

Migrations de base de données avec Flyway dans Spring Boot

Voici un guide concis et pratique sur la manière d'utiliser Flyway avec une application Spring Boot pour gérer votre schéma de base de données :

Qu'est-ce que Flyway ?

Flyway est un outil de migration de base de données qui vous aide à contrôler les versions de votre schéma de base de données. Il applique des scripts SQL (migrations) dans un ordre spécifique pour faire évoluer votre base de données, ce qui en fait un outil idéal pour les projets Spring Boot où la cohérence et l'automatisation sont essentielles.

Étape 1 : Ajouter la dépendance Flyway

Pour commencer, ajoutez Flyway à votre projet. Si vous utilisez **Maven**, mettez à jour votre `pom.xml` :

```
<dependency>
  <groupId>org.flywaydb</groupId>
  <artifactId>flyway-core</artifactId>
</dependency>
```

Pour **Gradle**, ajoutez ceci à votre `build.gradle` :

```
implementation 'org.flywaydb:flyway-core'
```

Cela intègre Flyway dans votre application Spring Boot.

Étape 2 : Configurer la connexion à la base de données

Flyway doit savoir quelle base de données gérer. Définissez votre connexion à la base de données dans `application.properties` (ou `application.yml`). Par exemple, avec PostgreSQL :

```
spring.datasource.url=jdbc:postgresql://localhost:5432/mydb
spring.datasource.username=myuser
spring.datasource.password=mypassword
```

Spring Boot configure automatiquement Flyway pour utiliser cette source de données, donc aucune configuration supplémentaire n'est nécessaire ici.

Étape 3 : Créer des scripts de migration

Flyway utilise des scripts SQL versionnés pour mettre à jour votre base de données. Placez ces scripts dans le répertoire par défaut : `src/main/resources/db/migration`. Chaque script doit suivre ce modèle de nommage : `V<version>__<description>.sql`. Le numéro de version garantit que les migrations s'exécutent dans l'ordre.

Par exemple, créez un fichier nommé `V1__create_users_table.sql` :

```
CREATE TABLE users (  
    id SERIAL PRIMARY KEY,  
    username VARCHAR(255) NOT NULL,  
    email VARCHAR(255) NOT NULL  
);
```

Enregistrez-le dans `src/main/resources/db/migration`.

Étape 4 : Exécuter votre application

Démarrez votre application Spring Boot (par exemple, via `mvn spring-boot:run` ou votre IDE). Flyway : - Vérifiera la base de données pour un historique de migration. - Appliquera toute migration en attente (comme `V1__create_users_table.sql`) dans l'ordre.

Cela se fait automatiquement au démarrage grâce à l'intégration de Spring Boot—aucune intervention manuelle n'est requise.

Étape 5 : Personnaliser Flyway (Optionnel)

Vous pouvez ajuster le comportement de Flyway avec des propriétés dans `application.properties`. Voici quelques options courantes :

```
spring.flyway.locations=classpath:db/migration # Où trouver les scripts  
spring.flyway.baseline-on-migrate=true         # Gérer les bases de données existantes  
spring.flyway.enabled=true                     # Activer/Désactiver Flyway
```

Cela vous permet d'adapter Flyway aux besoins de votre projet.

Fonctionnalités avancées

Flyway offre plus que de simples migrations : - **Migrations basées sur Java** : Écrivez des migrations en Java pour une logique complexe. - **Migrations répétables** : Utilisez `R_<description>.sql` pour des scripts qui s'exécutent à nouveau lorsqu'ils sont modifiés (par exemple, des vues). - **Annuler les migrations** : Revenez en arrière sur les modifications (nécessite Flyway Pro).

Consultez la documentation Flyway pour une exploration plus approfondie.

Résumé

Pour utiliser Flyway dans Spring Boot : 1. Ajoutez la dépendance Flyway à votre projet. 2. Configurez votre connexion à la base de données dans `application.properties`. 3. Rédigez des scripts SQL versionnés dans `src/main/resources/db/migration`. 4. Lancez votre application—Flyway applique les migrations automatiquement.

Flyway simplifie la gestion des schémas de base de données avec le versionnement et l'automatisation, complétant parfaitement l'expérience de développement simplifiée de Spring Boot.