Formula Evaluation

by Amol Deshmukh

1. Introduction

The POI formula evaluation code enables you to calculate the result of formulas in Excels sheets read-in, or created in POI. This document explains how to use the API to evaluate your formulas.

> **Note:**
> .xlsx format is suported since POI 3.5, make sure yoy upgraded to that version before experimenting with this code. Users of all versions of POI may wish to make use of a recent SVN checkout, as new functions are currently being added fairly frequently.

2. Status

The code currently provides implementations for all the arithmatic operators. It also provides implementations for approx. 100 built in functions in Excel. The framework however makes is easy to add implementation of new functions. See the Formula evaluation development guide and javadocs for details.

Both HSSFWorkbook and XSSFWorkbook are supported, so you can evaluate formulas on both .xls and .xlsx files.

Note that user-defined functions are not supported, and is not likely to done any time soon... at least, not till there is a VB implementation in Java!

3. User API How-TO

The following code demonstrates how to use the FormulaEvaluator in the context of other POI excel reading code.

There are several ways in which you can use the FormulaEvalutator API.

3.1. Using FormulaEvaluator.evaluate(Cell cell)

This evaluates a given cell, and returns the new value, without affecting the cell

```
FileInputStream fis = new FileInputStream("c:/temp/test.xls");
Workbook wb = new HSSFWorkbook(fis); //or new XSSFWorkbook("c:/temp/test.xls")
Sheet sheet = wb.getSheetAt(0);
FormulaEvaluator evaluator = wb.getCreationHelper().createFormulaEvaluator();

// suppose your formula is in B3
CellReference cellReference = new CellReference("B3");
Row row = sheet.getRow(cellReference.getRow());
Cell cell = row.getCell(cellReference.getCol());

CellValue cellValue = evaluator.evaluate(cell);

switch (cellValue.getCellType()) {
    case Cell.CELL_TYPE_BOOLEAN:
        System.out.println(cellValue.getBooleanValue());
        break;
    case Cell.CELL_TYPE_NUMERIC:
        System.out.println(cellValue.getNumberValue());
        break;
    case Cell.CELL_TYPE_STRING:
        System.out.println(cellValue.getStringValue());
        break;
    case Cell.CELL_TYPE_BLANK:
        break;
    case Cell.CELL_TYPE_ERROR:
        break;

    // CELL_TYPE_FORMULA will never happen
    case Cell.CELL_TYPE_FORMULA:
        break;
}
```

Thus using the retrieved value (of type FormulaEvaluator.CellValue - a nested class) returned by FormulaEvaluator is similar to using a Cell object containing the value of the formula evaluation. CellValue is a simple value object and does not maintain reference to the original cell.

3.2. Using FormulaEvaluator.evaluateFormulaCell(Cell cell)

evaluateFormulaCell(Cell cell) will check to see if the supplied cell is a formula cell. If it isn't, then no changes will be made to it. If it is, then the formula is evaluated. The value for the formula is saved alongside it, to be displayed in excel. The formula remains in the cell, just with a new value

The return of the function is the type of the formula result, such as Cell.CELL_TYPE_BOOLEAN

```
FileInputStream fis = new FileInputStream("/somepath/test.xls");
```

```
Workbook wb = new HSSFWorkbook(fis); //or new XSSFWorkbook("/somepath/test.xls")
Sheet sheet = wb.getSheetAt(0);
FormulaEvaluator evaluator = wb.getCreationHelper().createFormulaEvaluator();

// suppose your formula is in B3
CellReference cellReference = new CellReference("B3");
Row row = sheet.getRow(cellReference.getRow());
Cell cell = row.getCell(cellReference.getCol());

if (cell!=null) {
    switch (evaluator.evaluateFormulaCell(cell)) {
        case Cell.CELL_TYPE_BOOLEAN:
            System.out.println(cell.getBooleanCellValue());
            break;
        case Cell.CELL_TYPE_NUMERIC:
            System.out.println(cell.getNumericCellValue());
            break;
        case Cell.CELL_TYPE_STRING:
            System.out.println(cell.getStringCellValue());
            break;
        case Cell.CELL_TYPE_BLANK:
            break;
        case Cell.CELL_TYPE_ERROR:
            System.out.println(cell.getErrorCellValue());
            break;

        // CELL_TYPE_FORMULA will never occur
        case Cell.CELL_TYPE_FORMULA:
            break;
    }
}
```

