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:**
> This code currently lives the scratchpad area of the POI CVS repository. Ensure that you have the scratchpad jar or the scratchpad build area in your classpath before experimenting with this code.

2. Status

The code currently provides implementations for all the arithmatic operators. It also provides implementations for approx. 20 built in functions in Excel. The framework however makes is 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 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 two ways in which you can use the HSSFFormulaEvalutator API.

3.1. Using HSSFFormulaEvaluator.evaluate(HSSFCell cell)

```
FileInputStream fis = new FileInputStream("c:/temp/test.xls");
HSSFWorkbook wb = new HSSFWorkbook(fis);
HSSFSheet sheet = wb.getSheetAt(0);
HSSFFormulaEvaluator evaluator = new HSSFFormulaEvaluator(sheet, wb);
```

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

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

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

```
                    break;
                case HSSFCell.CELL_TYPE_STRING:
                    System.out.println(cell.getStringCellValue());
                    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;
        }
}
```

4. Performance Notes

- Generally you should have to create only one HSSFFormulaEvaluator instance per sheet, but there really is no overhead in creating multiple HSSFFormulaEvaluators per sheet other than that of the HSSFFormulaEvaluator object creation.
- Also note that HSSFFormulaEvaluator 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 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.