```
//--------------------------------------------------------------------
--------
//BLOBFileExample.java
//--------------------------------------------------------------------
--------

/*
 *
=====================================================================
======
 * Copyright (c) 1998-2011 Jeffrey M. Hunter. All rights reserved.
 *
 * All source code and material located at the Internet address of
 * http://www.idevelopment.info is the copyright of Jeffrey M. Hunter
and
 * is protected under copyright laws of the United States. This source
code may
 * not be hosted on any other site without my express, prior, written
 * permission. Application to host any of the material elsewhere can be
made by
 * contacting me at jhunter@idevelopment.info.
 *
 * I have made every effort and taken great care in making sure that
the source
 * code and other content included on my web site is technically
accurate, but I
 * disclaim any and all responsibility for any loss, damage or
destruction of
 * data or any other property which may arise from relying on it. I
will in no
 * case be liable for any monetary damages arising from such loss,
damage or
 * destruction.
 *
 * As with any code, ensure to test this code in a development
environment
 * before attempting to run it in production.
 *
=====================================================================
======
 */

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
//Needed since we will be using Oracle's BLOB, part of Oracle's JDBC
extended
//classes. Keep in mind that we could have included Java's JDBC
interfaces
```

```
//java.sql.Blob which Oracle does implement. The oracle.sql.BLOB class
//provided by Oracle does offer better performance and functionality.
//Needed for Oracle JDBC Extended Classes

import oracle.sql.*;
import oracle.jdbc.*;

/**
 * ------------------------------------------------------------------------
---------
 * Used to test the functionality of how to load and unload binary data
from an
 * Oracle BLOB.
 *
 * This example uses an Oracle table with the following definition:
 *
 * CREATE TABLE test_blob ( id NUMBER(15) , image_name VARCHAR2(1000) ,
image
 * BLOB , timestamp DATE );
 * ------------------------------------------------------------------------
---------
 *
 * @version 1.0
 * @author Jeffrey M. Hunter (jhunter@idevelopment.info)
 * @author http://www.idevelopment.info
 *         ----------------------------------------
 *         ------------------------------------
 */

public class BLOBFileExample {

        private String inputBinaryFileName = null;
        private File inputBinaryFile = null;

        private String outputBinaryFileName1 = null;
        private File outputBinaryFile1 = null;

        private String outputBinaryFileName2 = null;
        private File outputBinaryFile2 = null;

        private String dbUser = "aw";
        private String dbPassword = "aw";
        private Connection conn = null;

        /**
         * Default constructor used to create this object. Responsible
for setting
         * this object's creation date, as well as incrementing the
number instances
         * of this object.
         *
         * @param args
         *            Array of string arguments passed in from the
command-line.
         * @throws java.io.IOException
         */
        public BLOBFileExample(String[] args) throws IOException {
```

```
                inputBinaryFileName = args[0];
                inputBinaryFile = new File(inputBinaryFileName);

                if (!inputBinaryFile.exists()) {
                        throw new IOException("File not found. " +
inputBinaryFileName);
                }

                outputBinaryFileName1 = inputBinaryFileName +
".getBytes.out";
                outputBinaryFileName2 = inputBinaryFileName +
".Streams.out";

        }

        /**
         * Obtain a connection to the Oracle database.
         *
         * @throws java.sql.SQLException
         */
        public void openOracleConnection() throws SQLException,
                        IllegalAccessException, InstantiationException,
                        ClassNotFoundException {

                String driver_class = "oracle.jdbc.driver.OracleDriver";
                String connectionURL = null;

                try {
                        Class.forName(driver_class).newInstance();
                        connectionURL =
"jdbc:oracle:thin:@SOMEURL:1521:xe";

                        conn = DriverManager.getConnection(connectionURL,
dbUser,
                                        dbPassword);
                        conn.setAutoCommit(false);
                        System.out.println("Connected.\n");
                } catch (IllegalAccessException e) {
                        System.out.println("Illegal Access Exception:
(Open Connection).");
                        e.printStackTrace();
                        throw e;
                } catch (InstantiationException e) {
                        System.out.println("Instantiation Exception:
(Open Connection).");
                        e.printStackTrace();
                        throw e;
                } catch (ClassNotFoundException e) {
                        System.out.println("Class Not Found Exception:
(Open Connection).");
                        e.printStackTrace();
                        throw e;
                } catch (SQLException e) {
                        System.out.println("Caught SQL Exception: (Open
Connection).");
                        e.printStackTrace();
```

```
                        throw e;
                }

        }

        /**
         * Close Oracle database connection.
