 3 ans de Java Spring Boot en entreprise
- 2 ans de formation
- clement.hamon35@gmail.com

Horaire et convocations

- 9h30 - 17h30
- 15 mn de pause le matin
- 1h de pause déjeuner
- 15 mn de pause l’après midi

- QCM à la fin pour vous évaluer

Déroulé et structure de la formation et formalités

- Emargement le matin et l’après-midi
- Google Forms de demi-journée pour la validation des acquis et l'adaptabilité
- Notions théoriques suivis de mise en pratique

Tour de table et pré-requis

- Qui êtes-vous et quel est votre expérience dans l'informatique ?
- Vos attentes à l'issue de cette formation ?

- Bonne connaissance de Java

Structure et versionning Github

- Pour chaque notion montrée, un commit sera disponible sur ce git
- https://github.com/ClementHamonDev/Formation-Spring

- Support de cours sur Teams et Github

Objectifs de la formation

1. Mettre en œuvre le module Spring boot
2. Mettre en place une API et des échanges avec une base de données
3. Maîtriser la configuration et la sécurité

Installation des outils

- Installation Java 21
- IDE Visual Code Studio
- Plugins
- Postman

1. Spring

Pourquoi est-on passé de Java EE à Spring ?

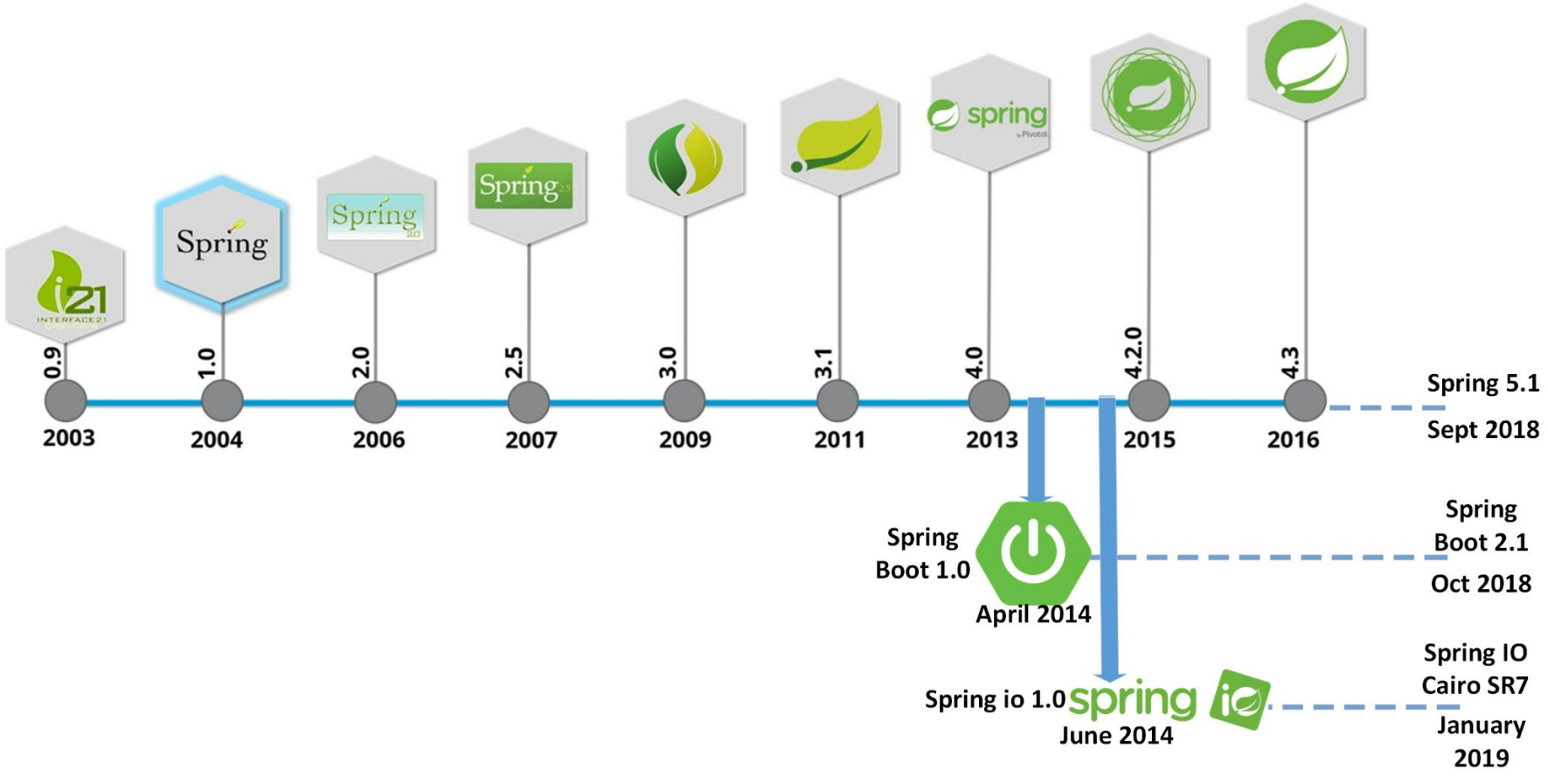
- **Légèreté et Flexibilité :** Spring a été conçu pour être un framework léger, contrairement à Java EE qui est plus lourd et rigide dans sa structure.

- **Simplicité de Configuration :** Java EE nécessitait une configuration complexe et fastidieuse. Spring, avec des annotations et une approche convention-over-configuration, a simplifié ce processus.

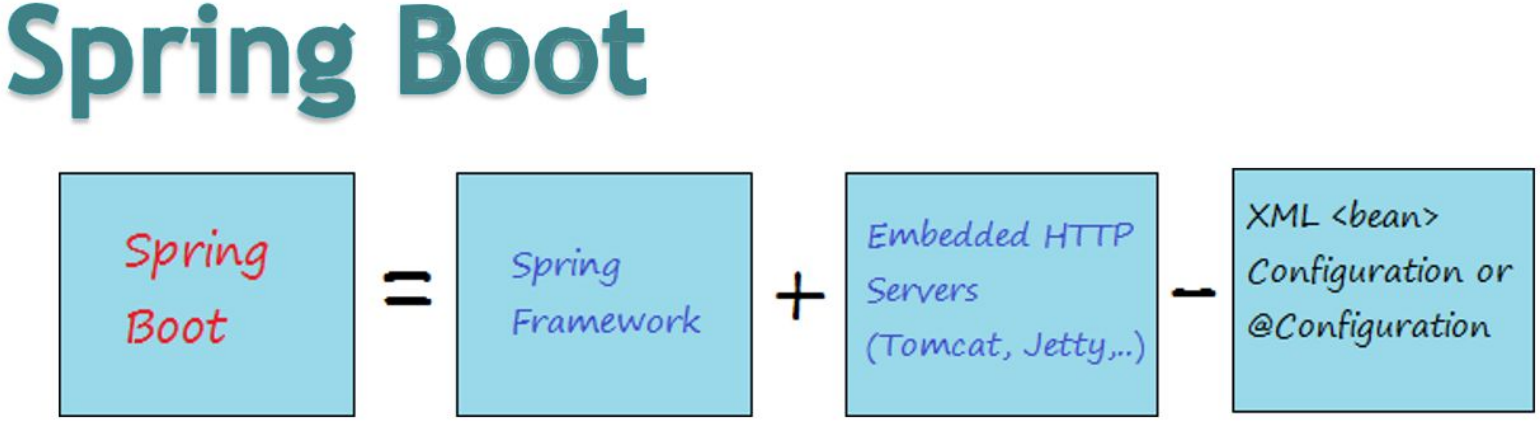
- **Gestion des Dépendances :** Avec l'injection de dépendances, Spring a apporté une meilleure modularité et maintenabilité du code.
- **Support Communautaire et Évolutivité :** Spring est soutenu par une grande communauté et propose des mises à jour régulières

