Spring - Using SQL Scripts with Spring JDBC + JPA + HSQLDB

Last Updated : 30 Apr, 2025

In this article, we will be running the **SQL scripts with Spring JDBC +JPA + HSQLDB**. These scripts are used to perform **SQL** commands at the time of application start. **Java Database Connectivity (JDBC)** is an application programming interface (API) for the programming language Java, which defines how a client may access a database. **JPA (Java Persistence API)** is a tool that facilitates Object-Relational Mapping (ORM) to manage relational data in Java applications.

We will be using JDBC as a connector and JPA with **HSQLDB,** which is the leading SQL relational database system written in Java. It offers a small, fast multithreaded, and transactional database engine with in-memory and disk-based tables and supports embedded and server modes. It includes a powerful command-line SQL tool and simple GUI query tools. For writing SQL scripts, we need to change the **DataSource** configuration. **DataSource** is a name given to the connection set up to a database from a server.

In short, Spring JDBC allows Java applications to interact with databases using SQL commands.

Step-by-Step Implementation

Step 1: Add Dependencies

First, we will start by adding the required dependencies into the **pom.xml** file.

```
<dependencies>
    <!-- Spring Boot Starter Web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Spring Boot Data JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency> <!-- No version specified as Spring Boot manages it
automatically -->

    <!-- HSQLDB (In-memory database) -->
    <dependency>
        <groupId>org.hsqldb</groupId>
        <artifactId>hsqldb</artifactId>
        <version>2.7.1</version> <!-- Latest version -->
        <scope>runtime</scope>
    </dependency>

    <!-- Flyway for database migrations (optional) -->
    <dependency>
        <groupId>org.flywaydb</groupId>
        <artifactId>flyway-core</artifactId>
        <version>9.x.x</version> <!-- Latest Flyway version -->
    </dependency>
</dependencies>
```

Step 3: Configure Application Properties

We can configure the database settings, including data initialization, through **application.properties**. This configuration will point to the SQL scripts for database schema creation and data insertion.

```
# DataSource Configuration for HSQLDB in-memory database
spring.datasource.url=jdbc:hsqldb:mem:geeksforgeeks
spring.datasource.driverClassName=org.hsqldb.jdbc.JDBCDriver
spring.datasource.username=root
spring.datasource.password=pass


# Location of the SQL scripts for schema creation and data
population
```

```
spring.datasource.schema=classpath:schema.sql
spring.datasource.data=classpath:data.sql

# Hibernate configuration to avoid schema generation by Hibernate
spring.jpa.hibernate.ddl-auto=none  <!-- **This prevents Hibernate from overwriting the schema** -->
spring.jpa.show-sql=true

# Enable Flyway (optional)
spring.flyway.enabled=true  <!-- **Enabled Flyway migrations** -->
spring.flyway.locations=classpath:db/migration  <!-- **Location for Flyway migrations** -->
```

Step 4: Create the SQL Scripts

Now, lets create the **schema.sql** and **data.sql** files to initialize the database schema.

schema.sql (for creating tables):

```
-- Creating car table
CREATE TABLE car (
    id INT IDENTITY PRIMARY KEY,
    name VARCHAR(255),
    price INT
);

-- Creating book table
CREATE TABLE book (
    id INT IDENTITY PRIMARY KEY,
    name VARCHAR(255),
```

```
    price INT
);
```

data.sql (for inserting initial data):

```
-- Insert data into car table
INSERT INTO car (id, name, price) VALUES (DEFAULT, 'Audi', 3000000);
INSERT INTO car (id, name, price) VALUES (DEFAULT, 'BMW', 4000000);
INSERT INTO car (id, name, price) VALUES (DEFAULT, 'Jaguar', 3500000);

-- Insert repeated data into car table
INSERT INTO car (id, name, price) VALUES (DEFAULT, 'Audi', 3000000);
INSERT INTO car (id, name, price) VALUES (DEFAULT, 'BMW', 4000000);
INSERT INTO car (id, name, price) VALUES (DEFAULT, 'Jaguar', 3500000);

-- Insert data into book table
INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-1', 600);
INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-2', 500);
INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-3', 800);

-- Insert repeated data into book table
INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-1', 600);
```

```
INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-2', 500);
INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-3', 800);
```

Step 5: Enable Database Migration (Optional but Recommended)

For better schema management, we can use Flyway for versioned migrations. This make sure that the database schema changes are applied consistently and in a version-controlled manner.

Note: Create migration scripts under **src/main/resources/db/migration**

V1_create_schema.sql (for schema creation):

```
-- Creating car table
CREATE TABLE car (
    id INT IDENTITY PRIMARY KEY,
    name VARCHAR(255),
    price INT
);
-- Creating book table
CREATE TABLE book (
    id INT IDENTITY PRIMARY KEY,
    name VARCHAR(255),
    price INT
);
```

V2_insert_data.sql (for data insertion):

```
-- Insert data into car table
```

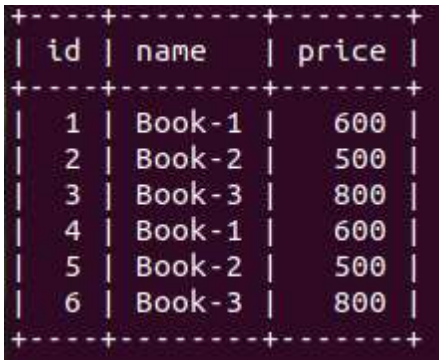
```
INSERT INTO car (id, name, price) VALUES (DEFAULT, 'Audi', 3000000);

INSERT INTO car (id, name, price) VALUES (DEFAULT, 'BMW', 4000000);

INSERT INTO car (id, name, price) VALUES (DEFAULT, 'Jaguar', 3500000);


-- Insert repeated data into car table

INSERT INTO car (id, name, price) VALUES (DEFAULT, 'Audi', 3000000);

INSERT INTO car (id, name, price) VALUES (DEFAULT, 'BMW', 4000000);

INSERT INTO car (id, name, price) VALUES (DEFAULT, 'Jaguar', 3500000);


-- Insert data into book table

INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-1', 600);

INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-2', 500);

INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-3', 800);


-- Insert repeated data into book table

INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-1', 600);

INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-2', 500);

INSERT INTO book (id, name, price) VALUES (DEFAULT, 'Book-3', 800);
```

Step 6: Run the Application

When we run the application, it will automatically execute the SQL scripts at startup. The schema will be created and populated with data from the schema.sql and data.sql files.

Output:

1. Book Table:



id	name	price
1	Book-1	600
2	Book-2	500
3	Book-3	800
4	Book-1	600
5	Book-2	500
6	Book-3	800

2. Car Table:

id	name	price
1	Audi	3000000
2	BMW	4000000
3	Jaguar	3500000
4	Audi	3000000
5	BMW	4000000
6	Jaguar	3500000