Spring JDBC:

DataSource: một mẫu factory vật lý tạo connection

JdbcTemplate: lớp giúp đơn giản hóa việc dùng jdbc

1. Tạo kết nối thêm bảng: với beans.xml

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.jdbc.Driver"/>

<property name ="url" value="jdbc:mysql://localhost:3306/sampledb" />

<property name="username" value="root"/>

<property name="password" value="Leu1234567"/>

</bean>

<bean id="studentJdbcDAO" class="thaythuan.springjdbc.StudentJdbcDAO" init-method="createTableIfNotExist">

<property name="dataSource" ref="dataSource"></property>

</bean>

Sau đó tạo bảng student:

* Khai báo:

private DataSource dataSource;

private JdbcTemplate jdbcTemplate;

* Set giá trị:

public void setDataSource(DataSource dataSource){

this.dataSource = dataSource;

this.jdbcTemplate = new JdbcTemplate(dataSource);

}

* Tạo bảng:

DatabaseMetaData dbmd = null; // lấy meta data mà datasource này chứa

ResultSet rs = null;

try {

dbmd = dataSource.getConnection().getMetaData();

rs = dbmd.getTables(null,null,"student",null);

if(rs.next()){

System.out.println("table" + rs.getString("TABLE\_NAME") + "exist ");

return;

}

String sql="create table student (id bigint primary key , name varchar(1000), age integer)";

jdbcTemplate.execute(sql);

} catch (SQLException e) {

e.printStackTrace();

}

Hàm get table sẽ lấy tất cả các bảng của tất cả các database chứ ko phải chỉ bảng của db hiện dùng.

1. Insert record:

Dùng hàm:

jdbcTemplate.update(insertQuery,…);

dấu .. là thứ tự các thuộc tính theo câu insertquery câu query sẽ đc thêm vào theo file beans bằng DI như 1 thuộc tính của JdbcDAO.

<bean id="studentJdbcDAO" class="thaythuan.springjdbc.StudentJdbcDAO" init-method="createTableIfNotExist">

<property name="dataSource" ref="dataSource"></property>

<property name="insertQuery" value="insert into student(name,age) values(?,?)"/>

</bean>

Nhớ thêm phương thức set tương ứng với queryInsert.

1. Sử dụng statement:

Dùng câu lệnh execute với jdbc template và có một call back bên trong vì sao dùng call back: việc execute chia làm nhiều giai đoạn và sau khi xong giai đoạn lấy statement thì nó sẽ gọi trở lại vào hàm call back để xử lí statement ấy:

jdbcTemplate.execute(new StatementCallback<Integer>() {

@Override

public Integer doInStatement(Statement stmt) throws SQLException, DataAccessException {

ResultSet rs = stmt.executeQuery("select count(\*) from student");

return rs.next()?rs.getInt(1):0;

}

});

1. Select data vào list:

public List loadStudent(String name){

return jdbcTemplate.query("select \* from student where name like '%" + name + "%'", new StudentRowMapper());

}

Tìm tất cả student có name, dùng mapper nó là một adapter dùng để chuyển đổi dữ liệu trong quá trình sử lý, với mỗi lần select thì sẽ ra một resultset và số hàng tương ứng đang ở, thông qua adapter mapper này để chuyển về vật thể student

class StudentRowMapper implements RowMapper<Student>{

@Override

public Student mapRow(ResultSet rs, int rowNum) throws SQLException {

return new Student(rs.getInt(1),rs.getString(2),rs.getInt(3));

}

}

1. Update dữ liệu:

jdbcTemplate.execute(updateAgeByNameSQL, new PreparedStatementCallback<Integer>() {

@Override

public Integer doInPreparedStatement(PreparedStatement ps) throws SQLException, DataAccessException {

ps.setInt(1, age);

ps.setString(2, name);

return ps.executeUpdate();

}

});

Nhớ dùng DI với beans.xml cho updateAgeByNameSQL

1. Add theo batch:

public int[] add(List<Student> students){

List<Object[]> batch = new ArrayList<>();

students.forEach(s->batch.add(new Object[]{s.getName(),s.getAge()}));

return jdbcTemplate.batchUpdate(insertQuery,batch);

