1 Spring 数据库管理

`JdbcTemplate` 的五种主要方法:

- `execute` 方法:可以用于执行任何 `SQL` 语句,一般用于执行 `DDL` 语句;

```
String sql = "";
Object param = {}; // sql 语句中变量
jdbcTemplate.update(sql, param);
```

- `query` 方法及 `queryForXXX` 方法:用于执行查询相关语句;
- `BeanPropertyRowMapper` 要求 `sql` 数据查询出来的列和实体属性一一对应，不一致，则在sql 语句中使用 as 取别名

```
String sql = "select from User";
RowMapper<User> rowMapper = new BeanPropertyRowMapper<User>(User.class);
User list = jdbcTemplate.queryForObject(sql, rowMapper, null);
```

- `update` 方法及 `batchUpdate` 方法: `update` 方法用于执行新增、修改、删除等语句; `batchUpdate` 方法用于执行批处理相关语句;

```
String sql = "insert into xxx values ()";
List <Object[]> batchArgs = new ArrayList<Object[]> ();
batchArgs.add(new Object[]{"..", ".."});
batchArgs.add(new Object[]{"..", ".."})
jdbcTemplate.batchUpdate(sql, batchArgs);
```

- `call` 方法:用于执行存储过程、函数相关语句。

2 JdbcTemplate 数据库操作

1. 添加 `Spring jdbcTemplate` 依赖的 jar 包
    - Mysql 驱动
    - C3p0 连接池
    - `Spring` 事务
2. 配置数据源和 JdbcTemplate
    1. 注入 `DataSource`，配置数据源

```
 <bean id = "ds" class = "com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name = "driverClass" value = "com.mysql.cj.jdbc.Driver"/>
        <property name = "jdbcUrl" value = "jdbc:mysql://localhost:3306/Spring?
characterEncoding=utf8"/>
        <property name = "user" value = "root"/>
        <property name = "password" value = "qwerasdf811013"/>
</bean>
```

2）配置 `JdbcTemplate`

```
<!-- 2. 配置JdbcTemplate-->
<bean id = "jt" class = "org.springframework.jdbc.core.JdbcTemplate">
  <property name = "dataSource" ref = "ds"/>
</bean>
```

3）配置事务管理器

```
<!-- 3、配置事务管理器 -->
<bean id = "txman" class =
"org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name = "dataSource" ref = "ds"></property>
</bean>
```

3. 创建数据访问层（Dao类）
    - `@Autowired`：使`JdbcTemplate`自动生成
    - 通过`JdbcTemplate`的方法执行数据库操作

3 Spring 事务管理

3.1 事务特性

ACID属性

- **原子性**：要么都成功，要么都失败
- **一致性**：操作前与操作后的数据总量不变
- **隔离性**：多事务操作时，彼此间不相互影响
- **持久性**：事务提交后，表中的数据真正改变

常见的数据失效

- **脏读**：事务回滚导致数据失效，即一个事务看到了另一个事务未提交的更新数据
- **幻读**：从数据量看问题
    - 一个事务在执行过程中读取到了另一个事务已提交的插入数据，即:

    - 第一步:第一个事务刚开始时读取到一批数据，
    - 第二步:第二个事务插入了新数据并提交，
    - 第三步:第一个事务又读取这批数据，发现多了一条，好像发生幻觉。
- **不可重复读**：从数据值看问题
    - 在同一事务中，多次读取同一数据却返回不同的结果:即有其他事务更改了这些数据
      数据库通过锁机制解决并发问题，分为表锁定和行锁定

数据库通过锁机制解决并发问题，分为表锁定和行锁定

标准SQL的四种数据库隔离级别: （了解）

- `READ_ UNCOMITED`： 未提交读，导致脏读幻读、不可重复读
- `READ COMMITTED` ： 提交读，可防脏读，导致幻读、不可重复读
- `REPEATABLE_ READ`： 重复读，可防脏读，不可重复读，导致幻读
- `SERIALIZABLE` ： 序列化，完全服 ACID 的隔离级别， 可防止脏读、幻读、不可重复读

隔离级别越[高]，事务并发性能越[差]，处理的操作就越[少1]
因此在实际项目中，基于并发性能的考虑，一般使用[提交读]隔离级别
[提交读]可以避免脏读，不能避免不可重复读和幻读
[提交读]+悲观锁或乐观锁可以解决这些问题。

提交读（READ_COMMITTED）为什么不能防止幻读、不可重复读?
读写锁只有读读可以并发，读写、写读、写写都不能并发，降低了并发度
MySQL的提交读和可重复读使用多版本并发控制（MVCC）实现，而非读写锁
MVCC的读不加锁，读写和写读可以同时进行，只有写写需要加锁，提高了并发度
MVCC通过可见性算法、undo日志以及read view控制每个读操作的数据历史版本
提交读，每次读取最新的数据版本，多次读取同一行数据时，会出现不一致的现象，即不可重复读
重复读，第一次读的时候就确定了数据版本，整个事务过程中都使用这一版本，所以每次读取的数据都一致

3.2 使用步骤

1. 在 `Spring` 命名空间加入 `xmlns:tx="http://www.springframework.org/schema/tx"`
2. 在 `Dao` 实现类里加入使用 `@Autowired` 注入一个 `JdbcTemplate` 类
3. 在需要的方法上加上 `@Transactional` 注解或为整个类所有方法添加事务管理
4. Spring 配置文件中配置事务管理器
5. 使用事务管理器

```
<!-- 1、包自动扫描 -->
<context:component-scan base-package="com.qst"></context:component-scan>
<!-- 2、配置数据源 -->
<bean id="ds" class="com.mchange.v2.c3p0.ComboPooledDataSource">
  <property name="driverClass" value=""></property>
  <property name="jdbcUrl" value="${url}"></property>
  <property name="user" value="${user}"></property>
  <property name="password" value="${pwd}"></property>
</bean>
<!-- 3、注入JdbcTemplate -->
<bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate">
```

```
    <property name="dataSource" ref="ds"></property>
  </bean>
  <!-- 4、注入事务管理器 -->
  <bean id="txman"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="ds"></property>
  </bean>
  <!-- 5、使用事务管理器 -->
  <tx:annotation-driven transaction-manager="txman" />
```