3.3. Using FormulaEvaluator.evaluateInCell(Cell cell)

evaluateInCell(Cell cell) will check to see if the supplied cell is a formula cell. If it isn't, then no changes will be made to it. If it is, then the formula is evaluated, and the new value saved into the cell, in place of the old formula.

```
FileInputStream fis = new FileInputStream("/somepath/test.xls");
Workbook wb = new HSSFWorkbook(fis); //or new XSSFWorkbook("/somepath/test.xls")
Sheet sheet = wb.getSheetAt(0);
FormulaEvaluator evaluator = wb.getCreationHelper().createFormulaEvaluator();

// suppose your formula is in B3
CellReference cellReference = new CellReference("B3");
Row row = sheet.getRow(cellReference.getRow());
Cell cell = row.getCell(cellReference.getCol());

if (cell!=null) {
    switch (evaluator.evaluateInCell(cell).getCellType()) {
        case Cell.CELL_TYPE_BOOLEAN:
```

```
            System.out.println(cell.getBooleanCellValue());
            break;
        case Cell.CELL_TYPE_NUMERIC:
            System.out.println(cell.getNumericCellValue());
            break;
        case Cell.CELL_TYPE_STRING:
            System.out.println(cell.getStringCellValue());
            break;
        case Cell.CELL_TYPE_BLANK:
            break;
        case Cell.CELL_TYPE_ERROR:
            System.out.println(cell.getErrorCellValue());
            break;

        // CELL_TYPE_FORMULA will never occur
        case Cell.CELL_TYPE_FORMULA:
            break;
    }
}
```

3.4. Re-calculating all formulas in a Workbook

```
FileInputStream fis = new FileInputStream("/somepath/test.xls");
Workbook wb = new HSSFWorkbook(fis); //or new XSSFWorkbook("/somepath/test.xls")
FormulaEvaluator evaluator = wb.getCreationHelper().createFormulaEvaluator();
for(int sheetNum = 0; sheetNum < wb.getNumberOfSheets(); sheetNum++) {
    Sheet sheet = wb.getSheetAt(sheetNum);
    for(Row r : sheet) {
        for(Cell c : r) {
            if(c.getCellType() == Cell.CELL_TYPE_FORMULA) {
                evaluator.evaluateFormulaCell(c);
            }
        }
    }
}
```

Alternately, if you know which of HSSF or XSSF you're working with, then you can call the static **evaluateAllFormulaCells** method on the appropriate HSSFFormulaEvaluator or XSSFFormulaEvaluator class.

4. Recalculation of Formulas

In certain cases you may want to force Excel to re-calculate formulas when the workbook is opened. Consider the following example:

Open Excel and create a new workbook. On the first sheet set A1=1, B1=1, C1=A1+B1. Excel automatically calculates formulas and the value in C1 is 2. So far so good.

Now modify the workbook with POI:

```
Workbook wb = WorkbookFactory.create(new FileInputStream("workbook.xls"));

Sheet sh = wb.getSheetAt(0);
sh.getRow(0).getCell(0).setCellValue(2);  // set A1=2

FileOutputStream out = new FileOutputStream("workbook2.xls");
wb.write(out);
out.close();
```

Now open workbook2.xls in Excel and the value in C1 is still 2 while you expected 3. Wrong? No! The point is that Excel caches previously calculated results and you need to trigger recalculation to updated them. It is not an issue when you are creating new workbooks from scratch, but important to remember when you are modifing existing workbooks with formulas. This can be done in two ways:

1. Re-evaluate formuals with POI's FormulaEvaluator:

```
Workbook wb = WorkbookFactory.create(new FileInputStream("workbook.xls"));

Sheet sh = wb.getSheetAt(0);
sh.getRow(0).getCell(0).setCellValue(2);  // set A1=2

wb.getCreationHelper().createFormulaEvaluator().evaluateAll();
```

2. Delegate re-calculation to Excel. The application will perform a full recalculation when the workbook is opened:

```
Workbook wb = WorkbookFactory.create(new FileInputStream("workbook.xls"));

Sheet sh = wb.getSheetAt(0);
sh.getRow(0).getCell(0).setCellValue(2);  // set A1=2

wb.setForceFormulaRecalculation(true);
```

5. Performance Notes

- Generally you should have to create only one FormulaEvaluator instance per sheet, but there really is no overhead in creating multiple FormulaEvaluators per sheet other than that of the FormulaEvaluator object creation.
- Also note that FormulaEvaluator maintains a reference to the sheet and workbook, so ensure that the evaluator instance is available for garbage collection when you are done with it (in other words don't maintain long lived reference to FormulaEvaluator if you don't really need to - unless all references to the sheet and workbook are removed, these

don't get garbage collected and continue to occupy potentially large amounts of memory).

- CellValue instances however do not maintain reference to the Cell or the sheet or workbook, so these can be long-lived objects without any adverse effect on performance.