How to get the insert ID in JDBC?

Asked 15 years ago Modified 7 months ago Viewed 379k times

442

I want to `INSERT` a record in a database (which is Microsoft SQL Server in my case) using JDBC in Java. At the same time, I want to obtain the insert ID. How can I achieve this using JDBC API?

java sql jdbc insert-id

Share

Improve this question

Follow

edited Oct 29, 2017 at 3:51

Aurgho Bhattacharjee
436 ● 2 ● 17

asked Dec 16, 2009 at 14:57

Satya
8,346 ● 10 ● 42 ● 44

1 Leave the **id** which is AutoGenrerated in the Query `String sql = "INSERT INTO 'yash'.'mytable' ('name') VALUES (?)"; int primkey = 0 ; PreparedStatement pstmt = con.prepareStatement(sql, new String[] { "id" }/*Statement.RETURN_GENERATED_KEYS*/); pstmt.setString(1, name); if (pstmt.executeUpdate() > 0) { java.sql.ResultSet generatedKeys = pstmt.` getGeneratedKeys(); `if (generatedKeys.next()) primkey =`

`generatedKeys.getInt(1);                 }` – Yash Nov 23, 2015 at 11:40

Just a note for everyone. You can only get Generated Keys with AUTO INC type. UUID or char or other types which use defaults will not work with MSSQL. – sproketboy Feb 3, 2021 at 14:04

16 Answers

Sorted by: Highest score (default)

757

If it is an auto generated key, then you can use `Statement#getGeneratedKeys()` for this. You need to call it on the same `Statement` as the one being used for the `INSERT`. You first **need** to create the statement using `Statement.RETURN_GENERATED_KEYS` to notify the JDBC driver to return the keys.

Here's a basic example:

```
public void create(User user) throws SQLException {
    try (
        Connection connection = dataSource.getConnecti
        PreparedStatement statement = connection.prepa
                                      Statement.RETURN
    ) {
        statement.setString(1, user.getName());
        statement.setString(2, user.getPassword());
        statement.setString(3, user.getEmail());
        // ...

        int affectedRows = statement.executeUpdate();

        if (affectedRows == 0) {
            throw new SQLException("Creating user fail
        }
```

```
            try (ResultSet generatedKeys = statement.getGe
                if (generatedKeys.next()) {
                    user.setId(generatedKeys.getLong(1));
                }
                else {
                    throw new SQLException("Creating user
obtained.");
                }
            }
        }
    }
```

Note that you're dependent on the JDBC driver as to whether it works. Currently, most of the last versions will work, but if I am correct, Oracle JDBC driver is still somewhat troublesome with this. MySQL and DB2 already supported it for ages. PostgreSQL started to support it not long ago. I can't comment about MSSQL as I've never used it.

For Oracle, you can invoke a `CallableStatement` with a `RETURNING` clause or a `SELECT CURRVAL(sequencename)` (or whatever DB-specific syntax to do so) directly after the `INSERT` in the same transaction to obtain the last generated key. See also this answer.

Share Improve this answer Follow

edited May 2, 2018 at 18:59

rogerdpack
66.5k ● 39 ● 282 ● 401

answered Dec 16, 2009 at 15:03

BalusC
1

5 It's better to get the next value in a sequence before the insert than to get the curval after the insert, because the latter might return the wrong value in a multi-threaded environment (e.g., any web app container). The JTDS MSSQL driver supports getGeneratedKeys. – JeeBee Dec 16, 2009 at 15:44

5 (should clarify that I usually use Oracle, so have very low expectations of a JDBC driver's capabilities normally). – JeeBee Dec 16, 2009 at 15:45

8 An interesting side-effect of NOT setting the Statement.RETURN_GENERATED_KEYS option is the error message, which is the completely obscure "The statement must be executed before any results can be obtained." – Chris Winters Mar 3, 2011 at 16:15

8 The `generatedKeys.next()` returns `true` if the DB returned a generated key. Look, it's a `ResultSet`. The `close()` is just to free resources. Otherwise your DB will run out of them on long run and your application will break. You just have to write up some utility method yourself which does the closing task. See also this and this answer. – BalusC Mar 9, 2011 at 18:55

5 Correct answer for most databases/drivers. For Oracle this does not work however. For Oracle, change to: connection.prepareStatement(sql,new String[] {"PK column name"}); – Darrell Teague Oct 13, 2014 at 13:37

