Spring Boot

- Découverte
- Gestion de la persistance avec JPA et Hibernate
- Clean architecture
- TP Spring Boot

Découverte

https://devstory.net/11267/tutoriel-spring-boot-pour-debutant

https://bnguimgo.developpez.com/tutoriels/spring/services-rest-avec-springboot-et-spring-resttemplate/?page=premiere-partie-le-serveur

https://www.youtube.com/playlist?list=PLtyD11a_24egpX6V-BXtVnBmrbR60I1P6

Gestion de la persistance avec JPA et Hibernate

https://gayerie.dev/docs/spring/spring/spring_data_jpa.html

https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa

https://www.baeldung.com/spring-boot-hibernate

https://www.javadevjournal.com/spring-boot/spring-boot-with-hibernate/

Clean architecture

https://www.baeldung.com/spring-boot-clean-architecture

https://medium.com/swlh/clean-architecture-java-spring-fea51e26e00

https://reflectoring.io/java-components-clean-boundaries/

TP Spring Boot

Dans ce TP, vous allez développer un système de gestion pour une bibliothèque en utilisant Spring Boot et Spring Data JPA. Nous allons créer plusieurs classes, des interfaces pour les dépôts de données et des contrôleurs pour exposer nos services via une API REST.

Partie 1: Initialisation du projet Spring Boot

1. Allez sur Spring Initializr pour générer un nouveau projet Spring Boot. Choisissez "Maven Project", "Java", et la dernière version de Spring Boot.
2. Dans la section "Project Metadata", donnez un nom de groupe et d'artefact approprié, par exemple "com.example" et "library".
3. Dans la section "Dependencies", ajoutez les dépendances suivantes: Spring Web, Spring Data JPA, et Spring Boot DevTools.
4. Cliquez sur "Generate" pour télécharger un fichier zip du projet. Décompressez le fichier et ouvrez le projet dans votre IDE favori.

Partie 2: Création de la classe `Publisher`

1. Dans le package `model`, créez une classe `Publisher` qui représente un éditeur de livres. Cette classe doit avoir les attributs suivants :
   - `id` : l'ID de l'éditeur (Long)
   - `name` : le nom de l'éditeur (String)
   - `address` : l'adresse de l'éditeur (String)
2. Annoter la classe avec `@Entity` pour indiquer qu'il s'agit d'une entité JPA.
3. Annoter l'attribut `id` avec `@Id` et `@GeneratedValue` pour indiquer qu'il s'agit de la clé primaire et qu'elle est générée automatiquement.

Partie 3: Création de l'interface `PublisherRepository`

1. Dans le package `repository`, créez une interface `PublisherRepository` qui étend `JpaRepository<Publisher, Long>`. Cela nous donne gratuitement plusieurs méthodes pour interagir avec la base de données comme `findAll()`, `findById()`, `save()`, `delete()`, etc.

Partie 4: Création de la classe `Book`

1. Dans le package `model`, créez une classe `Book` qui représente un livre. Cette classe doit avoir les attributs suivants :
   - `id` : l'ID du livre (Long)
   - `title` : le titre du livre (String)
   - `author` : l'auteur du livre (String)
   - `publisher` : l'éditeur du livre (`Publisher`)
2. Annoter la classe avec `@Entity` pour indiquer qu'il s'agit d'une entité JPA.
3. Annoter l'attribut `id` avec `@Id` et `@GeneratedValue` pour indiquer qu'il s'agit de la clé primaire et qu'elle est générée automatiquement.
4. Annoter l'attribut `publisher` avec `@ManyToOne` pour indiquer la relation entre `Book` et `Publisher`.

Partie 5: Création de l'interface `BookRepository`

1. Dans le package `repository`, créez une interface `BookRepository` qui étend `JpaRepository<Book, Long>`.

Partie 6: Création des contrôleurs

1. Dans le package `controller`, créez une classe `PublisherController` avec une méthode `getAllPublishers()` qui retourne tous les éditeurs, et une méthode `addPublisher()` qui ajoute un nouvel éditeur.
2. Dans le même package, créez une classe `BookController` avec une méthode `getAllBooks()` qui retourne tous les livres, et une méthode `addBook()` qui ajoute un nouveau livre.

Partie 7: Test de l'application

1. Exécutez l'application et utilisez un outil comme Postman pour tester votre API REST.

Arborescence de fichiers

Voici une possible arboressence de fichiers pour ce TP :

```
src/
├── main/
│   ├── java/
│   │   ├── com.example.library
│   │   │   ├── LibraryApplication.java
│   │   │   ├── model/
│   │   │   │   ├── Book.java
```

```
│   │   │   │   ├── Publisher.java
│   │   │   ├── repository/
│   │   │   │   ├── BookRepository.java
│   │   │   │   ├── PublisherRepository.java
│   │   │   ├── controller/
│   │   │   │   ├── BookController.java
│   │   │   │   ├── PublisherController.java
│   ├── resources/
│   │   ├── application.properties
```

Code

Model

Book

```
package com.example.library.model;

import javax.persistence.*;

@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
    private String author;

    @ManyToOne
    private Publisher publisher;

    public Book() {
    }

    public Book(String title, String author, Publisher publisher) {
        this.title = title;
        this.author = author;
        this.publisher = publisher;
    }
```

```
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public Publisher getPublisher() {
        return publisher;
    }

    public void setPublisher(Publisher publisher) {
        this.publisher = publisher;
    }
}
```

Publisher

```
package com.example.library.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;

@Entity
public class Publisher {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String address;

    public Publisher() {
    }

    public Publisher(String name, String address) {
        this.name = name;
        this.address = address;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
```

```
    }
}
```

Repository

BookRepository

```
package com.example.library.repository;

import com.example.library.model.Book;
import org.springframework.data.jpa.repository.JpaRepository;

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

PublisherRepository

```
package com.example.library.repository;

import com.example.library.model.Publisher;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PublisherRepository extends JpaRepository<Publisher, Long> {
}
```

Controller

BookController

```
package com.example.library.controller;

import com.example.library.model.Book;
import com.example.library.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class BookController {
    @Autowired
```

```
    private BookRepository bookRepository;

    @GetMapping("/books")
    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    @PostMapping("/books")
    public Book addBook(@RequestBody Book book) {
        return bookRepository.save(book);
    }
}
```

PublisherController

```
package com.example.library.controller;

import com.example.library.model.Publisher;
import com.example.library.repository.PublisherRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class PublisherController {
    @Autowired
    private PublisherRepository publisherRepository;

    @GetMapping("/publishers")
    public List<Publisher> getAllPublishers() {
        return publisherRepository.findAll();
    }

    @PostMapping("/publishers")
    public Publisher addPublisher(@RequestBody Publisher publisher) {
        return publisherRepository.save(publisher);
    }
}
```