УДК 004

Загрузка исходных данных с помощью Spring

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описывается процесс загрузки исходных данных с помощью Spring. Будут созданы классы создания таблиц в базе данных. Практическим результатом являются рассмотренные примеры загрузки данных.
Ключевые слова: Java, JPA, Spring, Spring boot, Spring Data

Loading initial data with Spring

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of loading raw data using Spring. The classes for creating tables in the database will be created. The practical result is the considered examples of data loading.
Keywords: Java, JPA, Spring, Spring boot, Spring Data

Spring Data - предоставляет знакомую и последовательную модель программирования на основе Spring для доступа к данным, сохраняя при этом особые черты базового хранилища данных.

Это упрощает использование технологий доступа к данным, реляционных и не реляционных баз данных, фреймворков с сокращением карт и облачных сервисов данных. Это зонтичный проект, который содержит множество подпроектов, специфичных для данной базы данных. Проекты разрабатываются в сотрудничестве со многими компаниями и разработчиками, стоящими за этими захватывающими технологиями.

С помощью Spring Data очень легко разработать простое приложение CRUD по созданию собственных DAO. Spring Boot упрощает задачу, так что даже не нужно создавать файл «EntityManager» самостоятельно.

Прохоров П.В., Разговоров Н.В. рассмотрели в своей работе современные подходы и технологии в разработке серверных приложений на примере онлайн-магазина с использованием Spring [1]. В своей статье Жданова В.С. описала подход в реализации серверной части клиент-серверного мобильного приложения для просмотра расписания [2]. Так же Нарижный А.Д., Губенко Н.Е. провели сравнительный анализ технологий, которые имеют схожую функциональность и которые предназначены для одних и тех же задач. Это технологии стеков Spring и JavaEE, которые предназначены для разработки Enterprise-приложений [3]. Волушкова В.Л. рассмотрела в своей работе технологии программирования на примере языка Java [4]. В своей статье Егунова А.И., Аббакумов А.А., Воропаева М.А., Вечканова Ю.С. рассматривают проблемы повышения эффективности образовательной деятельности вуза и использование научной интеллектуальной собственности в преподавательской деятельности. Так же предложили реализацию информационного хранилища и поисковой системы в виде J2EE-приложения с использованием фреймворка Spring [5].

Одна из приятных особенностей SpringBoot заключается в том, что она автоматически выбирает любые файлы schema.sql и data.sql в пути к классам, если используется встроенный источник данных.

Допустим, у есть следующая сущность(рис.1)

```
@Data
@Builder
@NoArgConstructor
@AllArgConstructor
@Entity
@Table(name = "marvel_character")
public class MarvelCharacter {
    @Id
    @Column(name = "hero_name")
    private String heroName;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
    private String lastName;
}
```

Рисунок 1 – Имеющаяся сущность

В этом случае можно настроить схему, создав файл schema.sql следующим образом (Рис.2)

```
create table marvel_character (
    name        varchar(100)    not null,
    first_name  varchar(100)    not null,
    last_name   varchar(100)    not null,
    primary key (name)
);
```

Рисунок 2 – Создание схемы

Кроме того, создадим файл data.sql следующим образом (рис.3)

```
insert into marvel_character (hero_name, first_name, last_name) values ('Iron man', 'Tony', 'Stark');
insert into marvel_character (hero_name, first_name, last_name) values ('Thor', 'Thor', 'Odinson');
insert into marvel_character (hero_name, first_name, last_name) values ('Black widow', 'Natasha', 'Romanova');
insert into marvel_character (hero_name, first_name, last_name) values ('Hawkeye', 'Clint', 'Barton');
insert into marvel_character (hero_name, first_name, last_name) values ('Spider-man', 'Peter', 'Parker');
insert into marvel_character (hero_name, first_name, last_name) values ('Captain America', 'Steve', 'Rogers');
insert into marvel_character (hero_name, first_name, last_name) values ('Hulk', 'Bruce', 'Banner');
insert into marvel_character (hero_name, first_name, last_name) values ('Ant-man', 'Scott', 'Lang');
```

Рисунок 3 – Добавление данных

Однако есть несколько предостережений. Прежде всего, если используется JPA и встроенный источник данных, то schema.sql не вступит в силу, потому что генерация DDL Hibernate имеет приоритет над ним.

Для решения этой проблемы - нужно установить следующее свойство (рис.4).

```
spring.jpa.hibernate.ddl-auto=none
```

Рисунок 4 – Свойство для Hibernate

Кроме того, начиная с SpringBoot2, схема инициализируется по умолчанию только для встроенных источников данных. Чтобы разрешить загрузку данных для всех типов источников данных,нужно установить следующее свойство (Рис.5).

```
spring.datasource.initialization-mode=always
```

Рисунок 5 – Свойство загрузки

Если необходимо использовать несколько источников данных, например, базу данных H2 в памяти для разработки и базу данных MySQL для производства, то можно назвать файлы, например, schema-h2.sql и data-h2.sql.

