Busy Developers' Guide to HSSF Features

by Glen Stampoultzis

1. Busy Developers' Guide to Features

Want to use HSSF read and write spreadsheets in a hurry? This guide is for you. If you're after more in-depth coverage of the HSSF user-API please consult the HOWTO guide as it contains actual descriptions of how to use this stuff.

1.1. Index of Features

- Headers and Footers.

1.2. Features

1.2.1. New Workbook

```
HSSFWorkbook wb = new HSSFWorkbook();
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.2. New Sheet

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet1 = wb.createSheet("new sheet");
HSSFSheet sheet2 = wb.createSheet("second sheet");
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.3. Creating Cells

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("new sheet");

// Create a row and put some cells in it. Rows are 0 based.
HSSFRow row = sheet.createRow((short)0);
// Create a cell and put a value in it.
HSSFCell cell = row.createCell((short)0);
cell.setCellValue(1);

// Or do it on one line.
row.createCell((short)1).setCellValue(1.2);
row.createCell((short)2).setCellValue("This is a string");
row.createCell((short)3).setCellValue(true);

// Write the output to a file
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.4. Creating Date Cells

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("new sheet");
```

```
    // Create a row and put some cells in it. Rows are 0 based.
    HSSFRow row = sheet.createRow((short)0);

    // Create a cell and put a date value in it.  The first cell is not styled
    // as a date.
    HSSFCell cell = row.createCell((short)0);
    cell.setCellValue(new Date());

    // we style the second cell as a date (and time).  It is important to
    // create a new cell style from the workbook otherwise you can end up
    // modifying the built in style and effecting not only this cell but other cells.
    HSSFCellStyle cellStyle = wb.createCellStyle();
    cellStyle.setDataFormat(HSSFDataFormat.getBuiltinFormat("m/d/yy h:mm"));
    cell = row.createCell((short)1);
    cell.setCellValue(new Date());
    cell.setCellStyle(cellStyle);

    // Write the output to a file
    FileOutputStream fileOut = new FileOutputStream("workbook.xls");
    wb.write(fileOut);
    fileOut.close();
```

1.2.5. Working with different types of cells

```
    HSSFWorkbook wb = new HSSFWorkbook();
    HSSFSheet sheet = wb.createSheet("new sheet");
    HSSFRow row = sheet.createRow((short)2);
    row.createCell((short) 0).setCellValue(1.1);
    row.createCell((short) 1).setCellValue(new Date());
    row.createCell((short) 2).setCellValue("a string");
    row.createCell((short) 3).setCellValue(true);
    row.createCell((short) 4).setCellType(HSSFCell.CELL_TYPE_ERROR);

    // Write the output to a file
    FileOutputStream fileOut = new FileOutputStream("workbook.xls");
    wb.write(fileOut);
    fileOut.close();
```

1.2.6. Demonstrates various alignment options

```
    public static void main(String[] args)
            throws IOException
    {
        HSSFWorkbook wb = new HSSFWorkbook();
        HSSFSheet sheet = wb.createSheet("new sheet");
        HSSFRow row = sheet.createRow((short) 2);
        createCell(wb, row, (short) 0, HSSFCellStyle.ALIGN_CENTER);
        createCell(wb, row, (short) 1, HSSFCellStyle.ALIGN_CENTER_SELECTION);
        createCell(wb, row, (short) 2, HSSFCellStyle.ALIGN_FILL);
        createCell(wb, row, (short) 3, HSSFCellStyle.ALIGN_GENERAL);
```

```
        createCell(wb, row, (short) 4, HSSFCellStyle.ALIGN_JUSTIFY);
        createCell(wb, row, (short) 5, HSSFCellStyle.ALIGN_LEFT);
        createCell(wb, row, (short) 6, HSSFCellStyle.ALIGN_RIGHT);

        // Write the output to a file
        FileOutputStream fileOut = new FileOutputStream("workbook.xls");
        wb.write(fileOut);
        fileOut.close();

    }

    /**
     * Creates a cell and aligns it a certain way.
     *
     * @param wb        the workbook
     * @param row       the row to create the cell in
     * @param column    the column number to create the cell in
     * @param align     the alignment for the cell.
     */
    private static void createCell(HSSFWorkbook wb, HSSFRow row, short column, short al
    {
        HSSFCell cell = row.createCell(column);
        cell.setCellValue("Align It");
        HSSFCellStyle cellStyle = wb.createCellStyle();
        cellStyle.setAlignment(align);
        cell.setCellStyle(cellStyle);
    }
```