33

1. Create Generated Column

   ```
   String generatedColumns[] = { "ID" };
   ```

2. Pass this geneated Column to your statement

   ```
   PreparedStatement stmtInsert = conn.prepareStateme
   generatedColumns);
   ```

3. Use `ResultSet` object to fetch the GeneratedKeys on Statement

   ```
   ResultSet rs = stmtInsert.getGeneratedKeys();

   if (rs.next()) {
       long id = rs.getLong(1);
       System.out.println("Inserted ID -" + id); // d
   }
   ```

Share Improve this answer Follow

edited Jan 16, 2017 at 16:10

Willi Mentzel
29.7k ● 21 ● 118 ● 126

answered Dec 6, 2016 at 4:59

Harsh Maheswari
507 ● 5 ● 3

15

When encountering an 'Unsupported feature' error while using `Statement.RETURN_GENERATED_KEYS`, try this:

```
String[] returnId = { "BATCHID" };
String sql = "INSERT INTO BATCH (BATCHNAME) VALUES ('a
PreparedStatement statement = connection.prepareStatem
int affectedRows = statement.executeUpdate();
```

```
if (affectedRows == 0) {
    throw new SQLException("Creating user failed, no r
}

try (ResultSet rs = statement.getGeneratedKeys()) {
    if (rs.next()) {
        System.out.println(rs.getInt(1));
    }
    rs.close();
}
```

Where `BATCHID` is the auto generated id.

Share Improve this answer

Follow

edited Dec 10, 2019 at 14:03

Miss Chanandler Bong
4,258 ● 11 ● 28 ● 39

answered Dec 14, 2015 at 8:39

Eitan Rimon
711 ● 1 ● 8 ● 14

you mean `BATCHID` – moolsbytheway Apr 3, 2018 at 20:20

9

I'm hitting Microsoft SQL Server 2008 R2 from a single-threaded JDBC-based application and pulling back the last ID without using the RETURN_GENERATED_KEYS property or any PreparedStatement. Looks something like this:

```
private int insertQueryReturnInt(String SQLQy) {
    ResultSet generatedKeys = null;
    int generatedKey = -1;

    try {
        Statement statement = conn.createStatement();
        statement.execute(SQLQy);
    } catch (Exception e) {
        errorDescription = "Failed to insert SQL query
e.toString() + ")";
        return -1;
    }

    try {
        generatedKey = Integer.parseInt(readOneValue("
    } catch (Exception e) {
        errorDescription = "Failed to get ID of just-i
SQLQy + "( " + e.toString() + ")";
        return -1;
    }

    return generatedKey;
}
```

This blog post nicely isolates three main SQL Server "last ID" options: http://msjawahar.wordpress.com/2008/01/25/how-to-find-the-last-identity-value-inserted-in-the-sql-server/ - haven't needed the other two yet.

Share Improve this answer Follow

edited Jul 3, 2015 at 8:41
Community Bot
1 ● 1

answered Jun 24, 2011 at 5:52
ftexperts
746 ● 7 ● 9

6 That the application has only one thread doesn't make a race condition impossible: if two clients insert a row and retrieve the ID with your method, it may fail. – 11684 Mar 11, 2013 at 8:54

Why would you? I'm just glad I'm not the poor sod who has to debug your code when allowing multiple threads! – mjaggard Sep 3, 2013 at 13:32

@11684 yes you are right. Some drivers just don't give the ID via `statement.getGeneratedKeys()`, which makes this attempt "understandable". However supplying the ID(s) during the prepareStatement solves this (e.g. `preapareStatement(query, new String[] {insertIdColumnName})`). See @Yash's slightly underrated answer for more details. – Levite Mar 3, 2022 at 12:53

7

Instead of a comment, I just want to answer post.

