IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. IV (Sep – Oct. 2014), PP 58-65
www.iosrjournals.org

Design and implementation of massive MYSQL data intelligent export system to excel by using Apache –POI libraries

Kamalakant L Bawankule, N B. Raut
M.Tech 4th (CSE) Student, N.U.V.A. College of Engineering, Nagpur
Assistant Prof., Gurunanak College of Engineering, Nagpur

Abstract: *To export nearly 1,0000 attribute from database to excel of 10-20 column information of students and papers of database format into the target Excel sheet, we developed the intelligent export system which distinguished the simple export system. The system uses Netbeans/Eclipse as the development tool, object-oriented language java as the programming language, MySQL as the background database, POI-HSSF as the apache Library for reading & writing data from excel, used AWT and SWING for design of an application ,and uses the mysql-connector as connection technology to database. The system realizes to export the massive data of database into the Excel format intellectually that convenient for integrating and managing the multiple data required for regular examination process in the institutes and universities.*
HSSF is the POI Project's pure Java implementation of the Excel '97(-2007) file format. XSSF is the POI Project's pure Java implementation of the Excel 2007 OOXML (.xlsx) file format.HSSF and XSSF provide ways to read spreadsheets create, modify, read and write XLS spreadsheets.
Index Terms: *Database, JAVA, POI-HSSF, HSSF, MySQL, XSSF.*

I. Introduction

Today in many industries employee data, salary data, purchase data and etc is present in database. To create reports, manipulate and update data we need to search data in the database. Data analysis in database is very tedious, while generating report and formats. Many industries also has their old data in database so they need an application to export data from database (MySQL) to excel database (MySQL).Industry should be able to reuse their old data. POIFSFileSystem is the complete file system which will be used to handle excel files.FileInputStream is used to load the excel file to the application .Data export application has facility to read data from database and export it into database (MySQL).Excel data is in table format same as database, apache library functions are used to read each column in the database and store data into Spreadsheet.

The data in vector or an array is compiled in a single vector. Direct export of data from database to the excel spreadsheet is done through connecting to mysql.Thier is no limit of writing data massive data can be read through database and can be write into Excel spreadsheet. Exporting data from database will help to generate report and make it easy way to analyze data. Data exported can be read, write and can be updated if needed. It becomes very much easy way for exporting data with apache library HSSF.

II. Apache Poi-Hssf

1) HSSF is the POI Project's pure Java implementation of the Excel '97(-2007) file format. XSSF is the POI Project's pure Java implementation of the Excel 2007 OOXML (.xlsx) file format.
2) HSSF and XSSF provide ways to write spreadsheets create, modify, read and write XLS spreadsheets. They provide:

- low level structures for those with special needs
- an event model api for efficient read-only access
- a full usermodel api for creating, reading and modifying XLS files

3) An alternate way of generating a spreadsheet is via the Cocoon serializer (yet you'll still be using HSSF indirectly). With Cocoon you can serialize any XML datasource (which might be a ESQL page outputting in SQL for instance) by simply applying the stylesheet and designating the serializer.
4) If you're merely reading spreadsheet data, then use the eventmodel api in either the org.apache.poi.hssf.eventusermodel package, or the org.apache.poi.xssf.eventusermodel package, depending on your file format.
5) If you're modifying spreadsheet data then use the usermodel api. You can also generate spreadsheets this way.Note that the usermodel system has a higher memory footprint than the low level eventusermodel, but have

the major advantage of being much simpler to work with. Also please be aware that as the new XSSF supported Excel 2007 OOXML (.xlsx) files are XML based, the memory footprint for processing them is higher than for the older HSSF supported (.xls) binary files.

6) SXSSF is an API-compatible streaming extension of XSSF to be used when very large spreadsheets have to be produced, and heap space is limited. SXSSF achieves its low memory footprint by limiting access to the rows that are within a sliding window, while XSSF gives access to all rows in the document. Older rows that are no longer in the window become inaccessible, as they are written to the disk.

In auto-flush mode the size of the access window can be specified, to hold a certain number of rows in memory. When that value is reached, the creation of an additional row causes the row with the lowest index to to be removed from the access window and written to disk. Or, the window size can be set to grow dynamically; it can be trimmed periodically by an explicit call to flushRows(int keepRows) as needed.