1.2.7. Working with borders

```
    HSSFWorkbook wb = new HSSFWorkbook();
    HSSFSheet sheet = wb.createSheet("new sheet");

    // Create a row and put some cells in it. Rows are 0 based.
    HSSFRow row = sheet.createRow((short) 1);

    // Create a cell and put a value in it.
    HSSFCell cell = row.createCell((short) 1);
    cell.setCellValue(4);

    // Style the cell with borders all around.
    HSSFCellStyle style = wb.createCellStyle();
    style.setBorderBottom(HSSFCellStyle.BORDER_THIN);
    style.setBottomBorderColor(HSSFColor.BLACK.index);
    style.setBorderLeft(HSSFCellStyle.BORDER_THIN);
    style.setLeftBorderColor(HSSFColor.GREEN.index);
    style.setBorderRight(HSSFCellStyle.BORDER_THIN);
    style.setRightBorderColor(HSSFColor.BLUE.index);
    style.setBorderTop(HSSFCellStyle.BORDER_MEDIUM_DASHED);
    style.setTopBorderColor(HSSFColor.BLACK.index);
    cell.setCellStyle(style);
```

```
    // Write the output to a file
    FileOutputStream fileOut = new FileOutputStream("workbook.xls");
    wb.write(fileOut);
    fileOut.close();
```

1.2.8. Fills and colors

```
    HSSFWorkbook wb = new HSSFWorkbook();
    HSSFSheet sheet = wb.createSheet("new sheet");

    // Create a row and put some cells in it. Rows are 0 based.
    HSSFRow row = sheet.createRow((short) 1);

    // Aqua background
    HSSFCellStyle style = wb.createCellStyle();
    style.setFillBackgroundColor(HSSFColor.AQUA.index);
    style.setFillPattern(HSSFCellStyle.BIG_SPOTS);
    HSSFCell cell = row.createCell((short) 1);
    cell.setCellValue("X");
    cell.setCellStyle(style);

    // Orange "foreground", foreground being the fill foreground not the font color.
    style = wb.createCellStyle();
    style.setFillForegroundColor(HSSFColor.ORANGE.index);
    style.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);
    cell = row.createCell((short) 2);
    cell.setCellValue("X");
    cell.setCellStyle(style);

    // Write the output to a file
    FileOutputStream fileOut = new FileOutputStream("workbook.xls");
    wb.write(fileOut);
    fileOut.close();
```

1.2.9. Merging cells

```
    HSSFWorkbook wb = new HSSFWorkbook();
    HSSFSheet sheet = wb.createSheet("new sheet");

    HSSFRow row = sheet.createRow((short) 1);
    HSSFCell cell = row.createCell((short) 1);
    cell.setCellValue("This is a test of merging");

    sheet.addMergedRegion(new Region(1,(short)1,1,(short)2));

    // Write the output to a file
    FileOutputStream fileOut = new FileOutputStream("workbook.xls");
    wb.write(fileOut);
    fileOut.close();
```

1.2.10. Working with fonts

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("new sheet");

// Create a row and put some cells in it. Rows are 0 based.
HSSFRow row = sheet.createRow((short) 1);

// Create a new font and alter it.
HSSFFont font = wb.createFont();
font.setFontHeightInPoints((short)24);
font.setFontName("Courier New");
font.setItalic(true);
font.setStrikeout(true);

