JAVA SE

Quiz yourself: Working with PreparedStatement and SQL NULL values in Java

It's not always easy to work with NULL values via SQL statements.

by Simon Roberts and Mikalai Zaikin

August 23, 2021

If you have worked on our quiz questions in the past, you know none of them is easy. They model the difficult questions from certification examinations. We write questions for the certification exams, and we intend that the same rules apply: Take words at their face value and trust that the questions are not intended to deceive you but to straightforwardly test your knowledge of the ins and outs of the language.

Imagine that your application uses the `PRODUCTS` relational database table with the following data definition language (DDL) description:

```
CREATE TABLE APP.PRODUCTS (
  ID INTEGER DEFAULT AUTOINCREMENT NOT NULL,
  DESCRIPTION VARCHAR(100),
  PRICE INTEGER
);
```

However, later you notice that some products have no description but have a price.

```
ID        DESCRIPTION    PRICE
1         <NULL>         10
2         <NULL>         20
```

You want to make those products' prices NULL for consistency and you have written the following not-yet-complete code:

```
var conn = DriverManager.getConnection(dbURL);
var pstmt = conn.prepareStatement("UPDATE APP.PROI
```

```
... // add code here
pstmt.execute();
conn.commit();
```

Which code, added individually, would complete the code above so that it updates the price fields as required? Choose two.

A.

```
Integer p = null;
     pstmt.setInt(1, p);
```

The answer is A.

B.

```
pstmt.setNull(1);
```

The answer is B.

C.

```
pstmt.setNull(1, Types.INTEGER);
```

The answer is C.

D.

```
Integer p = null;
     pstmt.setObject(1, p, Types.INTEGER);
```

The answer is D.

Answer. Option A uses the `setInt()` method. However, this method's argument type is a primitive `int`. The compilation succeeds by applying autounboxing. However, at runtime, because the wrapper reference is null, the conversion attempt will throw a `NullPointerException`. Because of this, option A is incorrect.

The type of the variable `pstmt` will be `PreparedStatement`, and that type provides two overloaded `setNull` methods. However, both require at least two parameters: The first is the index of the parameter to be set, and the second is an indication of the type of the table column. This is done because JDBC is supposed to work with databases from many vendors and some relational databases require that the type of the column be provided when setting the null value. From this, you know that there is no `setNull` method that takes a single parameter and, therefore, option B must be incorrect.

Option C is correct. In this option, the code explicitly specifies the SQL type of the column to be updated. The full signature of this `setNull` method is the following:

```
void setNull(int parameterIndex, int sqlType) thro
```

Option D is also correct. This option demonstrates another approach for assigning a null value to a parameter. The `setObject` method shown will correctly set a null value if that's what's in the argument it receives.

There are two additional observations here.

First, one of the five overloaded versions of the `setObject` method (see the documentation) does not require the column's SQL type as a parameter. However, the documentation notes that some databases do not allow a NULL value to be set without a type being specified. So for safety and portability, it's good practice to use one of the other four overloads and provide the SQL type explicitly, as was done in option D.

Second, the behavior of null in Java differs significantly from that of NULL in SQL. In Java you can test a value against null reliably, as shown in the following example:

```
if (obj == null) {...} // do something
```

However, if you use a similar approach to search for rows that have NULL fields in SQL, you might be tempted to write code like this.

```
SELECT * FROM APP.PRODUCT WHERE PRICE=?
```

Unfortunately, the result of that will always be an empty result set, because the two NULL fields are not treated as equal to each other in SQL. For this reason, SQL has special syntax for checking: `COLUMN IS NULL`.

Based on this information, you can conclude that the `setNull` method may reasonably be used in `INSERT` or `UPDATE` statements but should be avoided in `SELECT` queries.

Conclusion. The correct answers are options C and D.

Related quizzes

- Quiz yourself: Use PreparedStatement to perform database CRUD operations
- Quiz yourself: The Optional class and null values in Java

Simon Roberts

Simon Roberts joined Sun Microsystems in time to teach Sun's first Java classes in the UK. He created the Sun Certified Java Programmer and Sun Certified Java Developer exams. He wrote

several Java certification guides and is currently a freelance educator who publishes recorded and live video training through Pearson InformIT (available direct and through the O'Reilly Safari Books Online service). He remains involved with Oracle's Java certification projects.

Mikalai Zaikin

Mikalai Zaikin is a lead Java developer at IBA IT Park in Minsk, Belarus. During his career, he has helped Oracle with development of Java certification exams, and he has been a technical reviewer of several Java certification books, including three editions of the famous *Sun Certified Programmer for Java* study guides by Kathy Sierra and Bert Bates.

Share this Page