Documentation de l'API Spring Boot - Gestion des Utilisateurs

Turki Khaled

November 21, 2024

Contents

List of Figures

1 Introduction

Cette API Spring Boot permet de gérer les utilisateurs en enregistrant des informations telles que le nom d'utilisateur, le pays de résidence, la date de naissance, etc. Elle respecte certaines règles métiers, comme l'enregistrement limité aux résidents français majeurs.

1.1 Fonctionnalités principales

- Créer un utilisateur
- Afficher les détails d'un utilisateur
- Validation des entrées (seuls les résidents français majeurs sont autorisés à créer un compte)
- Enregistrement des appels API avec Spring AOP
- Base de données embarquée (H2)

2 Pré-requis

Avant de commencer, assurez-vous d'avoir les outils suivants installés :

- Java 17 ou une version plus récente
- Maven 3.8+
- Git
- Un client API comme Postman ou cURL

3 Installation et Configuration

3.1 Cloner le dépôt Git

```
https://github.com/TurkiKhaled/Test_Turki-Khaled.git
cd test
```

3.2 Compilation du projet

Pour compiler le projet, exécutez la commande suivante :

```
mvn clean install
```

3.3 Configuration du fichier application.properties

Dans le dossier `src/main/resources`, créez ou modifiez le fichier `application.properties` comme suit :

```
# Configuration de la base de donn es H2
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=test
spring.datasource.password=test
spring.jpa.hibernate.ddl-auto=update

# Affichage de la console H2
spring.h2.console.enabled=true
```

3.4 Lancer l'application

Une fois la configuration terminée, vous pouvez démarrer l'application en exécutant :

```
mvn spring-boot:run
```

L'application sera disponible à l'adresse suivante :

`http://localhost:8087/h2-console`

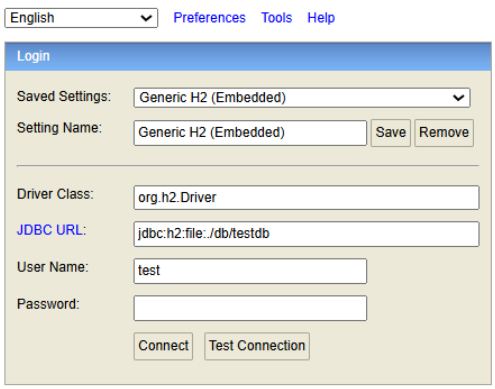

Figure 1: Console H2

4 Structure de l'API

4.1 1. Enregistrer un Utilisateur

Endpoint : `/api/users/register`
Méthode : `POST`
Description : Créer un nouvel utilisateur.
Headers :

- Content-Type: application/json

Body (JSON) :

```
{
    "username": "Khaled Turki",
    "country": "France",
    "dateOfBirth": "2000-05-15",
    "phoneNumber": "+33612345678",
    "gender": "M"
}
```

Réponse (201 Created) :

```
{
    "id": 1,
    "username": "Khaled Turki",
    "country": "France",
    "dateOfBirth": "2000-05-15",
    "phoneNumber": "+33612345678",
    "gender": "M"
}
```

4.2 2. Consulter un Utilisateur

Endpoint : `/api/users/id`
Méthode : `GET`
Description : Récupérer les détails d'un utilisateur.
Réponse (200 OK) :

```
{
    "id": 1,
    "username": "Khaled Turki",
    "country": "France",
    "dateOfBirth": "2000-05-15",
    "phoneNumber": "+33612345678",
    "gender": "M"
}
```

5 Gestion des Erreurs

Les erreurs sont renvoyées sous forme de codes HTTP appropriés avec des messages d'erreur détaillés.

- **400 Bad Request** : Si les données envoyées sont incorrectes (par exemple, un utilisateur mineur ou un pays incorrect).
- **404 Not Found** : Si l'utilisateur n'est pas trouvé.
- **500 Internal Server Error** : En cas d'erreur serveur.

6 Tests avec Postman

6.1 Ajouter Utilisateur

Importez la collection `Postman` fournie pour tester les endpoints de l'API. Voici un exemple de requête pour tester l'enregistrement d'un utilisateur :

- URL : `http://localhost:8087/users/register`
- Méthode : `POST`
- Body : Voir la section **1. Enregistrer un Utilisateur**.

