

Formula Evaluation

by Amol Deshmukh

1. Introduction

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

Note:

In versions of POI before 3.0.3, this code lived in the scratchpad area of the POI SVN repository. If using an such an older version of POI, ensure that you have the scratchpad jar or the scratchpad build area in your classpath 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 arithmetic operators. It also provides implementations for approx. 100 built in functions in Excel. The framework however makes it easy to add implementation of new functions. See the [Formula evaluation development guide](#) for details.

Note that user-defined functions are not supported, and is not likely to be 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 `HSSFFormulaEvaluator` in the context of other POI excel reading code.

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

3.1. Using `HSSFFormulaEvaluator.evaluate(HSSFCell cell)`

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

```
FileInputStream fis = new FileInputStream("c:/temp/test.xls");
```

```

HSSFWorkbook wb = new HSSFWorkbook(fis);
HSSFSheet sheet = wb.getSheetAt(0);
HSSFFormulaEvaluator evaluator = new HSSFFormulaEvaluator(wb);

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

HSSFFormulaEvaluator.CellValue cellValue = evaluator.evaluate(cell);

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

    // CELL_TYPE_FORMULA will never happen
    case HSSFCell.CELL_TYPE_FORMULA:
        break;
}

```

Thus using the retrieved value (of type `HSSFFormulaEvaluator.CellValue` - a nested class) returned by `HSSFFormulaEvaluator` is similar to using a `HSSFCell` 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 `HSSFFormulaEvaluator.evaluateFormulaCell(HSSFCell cell)`

`evaluateFormulaCell(HSSFCell 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 `HSSFCell.CELL_TYPE_BOOLEAN`

```

FileInputStream fis = new FileInputStream("/somepath/test.xls");
HSSFWorkbook wb = new HSSFWorkbook(fis);
HSSFSheet sheet = wb.getSheetAt(0);

```

Formula Evaluation

```
HSSFFormulaEvaluator evaluator = new HSSFFormulaEvaluator(wb);

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

if (cell!=null) {
    int valueType = evaluator.evaluateFormulaCell(cell);
    if (valueType == -1) {
        // Cell was not a formula cell
        // but we can read the plain value from the cell just the same as a for
        valueType = cell.getCellType();
    }
    switch (valueType) {
        case HSSFCell.CELL_TYPE_BOOLEAN:
            System.out.println(cell.getBooleanCellValue());
            break;
        case HSSFCell.CELL_TYPE_NUMERIC:
            System.out.println(cell.getNumericCellValue());
            break;
        case HSSFCell.CELL_TYPE_STRING:
            System.out.println(cell.getRichStringCellValue().getString());
            break;
        case HSSFCell.CELL_TYPE_BLANK:
            break;
        case HSSFCell.CELL_TYPE_ERROR:
            System.out.println(HSSFErrorConstants.getText(cell.getErrorCellValue()));
            break;

        case HSSFCell.CELL_TYPE_FORMULA:
            throw new IllegalStateException("Result-type 'formula' cannot o
    }
}
```

3.3. Using `HSSFFormulaEvaluator.evaluateInCell(HSSFCell cell)`

`evaluateInCell(HSSFCell 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");
HSSFWorkbook wb = new HSSFWorkbook(fis);
HSSFSheet sheet = wb.getSheetAt(0);
HSSFFormulaEvaluator evaluator = new HSSFFormulaEvaluator(wb);

// suppose your formula is in B3
CellReference cellReference = new CellReference("B3");
HSSFRow row = sheet.getRow(cellReference.getRow());
```

```

HSSFCell cell = row.getCell((int)cellReference.getCol());

if (cell!=null) {
    switch (evaluator.evaluateInCell(cell).getCellType()) {
        case HSSFCell.CELL_TYPE_BOOLEAN:
            System.out.println(cell.getBooleanCellValue());
            break;
        case HSSFCell.CELL_TYPE_NUMERIC:
            System.out.println(cell.getNumericCellValue());
            break;
        case HSSFCell.CELL_TYPE_STRING:
            System.out.println(cell.getRichStringCellValue().getString());
            break;
        case HSSFCell.CELL_TYPE_BLANK:
            break;
        case HSSFCell.CELL_TYPE_ERROR:
            System.out.println(cell.getErrorCellValue());
            break;

        // CELL_TYPE_FORMULA will never occur
        case HSSFCell.CELL_TYPE_FORMULA:
            break;
    }
}

```

3.4. Re-calculating all formulas in a Workbook

```

FileInputStream fis = new FileInputStream("/somepath/test.xls");
HSSFWorkbook wb = new HSSFWorkbook(fis);
HSSFFormulaEvaluator evaluator = new HSSFFormulaEvaluator(wb);
for(int sheetNum = 0; sheetNum < wb.getNumberOfSheets(); sheetNum++) {
    HSSFSheet sheet = wb.getSheetAt(sheetNum);

    for(Iterator rit = sheet.rowIterator(); rit.hasNext();) {
        HSSFRow r = (HSSFRow)rit.next();

        for(Iterator cit = r.cellIterator(); cit.hasNext();) {
            HSSFCell c = (HSSFCell)cit.next();
            if(c.getCellType() == HSSFCell.CELL_TYPE_FORMULA) {
                evaluator.evaluateFormulaCell(c);
            }
        }
    }
}
wb.write(new FileOutputStream("/somepath/changed.xls"));

```

4. Performance Notes

- Generally you should have to create only one HSSFFormulaEvaluator instance per

Formula Evaluation

workbook, but there really is no overhead in creating multiple `HSSFFormulaEvaluators` per workbook other than that of the `HSSFFormulaEvaluator` object creation.

- Also note that `HSSFFormulaEvaluator` maintains a reference to the 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 `HSSFFormulaEvaluator` 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 `HSSFCell` or the sheet or workbook, so these can be long-lived objects without any adverse effect on performance.