- **Facilité d'Intégration**
- **Adapté aux Microservices :** Contrairement à Java EE, Spring (notamment avec Spring Boot) offre une architecture plus adaptée aux microservices et aux applications modernes cloud-native.

Historique de Spring



C’est quoi Spring Boot

- 

C’est quoi Spring Boot

Framework conçu essentiellement pour développer des architectures à base de micro-services

- L’idée est de découper une application en petites unités implémentées sous forme de micro-services
- Chaque micro-service est responsable d’une fonctionnalité élémentaire, développé, testé et déployé indépendamment des autres
- Chaque micro-service peut être conçu à l’aide de n’importe quel langage et technologie

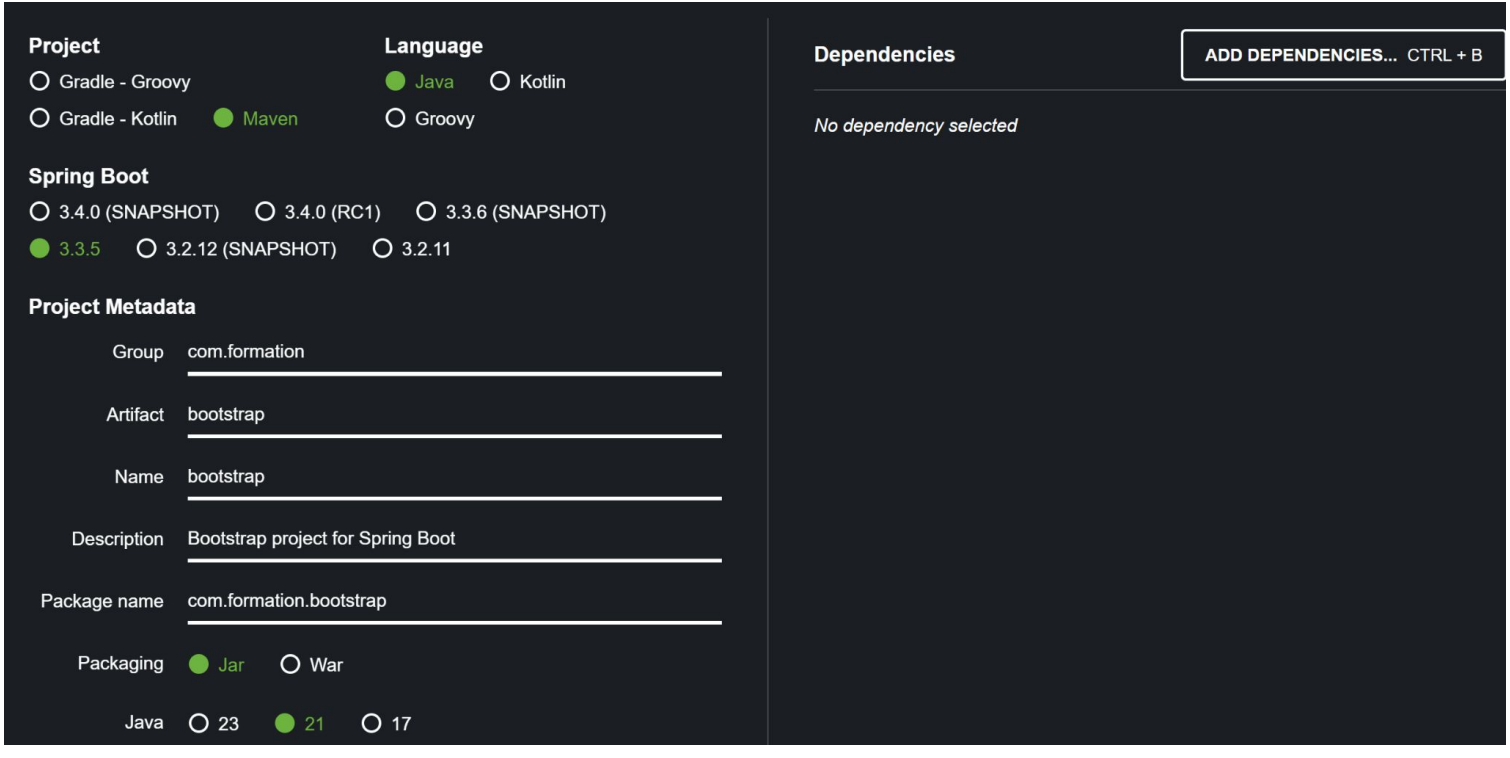
Avantages de Spring Boot

- Faciliter le développement d’applications complexes
- Faciliter l’injection des dépendances
- Faciliter la gestion des dépendances avec Maven
- Réduire les fichiers de configuration et supporter l’auto-configuration