Interface *java.sql.PreparedStatement*

1. **columnIndexes** « You can use prepareStatement function that accepts columnIndexes and SQL statement. *Where columnIndexes allowed constant flags are Statement.RETURN_GENERATED_KEYS1 or Statement.NO_GENERATED_KEYS[2], SQL statement that may contain one or more '?' IN parameter placeholders.*

   SYNTAX «

   ```
   Connection.prepareStatement(String sql, int autoGe
   Connection.prepareStatement(String sql, int[] colu
   ```

   Example:

```
PreparedStatement pstmt =
    conn.prepareStatement( insertSQL, Statement.RE
```

---

2. **columnNames** « *List out the columnNames like* `'id', 'uniqueID', ...`*. in the target table that contain the auto-generated keys that should be returned. The driver will ignore them if the SQL statement is not an* `INSERT` *statement.*

   SYNTAX «

   ```
   Connection.prepareStatement(String sql, String[] c
   ```

   Example:

   ```
   String columnNames[] = new String[] { "id" };
   PreparedStatement pstmt = conn.prepareStatement( i
   ```

Full Example:

```
public static void insertAutoIncrement_SQL(String User
String Message) {
    String DB_URL = "jdbc:mysql://localhost:3306/test"
DB_Password = "";

    String insertSQL = "INSERT INTO `unicodeinfo`( `Us
`Message`) VALUES (?,?,?)";
            //"INSERT INTO `unicodeinfo`(`id`, `UserNa
`Message`) VALUES (?,?,?,?)";
    int primkey = 0 ;
    try {
        Class.forName("com.mysql.jdbc.Driver").newInst
        Connection conn = DriverManager.getConnection(
DB_Password);
```

```
        String columnNames[] = new String[] { "id" };

        PreparedStatement pstmt = conn.prepareStatemen
);
        pstmt.setString(1, UserName );
        pstmt.setString(2, Language );
        pstmt.setString(3, Message );

        if (pstmt.executeUpdate() > 0) {
            // Retrieves any auto-generated keys creat
executing this Statement object
            java.sql.ResultSet generatedKeys = pstmt.g
            if ( generatedKeys.next() ) {
                primkey = generatedKeys.getInt(1);
            }
        }
        System.out.println("Record updated with id = "
    } catch (InstantiationException | IllegalAccessExc
ClassNotFoundException | SQLException e) {
        e.printStackTrace();
    }
}
```

Share Improve this answer Follow

answered Jan 18, 2018 at 13:20

Yash
9,548 ● 2 ● 73 ● 79

1 Is it safe to use this solution in a multithreaded runtime environment? – The Prototype Jun 14, 2020 at 16:55

This deserves way more upvotes!! It solves returning of IDs even for older drivers - no need to use `@@IDENTIY` (when supplying a `String Array` of requested IDs). – Levite Mar 3, 2022 at 12:44

I'm using **SQLServer** 2008, but I have a development limitation: I cannot use a new driver for it, I have to use

3

"com.microsoft.jdbc.sqlserver.SQLServerDriver" (I cannot use "com.microsoft.sqlserver.jdbc.SQLServerDriver").

That's why the solution `conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)` threw a **java.lang.AbstractMethodError** for me. In this situation, a possible solution I found is the old one suggested by Microsoft: How To Retrieve @@IDENTITY Value Using JDBC

```
import java.sql.*;
import java.io.*;

public class IdentitySample
{
    public static void main(String args[])
    {
        try
        {
            String URL =
"jdbc:microsoft:sqlserver://yourServer:1433;databasena
            String userName = "yourUser";
            String password = "yourPassword";

            System.out.println( "Trying to connect to:

            //Register JDBC Driver

Class.forName("com.microsoft.jdbc.sqlserver.SQLServerD

            //Connect to SQL Server
            Connection con = null;
            con = DriverManager.getConnection(URL,user
            System.out.println("Successfully connected

            //Create statement and Execute using eithe
batch statement
            CallableStatement callstmt = null;

            callstmt = con.prepareCall("INSERT INTO my
(?);SELECT @@IDENTITY");
            callstmt.setString(1, "testInputBatch");
            System.out.println("Batch statement succes
            callstmt.execute();

            int iUpdCount = callstmt.getUpdateCount();
            boolean bMoreResults = true;
            ResultSet rs = null;
            int myIdentVal = -1; //to store the @@IDEN

            //While there are still more results or up
            //available, continue processing resultset
            while (bMoreResults || iUpdCount!=-1)
            {
                //NOTE: in order for output parameters
                //all resultsets must be processed

                rs = callstmt.getResultSet();

                //if rs is not null, we know we can ge
SELECT @@IDENTITY
                if (rs != null)
                {
                    rs.next();
                    myIdentVal = rs.getInt(1);
                }

                //Do something with the results here (

                //get the next resultset, if there is
                //this call also implicitly closes the
ResultSet
                bMoreResults = callstmt.getMoreResults
                iUpdCount = callstmt.getUpdateCount();
            }

            System.out.println( "@@IDENTITY is: " + my

            //Close statement and connection
            callstmt.close();
            con.close();
        }
        catch (Exception ex)
        {
            ex.printStackTrace();
        }

        try
        {
            System.out.println("Press any key to quit.
            System.in.read();
        }
        catch (Exception e)
        {
        }
    }
}
```

This solution worked for me!