         *
         * @throws java.sql.SQLException
         */
        public void closeOracleConnection() throws SQLException {

                try {
                        conn.close();
                        System.out.println("Disconnected.\n");
                } catch (SQLException e) {
                        System.out.println("Caught SQL Exception:
(Closing Connection).");
                        e.printStackTrace();
                        if (conn != null) {
                                try {
                                        conn.rollback();
                                } catch (SQLException e2) {
                                        System.out
                                                        .println("Caught
SQL (Rollback Failed) Exception.");
                                        e2.printStackTrace();
                                }
                        }
                        throw e;
                }

        }

        /**
         * Method used to print program usage to the console.
         */
        static public void usage() {
                System.out
                                .println("\nUsage: java BLOBFileExample
\"Binary File Name\"\n");
        }

        /**
         * Validate command-line arguments to this program.
         *
         * @param args
         *            Array of string arguments passed in from the
command-line.
         * @return Boolean - value of true if correct arguments, false
otherwise.
         */
        static public boolean checkArguments(String[] args) {

                if (args.length == 1) {
                        return true;
                } else {
```

```
                        return false;
                }

        }

        /**
         * Override the Object toString method. Used to print a version
of this
         * object to the console.
         *
         * @return String - String to be returned by this object.
         */
        public String toString() {

                String retValue;

                retValue = "Input File         : " + inputBinaryFileName
+ "\n"
                                + "Output File (1)    : " +
outputBinaryFileName1 + "\n"
                                + "Output File (2)    : " +
outputBinaryFileName2 + "\n"
                                + "Database User      : " + dbUser;
                return retValue;

        }

        /**
         * Method used to write binary data contained in a file to an
Oracle BLOB
         * column. The method used to write the data to the BLOB uses
the putBytes()
         * method. This is one of two types of methods used to write
binary data to
         * a BLOB column. The other method uses Streams.
         *
         * @throws java.io.IOException
         * @throws java.sql.SQLException
         */
        public void writeBLOBPut() throws IOException, SQLException {

                FileInputStream inputFileInputStream = null;
                String sqlText = null;
                Statement stmt = null;
                ResultSet rset = null;
                BLOB image = null;
                int chunkSize;
                byte[] binaryBuffer;
                long position;
                int bytesRead = 0;
                int bytesWritten = 0;
                int totbytesRead = 0;
                int totbytesWritten = 0;

                try {

                        stmt = conn.createStatement();
```

```
                        inputBinaryFile = new File(inputBinaryFileName);
                        inputFileInputStream = new
FileInputStream(inputBinaryFile);

                        sqlText = "INSERT INTO test_blob (fileblobid,
image_name, image, timestamp) "
                                        + "   VALUES(33, '"
                                        + inputBinaryFile.getName()
                                        + "', EMPTY_BLOB(), SYSDATE)";
                        stmt.executeUpdate(sqlText);

                        sqlText = "SELECT image " + "FROM   test_blob " +
"WHERE  fileblobid = 33 "
                                        + "FOR UPDATE";
                        rset = stmt.executeQuery(sqlText);
                        rset.next();
                       image = ((OracleResultSet) rset).getBLOB("image");

                        chunkSize = image.getChunkSize();
                        binaryBuffer = new byte[chunkSize];

                        position = 1;
                        while ((bytesRead =
inputFileInputStream.read(binaryBuffer)) != -1) {
                                bytesWritten = image
                                                .putBytes(position,
binaryBuffer, bytesRead);
                                position += bytesRead;
                                totbytesRead += bytesRead;
                                totbytesWritten += bytesWritten;
                        }

                        inputFileInputStream.close();

                        conn.commit();
                        rset.close();
                        stmt.close();

                        System.out

        .println("=========================================================
====\n"
                                                        + "  PUT METHOD\n"
                                                        +
"============================================================\n"
                                                        + "Wrote file "
                                                        +
inputBinaryFile.getName()
                                                        + " to BLOB
column.\n"
                                                        + totbytesRead
                                                        + " bytes read.\n"
                                                        + totbytesWritten
                                                        + " bytes
written.\n");
```

```
                } catch (IOException e) {
                        System.out
                                        .println("Caught I/O Exception:
(Write BLOB value - Put Method).");
                        e.printStackTrace();
                        throw e;
                } catch (SQLException e) {
                        System.out
                                        .println("Caught SQL Exception:
(Write BLOB value - Put Method).");
                        System.out.println("SQL:\n" + sqlText);
                        e.printStackTrace();
                        throw e;
                }

        }

        /**
         * Method used to write the contents (data) from an Oracle BLOB
column to an
         * O/S file. This method uses one of two ways to get data from
the BLOB
         * column - namely the getBytes() method. The other way to read
data from an
         * Oracle BLOB column is to use Streams.
