You can declare in the @Transactional that you want Spring to perform a rollback when your MyException is thrown:

@Transactional(rollbackFor=MyException.class)

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/transaction.html#transaction-declarative-rolling-back>

<http://www.javatpoint.com/hibernate-transaction-management-example>

**How to begin transaction and commit and rollback on jdbc**

Connection conn = null;

Statement stmt = null;

try{

//STEP 2: Register JDBC driver

Class.forName("com.mysql.jdbc.Driver");

//STEP 3: Open a connection

System.out.println("Connecting to database...");

conn = DriverManager.getConnection(DB\_URL,USER,PASS);

//STEP 4: Set auto commit as false.

conn.setAutoCommit(false);

//STEP 5: Execute a query to create statment with

// required arguments for RS example.

System.out.println("Creating statement...");

stmt = conn.createStatement(

ResultSet.TYPE\_SCROLL\_INSENSITIVE,

ResultSet.CONCUR\_UPDATABLE);

//STEP 6: INSERT a row into Employees table

System.out.println("Inserting one row....");

String SQL = "INSERT INTO Employees " +

"VALUES (106, 20, 'Rita', 'Tez')";

stmt.executeUpdate(SQL);

//STEP 7: INSERT one more row into Employees table

SQL = "INSERT INTO Employees " +

"VALUES (107, 22, 'Sita', 'Singh')";

stmt.executeUpdate(SQL);

//STEP 8: Commit data here.

System.out.println("Commiting data here....");

conn.commit();

}catch(SQLException se){

conn.rollback();

}