I hope this helps!

Share Improve this answer Follow

answered Sep 10, 2013 at 9:41

xanblax
286 ● 2 ● 5

Try supplying `String[]` array of id names you want, instead of `RETURN_GENERATED_KEYS`. This should suddenly give you a valid ResultSet and the ID via getInt(1) therein.
– Levite Mar 3, 2022 at 12:48

3

You can use following java code to get new inserted id.

```
ps = con.prepareStatement(query, Statement.RETURN_GENE
ps.setInt(1, quizid);
ps.setInt(2, userid);
ps.executeUpdate();

ResultSet rs = ps.getGeneratedKeys();
if (rs.next()) {
    lastInsertId = rs.getInt(1);
}
```

Share Improve this answer

Follow

edited Dec 10, 2019 at 12:57

Miss Chanandler Bong
4,258 ● 11 ● 28 ● 39

answered May 21, 2019 at 10:35

user11533210

2

It is possible to use it with normal `Statement`'s as well (not just `PreparedStatement`)

```
Statement statement = conn.createStatement();
int updateCount = statement.executeUpdate("insert into
Statement.RETURN_GENERATED_KEYS);
try (ResultSet generatedKeys = statement.getGeneratedK
  if (generatedKeys.next()) {
    return generatedKeys.getLong(1);
  }
  else {
    throw new SQLException("Creating failed, no ID obt
  }
}
```

Share Improve this answer

Follow

answered May 4, 2018 at 19:33

rogerdpack
66.5k ● 39 ● 282 ● 401

that help for me. – Thilina Sampath Jun 15, 2021 at 19:37

1

With Hibernate's NativeQuery, you need to return a ResultList instead of a SingleResult, because Hibernate modifies a native query

```
INSERT INTO bla (a,b) VALUES (2,3) RETURNING id
```

like

```
INSERT INTO bla (a,b) VALUES (2,3) RETURNING id LIMIT
```

if you try to get a single result, which causes most databases (at least PostgreSQL) to throw a syntax error. Afterwards, you may fetch the resulting id from the list (which usually contains exactly one item).

Share Improve this answer Follow

edited Apr 23, 2018 at 8:26
NightOwl888
56.8k ● 27 ● 147 ● 220

answered Apr 23, 2018 at 8:01
Balin
81 ● 3

1

Most others have suggested to use JDBC API for this, but personally, I find it quite painful to do with most drivers. When in fact, you can just use a native T-SQL feature, [the `OUTPUT` clause]:

```
try (
    Statement s = c.createStatement();
    ResultSet rs = s.executeQuery(
        """
        INSERT INTO t (a, b)
        OUTPUT id
        VALUES (1, 2)
        """
    );
) {
    while (rs.next())
        System.out.println("ID = " + rs.getLong(1));
}
```

This is the simplest solution for SQL Server as well as a few other SQL dialects (e.g. Firebird, MariaDB, PostgreSQL, where you'd use `RETURNING` instead of `OUTPUT`).

I've blogged about this topic more in detail here.

Share Improve this answer Follow

answered Aug 23, 2022 at 15:18

Lukas Eder
220k ● 135 ● 718 ● 1.6k

0

In my case ->

```
ConnectionClass objConnectionClass=new ConnectionClass
con=objConnectionClass.getDataBaseConnection();
pstmtGetAdd=con.prepareStatement(SQL_INSERT_ADDRESS_QU
pstmtGetAdd.setString(1, objRegisterVO.getAddress());
pstmtGetAdd.setInt(2, Integer.parseInt(objRegisterVO.g
int addId=pstmtGetAdd.executeUpdate();
if(addId>0)
{
    ResultSet rsVal=pstmtGetAdd.getGeneratedKeys();
    rsVal.next();
    addId=rsVal.getInt(1);
}
```

Share Improve this answer

Follow

edited Aug 29, 2018 at 13:37

KarlR

1,615 ● 14 ● 29

answered Aug 29, 2018 at 12:10

TheSagya

45 ● 12

Still I think it's lengthy approach to get it. I think there will be more compressed solution also. – TheSagya Dec 31, 2018 at 10:12

0

If you are using Spring JDBC, you can use Spring's GeneratedKeyHolder class to get the inserted ID.