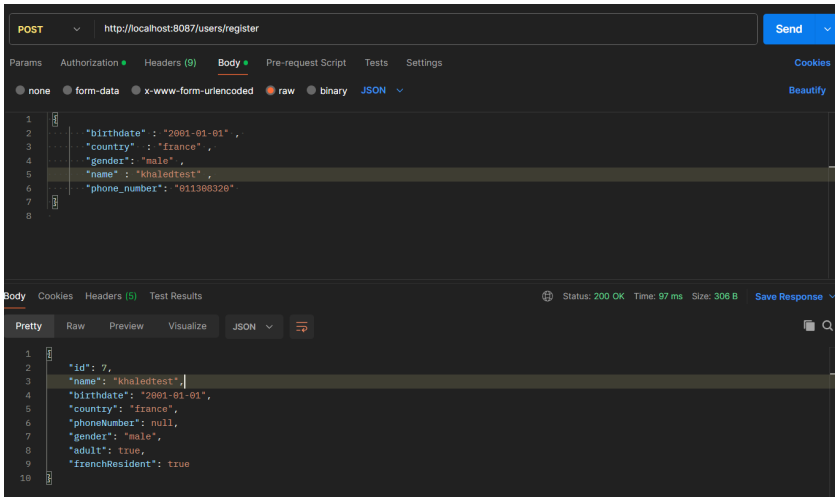

Figure 2: Ajouter Utilisateur via Postman

6.2 Récupérer Utilisateur

Une fois qu'un utilisateur a été créé, vous pouvez récupérer ses détails en utilisant son `id`. Cette opération est réalisée avec la méthode `GET` et l'ID de l'utilisateur comme paramètre dans l'URL.

Endpoint : `/api/users/id`
Méthode : `GET`
Description : Cette requête permet de récupérer les détails d'un utilisateur spécifique en utilisant son identifiant `id`. L'ID est récupéré à partir de la réponse de la création de l'utilisateur, comme montré dans la section précédente.

URL :

```
http://localhost:8087/users/{id}
```

Remplacez `id` par l'ID réel de l'utilisateur que vous souhaitez consulter, par exemple `1` si vous avez créé un utilisateur avec cet ID.

Réponse (200 OK) : Si l'utilisateur est trouvé, vous recevrez une réponse contenant les détails de l'utilisateur.

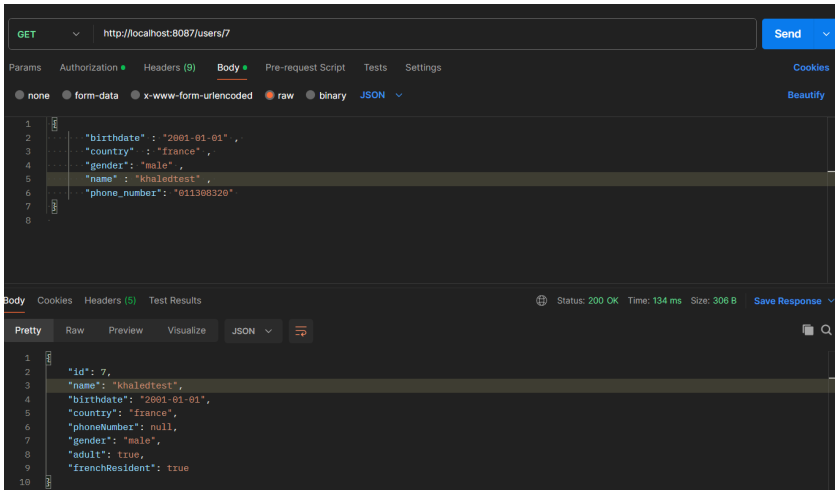

Figure 3: Récupération utilisateur par ID

7 Tests Unitaires (TU) et Tests d'Intégration (TI)

7.1 Tests Unitaires (TU)

Les tests unitaires permettent de tester chaque unité de manière isolée. Ils vérifient que chaque méthode ou service fonctionne correctement indépendamment des autres composants. Les tests unitaires sont particulièrement utilisés pour tester les logiques métier, les méthodes de services et les interactions avec des objets simulés (mocked).

7.2 Tests d'Intégration (TI)

Les tests d'intégration vérifient le bon fonctionnement de l'application dans son ensemble. Ces tests assurent que les différents composants, tels que les services, les contrôleurs, et la base de données, fonctionnent correctement ensemble. Les tests d'intégration sont utilisés pour tester les appels API réels et la communication avec la base de données.

8 Conclusion

Ce document fournit une vue d'ensemble détaillée de l'API Spring Boot pour la gestion des utilisateurs. Vous pouvez maintenant configurer, utiliser, et tester l'API à partir mon github