**Java code to insert data from CSV**

import java.io.\*;

import java.sql.\*;

import org.supercsv.cellprocessor.Optional;

import org.supercsv.cellprocessor.ParseDouble;

import org.supercsv.cellprocessor.constraint.NotNull;

import org.supercsv.cellprocessor.ift.CellProcessor;

import org.supercsv.io.CsvBeanReader;

import org.supercsv.io.ICsvBeanReader;

import org.supercsv.prefs.CsvPreference;

public class ComplexCsv2DbInserter {

    public static void main(String[] args) {

        String jdbcURL = "jdbc:mysql://localhost:3306/sales";

        String username = "user";

        String password = "password";

        String csvFilePath = "Reviews-complex.csv";

        int batchSize = 20;

        Connection connection = null;

        ICsvBeanReader beanReader = null;

        CellProcessor[] processors = new CellProcessor[] {

                new NotNull(), // course name

                new NotNull(), // student name

                new ParseTimestamp(), // timestamp

                new ParseDouble(), // rating

                new Optional()// comment

        };

        try {

            long start = System.currentTimeMillis();

            connection = DriverManager.getConnection(jdbcURL, username, password);

            connection.setAutoCommit(false);

            String sql = "INSERT INTO review (course\_name, student\_name, timestamp, rating, comment) VALUES (?, ?, ?, ?, ?)";

            PreparedStatement statement = connection.prepareStatement(sql);

            beanReader = new CsvBeanReader(new FileReader(csvFilePath),

                    CsvPreference.STANDARD\_PREFERENCE);

            beanReader.getHeader(true); // skip header line

            String[] header = {"courseName", "studentName", "timestamp", "rating", "comment"};

            Review bean = null;

            int count = 0;

            while ((bean = beanReader.read(Review.class, header, processors)) != null) {

                String courseName = bean.getCourseName();

                String studentName = bean.getStudentName();

                Timestamp timestamp = bean.getTimestamp();

                double rating = bean.getRating();

                String comment = bean.getComment();

                statement.setString(1, courseName);

                statement.setString(2, studentName);

                statement.setTimestamp(3, timestamp);

                statement.setDouble(4, rating);

                statement.setString(5, comment);

                statement.addBatch();

                if (count % batchSize == 0) {

                    statement.executeBatch();

                }

            }

            beanReader.close();

            // execute the remaining queries

            statement.executeBatch();

            connection.commit();

            connection.close();

            long end = System.currentTimeMillis();

            System.out.println("Execution Time: " + (end - start));

        } catch (IOException ex) {

            System.err.println(ex);

        } catch (SQLException ex) {

            ex.printStackTrace();

            try {

                connection.rollback();

            } catch (SQLException e) {

                e.printStackTrace();

            }

        }

    }

}

<dependency>

    <groupId>net.sf.supercsv</groupId>

    <artifactId>super-csv</artifactId>

    <version>2.4.0</version>

</dependency>

|  |  |
| --- | --- |
| import java.sql.Timestamp;    public class Review {      private String courseName;      private String studentName;      private Timestamp timestamp;      private double rating;      private String comment;          // constructors...        // getters and setters...    } | |
|  | import java.sql.Timestamp;    import org.supercsv.cellprocessor.CellProcessorAdaptor;  import org.supercsv.cellprocessor.ift.CellProcessor;  import org.supercsv.util.CsvContext;    public class ParseTimestamp extends CellProcessorAdaptor {          public ParseTimestamp() {          super();      }        public ParseTimestamp(CellProcessor next) {          super(next);      }        @Override      public Object execute(Object value, CsvContext context) {          return Timestamp.valueOf((String) value);      }  } |