Однако это будет работать, только если установить следующее свойство (рис.6).

```
spring.datasource.platform=h2
```

Рисунок 6 – Свойство для h2

И последнее, но не менее важное: если хочется изменить имена файлов по-другому или выполнить несколько сценариев SQL, то можно использовать «spring.datasource.data» (Рис.7).

```
spring.datasource.data=classpath:script1.sql, classpath:script2.sql
```

Рисунок 7 – Свойство сценариев

В Hibernate также можно автоматически сгенерировать DDL на основе сущностей, установив свойство «spring.jpa.hibernate.ddl-auto=create-drop».

После этого Hibernate выберет файл с именем import.sql в пути к классам, в который можно добавить свои операторы вставки (рис.8).

```
insert into marvel_character (hero_name, first_name, last_name) values ('Iron man', 'Tony', 'Stark');
insert into marvel_character (hero_name, first_name, last_name) values ('Thor', 'Thor', 'Odinson');
insert into marvel_character (hero_name, first_name, last_name) values ('Black widow', 'Natasha', 'Romanova');
insert into marvel_character (hero_name, first_name, last_name) values ('Hawkeye', 'Clint', 'Barton');
insert into marvel_character (hero_name, first_name, last_name) values ('Spider-man', 'Peter', 'Parker');
insert into marvel_character (hero_name, first_name, last_name) values ('Captain America', 'Steve', 'Rogers');
insert into marvel_character (hero_name, first_name, last_name) values ('Hulk', 'Bruce', 'Banner');
insert into marvel_character (hero_name, first_name, last_name) values ('Ant-man', 'Scott', 'Lang');
```

Рисунок 8 – Вставка данных

В предыдущих решениях для вставки данных в базу данных использовались операторы SQL, также можно использовать репозиторий данных Spring, если он был создан.

Чтобы иметь возможность вставлять данные в базу данных при запуске приложения, все, что нужно сделать, это создать bean-компонент типа «ApplicationRunner» или «CommandLineRunner» (рис.9).

```
@Bean
public ApplicationRunner initializer(MarvelCharacterRepository repository) {
    return args -> repository.saveAll(Arrays.asList(
        MarvelCharacter.builder().heroName("Iron man").firstName("Tony").lastName("Stark").build(),
        MarvelCharacter.builder().heroName("Thor").firstName("Thor").lastName("Odinson").build(),
        MarvelCharacter.builder().heroName("Black widow").firstName("Natasha").lastName("Romanova").build(),
        MarvelCharacter.builder().heroName("Hawkeye").firstName("Clint").lastName("Barton").build(),
        MarvelCharacter.builder().heroName("Spider-man").firstName("Peter").lastName("Parker").build(),
        MarvelCharacter.builder().heroName("Captain America").firstName("Steve").lastName("Rogers").build(),
        MarvelCharacter.builder().heroName("Hulk").firstName("Bruce").lastName("Banner").build(),
        MarvelCharacter.builder().heroName("Ant-man").firstName("Scott").lastName("Lang").build()
    ));
}
```

Рисунок 9 – Создание компонента

Преимущество этого подхода в том, что можно делать это программно. Обратной стороной, однако, является то, что если допустить ошибку в

отображении сущностей, то можно не заметить ее сразу, потому что сущности могут по-прежнему работать нормально.

Вместо того, чтобы полагаться на «DataSource» то, что предоставляется авто конфигурацией загрузки Spring, можно создать собственный «DataSource». Используя «EmbeddedDataSourceBuilder», можно легко добавлять скрипты, которые необходимо выполнить (рис.10).

```
@Bean
public DataSource dataSource() {
    return new EmbeddedDatabaseBuilder()
        .setName("marvelDB")
        .setType(EmbeddedDatabaseType.H2)
        .addScript("classpath:script1.sql")
        .addScript("classpath:script2.sql")
        // ...
        .build();
}
```

Рисунок 10 – Создание DataSource

Необходимо иметь в виду то, что этот подход работает только для встроенных источников данных, у обычных «DataSourceBuilder» нет удобного способа добавления сценариев SQL для загрузки при запуске.

Есть множество способов настроить исходные данные. Для каждого конкретного требования есть решение. Независимо от того, выбирается полноценная среда миграции базы данных или простой файл SQL.

В данной статье были рассмотрены примеры загрузки исходных данных с использованием SpringBoot.

Библиографический список

1. Прохоров П.В., Разговоров Н.В. Современные подходы в backend разработке на примере онлайн-магазина // Прикладная математика и фундаментальная информатика. 2020. №2. С. 23-28.
2. Жданова В.С. Серверный модуль системы уведомления об изменениях в расписании занятий // Молодость. Интеллект. Инициатива. 2017. С. 21-22.
3. Нарижный А.Д., Губенко Н.Е. Сравнительный анализ стеков технологий spring и javaee (jakartaee) для разработки enterprise приложений // Информатика, управляющие системы, математическое и компьютерное моделирование 2020. №3. С. 549-462.
4. Волушкова В.Л. Архитектурные решения java для доступа к данным // Теоретические основы программирования. Учебное пособие 2019. 137с.
5. Егунова А.И., Аббакумов А.А., Воропаева М.А., Вечканова Ю.С. Система управления хранилищем электронных образовательных ресурсов // Образовательные технологии и общество. 2019. №3. С. 145-154.