Due to the streaming nature of the implementation, there are the following limitations when compared to XSSF:

- Only a limited number of rows are accessible at a point in time.
- Sheet.clone() is not supported.
- Formula evaluation is not supported

	HSSF(.XLS)		**XSSF(.XLSX)**		
	event model	**userm odel**	**event model**	**userm odel**	**SXSSF**
API Type	**Strea ming**	**In memor y tree**	**Strea ming**	**In memor y tree**	**Buffered Streamin g**
CPU and Memory Efficency	Good	Varies	Good	Varies	Good
Forward Only	Yes	No	Yes	No	Yes
Read Files	Yes	Yes	Yes	Yes	No
Write Files	No	Yes	No	Yes	Yes
Features :					
create sheets/rows/c ells	No	Yes	No	Yes	Yes
styling cells	No	Yes	No	Yes	Yes
delete sheets/rows/c ells	No	Yes	No	Yes	No
shift rows	No	Yes	No	Yes	No
doning sheets	No	Yes	No	Yes	No
formula evalution	No	Yes	No	Yes	No
cell comments	No	Yes	No	Yes	No
pictures	No	Yes	No	Yes	Yes

III. Database (MySQL)

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack (and other 'AMP' stacks). LAMP is an acronym for "Linux, Apache, MySQL, and Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

Database has to be maintained by the system to store information about students, invigilators, class rooms etc. The various updates must be saved and stored in the database of the system. Therefore MySQL, relational database that can handle large amount of data on relatively cheap hardware has been used.

All the reports, formats will be made available on a single button click. The automated system helps to save the time and the laborious work. MySQL is the world's most used open source relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases.

The SQL phrase stands for Structured Query Language. Universities, internet service providers and nonprot organizations are the main users of MySQL, mainly because of its price. Free software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available and over additional functionality.

IV. Design and Implementation

I] Design

Create a blank Excel workbook which is in the form of table. Create object of HSSFWorkbook, HSSFSheet and row Iterator which will help to write in the excel sheet present in the workbook. Create object of HSSFRow, and cell Iterator to read excel sheet row by row and each cell in each row.

Declare an object of class TreeMap class to map the values in particular row and column in the excel spreadsheet. Mapping values from database to the excel need to connect to the database. Connect with database and read the particular date and time and particular data which we want to export to the excel sheet.

Once the data collected in the particular vectors after reading the particular data from the database write/export the massive amount of data with the help of Apache –Poi function. Exporting data can be done with the help of put function of Map class. In the below given implementation we need to iterate the complete excel sheet to put the data with the help of Keyset object.