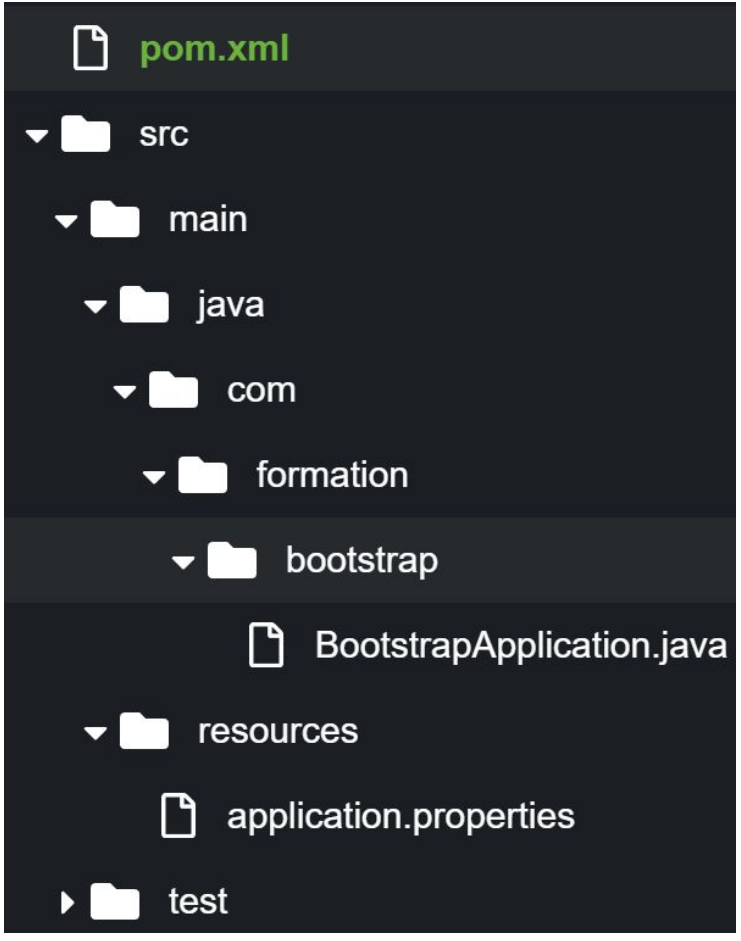
2. Introduction à Spring Boot

Spring Initializer

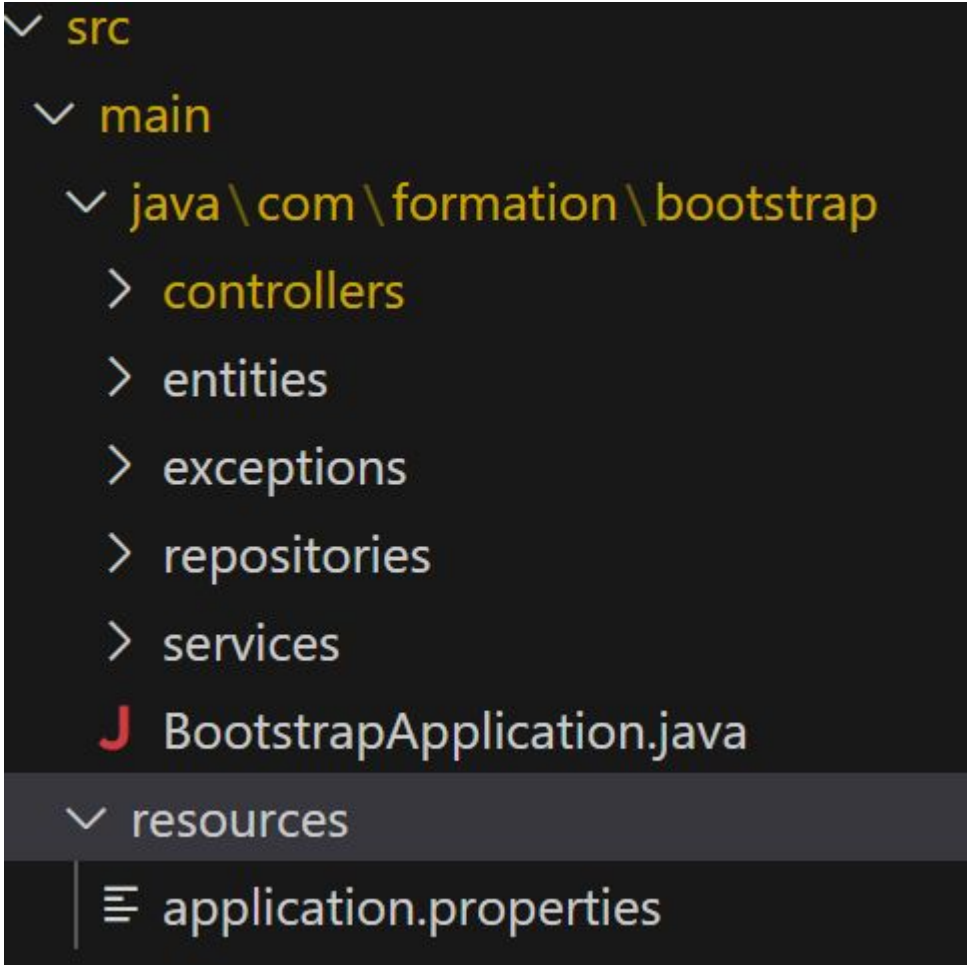
- **En ligne ou via plugin VS Code**



Différents fichiers



Différents fichiers



Dépendances

https://mvnrepository.com/

```xml
<dependencies>

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

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
    </dependency>
</dependencies>
```

Exercices

Création de l’architecture Spring et ajouts des dépendances Maven

Inversion of Controls (IoC)

Définition : Un **conteneur IoC** permet de gérer la création et l'injection des dépendances entre les différentes parties d'une application.

Principe de l'Inversion de Contrôle :

- Au lieu que chaque objet crée lui-même ses dépendances, le conteneur IoC prend en charge cette responsabilité.
- Les objets reçoivent (ou se voient "injecter") leurs dépendances par le conteneur, ce qui facilite la flexibilité et le découplage.

Avantages :

- **Découplage** : Les composants ne dépendent plus directement les uns des autres.
- **Testabilité** : Facilite les tests unitaires grâce à l'injection de dépendances simulées.
- **Maintenance** : Simplifie les mises à jour et les modifications de dépendances.

Le framework **Spring** utilise un conteneur IoC pour injecter des dépendances dans des services, des contrôleurs, et d'autres composants, avec des annotations comme `@Autowired`.

Inversion of Controls (IoC)

Exemple :

Les beans

Définition : Un **bean** est un objet géré par le conteneur IoC de Spring. Il est instancié, configuré et injecté là où il est nécessaire dans l'application.

Cycle de Vie d’un Bean :

1. **Instanciation**
2. **Injection des dépendances**
3. **Initialisation**
4. **Destruction**

`@Component`, `@Service`, `@Repository`, ou `@Controller` indiquent au conteneur que la classe doit être traitée comme un bean.

`@Autowired` est utilisée pour injecter des dépendances dans les beans.

Les beans - Scope

Par défaut, un Bean est défini comme **Singleton** (une instance est partagée pour tout l'app) mais il en existe d'autres :

- **Prototype** : Une instance est créée à chaque injection
- **Request**
- **Session**
- **Application**
- **Websocket**

```java
@RestController
@Scope("prototype")
@RequestMapping("/api/books")
public class BookController {

    @Autowired
    private BookRepository bookRepository;
```

3. CRUD

Création d’une route basique

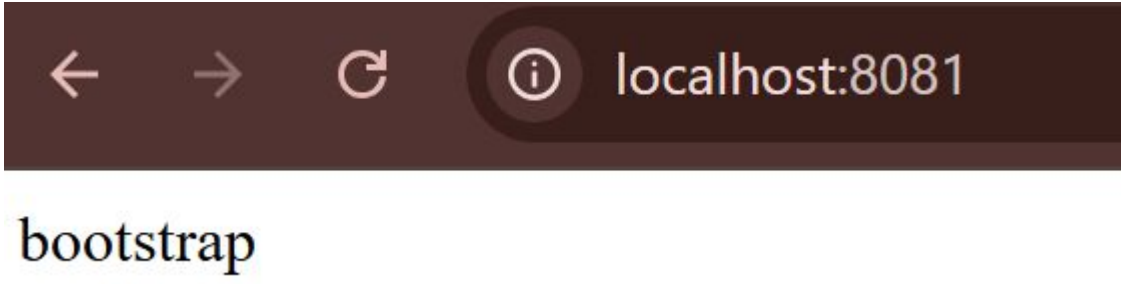
Annotations :

