Database Migrations with Flyway in Spring Boot

Here's a concise, practical guide on how to use Flyway with a Spring Boot application to manage your database schema:

What is Flyway?

Flyway is a database migration tool that helps you version-control your database schema. It applies SQL scripts (migrations) in a specific order to evolve your database, making it ideal for Spring Boot projects where consistency and automation are key.

Step 1: Add Flyway Dependency

To get started, add Flyway to your project. If you're using **Maven**, update your `pom.xml`:

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

For **Gradle**, add this to your `build.gradle`:

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

This pulls Flyway into your Spring Boot application.

Step 2: Configure Your Database Connection

Flyway needs to know which database to manage. Define your database connection in `application.properties` (or `application.yml`). For example, with PostgreSQL:

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

Spring Boot automatically configures Flyway to use this data source, so no extra setup is needed here.

Step 3: Create Migration Scripts

Flyway uses versioned SQL scripts to update your database. Place these scripts in the default directory: `src/main/resources/db/migration`. Each script must follow this naming pattern: `V<version>__<description>.sql`. The version number ensures migrations run in order.

For example, create a file named `V1__create_users_table.sql`:

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

Save it in `src/main/resources/db/migration`.

Step 4: Run Your Application

Start your Spring Boot application (e.g., via `mvn spring-boot:run` or your IDE). Flyway will: - Check the database for a migration history. - Apply any pending migrations (like `V1__create_users_table.sql`) in sequence.

This happens automatically on startup thanks to Spring Boot's integration—no manual intervention required.

Step 5: Customize Flyway (Optional)

You can adjust Flyway's behavior with properties in `application.properties`. Here are some common options:

```
spring.flyway.locations=classpath:db/migration  # Where to find scripts
spring.flyway.baseline-on-migrate=true          # Handle existing databases
spring.flyway.enabled=true                      # Toggle Flyway on/off
```

These let you tailor Flyway to your project's needs.

Advanced Features

Flyway offers more than just basic migrations: - **Java-based migrations**: Write migrations in Java for complex logic. - **Repeatable migrations**: Use `R__<description>.sql` for scripts that rerun when changed (e.g., views). - **Undo migrations**: Roll back changes (requires Flyway Pro).

Check the Flyway documentation for deeper exploration.

Summary

To use Flyway in Spring Boot: 1. Add the Flyway dependency to your project. 2. Set up your database connection in `application.properties`. 3. Write versioned SQL scripts in `src/main/resources/db/migration`. 4. Launch your app—Flyway applies the migrations automatically.

Flyway simplifies database schema management with versioning and automation, perfectly complementing Spring Boot’s streamlined development experience.