Ii] Implementation/Code

```
import com.mysql.jdbc.Connection;
import com.mysql.jdbc.ResultSet;
import com.mysql.jdbc.Statement;
import java.io.File;
import java.io.FileOutputStream;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;

public class MastersFile extends javax.swing.JFrame {

    Vector seatno = new Vector(0);
    Statement pst1 = null;
    ResultSet rs1 = null;
    Statement pst = null;
    ResultSet rs = null;
    String roomno;
    String dates;
    String timesel;
    String instcode;
    String coursecode;
    String yearcode;
```

```
    String mastercode;
    String subjcode;
    String subjabbr;
    String seatfrom;
    String seatto;
    String total;
    String ctotal;
    int counter = 0;
    String date, time;
    //Blank workbook
    HSSFWorkbook workbook = new HSSFWorkbook();
    //Create a blank sheet
     HSSFSheet sheet = workbook.createSheet("MASTERSEATING");

     public MastersFile() throws SQLException, ClassNotFoundException {
        initComponents();
        jProgressBar1.setVisible(false);
        //Fetch Data for Combo Box from DB
        String[] columnNameroom = columName_room("date", 1);
        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(columnNameroom));
        String[] columnNameroom2 = columName_room("time", 1);
        jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(columnNameroom2));
jComboBox1.setSelectedIndex(-1);
        jComboBox2.setSelectedIndex(-1);
        //Fetch Date and tme from combo box
    }
    public void CreateExcel() throws ClassNotFoundException {
        date = (String) jComboBox1.getSelectedItem();
        time = (String) jComboBox2.getSelectedItem();
        //Blank workbook
        HSSFWorkbook workbook = new HSSFWorkbook();

        //Create a blank sheet
        HSSFSheet sheet = workbook.createSheet("MASTERSEATING");

        //This data needs to be written (Object[])
        Map<String, Object[]> data = new TreeMap<String, Object[]>();
        data.put(Integer.toString(counter), new Object[]{"Dates", "Time", "MASTERSEATING CHART"});
        counter++;
        data.put(Integer.toString(counter), new Object[]{"ROOMNO", "INSTCODE", "COURSECODE",
"PAPERCODE", "FROM", "TO", "TOTAL", "CLASSTOTAL"});
        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con1 = null;
            con1 = (Connection) DriverManager.getConnection("jdbc:mysql://localhost/examination", "root",
"root");
            String query = "(SELECT * FROM masterseating where Date='" + date + "' AND Time='" + time + "')";
            pst = (Statement) con1.createStatement();
            rs = (ResultSet) pst.executeQuery(query);
            while (rs.next()) {
                counter++;
                roomno = rs.getString(3);
                System.out.println(roomno);
                instcode = Integer.toString(rs.getInt(4));
                coursecode = rs.getString(5);
                yearcode = Integer.toString(rs.getInt(6));
                mastercode = rs.getString(7);
```

```
        subjcode = rs.getString(8);
        subjabbr = rs.getString(9);
        seatfrom = Integer.toString(rs.getInt(10));
        seatto = Integer.toString(rs.getInt(11));
        total = rs.getString(12);
        ctotal = rs.getString(13);
        String code = coursecode + yearcode + mastercode;
        System.out.println("Data Collected");
              data.put(Integer.toString(counter), new Object[]{roomno, instcode, code, subjcode, seatfrom,
seatto, total, ctotal});
        //Iterate over data and write to sheet
        Set<String> keyset = data.keySet();
        int rownum = 0;
        for (String key : keyset) {
           Row row = sheet.createRow(rownum++);
           Object[] objArr = data.get(key);
           int cellnum = 0;
           for (Object obj : objArr) {
              Cell cell = row.createCell(cellnum++);
              if (obj instanceof String) {
                 cell.setCellValue((String) obj);
              } else if (obj instanceof Integer) {
                 cell.setCellValue((Integer) obj);
              }
           }
        }
        try {
           //Write the workbook in file system
                     FileOutputStream out = new FileOutputStream(new
       File("C:/Users/Documents/NetBeansProjects/ReadExel/src/WriteExcel6.xls"));
           workbook.write(out);
           out.close();
        } catch (Exception e) {
           e.printStackTrace();
        }
        }
       } catch (SQLException ex) {
              Logger.getLogger(MasterClasssromplan.class.getName()).log(Level.SEVERE, null, ex);
    }
     }

    public String[] columName_room(String colName, int index) throws SQLException,
    ClassNotFoundException {
     Vector str = new Vector(0);
     str.removeAllElements();
     Class.forName("com.mysql.jdbc.Driver");
com.mysql.jdbc.Connection con1 = (com.mysql.jdbc.Connection)
DriverManager.getConnection("jdbc:mysql://localhost/examination", "root", "root");
String query1 = "(SELECT DISTINCT " + colName + " FROM masterseating)";
pst1 = (Statement) con1.createStatement();
rs1 = (ResultSet) pst1.executeQuery(query1);
while (rs1.next()) {
    String accno2 = rs1.getString(index);
    str.addElement(accno2);
}
int length = str.capacity();
String arrayinst[] = new String[length];
```

```
    str.copyInto(arrayinst);
    return arrayinst;
}

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String arg = evt.getActionCommand();
    if (arg.equals("Cancel")) {
        System.exit(0);
    }
}

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    String arg = evt.getActionCommand();
    if (arg.equals("Download")) {
        try {
            jProgressBar1.setVisible(true);
            CreateExcel();
     JOptionPane.showMessageDialog(null, "FileSaved:");
            jProgressBar1.setVisible(false);
                JOptionPane.showMessageDialog(null, "File Location :C:/Users/govt. poly
    sakoli/Documents/NetBeansProjects/ReadExel/src/WriteExcel6.xls");
            System.exit(0);
        } catch (ClassNotFoundException ex) {
                    Logger.getLogger(MastersFile.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

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

   java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                new MastersFile().setVisible(true);
            } catch (SQLException ex) {
                    Logger.getLogger(MastersFile.class.getName()).log(Level.SEVERE, null, ex);
            } catch (ClassNotFoundException ex) {
                    Logger.getLogger(MastersFile.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });
}
// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JComboBox jComboBox2;
```

