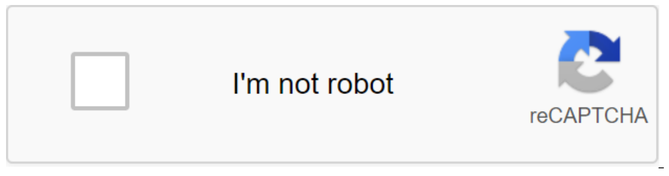

Continue

Java write blob to pdf file

The use of large LOB (Large Object Bynary) often has to be closely watched. This is due to the storage of various files in the database tables (DB). The question immediately arises, why store a file in the database, if then you have to extract it back again in the form of a file? After all, you can create a separate file store. The answer is also simple and trivial - file storage requires additional issues of replication and synchronization, granting access rights, and creating backup. Thus, if you can not do without LOB objects, it remains to choose their type - binary (BLOB Binary Large Object) or symbolic, called in the database of different providers in different ways (CLOB, TEXT). NOTE: When dealing with symbolic data, you need to consider encoding. This question will be taken into account when you write a text file in the database tables and unload from the database. Description of the example In the article looks at the example of writing and reading several types of files in the BLOB and CLOB (TEXT) fields. Oracle 10g Enterprise Edition, Oracle 10g Express Edition and MyS'L 5.1 were used as BD servers. An example is the Eclipse project, which implements all the principles of OOP (object-oriented programming) - inheritance, encapsulation and polymorphism. The structure of the project is presented in the following screenshot. The lib directory hosts JDBC libraries to connect to BD servers. The ojdbc7.jar was used to connect to the 10g Enterprise Edition. In the case of 10g Express Edition, the libraries of ojdbc14_g.jar, ojdbc14.jar were used. To connect the library to the IDE Eclipse, you need to highlight it in the Choose Build Path/Add to Build Path. To disable the use of the library in the project, use the Java Build Path tab in the project property window. The example includes the following software modules: DAOBase.java base module to connect to the database server and work with BLOB and CLOB (TEXT) fields; MainTest.java is the main testing module for LOB reading and writing methods; MyS'LDAO.java module to create a connection to the server MyS'L, inherits DAOBase.java; OracleDAO.java plug-in to Oracle server, inherits DAOBase.java; The aircraft.jpg, report.xlsx, text.txt Application in two different types of databases creates tables with BLOB and CLOB (TEXT) fields, in which files are first recorded, then read them. Presented procedures of writing and reading large objects you can use in their applications with almost no refinement. The creation of Connection connections to various database servers is described here and is not considered in this article (the code is presented). You can download an example here. Description of the base module, DAOBase.java Base Module includes the following basic methods: MethodInationIndation Public void createConnection () Creating Connection, Connection to BD Is Redefined for each BD provider public Connection getConnection () Getting Connection public void closeConnection () Closing Connection public boolean execS'L (sql) Performing the S'L request public boolean writeBlob (table, field, pk, id, fpath) Record file in the BLOB table box public long readbBloFile (table, field, field, fpath) Extracting the BLOB object into the public boolean writeClob file (table, field, id, fpath) Record the file in the CLOB table box public readClobToFile (table, field, id, path) Extracting the CLOB object into the public String readCDlobata file (table, field, id) Reading the CLOB Settings of recording procedures and reading large objects LOB Table table name, field LOB name, primary pk field name, id id id, and way to fpath file. If necessary, you can refine these procedures and include as a parameter the name of the Schema (Oracle) scheme or database Database (MyS-L). It is possible that the primary key may contain several fields. NOTE : 1. Please note that the BLOB and CLOB fields are not overridden according to the table above. But this applies to the Oracle and MyS'L databases used in the example. That is, we can say that JDBC methods cope with the fields of LOB data providers. In addition, myS'L uses a text type, and In Oracle CLOB. 2. You can use different methods to write file content into the LOB fields, which will be demonstrated by the example of MyS'L. Below is the DAOBase.java listing, where BLOB and CLOB fields do not include code. This was done intentionally to visualize the overall structure of the DAOBase.java module. Otherwise, the listing will increase significantly and the forest will not be seen behind the trees. The code for how to work with LOB fields is described as following. Listing DAOBase.java import java.io.IOException; import java.sql.Connection; import java.sql.SQLException; import