@RestController
@RequestMapping("/")
@GetMapping(“/”)

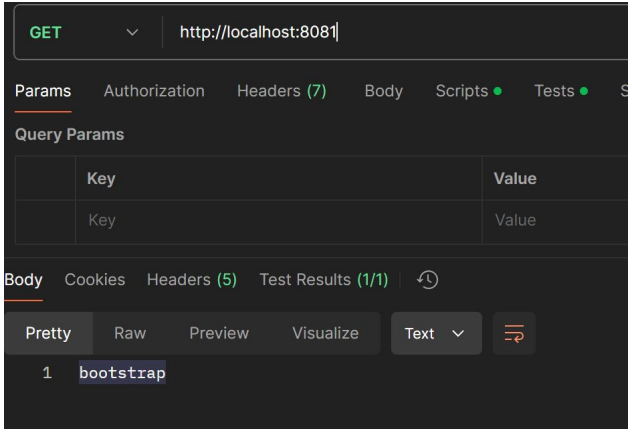
```java
@RestController
@RequestMapping("/")
class SimpleController {

    @Value("${spring.application.name}")
    String appName;

    @GetMapping("/")
    public String homePage() {
        return appName;
    }
}
```

Création d’une route basique

Dans une page web :



Dans Postman :



Création d’une CRUD Static

Create Retrieve Update Delete

- Création d’un tableau pour mettre nos données
- Création de routes pour créer, récupérer, mettre à jour et supprimer des données au sein d’un controller avec l’annotation @RestController

Exercice

- *Créer une route qui renvoie votre “Bonjour” + prénom*
- *Créer un tableau avec quelque prénoms de personnes de votre promo*
- *Créer le CRUD associé*

Bonus :

- *Créer un tableau avec des produits et un tableau panier vide*
- *Créer les routes associées avec un système de panier (pouvoir ajouter et supprimer un article)*

Création d’un CRUD

Create Retrieve Update Delete

Commençons par créer une entity Book

```
Entity
public class Book {

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

        @Column(nullable = false, unique = true)
        private String title;

        @Column(nullable = false)
        private String author;
```

Création d’un CRUD

Créons ensuite un repository

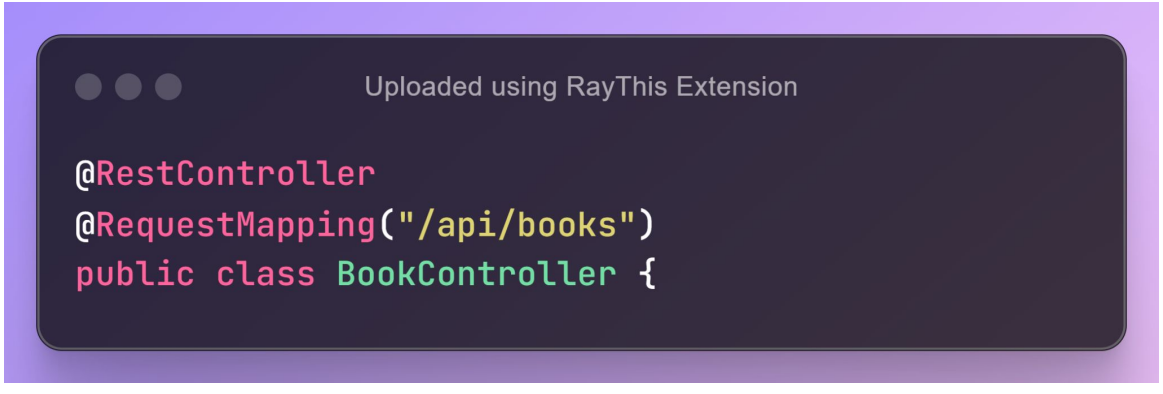
```java
@Repository
public interface BookRepository extends JpaRepository<Book, Long> {
}
```

Méthode de JPARepository

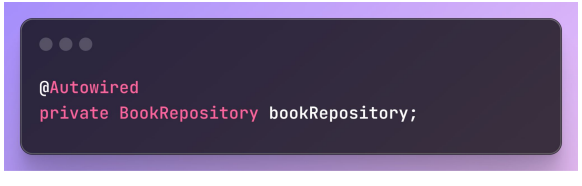
1. findBy<Attribut> => findByTitle
2. findBy<Attribut>GreaterThan
3. findBy<Attribut>GreaterThanEqual
4. findBy<Attribut>LessThan
5. findBy<Attribut>LessThanEqual
6. findBy<Attribut>Between
7. findBy<Attribut>Like
8. findBy<Attribut>NotLike
9. findBy<Attribut>StartingWith
10. findBy<Attribut>EndingWith
11. findBy<Attribut>IsNull
12. findBy<Attribut>IsNotNull
13. findBy<Attribut>And<AutreAttribut>
14. findBy<Attribut>Or<AutreAttribut>
15. countBy<Attribut>
16. findBy<Attribut>In
17. findBy<Attribut>NotIn
18. findBy<Attribut>OrderBy<Attribut>Asc
19. findBy<Attribut>OrderBy<Attribut>Desc
20. findBy<Relation><Attribut>

