Upgrading to POI 3.5, including converting existing HSSF Usermodel code to SS Usermodel (for XSSF and HSSF)

by Nick Burch

1. Things that have to be changed when upgrading to POI 3.5

Wherever possible, we have tried to ensure that you can use your existing POI code with POI 3.5 without requiring any changes. However, Java doesn't always make that easy, and unfortunately there are a few changes that may be required for some users.

1.1. org.apache.poi.hssf.usermodel.HSSFFormulaEvaluator.CellValue

Annoyingly, java will not let you access a static inner class via a child of the parent one. So, all references to *org.apache.poi.hssf.usermodel.HSSFFormulaEvaluator.CellValue* will need to be changed to *org.apache.poi.ss.usermodel.FormulaEvaluator.CellValue*

1.2. org.apache.poi.hssf.usermodel.HSSFRow.MissingCellPolicy

Annoyingly, java will not let you access a static inner class via a child of the parent one. So, all references to *org.apache.poi.hssf.usermodel.HSSFRow.MissingCellPolicy* will need to be changed to *org.apache.poi.ss.usermodel.Row.MissingCellPolicy*

1.3. DDF and org.apache.poi.hssf.record.RecordFormatException

Previously, record level errors within DDF would throw an exception from the hssf class heirachy. Now, record level errors within DDF will throw a more general RecordFormatException, *org.apache.poi.util.RecordFormatException*

In addition, org.apache.poi.hssf.record.RecordFormatException has been changed to inherit from the new *org.apache.poi.util.RecordFormatException*, so you may wish to change

catches of the hssf version to the new util version.

2. Converting existing HSSF Usermodel code to SS Usermodel (for XSSF and HSSF)

2.1. Why change?

If you have existing HSSF usermodel code that works just fine, and you don't want to use the new OOXML XSSF support, then you probably don't need to. Your existing HSSF only code will continue to work just fine.

However, if you want to be able to work with both HSSF for your .xls files, and also XSSF for .xslx files, then you will need to make some slight tweaks to your code.

2.2. org.apache.poi.ss.usermodel

The new SS usermodel (org.apache.poi.ss.usermodel) is very heavily based on the old HSSF usermodel (org.apache.poi.hssf.usermodel). The main difference is that the package name and class names have been tweaked to remove HSSF from them. Otherwise, the new SS Usermodel interfaces should provide the same functionality.

2.3. Constructors

Calling the empty HSSFWorkbook remains as the way to create a new, empty Workbook object. To open an existing Worbook, you should now call WorkbookFactory.create(inp).

For all other cases when you would have called a Usermodel constructor, such as 'new HSSFRichTextString()' or 'new HSSFDataFormat', you should instead use a CreationHelper. There's a method on the Workbook to get a CreationHelper, and the CreationHelper will then handle constructing new objects for you.

2.4. Other Code

For all other code, generally change a reference from org.apache.poi.hssf.usermodel.HSSFFoo to a reference to org.apache.poi.ss.usermodel.Foo. Method signatures should otherwise remain the same, and it should all then work for both XSSF and HSSF.

3. Worked Examples

3.1. Old HSSF Code

```
// import org.apache.poi.hssf.usermodel.*;

HSSFWorkbook wb = new HSSFWorkbook();
// create a new sheet
HSSFSheet s = wb.createSheet();
// declare a row object reference
HSSFRow r = null;
// declare a cell object reference
HSSFCell c = null;
// create 2 cell styles
HSSFCellStyle cs = wb.createCellStyle();
HSSFCellStyle cs2 = wb.createCellStyle();
HSSFDataFormat df = wb.createDataFormat();

// create 2 fonts objects
HSSFFont f = wb.createFont();
HSSFFont f2 = wb.createFont();

// Set font 1 to 12 point type, blue and bold
f.setFontHeightInPoints((short) 12);
f.setColor( HSSFColor.RED.index );
f.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);

// Set font 2 to 10 point type, red and bold
f2.setFontHeightInPoints((short) 10);
f2.setColor( HSSFFont.RED.index );
f2.setBoldweight(HSSFFont.BOLDWEIGHT_BOLD);

// Set cell style and formatting
cs.setFont(f);
cs.setDataFormat(df.getFormat("#,##0.0"));

// Set the other cell style and formatting
cs2.setBorderBottom(cs2.BORDER_THIN);
cs2.setDataFormat(HSSFDataFormat.getBuiltinFormat("text"));
cs2.setFont(f2);


// Define a few rows
for(short rownum = (short)0; rownum < 30; rownum++) {
        HSSFRow r = s.createRow(rownum);
        for(short cellnum = (short)0; cellnum < 10; cellnum += 2) {
                HSSFCell c = r.createCell(cellnum);
                HSSFCell c2 = r.createCell(cellnum+1);

                c.setCellValue((double)rownum + (cellnum/10));
                c2.setCellValue(new HSSFRichTextString("Hello! " + cellnum);
        }
}

// Save
FileOutputStream out = new FileOutputStream("workbook.xls");
wb.write(out);
```

```
out.close();
```

3.2. New, generic SS Usermodel Code

```
// import org.apache.poi.ss.usermodel.*;

Workbook[] wbs = new Workbook[] { new HSSFWorkbook(), new XSSFWorkbook() };
for(int i=0; i<wbs.length; i++) {
   Workbook wb = wbs[i];
   CreationHelper createHelper = wb.getCreationHelper();

   // create a new sheet
   Sheet s = wb.createSheet();
   // declare a row object reference
   Row r = null;
   // declare a cell object reference
   Cell c = null;
   // create 2 cell styles
   CellStyle cs = wb.createCellStyle();
   CellStyle cs2 = wb.createCellStyle();
   DataFormat df = wb.createDataFormat();

   // create 2 fonts objects
   Font f = wb.createFont();
   Font f2 = wb.createFont();

   // Set font 1 to 12 point type, blue and bold
   f.setFontHeightInPoints((short) 12);
   f.setColor( IndexedColors.RED.getIndex() );
   f.setBoldweight(Font.BOLDWEIGHT_BOLD);

   // Set font 2 to 10 point type, red and bold
   f2.setFontHeightInPoints((short) 10);
   f2.setColor( IndexedColors.RED.getIndex() );
   f2.setBoldweight(Font.BOLDWEIGHT_BOLD);

   // Set cell style and formatting
   cs.setFont(f);
   cs.setDataFormat(df.getFormat("#,##0.0"));

   // Set the other cell style and formatting
   cs2.setBorderBottom(cs2.BORDER_THIN);
   cs2.setDataFormat(df.getFormat("text"));
   cs2.setFont(f2);


   // Define a few rows
   for(int rownum = 0; rownum < 30; rownum++) {
           Row r = s.createRow(rownum);
           for(int cellnum = 0; cellnum < 10; cellnum += 2) {
                   Cell c = r.createCell(cellnum);
                   Cell c2 = r.createCell(cellnum+1);
```

```
                    c.setCellValue((double)rownum + (cellnum/10));
                    c2.setCellValue(
                          createHelper.createRichTextString("Hello! " + cellnum)
                    );
              }
   }

   // Save
   String filename = "workbook.xls";
   if(wb instanceof XSSFWorkbook) {
     filename = filename + "x";
   }

   FileOutputStream out = new FileOutputStream(filename);
   wb.write(out);
   out.close();
}
```