

Forum:[Hack Forums](#)

Thread: [Get current date java.sql.Date](#)

Nemo2478 - May 26, 2020, 2:21 am

Hello everybody,

I have to get the current date in java.sql.Date format to pass the data to a table in MySQL. But I am completely lost:

How to insert the date in MySQL from java?

Code:

Code private Date date;

How would you define the "date" within the following constructor to get its value?

Code:

```
Code Sale s=new Sale(0, Integer.parseInt(request.getParameter("idClient")),
                    Integer.parseInt(request.getParameter("idBook")), **date** ) {
    };
gsales.registerSale(s);
```

Thank you

Regards

Sterling Malory Archer - May 26, 2020, 7:00 am

Hi there, don't know if you still need but anyway.

Use Calendar package first to get the instance and then time.

Code java.util.Calendar
and then use it in the constructor.

Code java.sql.Date
This should look like...

Code java.sql.Date date = new java.sql.Date(Calendar.getInstance().getTime().getTime());
so the final code may look like like below

```
Code Sale s=new Sale(0, Integer.parseInt(request.getParameter("idClient")),
                    Integer.parseInt(request.getParameter("idBook")), new
java.sql.Date(Calendar.getInstance().getTime().getTime()) ) {
    };
gsales.registerSale(s);
```

MavenGradle - May 26, 2020, 7:13 am

(May 26, 2020 02:21 AM)Nemo2478 Wrote: [\[CITE\]](#) How to insert the date in MySQL from java?
You insert dates, or rather data in Java to a database using PreparedStatement. The 'official' oracle tutorial on how to achieve such a thing can be found here:

<https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html>

An example on how to insert a date using PreparedStatement:

```
Code final java.sql.Date sqlDate = new java.sql.Date();
```

```
final java.sql.Timestamp sqlTime = new java.sql.Timestamp();
```

```
final PreparedStatement preparedStatement = connection.prepareStatement("insert into tblDates  
(date,time) values(?,?)");
```

```
preparedStatement.setDate(1,sqlDate);
```

```
preparedStatement.setTimestamp(2,sqlTime);
```

```
preparedStatement.executeUpdate();
```

```
preparedStatement.close();
```

```
connection.close();
```

Do note that the `executeUpdate()` method on a `PreparedStatement` instance returns an integer value that indicates how many rows were affected by the statement or returns the value 0 for SQL statements that return nothing. It might be wise to make use of that value for, for example error handling or simple logging just to name a couple of examples.

Ideally speaking, you would utilize a try-with resource block statement in addition to proper error handling in order to ensure proper resource disposal and logging, even in the case of exceptions:

<https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html>

Example:

```
Code public final void viewTable(final Connection con) throws SQLException {
```

```
    final String query = "SELECT COF_NAME, SUP_ID, PRICE, SALES, TOTAL FROM COFFEES";
```

```
    try (final Statement stmt = con.createStatement()) {
```

```
        final ResultSet rs = stmt.executeQuery(query);
```

```
        while (rs.next()) {
```

```
            final String coffeeName = rs.getString("COF_NAME");
```

```
            final int supplierID = rs.getInt("SUP_ID");
```

```
            final float price = rs.getFloat("PRICE");
```

```
            final int sales = rs.getInt("SALES");
```

```
            final int total = rs.getInt("TOTAL");
```

```
            logger.info(coffeeName + ", " + supplierID + ", " +  
                        price + ", " + sales + ", " + total);
```

```
        }
```

```
    } catch (final SQLException e) {
```

```
        logger.error(e);
```

```
    }
```

```
}
```

This code fragment is taken from the [Oracle docs listed here](#) but slightly modified to include an [SLF4J](#) logging interface.

(May 26, 2020 02:21 AM)Nemo2478 Wrote: [\[CITE\]](#) How would you define the "date" within the following constructor to get its value?

Code:

```
Code Sale s=new Sale(0, Integer.parseInt(request.getParameter("idClient")),
```

```
Integer.parseInt(request.getParameter("idBook")), **date** ) {  
};  
gsales.registerSale(s);
```

I'm not sure what you are asking for in regards to your second question. Could you elaborate? I'm thinking you'd simply define it as the Date object that it is. You can also convert java.sql.Date and java.util.Date using their respective methods and constructors. For example, to convert a java.util.Date to a java.sql.Date:

```
Code new java.sql.Date(utilDate.getTime());
```

However, for modern application development, it is recommend that you make use of a Calendar object instance instead:

<https://docs.oracle.com/javase/7/docs/api/java/util/Calendar.html>

Or even better (actually preferred), [Joda-Time](#) (java.time).

Additionally, if I were you, I'd let the SQL part of your application be handled by [Hibernate](#) or other [JPA implementations](#) if you're working with specific entities/objects and data mapping. The inclusion of the Spring Framework will make your life a whole lot easier (not to mention make you a whole lot more interesting to hire if you can work in said framework):

<https://spring.io/>

<https://spring.io/projects/spring-data-jpa>

Nemo2478 - June 4, 2020, 1:23 pm

Doc Calendar api is old, new is java.time api.