Création d’un CRUD

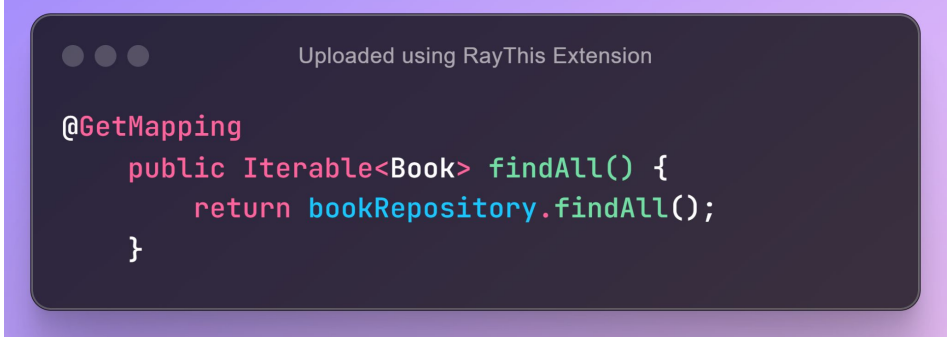
Création d’un controller



Appel au bean du Repository

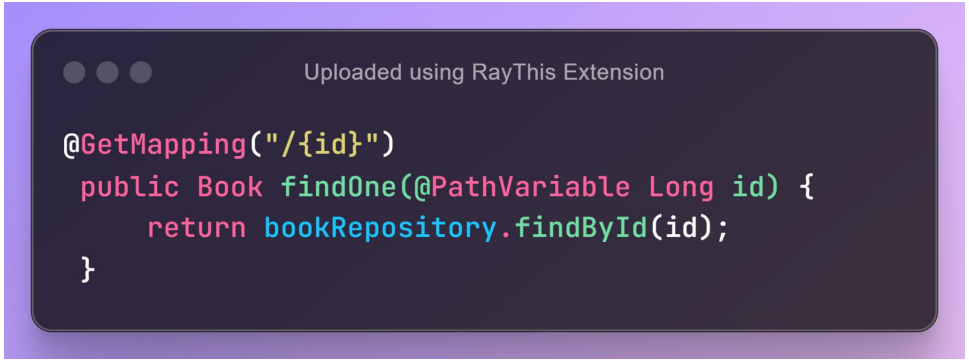


Création d’une route

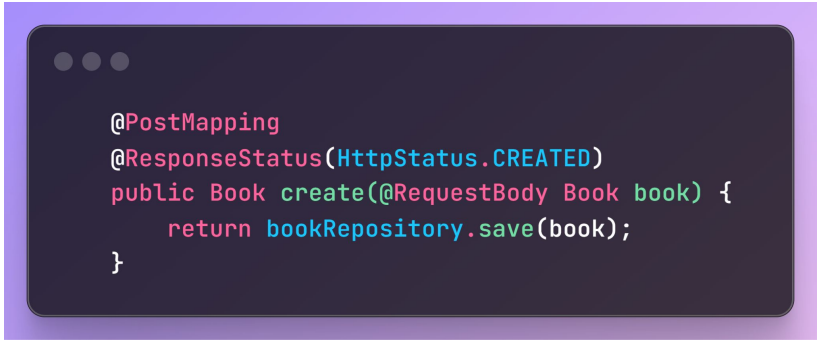


Création d’un CRUD

Création d’une route avec un paramètre :



Création d’une route avec un body :



Exercice

- *Créer un CRUD pour une entity représentant un produit quelconque d'un site ecommerce avec des colonnes id et nom*
- *Tester dans le navigateur*
- *Tester ces routes avec Postman*

Aller plus loin

Création de route custom dans le Repository et dans le Controller

```java
@Repository
public interface BookRepository extends JpaRepository<Book, Long> {
    List<Book> findByTitle(String title);
}
```

```java
@GetMapping("/title/{bookTitle}")
  public List<Book> findByTitle(@PathVariable String bookTitle) {
    return bookRepository.findByTitle(bookTitle);
  }
```

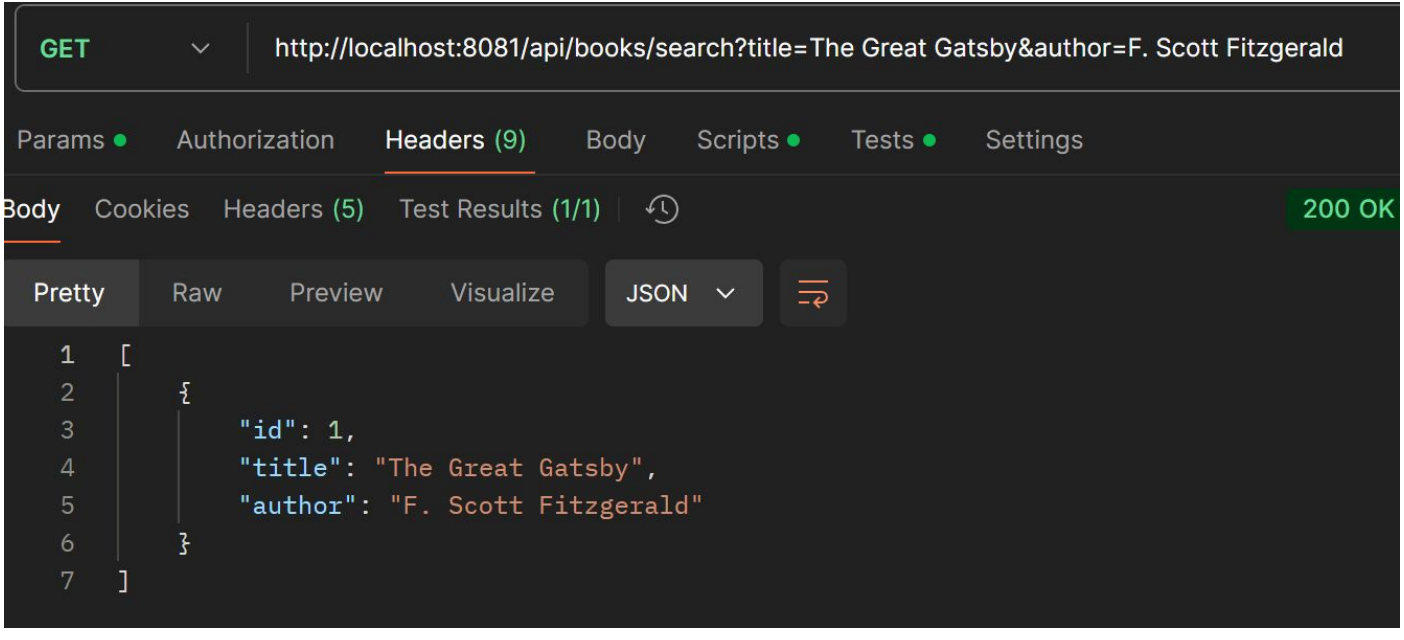
Aller plus loin

Création de route custom avec @PersitanceContext et EntityManager dans le Controller

```
@PersistenceContext
private EntityManager entityManager;
```

```
@GetMapping("/search")
  public List<Book> findByTitleAndAuthor(@RequestParam String title, @RequestParam String
author) {
    return entityManager.createQuery("SELECT b FROM Book b WHERE b.title = :title AND b.author
= :author", Book.class)
        .setParameter("title", title)
        .setParameter("author", author)
        .getResultList();
  }
```

Dans Postman



Exercice

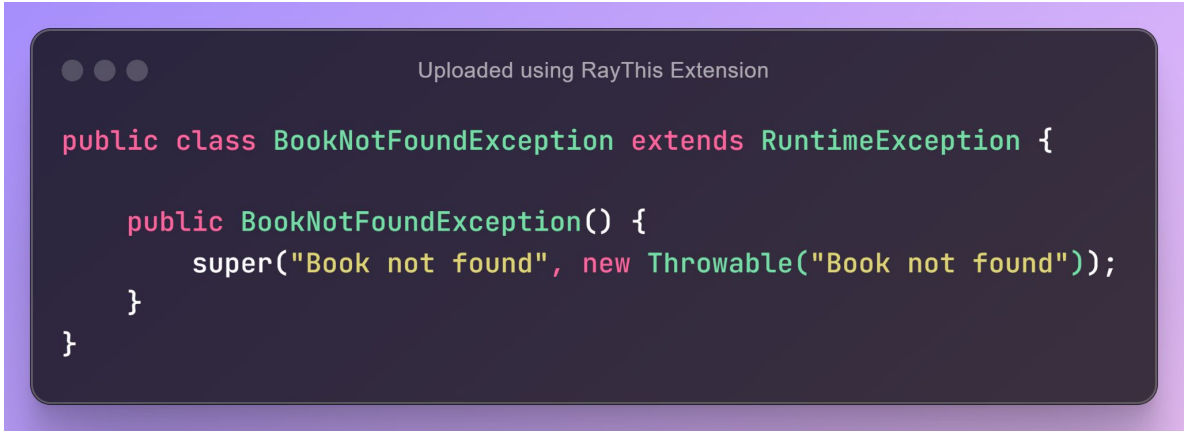
- *Créer une requête pour trouver un produit avec son nom et son prix (ajouter la colonne)*
- *Faites le avec JPA et avec @PersistanceContext*