// Fonts are set into a style so create a new one to use.
HSSFCellStyle style = wb.createCellStyle();
style.setFont(font);

// Create a cell and put a value in it.
HSSFCell cell = row.createCell((short) 1);
cell.setCellValue("This is a test of fonts");
cell.setCellStyle(style);

// Write the output to a file
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.11. Custom colors

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet();
HSSFRow row = sheet.createRow((short) 0);
HSSFCell cell = row.createCell((short) 0);
cell.setCellValue("Default Palette");

//apply some colors from the standard palette,
// as in the previous examples.
//we'll use red text on a lime background

HSSFCellStyle style = wb.createCellStyle();
style.setFillForegroundColor(HSSFColor.LIME.index);
style.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);

HSSFFont font = wb.createFont();
font.setColor(HSSFColor.RED.index);
style.setFont(font);

cell.setCellStyle(style);
```

```
//save with the default palette
FileOutputStream out = new FileOutputStream("default_palette.xls");
wb.write(out);
out.close();

//now, let's replace RED and LIME in the palette
// with a more attractive combination
// (lovingly borrowed from freebsd.org)

cell.setCellValue("Modified Palette");

//creating a custom palette for the workbook
HSSFPalette palette = wb.getCustomPalette();

//replacing the standard red with freebsd.org red
palette.setColorAtIndex(HSSFColor.RED.index,
        (byte) 153,  //RGB red (0-255)
        (byte) 0,    //RGB green
        (byte) 0     //RGB blue
);
//replacing lime with freebsd.org gold
palette.setColorAtIndex(HSSFColor.LIME.index, (byte) 255, (byte) 204, (byte) 102);

//save with the modified palette
// note that wherever we have previously used RED or LIME, the
// new colors magically appear
out = new FileOutputStream("modified_palette.xls");
wb.write(out);
out.close();
```

1.2.12. Reading and Rewriting Workbooks

```
POIFSFileSystem fs      =
        new POIFSFileSystem(new FileInputStream("workbook.xls"));
HSSFWorkbook wb = new HSSFWorkbook(fs);
HSSFSheet sheet = wb.getSheetAt(0);
HSSFRow row = sheet.getRow(2);
HSSFCell cell = row.getCell((short)3);
if (cell == null)
    cell = row.createCell((short)3);
cell.setCellType(HSSFCell.CELL_TYPE_STRING);
cell.setCellValue("a test");

// Write the output to a file
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.13. Using newlines in cells

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet s = wb.createSheet();
HSSFRow r = null;
HSSFCell c = null;
HSSFCellStyle cs = wb.createCellStyle();
HSSFFont f = wb.createFont();
HSSFFont f2 = wb.createFont();

cs = wb.createCellStyle();

cs.setFont( f2 );
//Word Wrap MUST be turned on
cs.setWrapText( true );

r = s.createRow( (short) 2 );
r.setHeight( (short) 0x349 );
c = r.createCell( (short) 2 );
c.setCellType( HSSFCell.CELL_TYPE_STRING );
c.setCellValue( "Use \n with word wrap on to create a new line" );
c.setCellStyle( cs );
s.setColumnWidth( (short) 2, (short) ( ( 50 * 8 ) / ( (double) 1 / 20 ) ) );

FileOutputStream fileOut = new FileOutputStream( "workbook.xls" );
wb.write( fileOut );
fileOut.close();
```

1.2.14. Data Formats

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("format sheet");
HSSFCellStyle style;
HSSFDataFormat format = wb.createDataFormat();
HSSFRow row;
HSSFCell cell;
short rowNum = 0;
short colNum = 0;

row = sheet.createRow(rowNum++);
cell = row.createCell(colNum);
cell.setCellValue(11111.25);
style = wb.createCellStyle();
style.setDataFormat(format.getFormat("0.0"));
cell.setCellStyle(style);

row = sheet.createRow(rowNum++);
cell = row.createCell(colNum);
cell.setCellValue(11111.25);
style = wb.createCellStyle();
style.setDataFormat(format.getFormat("#,##0.0000"));
cell.setCellStyle(style);

FileOutputStream fileOut = new FileOutputStream("workbook.xls");
```

```
    wb.write(fileOut);
    fileOut.close();
```

