# Writing and Reading MySQL BLOB Using JDBC

?

This tutorial shows you how to write and read MySQL BLOB data using JDBC API.

We will use the `candidates` table in the mysqljdbc sample database. For the sake of demonstration, we will add one more column named `resume` into the `candidates` table. The data type of this column will be `MEDIUMBLOB` that can hold up to 16MB.

The following ALTER TABLE statement adds resume column into the `candidates` table.

```
1 ALTER TABLE candidates
2 ADD COLUMN resume LONGBLOB NULL AFTER email;
```

We will use a sample resume in PDF format and load this file into the `resume` column of the `candidates` table later. You can download the sample PDF file for practicing via the following link:

Sample Resume (303.37 kB)
1004 downloads

## Writing BLOB data into MySQL database

The steps for writing BLOB data into MySQL database is as follows:

First, open a new connection to the database by creating a new`Connection` object.

```
Connection conn =
DriverManager.getConnection(url,username,password);
```

Then, construct an UPDATE statement and create a`PreparedStatement` from the `Connection` object.

```
String updateSQL = "UPDATE candidates "
                 + "SET resume = ? "
                 + "WHERE id=?";

PreparedStatement pstmt = conn.prepareStatement(updateSQL);
```

Next, read data from the sample resume file using`FileInputStream` and call `setBinaryStream()` method to set parameters for the `PreparedStatement` .

```
// read the file
File file = new File(filename);
FileInputStream input = new FileInputStream(file);

// set parameters
pstmt.setBinaryStream(1, input);
pstmt.setInt(2, candidateId);
```

After that, call the `executeUpdate()` method of the`PreparedStatement` object.

```
pstmt.executeUpdate();
```

Finally, close the `PreparedStatement` and `Connection` objects by calling the `close()` methods.

To simplify the `Connection` creation process, we use the`MySQLJDBCUtil` class that we developed in the previous tutorial to open a new connection. The complete example of writing BLOB data into MySQL database is as follows:

```
package org.mysqltutorial;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

```
6  import java.sql.Connection;
7  import java.sql.PreparedStatement;
8  import java.sql.SQLException;
9
10 /**
11  *
12  * @author mysqltutorial.org
13  */
14 public class Main {
15
16     /**
17      * Update resume for a specific candidate
18      *
19      * @param candidateId
20      * @param filename
21      */
22     public static void writeBlob(int candidateId,
23  String filename) {
24         // update sql
25         String updateSQL = "UPDATE candidates "
26                 + "SET resume = ? "
27                 + "WHERE id=?";
28
29         try (Connection conn = MySQLJDBCUtil.getC
   onnection();
30                 PreparedStatement pstmt = conn.pr
31 epareStatement(updateSQL)) {
32
33             // read the file
34             File file = new File(filename);
35             FileInputStream input = new FileInput
36 Stream(file);
37
38             // set parameters
39             pstmt.setBinaryStream(1, input);
40             pstmt.setInt(2, candidateId);
41
42             // store the resume file in database
43
```

```
            System.out.println("Reading file " +
file.getAbsolutePath());
            System.out.println("Store file in the
 database.");
            pstmt.executeUpdate();

        } catch (SQLException | FileNotFoundExcep
tion e) {
            System.out.println(e.getMessage());
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        writeBlob(122, "johndoe_resume.pdf");

    }

}
```

Let's run the program.

```
Output - MySQLJDBCBlobClob (run)
run:
Reading file C:\JDBC\MySQLJDBCBlobClob\johndoe_resume.pdf
Store file in the database.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Now we check the candidates table for candidate with id 122.

```
SELECT * FROM candidates WHERE id = 122;
```

| id | first_name | last_name | dob | phone | email | resume |
|---|---|---|---|---|---|---|
| 122 | John | Doe | 1990-01-04 | (408) 898-5641 | john.d@yahoo.com | BLOB |

As you see, we have BLOB data updated in the resume column of the `candidates` table for record with id 122.

# Reading BLOB data from MySQL database

The process of reading BLOB data from the database is similar to the process of writing BLOB except the part that we write BLOB data into the file.

First, open a new connection to the database.

```
Connection conn = MySQLJDBCUtil.getConnection(dbURL,username,password);
```

Then, construct a SELECT statement and create a`PreparedStatement` from the `Connection` object.

```
String selectSQL = "SELECT resume FROM candidates
 WHERE id=?";
PreparedStatement pstmt = conn.prepareStatement(selectSQL);
```

Next, set the parameters and execute the query:

```
pstmt.setInt(1, candidateId);
ResultSet rs = pstmt.executeQuery();
```

After than, get BLOB data from the `ResultSet` and write it into a file:

```
File file = new File(filename);
FileOutputStream output = new FileOutputStream(file);

System.out.println("Writing to file " + file.getAbsolutePath());
while (rs.next()) {
    InputStream input = rs.getBinaryStream("resume");
    byte[] buffer = new byte[1024];
    while (input.read(buffer) > 0) {
        output.write(buffer);
    }
}
```

Finally, call `close()` methods of `PreparedStatment` and`Connection` objects. If you use try-with-resources statement, you don't have to do it explicitly.

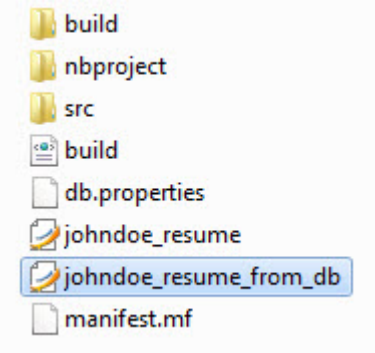
The following example illustrates how to read BLOB data from MySQL database.

```
1  package org.mysqltutorial;
2
3  import java.io.File;
4  import java.io.FileOutputStream;
5  import java.io.IOException;
6  import java.io.InputStream;
7  import java.sql.Connection;
8  import java.sql.PreparedStatement;
9  import java.sql.ResultSet;
10 import java.sql.SQLException;
11
12 /**
13  *
14  * @author Main.org
15  */
16 public class Main {
17
18     /**
19      * Read resume of a candidate and write it in
20 to a file
21      *
22      * @param candidateId
23      * @param filename
24      */
25     public static void readBlob(int candidateId,
26 String filename) {
27         // update sql
28         String selectSQL = "SELECT resume FROM ca
29 ndidates WHERE id=?";
30         ResultSet rs = null;
31         try (Connection conn = MySQLJDBCUtil.getC
32 onnection();
```

```
                PreparedStatement pstmt = conn.pr
epareStatement(selectSQL);) {
            // set parameter;
            pstmt.setInt(1, candidateId);
            rs = pstmt.executeQuery();

            // write binary stream into file
            File file = new File(filename);
            FileOutputStream output = new FileOut
putStream(file);

            System.out.println("Writing to file "
 + file.getAbsolutePath());
            while (rs.next()) {
                InputStream input = rs.getBinaryS
tream("resume");
                byte[] buffer = new byte[1024];
                while (input.read(buffer) > 0) {
                    output.write(buffer);
                }
            }
        } catch (SQLException | IOException e) {
            System.out.println(e.getMessage());
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
            } catch (SQLException e) {
                System.out.println(e.getMessage())
            }
        }

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
```

```
            //
            readBlob(122, "johndoe_resume_from_db.pd
f");
        }


}
```

After runing the program, browsing the project the folder, you will see that there is a new file named`johndoe_resume_from_db.pdf` created.




In this tutorial, we have shown you how to work with MySQL BLOB data from JDBC.