java.sql.Statement; public class DAOBase q protected Connection connection q null; protected final int BUFFER_length q 2 q 1024; private final String TABLE_DROP q DROP TABLE %s; ?'s public void to createConnection Public Connection getConnection () The public void closeion () if (connection! q null) println (e.getMessage ()@return @param); Issues / public boolean execS'L (final line sql) - boolean result - false; Try : Statement that (connection! - null) - statement - connection.createStatement (); statement.execute (sql); statement.close (); The statement is invalid; The result is true; System.err.println («TS»L : « s sql); The result of the return - /-- see zenit - zenith; (/) - public l readBlobToFile (final string table, final string table, final string field, final string pk, final int id, final string path) casts IOException, S'LException - long size No 0; // ... See the size of the refund. - /I am a public bulean letterClob (final int id, final thong-fpat) - drilling result . See zenith - zenit; -//-- see the size of the refund; I'm a public line readClobData (the final string table, Final String Field, Final int id) - StringBuffer (see zenith returns buffer.toString(); // -- //---TABLE_DROP---- //) Creation (sql) s.T.T. (sql) As a rule, the structure of the database changes rarely and there is no need to include in the application procedures that are not used. файла из БД. Зенит и Зенит - Зенит Квинси Промес читайте в материале BlobToFile (читайBlobField), Зенит частная строка INSERT_blob вставить в %s (%s) значения (%d) ; частные струнные UPDATE_blob и обновить %s набор %s ? где %s ??; частная строка SELECT_blob выбрать %s от %s, где %s и %d ; публичная boolean writeBlob (окончательный струнный стол, финальное струнное поле, окончательное string pk, final int id, final String fpath) - String sql - String.format (INSERT_blob, таблица, pk, id); boolean result - execS'L (sql); если (результат) - PreparedStatement ps - null; Файл - новый файл (fpath); попробуйте s sql и String.format (UPDATE_blob, таблица, поле, pk); FileInputStream — это новый FileInputStream (файл); ps - connection.prepareStatement (sql); ps.setBinaryStream(1, is, (int)file.length(); ps.setInt (2, id); ps.executeUpdate(); connection.commit(); ps.close(); - улов (FileNotFoundException e) - результат - ложный; e.printStackTrace (); результат возврата; (//) - частная Blob readBlobField (окончательный струнный стол, Окончательное струнное поле, окончательный String pk, окончательный int id) бросает S'LException - Струнный sql - String.format (SELECT_blob, поле, стол, pk, id); Заявление stmt - connection.createStatement(); ResultSet rs - stmt.execute'y (sql); Blob blob - нулевой; если (rs.next()) blob s rs.getBlob (1); возвращение капли; (//) - частная длинная записиFromBlob2Stream (Blob blob, OutputStream out) выбрасывает S'LException, IOException - InputStream - это blob.getBinaryStream (); int length -1; длинный размер - 0; byte ' buf - новый byte-BUFFER_length;; в то время как (длина - is.read(buf)) ! -1) - out.write(buf, 0, length; - есть.близко (); размер возврата; - публичная длинная readBlobToFile (окончательный струнный стол, финальное струнное поле, окончательный String pk, final int id, final String fpath) бросает IOException, S'LException - длинный размер No 0; OutputStream fwriter - новый FileOutputStream (fpath); Blob blob - readBlobField (таблица, поле, pk, id); размер - writeFromBlob2Stream (blob, fwriter); fwriter.close(); размер возврата; Квинси КЛОБ Read in the article of the magazine Ogonek KloboPhile, readclobData. Kinsey Promes and BufferedReader. Surab Tsereteli/TEXT, KLOB/TEXT, KLOB, zurab Tsereteli. Read in the article of the magazine Power of KLOB (readKlobField), zenith and zenith. protected line INSERT_clob and paste in %s (%s) values (%d) protected strings UPDATE_clob and update %s set %s ? where is %s ??; private line SELECT_clob select %s from %s where ID and %d; public boolean writeClob (final string table, final string field, final string pk, final int id, final String fpath) - String sql - String.format (INSERT_clob, table, pk, id); boolean result - execS'L (sql); If (result) - PreparedStatement ps - null; File - new file (fpath); Try - FileInputStream fis-new FileInputStream (file picture); InputStreamReader isr'new InputStreamReader (fis,UTF-8); BufferedReader br New BufferedReader (isr); sql and String.format (UPDATE_clob, table, field, pk); ps - connection.prepareStatement (sql); ps.setCharacterStream (1, br, (int)file.length (); ps.setInt (2, id); ps.executeUpdate (); connection.commit(); ps.close(); - catch (UnsupportedEncodingException e) - result - false; e.printStackTrace (); - catch (FileNotNotNException e) - result - e.print (/) - public long readClobToFile (final string table, final string box, final Int ID, final