1.2.15. Set Print Area to One Page

```
    HSSFWorkbook wb = new HSSFWorkbook();
    HSSFSheet sheet = wb.createSheet("format sheet");
    HSSFPrintSetup ps = sheet.getPrintSetup()

    sheet.setAutobreaks(true)

    ps.setFitHeight((short)1);
    ps.setFitWidth((short)1);


    // Create various cells and rows for spreadsheet.

    FileOutputStream fileOut = new FileOutputStream("workbook.xls");
    wb.write(fileOut);
    fileOut.close();
```

1.2.16. Set Print Area

```
    HSSFWorkbook wb = new HSSFWorkbook();
    HSSFSheet sheet = wb.createSheet("Sheet1");
    wb.setPrintArea(0, "$A$1:$C$2");
    //sets the print area for the first sheet
    //Alternatively:
    //wb.setPrintArea(0, 0, 1, 0, 0) is equivalent to using the name reference (See the

    // Create various cells and rows for spreadsheet.

    FileOutputStream fileOut = new FileOutputStream("workbook.xls");
    wb.write(fileOut);
    fileOut.close();
```

1.2.17. Set Page Numbers on Footer

```
    HSSFWorkbook wb = new HSSFWorkbook();
    HSSFSheet sheet = wb.createSheet("format sheet");
    HSSFFooter footer = sheet.getFooter()

    footer.setRight( "Page " + HSSFFooter.page() + " of " + HSSFFooter.numPages() );
```

```
// Create various cells and rows for spreadsheet.

FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.18. Using the Convenience Functions

The convenience functions live in contrib and provide utility features such as setting borders around merged regions and changing style attributes without explicitly creating new styles.

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet1 = wb.createSheet( "new sheet" );

// Create a merged region
HSSFRow row = sheet1.createRow( (short) 1 );
HSSFRow row2 = sheet1.createRow( (short) 2 );
HSSFCell cell = row.createCell( (short) 1 );
cell.setCellValue( "This is a test of merging" );
Region region = new Region( 1, (short) 1, 4, (short) 4 );
sheet1.addMergedRegion( region );

// Set the border and border colors.
final short borderMediumDashed = HSSFCellStyle.BORDER_MEDIUM_DASHED;
HSSFRegionUtil.setBorderBottom( borderMediumDashed,
    region, sheet1, wb );
HSSFRegionUtil.setBorderTop( borderMediumDashed,
    region, sheet1, wb );
HSSFRegionUtil.setBorderLeft( borderMediumDashed,
    region, sheet1, wb );
HSSFRegionUtil.setBorderRight( borderMediumDashed,
    region, sheet1, wb );
HSSFRegionUtil.setBottomBorderColor(HSSFColor.AQUA.index, region, sheet1, wb);
HSSFRegionUtil.setTopBorderColor(HSSFColor.AQUA.index, region, sheet1, wb);
HSSFRegionUtil.setLeftBorderColor(HSSFColor.AQUA.index, region, sheet1, wb);
HSSFRegionUtil.setRightBorderColor(HSSFColor.AQUA.index, region, sheet1, wb);

// Shows some usages of HSSFCellUtil
HSSFCellStyle style = wb.createCellStyle();
style.setIndention((short)4);
HSSFCellUtil.createCell(row, 8, "This is the value of the cell", style);
HSSFCell cell2 = HSSFCellUtil.createCell( row2, 8, "This is the value of the cell")
HSSFCellUtil.setAlignment(cell2, wb, HSSFCellStyle.ALIGN_CENTER);

// Write out the workbook
FileOutputStream fileOut = new FileOutputStream( "workbook.xls" );
wb.write( fileOut );
fileOut.close();
```

1.2.19. Shift rows up or down on a sheet

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("row sheet");

// Create various cells and rows for spreadsheet.