- *Tester ces routes avec Postman*

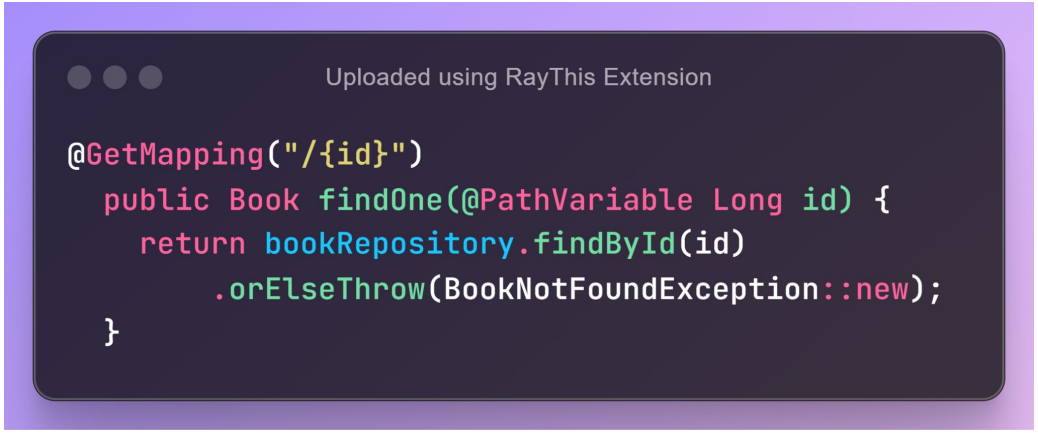
Bonus :

- *Créer une requête qui trouve tous les produits qui coûte plus de x euros avec x passé en paramètre*
- *Créer une requête qui trouve tous les produits ayant le même nom et qui coûte entre x et y euros avec le nom, x et y en paramètre*
- *Créer une requête qui renvoie tous les produits triés par ordre croissant selon leur prix*

Gérer les exceptions



Gérer les exceptions



Exercice

- *Créer une exception pour gérer le cas où il n'y a pas de Produit correspondant*
- *Créer une exception pour gérer le cas où le prix est inférieur à 0*

- *Tester ces exceptions avec Postman*

Relation OneToOne

Book.java

```java
// Relation One-to-One
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "detail_id", referencedColumnName = "id", nullable = false)
    private BookDetails bookDetail;
```

BookDetails.java


```java
@Entity
public class BookDetails {
    // Relation bidirectionnelle One-to-One
    @OneToOne(mappedBy = "bookDetail")
    private Book book;
```

Relation OneToOne

- `@OneToOne(cascade = CascadeType.ALL)` dans `Book` : crée une relation un-à-un avec `BookDetail`. Le `CascadeType.ALL` assure que les opérations effectuées sur `Book` sont également appliquées à `BookDetail`.

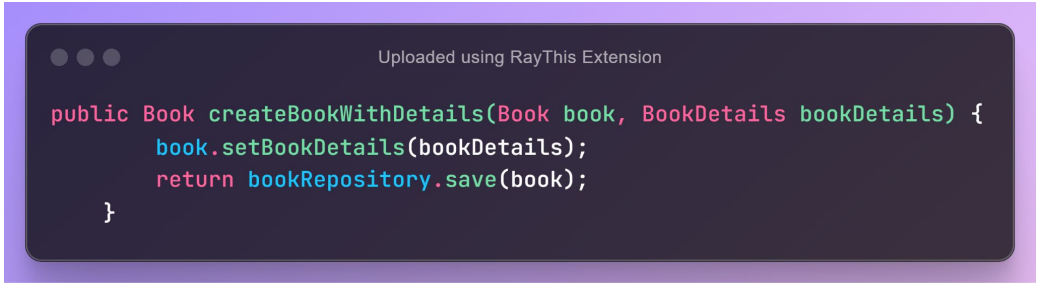
- `@JoinColumn(name = "detail_id", referencedColumnName = "id")` : spécifie que la colonne `detail_id` dans `Book` est la clé étrangère vers `BookDetail`.

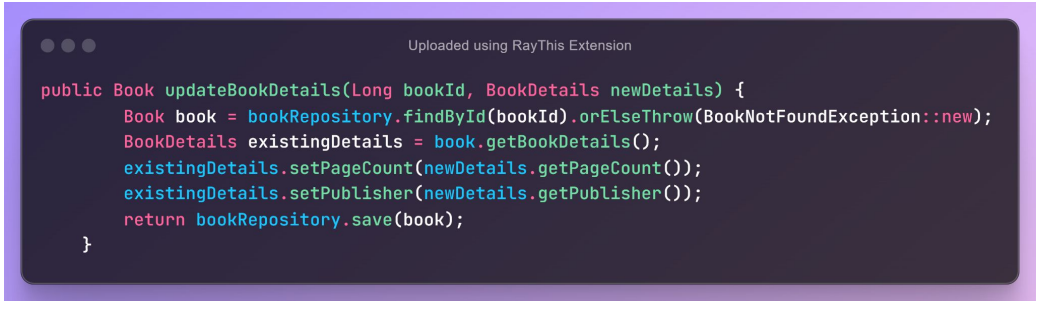
- `mappedBy = "bookDetail"` dans `BookDetail` : indique que `Book` possède la relation et que `BookDetail` est le côté inverse de la relation.

