Spring Data JDBC

Spring Data JDBC 是一个强大的库，通过提供一种轻量级、基于注解的方法来简化 Spring 应用程序中的数据库交互。它比 Spring Data JPA 更少意见，专注于一种更简单、更明确的将对象映射到数据库表的方式。下面，我将带你了解如何使用它的基础知识。

1. 设置您的项目

要开始，您需要在项目中包含 Spring Data JDBC。如果您使用的是 Maven，请将以下依赖项添加到您的 `pom.xml`：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jdbc</artifactId>
</dependency>
```

您还需要一个 JDBC 驱动程序（例如 H2、MySQL、PostgreSQL）。例如，对于 H2：

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```

如果您使用的是 Gradle，等效的依赖项将是：

```
implementation 'org.springframework.boot:spring-boot-starter-data-jdbc'
runtimeOnly 'com.h2database:h2'
```

2. 配置您的数据库

在 `application.properties` 或 `application.yml` 中配置数据库连接。对于 H2 内存数据库，它可能看起来像这样：

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=true
```

对于实际数据库（如 PostgreSQL），请相应地调整 URL、用户名和密码。

3. 定义您的领域模型

创建一个简单的实体类来表示数据库中的表。Spring Data JDBC 使用约定，类名映射到表名（默认情况下为小写），字段映射到列。

```
import org.springframework.data.annotation.Id;

public class Person {
    @Id
    private Long id;
    private String firstName;
    private String lastName;

    // 默认构造函数（Spring Data JDBC 所需）
    public Person() {}

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    // Getters 和 Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getFirstName() { return firstName; }
    public void setFirstName(String firstName) { this.firstName = firstName; }
    public String getLastName() { return lastName; }
    public void setLastName(String lastName) { this.lastName = lastName; }
}
```

- `@Id` 标记主键。
- Spring Data JDBC 期望一个无参构造函数。
- 表将命名为 `person`，除非被覆盖。

4. 创建一个仓库

定义一个扩展 `CrudRepository` 的接口来处理基本的 CRUD 操作：

```
import org.springframework.data.repository.CrudRepository;

public interface PersonRepository extends CrudRepository<Person, Long> {
}
```

就是这样！您不需要实现它——Spring Data JDBC 会在运行时生成实现。

5. 使用仓库

将仓库注入到服务或控制器中并使用它：

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class PersonService {
    private final PersonRepository repository;

    @Autowired
    public PersonService(PersonRepository repository) {
        this.repository = repository;
    }

    public void savePerson() {
        Person person = new Person("John", "Doe");
        repository.save(person);
        System.out.println("Saved person with ID: " + person.getId());
    }

    public void listPeople() {
        Iterable<Person> people = repository.findAll();
        people.forEach(p -> System.out.println(p.getFirstName() + " " + p.getLastName()));
    }
}
```

6. 运行您的应用程序

如果您使用的是 Spring Boot，创建一个带有 `@SpringBootApplication` 的主类并运行它：

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
```

```
        SpringApplication.run(Application.class, args);
    }
}
```

Spring Boot 将自动配置数据源和 Spring Data JDBC。

7. 可选：自定义查询

如果您需要自定义查询，可以在仓库中添加带有 `@Query` 的方法：

```
import org.springframework.data.jdbc.repository.query.Query;
import org.springframework.data.repository.CrudRepository;

public interface PersonRepository extends CrudRepository<Person, Long> {
    @Query("SELECT * FROM person WHERE first_name = :name")
    List<Person> findByFirstName(String name);
}
```

需要记住的关键点

- **无延迟加载：**与 JPA 不同，Spring Data JDBC 不支持延迟加载。关系（例如一对多）需要显式建模，使用聚合根。
- **聚合根：**每个实体都是聚合的一部分，只有根才能直接保存或检索。
- **模式创建：** Spring Data JDBC 不会自动创建表。您需要提供模式（例如，通过 `src/main/resources` 中的 `schema.sql`）：

```
CREATE TABLE person (
    id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255)
);
```

示例工作流程

1. 启动您的应用程序。
2. 使用 `repository.save()` 保存一个 `Person` 对象。
3. 使用 `repository.findById()` 或 `findAll()` 检索它。

这就是基础知识！如果您想深入了解关系、自定义映射或某些特定内容，请告诉我您的目标，我将为您量身定制解释。到目前为止，这对您有帮助吗？