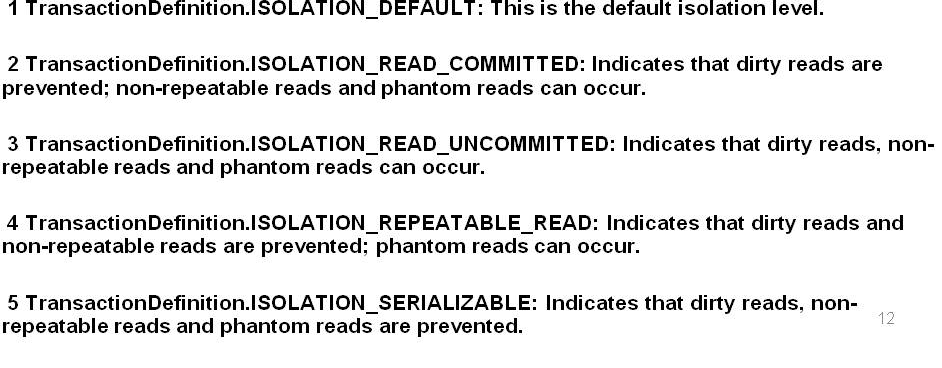
}

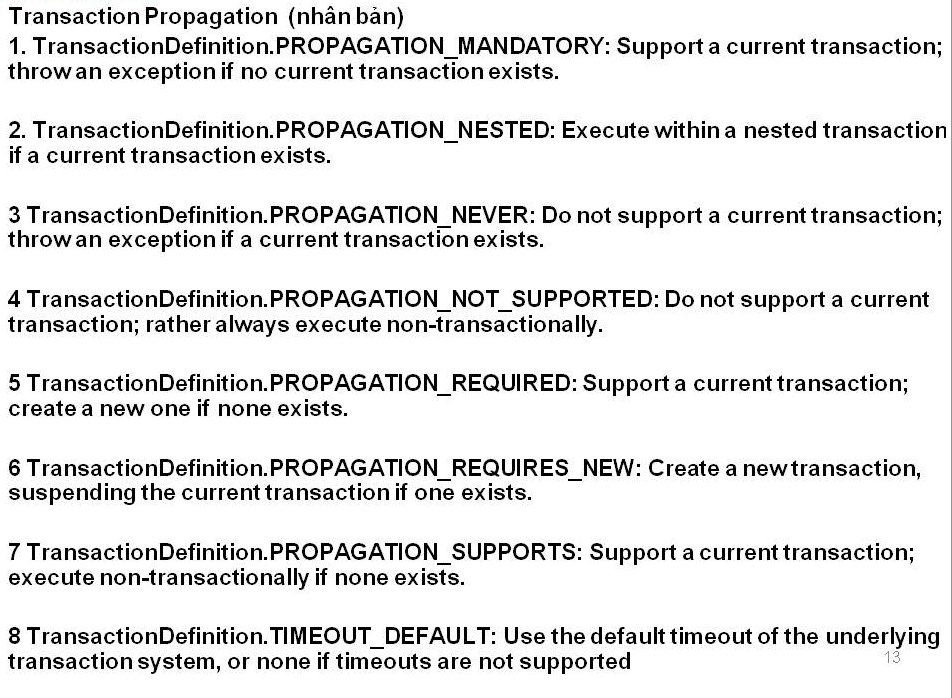
Add số lượng lớn với list các mảng object , một mảng object sẽ chứa các phần tử của một vật thể bị add.

1. Transaction local đc chỉ định cho tài nguyên hướng giao dịch đơn như jdbc connection

Transaction toàn cục có thể trải dài ra các tài nguyên hướng giao dịch như transaction trong hệ thống phân tán.

* Dirty read: đọc dữ liệu chưa commit
* Non-repeatable read: đọc ko nhất quán -> nó sẽ block việc update dữ liệu cho đến khi hoàn thành giao dịch
* Phantom read: đọc dl ko có thực.





1. Dùng transaction:

public void save(Object name,Object age){

TransactionDefinition def = new DefaultTransactionDefinition();

TransactionStatus status = transactionManager.getTransaction(def);

String sql = "";

int total ;

String countQuery = "select count(\*) from student";

try{

total = jdbcTemplate.queryForObject(countQuery, Integer.class);

System.out.println("before save data, total record is " + total);

sql = "insert into student(name,age) values(?,?)";

jdbcTemplate.update(sql,name,age);

total = jdbcTemplate.queryForObject(countQuery, Integer.class);

System.out.println("after save data, total rec is "+ total);

throw new Exception();

}catch(Exception e){

transactionManager.rollback(status);

total = jdbcTemplate.queryForObject(countQuery, Integer.class);

System.out.println("after rollback : " + total);

}

}

Đoạn code khai báo transaction:

TransactionDefinition def = new DefaultTransactionDefinition();

TransactionStatus status = transactionManager.getTransaction(def);

Với transaction manager:

@Autowired

@Qualifier("transactionManager")

private DataSourceTransactionManager transactionManager;

được xác định trong file bean: nó đc cung cấp datasource

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource"/>

</bean>

Trong đoạn code đầu việc thực thi sẽ chưa đc thực hiện khi transaction chưa commit.