Relation OneToOne

Création d'un Repository pour BookDetails

Création d'un service pour Book avec une méthode pour créer un livre avec ses détails et une pour update





Exercice

- *Créer une relation OneToOne entre Produit et Vendeur avec les champs nom et prénom et les routes du CRUD*

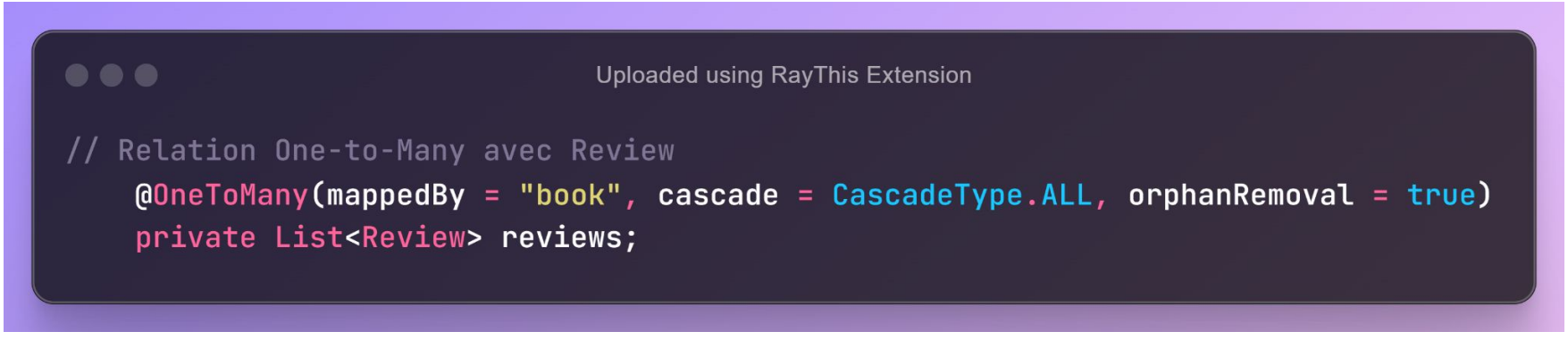
- *Tester les routes avec Postman*

Bonus :

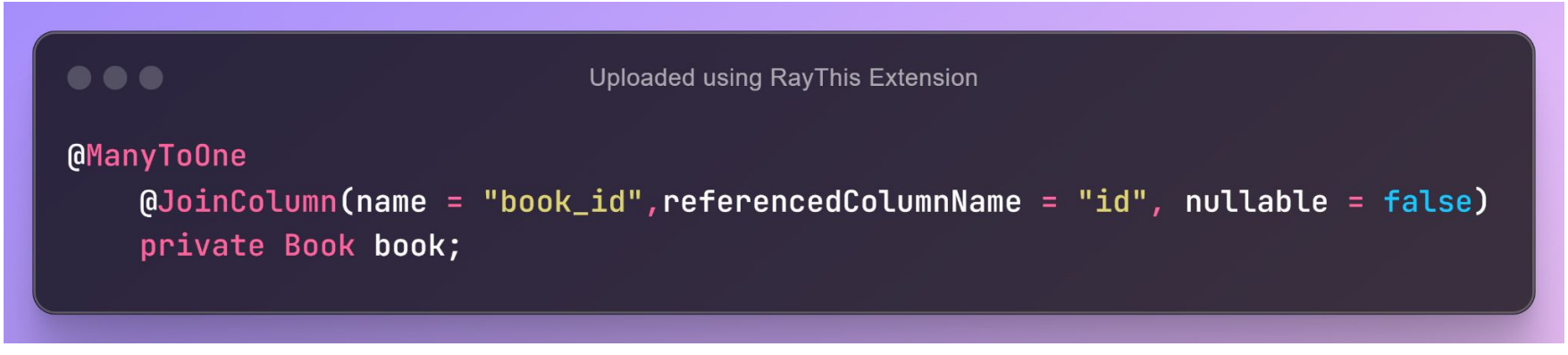
- *Rajouter des routes qui recherchent*
  - *tous les vendeurs qui ont le même prénom*
  - *selon le nom d'un produit, tous les vendeurs qui en vendent*

Relation OneToMany

Book.java



Review.java



Relation OneToMany

`@OneToMany(mappedBy = "book", cascade = CascadeType.ALL, orphanRemoval = true)` dans `Book` :

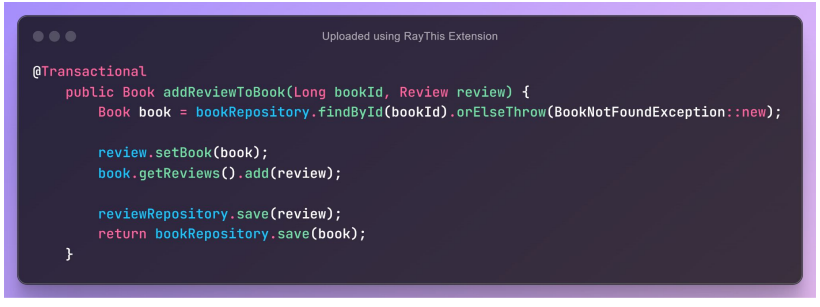
- `mappedBy = "book"` indique que la clé étrangère est définie dans l'entité `Review`.
- `cascade = CascadeType.ALL` signifie que toutes les opérations effectuées sur `Book` seront également appliquées aux `Review` associés.
- `orphanRemoval = true` permet de supprimer les `Review` orphelins (ceux qui ne sont plus associés à aucun `Book`).

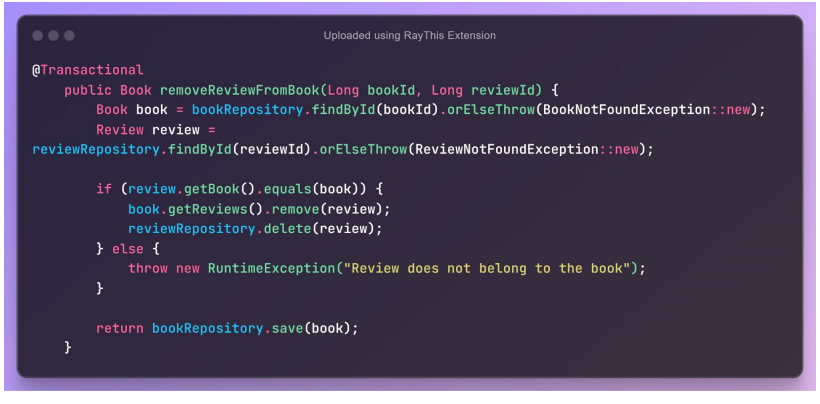
`@ManyToOne` dans `Review` : relie chaque `Review` à un `Book`. La colonne `book_id` est la clé étrangère dans `Review` qui référence l'entité `Book`.