         *
         * @throws java.io.IOException
         * @throws java.sql.SQLException
         */
        public void readBLOBToFileGet() throws IOException, SQLException
{

                FileOutputStream outputFileOutputStream = null;
                String sqlText = null;
                Statement stmt = null;
                ResultSet rset = null;
                BLOB image = null;
                long blobLength;
                long position;
                int chunkSize;
                byte[] binaryBuffer;
                int bytesRead = 0;
                int bytesWritten = 0;
                int totbytesRead = 0;
                int totbytesWritten = 0;

                try {

                        stmt = conn.createStatement();

                        outputBinaryFile1 = new
File(outputBinaryFileName1);
                        outputFileOutputStream = new
FileOutputStream(outputBinaryFile1);

                        sqlText = "SELECT image " + "FROM   test_blob " +
"WHERE  fileblobid = 33 "
```

```
                                    + "FOR UPDATE";
                        rset = stmt.executeQuery(sqlText);
                        rset.next();
                       image = ((OracleResultSet) rset).getBLOB("image");

                        blobLength = image.length();
                        chunkSize = image.getChunkSize();
                        binaryBuffer = new byte[chunkSize];

                        for (position = 1; position <= blobLength;
position += chunkSize) {

                                // Loop through while reading a chunk of
data from the BLOB
                                // column using the getBytes() method.
This data will be stored
                                // in a temporary buffer that will be
written to disk.
                                bytesRead = image.getBytes(position,
chunkSize, binaryBuffer);

                                // Now write the buffer to disk.
                               outputFileOutputStream.write(binaryBuffer,
0, bytesRead);

                                totbytesRead += bytesRead;
                                totbytesWritten += bytesRead;

                        }

                        outputFileOutputStream.close();

                        conn.commit();
                        rset.close();
                        stmt.close();

                        System.out

        .println("=========================================================
====\n"
                                                        + "  GET METHOD\n"
                                                        +
"============================================================\n"
                                                        + "Wrote BLOB
column data to file "
                                                        +
outputBinaryFile1.getName()
                                                        + ".\n"
                                                        + totbytesRead
                                                        + " bytes read.\n"
                                                        + totbytesWritten
                                                        + " bytes
written.\n");

                } catch (IOException e) {
                        System.out
```

```
                                        .println("Caught I/O Exception:
(Write BLOB value to file - Get Method).");
                        e.printStackTrace();
                        throw e;
                } catch (SQLException e) {
                        System.out
                                        .println("Caught SQL Exception:
(Write BLOB value to file - Get Method).");
                        System.out.println("SQL:\n" + sqlText);
                        e.printStackTrace();
                        throw e;
                }

        }

        /**
         * Method used to write binary data contained in a file to an
Oracle BLOB
         * column. The method used to write the data to the BLOB uses
Streams. This
         * is one of two types of methods used to write binary data to a
BLOB
         * column. The other method uses the putBytes() method.