// Shift rows 6 - 11 on the spreadsheet to the top (rows 0 - 5)
sheet.shiftRows(5, 10, -5);

FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.20. Set a sheet as selected

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("row sheet");
sheet.setSelected(true);

// Create various cells and rows for spreadsheet.

FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.21. Set the zoom magnification

The zoom is expressed as a fraction. For example to express a zoom of 75% use 3 for the numerator and 4 for the denominator.

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet1 = wb.createSheet("new sheet");
sheet1.setZoom(3,4);   // 75 percent magnification
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.22. Splits and freeze panes

There are two types of panes you can create; freeze panes and split panes.

A freeze pane is split by columns and rows. You create a freeze pane using the following mechanism:

sheet1.createFreezePane(3, 2, 3, 2);

The first two parameters are the columns and rows you wish to split by. The second two

parameters indicate the cells that are visible in the bottom right quadrant.

Split pains appear differently. The split area is divided into four separate work area's. The split occurs at the pixel level and the user is able to adjust the split by dragging it to a new position.

Split panes are created with the following call:

sheet2.createSplitPane(2000, 2000, 0, 0, HSSFSheet.PANE_LOWER_LEFT);

The first parameter is the x position of the split. This is in 1/20th of a point. A point in this case seems to equate to a pixel. The second parameter is the y position of the split. Again in 1/20th of a point.

The last parameter indicates which pane currently has the focus. This will be one of HSSFSheet.PANE_LOWER_LEFT, PANE_LOWER_RIGHT, PANE_UPPER_RIGHT or PANE_UPPER_LEFT.

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet1 = wb.createSheet("new sheet");
HSSFSheet sheet2 = wb.createSheet("second sheet");
HSSFSheet sheet3 = wb.createSheet("third sheet");
HSSFSheet sheet4 = wb.createSheet("fourth sheet");

// Freeze just one row
sheet1.createFreezePane( 0, 1, 0, 1 );
// Freeze just one column
sheet2.createFreezePane( 1, 0, 1, 0 );
// Freeze the columns and rows (forget about scrolling position of the lower right
sheet3.createFreezePane( 2, 2 );
// Create a split with the lower left side being the active quadrant
sheet4.createSplitPane( 2000, 2000, 0, 0, HSSFSheet.PANE_LOWER_LEFT );

FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.23. Repeating rows and columns

It's possible to set up repeating rows and columns in your printouts by using the setRepeatingRowsAndColumns() function in the HSSFWorkbook class.

This function Contains 5 parameters. The first parameter is the index to the sheet (0 = first sheet). The second and third parameters specify the range for the columns to repeat. To stop the columns from repeating pass in -1 as the start and end column. The fourth and fifth parameters specify the range for the rows to repeat. To stop the columns from repeating pass in -1 as the start and end rows.

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet1 = wb.createSheet("new sheet");
HSSFSheet sheet2 = wb.createSheet("second sheet");

// Set the columns to repeat from column 0 to 2 on the first sheet
wb.setRepeatingRowsAndColumns(0,0,2,-1,-1);
// Set the the repeating rows and columns on the second sheet.
wb.setRepeatingRowsAndColumns(1,4,5,1,2);

FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

1.2.24. Headers and Footers

Example is for headers but applies directly to footers.

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("new sheet");

HSSFHeader header = sheet.getHeader();
header.setCenter("Center Header");
header.setLeft("Left Header");
header.setRight(HSSFHeader.font("Stencil-Normal", "Italic") +
                HSSFHeader.fontSize((short) 16) + "Right w/ Stencil-Normal Italic f

FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```