Relation OneToMany

Création d'un Repository pour Review

Création d'un service pour Book avec une méthode pour associer et dissocier une review d'un livre





Exercice

- *Créer une relation OneToMany entre Produit et Promotion avec les champs quantité et les routes du CRUD*

- *Tester les routes avec Postman*

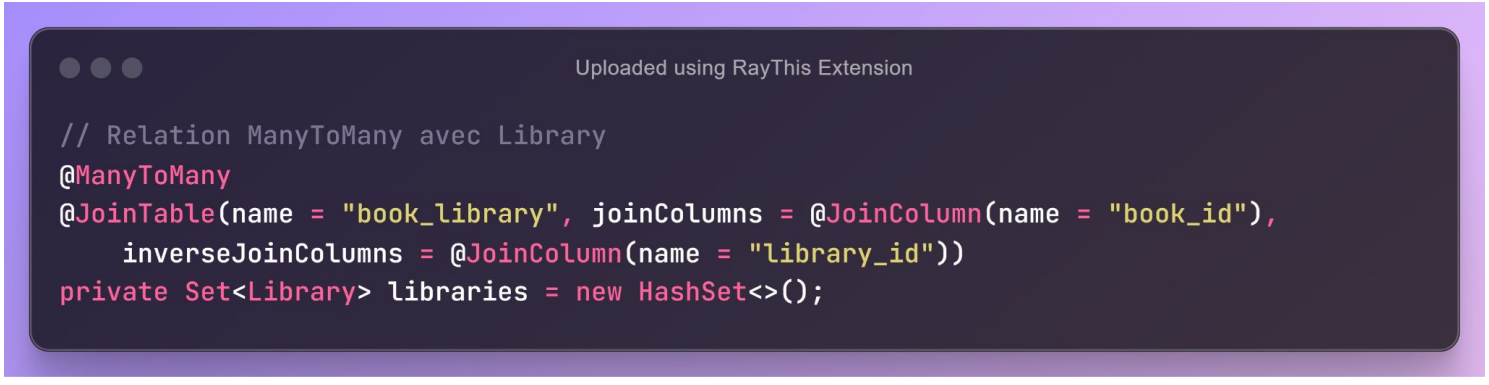
Bonus :

- *Rajouter des routes qui recherchent*
  - *Toutes les promos associées à un produit*
  - *toutes les promos de plus de -15%*
  - *Tous les vendeurs qui ont une promo sur un de leur produit*
  - *Tous les vendeurs qui ont une super promo ! (50% ou +)*

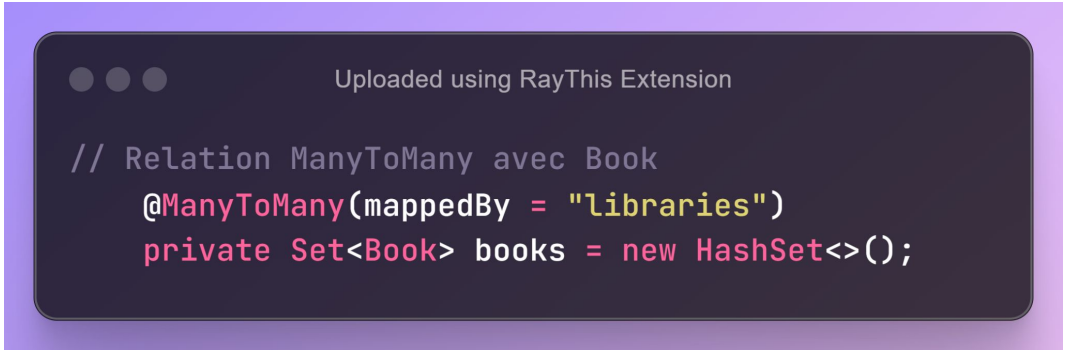
- *Créer une exception au cas où un promo à un pourcentage positif ou de plus de 100%*

Relation ManyToMany

Book.java



Librairy.java

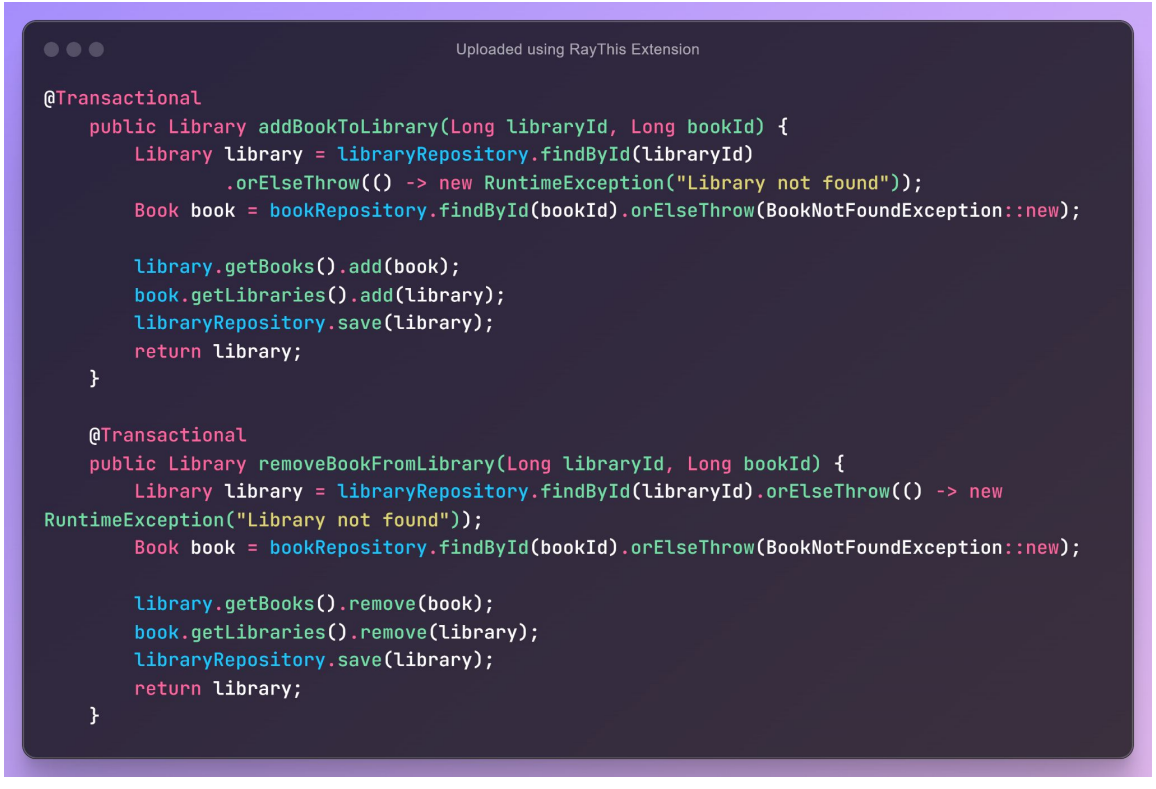


Relation ManyToMany

Relation ManyToMany

Création d'un Repository pour Library

Création d'un service et controller pour Library avec une méthode pour ajouter un livre et une méthode pour en supprimer un + CRUD classique



Exercice

- *Créer une relation ManyToMany entre Produit et Panier avec les champs prix total et nombre d'article et les routes du CRUD*

- *Tester les routes avec Postman*

Bonus :

- *Calculer automatiquement le prix total avec la somme de tous les prix*
- *Rajouter des routes qui recherchent*
  - *tous les paniers qui ont un prix total de plus de 50€ et le produit "PC"*
  - *tous les paniers qui ont un prix total de moins de 100€ après avoir appliquer les promotions*
- *Créer une exception quand le prix total est négatif*

4. Les événements et conditions

Questions / réponses

- Revenons sur les questions hors plan de cours que vous m’avez posé durant la formation pour y répondre

Votre formateur

Clément Hamon
clement.hamon35@gmail.com
https://www.linkedin.com/in/clément-hamon-135485209/

Merci d’avoir suivi cette formation !