         *
         * @throws java.io.IOException
         * @throws java.sql.SQLException
         */
        public void writeBLOBStream() throws IOException, SQLException {

                FileInputStream inputFileInputStream = null;
                OutputStream blobOutputStream = null;
                String sqlText = null;
                Statement stmt = null;
                ResultSet rset = null;
                BLOB image = null;
                int bufferSize;
                byte[] byteBuffer;
                int bytesRead = 0;
                int bytesWritten = 0;
                int totBytesRead = 0;
                int totBytesWritten = 0;

                try {

                        stmt = conn.createStatement();

                        inputBinaryFile = new File(inputBinaryFileName);
                        inputFileInputStream = new
FileInputStream(inputBinaryFile);

                        sqlText = "INSERT INTO test_blob (fileblobid,
image_name, image, timestamp) "
                                        + "   VALUES(34, '"
                                        + inputBinaryFile.getName()
                                        + "', EMPTY_BLOB(), SYSDATE)";
                        stmt.executeUpdate(sqlText);
```

```
                        sqlText = "SELECT image " + "FROM   test_blob " +
"WHERE  fileblobid = 34 "
                                        + "FOR UPDATE";
                        rset = stmt.executeQuery(sqlText);
                        rset.next();
                       image = ((OracleResultSet) rset).getBLOB("image");

                        bufferSize = image.getBufferSize();

                        // Notice that we are using an array of bytes.
This is required
                        // since we will be streaming the content (to
either a CLOB or BLOB)
                        // as a stream of bytes using an OutputStream
Object. This requires
                        // that a byte array to be used to temporarily
store the contents
                        // that will be sent to the LOB. Note that the
use of the byte
                        // array can be used even if we were reading
contents from an
                        // ASCII text file that would be sent to a CLOB.
                        byteBuffer = new byte[bufferSize];

                        blobOutputStream = image.getBinaryOutputStream();

                        while ((bytesRead =
inputFileInputStream.read(byteBuffer)) != -1) {

                                // After reading a buffer from the binary
file, write the
                                // contents
                                // of the buffer to the output stream
using the write()
                                // method.
                                blobOutputStream.write(byteBuffer, 0,
bytesRead);


                                totBytesRead += bytesRead;
                                totBytesWritten += bytesRead;

                        }

                        // Keep in mind that we still have the stream
open. Once the stream
                        // gets open, you cannot perform any other
database operations
                        // until that stream has been closed. This even
includes a COMMIT
                        // statement. It is possible to loose data from
the stream if this
                        // rule is not followed. If you were to attempt
to put the COMMIT in
                        // place before closing the stream, Oracle will
raise an
                        // "ORA-22990: LOB locators cannot span
transactions" error.
```

```
                        inputFileInputStream.close();
                        blobOutputStream.close();

                        conn.commit();
                        rset.close();
                        stmt.close();

                        System.out

        .println("============================================================
====\n"
                                                        + "  OUTPUT
STREAMS METHOD\n"
                                                        +
"============================================================\n"
                                                        + "Wrote file "
                                                        +
inputBinaryFile.getName()
                                                        + " to BLOB
column.\n"
                                                        + totBytesRead
                                                        + " bytes read.\n"
                                                        + totBytesWritten
                                                        + " bytes
written.\n");

                } catch (IOException e) {
                        System.out
                                                .println("Caught I/O Exception:
(Write BLOB value - Stream Method).");
                        e.printStackTrace();
                        throw e;
                } catch (SQLException e) {
                        System.out
                                                .println("Caught SQL Exception:
(Write BLOB value - Stream Method).");
                        System.out.println("SQL:\n" + sqlText);
                        e.printStackTrace();
                        throw e;
                }

        }

        /**
         * Method used to write the contents (data) from an Oracle BLOB
column to an
         * O/S file. This method uses one of two ways to get data from
the BLOB
         * column - namely using Streams. The other way to read data
from an Oracle
         * BLOB column is to use getBytes() method.