string fpath) casts IOException, S'LException - long size No 0; BufferedWriter fwriter-new BufferedWriter (new FileWriter (fpath)); Clob clob - readClobField (table, field, id); Size - readFromClob2Stream (clob, fwriter); fwriter.close(); Return size (/) - Private Clob readClobField (final string table, final string box, final int id) throws S'LException - String sql - String.format (SELECT_clob, field, table, id); Statement stmt - connection.createStatement(); ResultSet rs - stmt.execute'y (sql); Clob clob - null; If (rs.next)) a clob and rs.getClob (1); Return of clob; } Private long readFromClob2Stream (Clob clob, Writer out) throws S'LException, IOException - BufferedReader breader; Bakery; - new BufferedReader (clob.getCharacterStream ();int length -1; long size - 0; char'buf - new char'BUFFER_length;; while (length - breader.read (buf, 0, BUFFER_length)) length); Size - length; - breader.close (); Return size - public line readClobData (final string table, final string field, final int id) - StringBuffer buffer - new StringBuffer (); Try Clob clob readClobField (table, box, id); BufferedReader reader; reader - the new BufferedReader (clob.getCharacterStream); The charm of the buff and the new BUFFER_length; Length int -1; try while ((length'reader.read (buf,0,BUFFER_length) !- -1) if (length - BUFFER_length) buffer.append (String.valueOf (buf)); still - Tmp line - String.valueOf (buf) .substring (0, length); buffer.append (tmp); Catch (S'LException e) - e.printStackTrace - return buffer.toString - OracleDAO.java. Kinsey TABLE_blobs TABLE_files, zenith and zenith. When creating an object in the designer, the designer immediately creates a connection to the database server. The zenith is utf8. private final string DRIVER_oracle - oracle.jdbc.OracleDriver; private final string URL_oracle and jdbc:oracle:thin:'%s:%d:%s; Private final string URL_host - localhost; private final int PORT_oracle No. 1521; private final strings SCHEMA_oracle and ...; Private final string login - ...; Private final password strings - ...; private final string TABLE_blobs - TABLE CREATE blobs - ID INT PRIMARY KEY, BLOB NULL data) ; private final lines TABLE_files CREATE TABLE files, ID INT PRIMARY KEY, data Clob NULL) ; -I am a public OracleDAO () - creation Communications (); (//I) - attempt (//) @Override public emptiness Class.forName (DRIVER_oracle). Identify Connection Properties properties properties.setProperty(password , password); properties.setProperty(user , login); properties.setProperty(useUnicode , true); properties.setProperty(characterEncoding, utf8); String url = String.format(URL_oracle, URL_host, PORT_oracle, SCHEMA_oracle); connection_ora = (OracleConnection) DriverManager.getConnection(url, properties); connection_ora.setAutoCommit(false); } catch (InstantiationException e) { } catch (IllegalAccessException e) { } catch (ClassNotFoundException e) { } catch (SQLException e) { connection_ora = null; } }; It's not a good @Override one. the public boolean createTable (final String table) (table.equalsIgnoreCase (blobs)) return execS'L (TABLE_blobs); else if (table.equalsIgnoreCase (files)) return execS'L (TABLE_files); Else return false; Listing MyS'LDAO.java Listing of the MyS'LDAO.java module includes a redefined createConnection method, a createTable procedure method, and a writeClob method that demonstrates a different approach to recording a symbolic file in the TEXT box. In this method, fileInputStream file stream is immediately redirected to setAsciiStream. But in order not to distort the symbols, the appropriate encoding was used in the database table and connected to the server. The parent method of writeClob class DAOBase.java also worked without complaints. Thus, you have at your disposal 2 approaches of recording a text file in the database MyS'L. The table structures described in the TABLE_blobs constants and TABLE_files in the form of S'L scripts include only id id fields and data fields of relevant LOB types. The files table is encoded UTF8. By default, date fields have NULL values because they are used in upgrade methods. When you create an object in the designer, you immediately create a connection to the database server, which as a property is transferred the encoding of the installed connection utf8. private String DRIVER_MySQL = com.mysql.jdbc.Driver; private String jdbc:mysql://%s:%d/%; частные струнные URL_host и localhost ; частные int PORT_mysql No 3306 ; частная строка DATABASE - тест; частный струнный логин - ... ; частный струнный пароль - ... ; частные струнные TABLE_blobs - CREATE TABLE blobs; частная струнная TABLE_files - CREATE TABLE files - id INT PRIMARY KEY, - данные TEXT NULL) - ENGINE и INNODB - CHARACTER SET utf8 - CollatE utf8_general_ci;; -Я общественный MyS-LDAO () - createConnection (); //я - публичная пустота, создающаяСвязь (DRIVER_MySQL) - попробуйте @Override () // Свойства подключения - новые свойства (); properties.setProperty (пароль, пароль); properties.setProperty (пользователь, логин); properties.setProperty (использованиеУникод, правда) ; properties.setProperty (characterEncoding, utf8) ; Url-адрес строки - String.format (URL_mysql, URL_host, PORT_mysql, DATABASE); подключение - DriverManager.getConnection (url, свойства); connection.setAutoCommit (ложный); - улов (InstantiationException e) - улов (IllegalAccessException e) - улов (ClassNotFoundException e) - улов (S'LException e) - соединение нулевое; ~~~~~~ @Override public boolean createTable(final String table) { if (table.equalsIgnoreCase(blobs)) return execSQL (TABLE_blobs); else if (table.equalsIgnoreCase(files)) return execSQL (TABLE_files); else return false; } //~~~~~~ @Override public boolean writeClob(final String table, final String field, final String pk, final int id, final String fpath) { String sql = String.format(INSERT_clob_rec, table, pk, id); boolean result = execSQL (sql); if (result) { PreparedStatement ps = null; Файл - новый файл (fpath); попробуйте - FileInputStream fis - новый FileInputStream (файл); sql - String.format (UPDATE_clob_rec, таблица, поле, pk); ps - connection.prepareStatement (sql); ps.setAsciiStream (1, fis, (int) file.length ()); ps.setInt (2, id); ps.executeUpdate (); connection.commit (); ps.close (); результат возврата; Квинси Квинси, Квинси Квинси, Квинси Квинси Зураб М. Зураб М.В. Зубков Если подключение установлено, то создаются таблицы, в которые записываются и извлекаются файлы. частные TEST_mysql и ложные; частные TEST_oracle и правда; частный финальный струнный TEXT_tbl_create - Таблица ,%s; частный финальный струнный TEXT_tbl_insert - %s, вставленный в таблицу; частный финальный струнный TEXT_tbl_upload - %s (размер %d byte) загружен; частная финальная строка TEXT_tbl_drop - Таблица упала; TEST_oracle // если (TEST_mysql) testMyS'L (); - частный недействительный тестDAO (DAOBase dao, окончательный заголовок строки) - Струнный текст; если (dao.getConnection() ! System.out.println (Подключение к серверу); если (dao.createTable (blobs)) - текст - String.format (TEXT_tbl_create, blobs); System.out.println (текст); если (dao.writeBlob (blobs, данные, id, 1, aircraft.jpg)) - текст - String.format (TEXT_tbl_insert, Изображение); System.out.println (текст); попробовать - длинный размер - dao.readBlobToFile (blobs, data, id, 1, aircraft1.jpg); текст - String.format (TEXT_tbl_upload, Изображение, размер); System.out.println (текст); - улов (IOException e) - улов (S'LException e) - если (dao.writeBlob (blobs, data, id, 2, xlsx)) - текст - String.format (TEXT_tbl_insert, XLSX); System.out.println (текст); попробовать - длинный размер - dao.readBlobToFile (blobs, data, id, 2, 1.xlsx); текст - String.format (TEXT_tbl_upload, XLSX, размер); System.out.println (текст); - улов (IOException e) - улов (S'LException e) - если (dao.dropTable (blobs)) - текст - String.format (TEXT_tbl_drop, blobs); System.out.println (текст); - если (dao.createTable (файлы)) - текст - String.format (TEXT_tbl_create, файлы); System.out.println (текст); если (dao.writeClob (файлы, данные, id, 1, txt)) - текст - String.format (TEXT_tbl_insert, Файл); System.out.println (текст); попробовать - // No 1, CLOB - dao.readClobToFile (файлы, данные, 1, Зенит); текст - String.format (TEXT_tbl_upload, Файл, размер); System.out.println (текст); поймать (IOException e) - поймать (S'LException e) - если (dao.dropTable (файлы)) - текст - String.format (TEXT_tbl_drop, файлы); System.out.println (текст); - dao.closeConnection(); - еще System.out.println (Соединение - NULL); - частный недействительный тестMyS'L () - DAOBase dao - новый MyS'LDAO (); testDAO (dao, ----- TEST MyS'L -----); - частный тест пустотыOracle() - DAOBase dao - новый OracleDAO (); testDAO (dao, ----- TEST Oracle -----) System.exit(0); } Результаты тестирования Приложение выводит сообщения и результатах тестирования методов записи и чтения LOB-объектов в консоль. ----- TEST Oracle ----- Connection to server Table <blobs> created Image inserted into table Image (size 3573 byte) uploaded XLSX inserted into table XLSX (size 9929 byte) uploaded Table <blobs> dropped Table <files> created File inserted into table File (size 2191 byte) uploaded Table <files> dropped Скачать пример Исходный код рассмотренного примера записи и чтения больших объектов LOB с использованием JDBC можно скачать здесь (7.27 Мб). При тестировании примера необходимо определить параметры подключения к Вашему серверу БД - схема (база данных), логин и пароль. пароль. </files></files></blobs></blobs> javascript write blob to file. write file to azure blob storage java. oracle write blob to file java. write java.sql.blob to file