See this answer... [How to get inserted id using Spring Jdbctemplate.update(String sql, obj...args)](#)

Share Improve this answer

Follow

answered Jan 25, 2019 at 2:45

Rob Breidecker

604 ● 1 ● 7 ● 13

0

If you are using JDBC (tested with MySQL) and you just want the last inserted ID, there is an easy way to get it. The method I'm using is the following:

```
public static Integer insert(ConnectionImpl connection
    Integer lastInsertId = -1;
    try{
        final PreparedStatement ps = connection.prepar
        ps.executeUpdate(insertQuery);
        final com.mysql.jdbc.PreparedStatement psFinal
(com.mysql.jdbc.PreparedStatement) ps;
        lastInsertId = (int) psFinal.getLastInsertID()
        connection.close();
    } catch(SQLException ex){
        System.err.println("Error: "+ex);
    }

    return lastInsertId;
}
```

Also, (and just in case) the method to get the `ConnectionImpl` is the following:

```
public static ConnectionImpl getConnectionImpl(){
    ConnectionImpl conexion = null;

    final String dbName = "database_name";
    final String dbPort = "3306";
    final String dbIPAddress = "127.0.0.1";
    final String connectionPath =
"jdbc:mysql://"+dbIPAddress+":"+dbPort+"/"+dbName+"?
autoReconnect=true&useSSL=false";

    final String dbUser = "database_user";
    final String dbPassword = "database_password";
    try{
        conexion = (ConnectionImpl) DriverManager.getC
dbUser, dbPassword);
    }catch(SQLException e){
        System.err.println(e);
    }

    return conexion;
}
```

Remember to add the connector/J to the project referenced libraries.

In my case, the connector/J version is the 5.1.42. Maybe you will have to apply some changes to the `connectionPath` if you want to use a more modern version of the connector/J such as with the version 8.0.28.

In the file, remember to import the following resources:

```
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import com.mysql.jdbc.ConnectionImpl;
```

Hope this will be helpful.

Share Improve this answer Follow

answered Apr 5, 2022 at 14:43

Pol
454 ● 2 ● 4 ● 12

0

You can use `executeQuery(query)` function

and add `RETURNING *` clause to your query instead `*` you can specify column name

For example

create table first

```
CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREME
```

create query

```
val query: String = "INSERT INTO users (name) VALUES (
("Adam") RETURNING id;"
```

apply query

```
val statement = connection.prepareStatement(query)
val result = statement.executeQuery()
val keys = buildList {
    while (result.next()) { add(result.getLong(1)) }
}
```

keys is your result. So you can return any column set or full object

Share Improve this answer Follow

answered Jun 8, 2023 at 18:17

Д Динар Исламов
21 ● 3

0

To get the last inserted Id on a table in MSSQL there are various options:

What I use is this one:

```
IDENT_CURRENT(‘TableName’)
```

There are some other options like **SCOPE_IDENTITY()** and **@@IDENTITY** as well but the above is much easier.

Example:

```
String sql = "SELECT IDENT_CURRENT('TableName') AS ID;
try ( ResultSet rs = con.execSQL (sql, sqlParameters))
    while (rs.next ()) {
        ret = rs.getInt ("ID");
    }
}
............
```

For more info [check this out]()

I hope this helps...

Share Improve this answer

Follow

answered May 9 at 13:26

MaVRoSCy

17.8k ● 15 ● 84 ● 128

-6

```
Connection cn = DriverManager.getConnection("Host","us
Statement st = cn.createStatement("Ur Requet Sql");
int ret  = st.execute();
```

Share Improve this answer

Follow

edited Mar 14, 2017 at 23:00

Dave2e

24k ● 18 ● 44 ● 54

answered Mar 14, 2017 at 22:35

Abdelkhalek Benhoumine

9

Excuse me, but what this is supposed to be ?
– moolsbytheway Apr 3, 2018 at 20:26

1. The `createStatement` method from `Connection` do not expect any params. 2. The `execute` method from `Statement` expects a String with a Query. 3. The `execute` method returns: `true` if the first result is a `ResultSet` object; `false` if it is an update count or there are no results. docs.oracle.com/javase/7/docs/api/java/sql/… – atilacamurca May 17, 2018 at 12:10