         *
         * @throws java.io.IOException
         * @throws java.sql.SQLException
         */
```

```
        public void readBLOBToFileStream() throws IOException,
SQLException {

                FileOutputStream outputFileOutputStream = null;
                InputStream blobInputStream = null;
                String sqlText = null;
                Statement stmt = null;
                ResultSet rset = null;
                BLOB image = null;
                int chunkSize;
                byte[] binaryBuffer;
                int bytesRead = 0;
                int bytesWritten = 0;
                int totBytesRead = 0;
                int totBytesWritten = 0;

                try {

                        stmt = conn.createStatement();

                        outputBinaryFile2 = new
File(outputBinaryFileName2);
                        outputFileOutputStream = new
FileOutputStream(outputBinaryFile2);

                        sqlText = "SELECT image " + "FROM   test_blob " +
"WHERE  fileblobid = 34 "
                                        + "FOR UPDATE";
                        rset = stmt.executeQuery(sqlText);
                        rset.next();
                       image = ((OracleResultSet) rset).getBLOB("image");

                        // Will use a Java InputStream object to read
data from a BLOB (can
                        // also be used for a CLOB) object. In this
example, we will use an
                        // InputStream to read data from a BLOB.
                        blobInputStream = image.getBinaryStream();

                        chunkSize = image.getChunkSize();
                        binaryBuffer = new byte[chunkSize];

                        while ((bytesRead =
blobInputStream.read(binaryBuffer)) != -1) {

                                // Loop through while reading a chunk of
data from the BLOB
                                // column using an InputStream. This data
will be stored
                                // in a temporary buffer that will be
written to disk.
                               outputFileOutputStream.write(binaryBuffer,
0, bytesRead);

                                totBytesRead += bytesRead;
                                totBytesWritten += bytesRead;
```

```
                        }

                        outputFileOutputStream.close();
                        blobInputStream.close();

                        conn.commit();
                        rset.close();
                        stmt.close();

                        System.out

        .println("=============================================================
====\n"
                                                        + "  INPUT STREAMS
METHOD\n"
                                                        +
"============================================================\n"
                                                        + "Wrote BLOB
column data to file "
                                                        +
outputBinaryFile2.getName()
                                                        + ".\n"
                                                        + totBytesRead
                                                        + " bytes read.\n"
                                                        + totBytesWritten
                                                        + " bytes
written.\n");

                } catch (IOException e) {
                        System.out
                                                .println("Caught I/O Exception:
(Write BLOB value to file - Streams Method).");
                        e.printStackTrace();
                        throw e;
                } catch (SQLException e) {
                        System.out
                                                .println("Caught SQL Exception:
(Write BLOB value to file - Streams Method).");
                        System.out.println("SQL:\n" + sqlText);
                        e.printStackTrace();
                        throw e;
                }

        }

        /**
         * Sole entry point to the class and application.
         *
         * @param args
         *             Array of string arguments passed in from the
command-line.
         */
        public static void main(String[] args) {

                BLOBFileExample blobFileExample = null;

                if (checkArguments(args)) {
```

```
                        try {

                                blobFileExample = new
BLOBFileExample(args);

                                System.out.println("\n" + blobFileExample
+ "\n");

                                blobFileExample.openOracleConnection();

                                blobFileExample.writeBLOBPut();
                                blobFileExample.readBLOBToFileGet();

                                blobFileExample.writeBLOBStream();
                                blobFileExample.readBLOBToFileStream();

                                blobFileExample.closeOracleConnection();

                        } catch (IllegalAccessException e) {
                                System.out
                                                .println("Caught Illegal
Accecss Exception. Exiting.");
                                e.printStackTrace();
                                System.exit(1);
                        } catch (InstantiationException e) {
                                System.out.println("Instantiation
Exception. Exiting.");
                                e.printStackTrace();
                                System.exit(1);
                        } catch (ClassNotFoundException e) {
                                System.out.println("Class Not Found
Exception. Exiting.");
                                e.printStackTrace();
                                System.exit(1);
                        } catch (SQLException e) {
                                System.out.println("Caught SQL Exception.
Exiting.");
                                e.printStackTrace();
                                System.exit(1);
                        } catch (IOException e) {
                                System.out.println("Caught I/O Exception.
Exiting.");
                                e.printStackTrace();
                                System.exit(1);
                        }

                } else {
                       System.out.println("\nERROR: Invalid arguments.");
                        usage();
                        System.exit(1);
                }

                System.exit(0);
        }

}
```