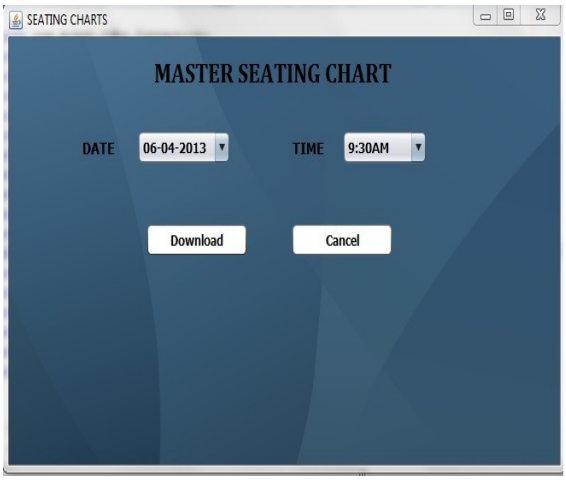
```
    private javax.swing.JDesktopPane jDesktopPane1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JProgressBar jProgressBar1;
    // End of variables declaration
}
```

V. System design

The below image shows the application system developed for the massive excel data intelligent export system to Excel Spreadsheet by using apache POI libraries.

The data read from the database is been mapped to excel cells with the help of use of Apache –POI library. After reading the data it is being exported to Excel Spreadsheet.

Application is very much useful during the massive export of data from database to excel, so that user can perform different types of operation on the data present in database.



The above developed system is useful to export the data related to examination different formats. Exam related data is always massive and it needs the rigid system. The system developed has many advantages and save much more time of resources required to copy and paste the data from web to excel and database to excel. Implementation of the complete system is done with the help of JAVA .To read data from database here used the apache poi library. Data from database (MySQL) with the help of J-Connector of MySQL is exported to excel .

VI. Conclusion

Massive Data read from database in table format and exported same directly to Excel with the help of APACHE POI-HSSF.Less time required for planning and no need of typing the complete details of particular seat number of candidate.Less man power required for planning and arranging the table data.No tiredness and No frustration.No chance for mistake as all the reports are system generated.Proposed System will be helping to save the time, man power and laborious work.Java application will occupy very less memory in the system in the bytes.The system will be user friendly which will help user to make the examination a grand success.

References

[1]. http://howtodoinjava.com/2013/06/19/readingwriting-excel-files-in-java-poi-tutorial/, "Reading/writing excel files in java : POI tutorial".
[2]. http://poi.apache.org/spreadsheet/quick-guide.html "Busy Developers' Guide to HSSF and XSSF Features".
[3]. http://www.vogella.com/articles/JavaExcel/article.html,"Excel and Java - Read and Write Excel with Java"
[4]. http://mrbool.com/reading-excel-file-with-java/24562,"Reading Excel file with Java".
[5]. http://www.javacoderanch.com/how-to-read-excel-file.html,"How to read Excel file ?".
[6]. http:// java-read-write-excel-file-apache,"Read / Write Excel File In Java Using Apache POI"
[7]. http://javabeginnerstutorial.com/code-base/write-excel-file/," Read and Write Excel with Java using PoI".
[8]. Andrew C. Oliver, Nicola Ken Barozzi "POI-HSSF and POI-XSSF - Java API To Access Microsoft Excel Format Files."
[9]. Zhang Ning, Jia Zi-Yan, Shi Zhong-Zhi, Research on Technology of ETL in Data Warehouse Computer Engineering and Applications, vol.24, no.12, 2002, pp.212-216. [10]
[10]. http://dev.mysql.com/doc/connector-j-usagenotes-connect-drivermanager.html,"Connecting to MySQL Using the JDBC DriverManager Interface".

[11]. http://java67.blogspot.in/2013/02/how-to-connect-mysql-database-from-java.html,“ Java program to connect MySQL database to execute query”.
[12]. http://www.javaworkspace.com/connectdatabase/connectMysql.do, “CONNECT TO MYSQL 5.1”.
[13]. http://poi.apache.org/apidocs/org/apache/poi/hssf/usermodel/HSSFDataFormat.html, “Class HSSFDataFormat”.
[14]. http://poi.apache.org/apidocs/org/apache/poi/hssf/usermodel/HSSFRichTextString.html “ Class HSSFRichTextString”.
[15]. http://poi.apache.org/spreadsheet/ ,“POI-HSSF and POI-XSSF – Java\
[16]. API To Access Microsoft Excel Format Files”.
[17]. http://docs.oracle.com/cd/B10501_01/server.920/a96652/ch02.htm,
[18]. “What Is the Import Utility?”.