

# SPOIWO

(Scala POI Wrapping Objects)

*How to Excel in Scala*

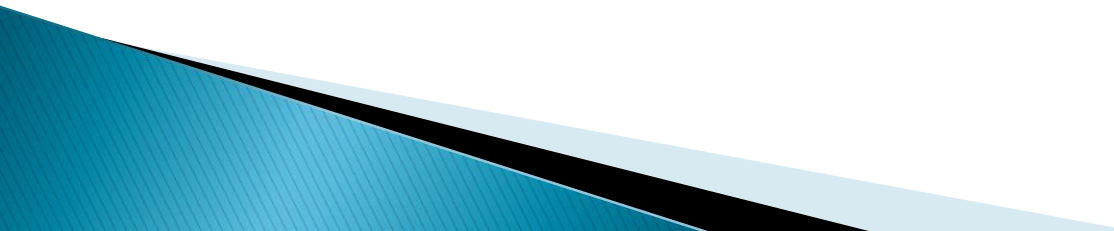
# Agenda

1. Issues with POI

2. Potential solution



# Norbert Radyk

- 2 years experience in Scala
  - 5 years experience in Java
  - Owner of Norbit Limited  
<http://www.norbitltd.com>
  - Author of SPOIWO since Jan'14
- 

# A few problems with creating a spreadsheet via Apache POI...


# 1. Mutability

```
val wb = new XSSFWorkbook
val sheet = wb.createSheet("new sheet")
val row = sheet.createRow(1.toShort);

val style = wb.createCellStyle
style.setFillBackgroundColor(IndexedColors.AQUA.getIndex())
style.setFillPattern(CellStyle.SOLID)

val cellA = row.createCell(0.toShort)
cell.setCellValue("SPOIWO")
cell.setCellStyle(style)

val cellB = row.createCell(1.toShort)
cell.setCellValue(2014)
cell.setCellStyle(style)
```



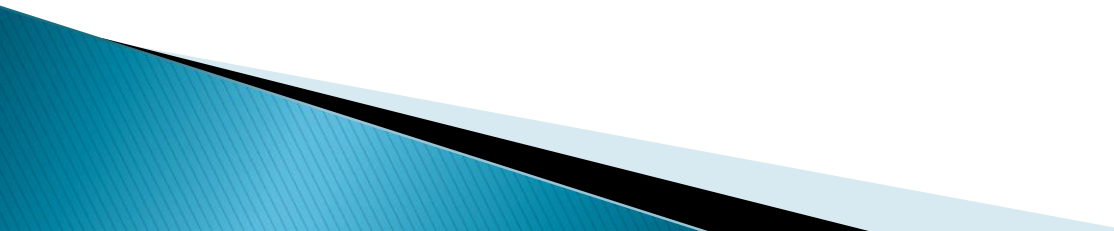
## 2. Enforced indexes

```
val wb = new XSSFWorkbook
val sheet = wb.createSheet("new sheet")
val row = sheet.createRow(1.toShort);

val style = wb.createCellStyle
style.setFillBackgroundColor(IndexedColors.AQUA.getIndex())
style.setFillPattern(CellStyle.SOLID)

val cellA = row.createCell(0.toShort)
cell.setCellValue("SPOIWO")
cell.setCellStyle(style)

val cellB = row.createCell(1.toShort)
cell.setCellValue(2014)
cell.setCellStyle(style)
```



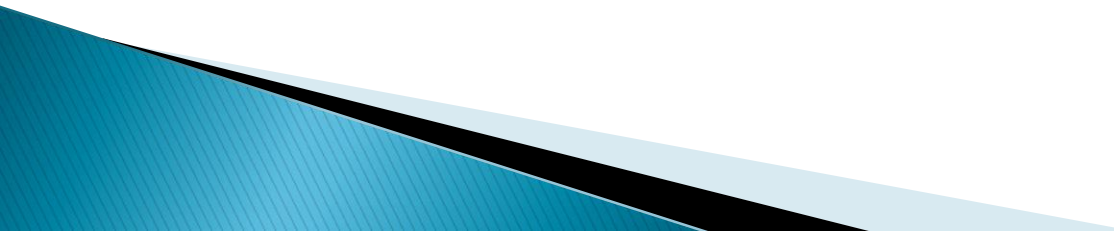
# 3. Structural coupling

```
val wb = new XSSFWorkbook
val sheet = wb.createSheet("new sheet")
val row = sheet.createRow(1.toShort);
```

```
val style = wb.createCellStyle
style.setFillBackgroundColor(IndexedColors.AQUA.getIndex())
style.setFillPattern(CellStyle.SOLID)
```

```
val cellA = row.createCell(0.toShort)
cell.setCellValue("SPOIWO")
cell.setCellStyle(style)
```

```
val cellB = row.createCell(1.toShort)
cell.setCellValue(2014)
cell.setCellStyle(style)
```



## 4. Order-dependent execution

```
sheet.setDefaultColumnStyle(0, style)
```

```
val row = sheet.createRow(0)
```

```
val cell = row.createCell(0)
```

```
cell.setCellValue("AAA")
```

```
val row = sheet.createRow(0)
```

```
val cell = row.createCell(0)
```

```
cell.setCellValue("AAA")
```

```
sheet.setDefaultColumnStyle(0, style)
```





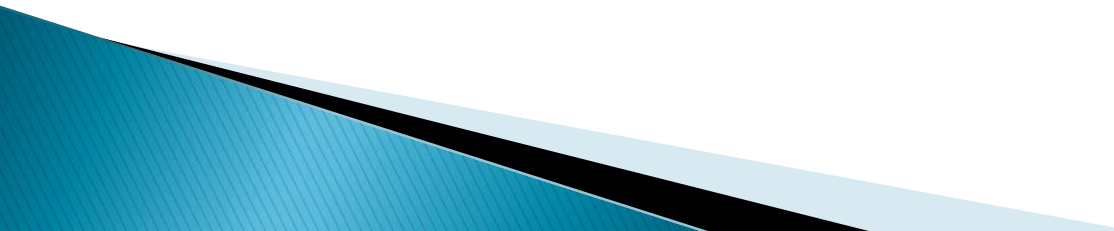
# 5. Objects duplication

```
def awesomeStyle : CellStyle = {  
    val style = wb.createCellStyle  
    style.setFillBackgroundColor(IndexedColors.AQUA.getIndex())  
    style.setFillPattern(CellStyle.SOLID)  
    return style  
}  
  
cells.foreach(_.setCellStyle(awesomeStyle))
```

**A SOLUTION...**

# SPOIWO

```
val style = CellStyle(  
    backgroundColor = AQUA,  
    fillPattern = SOLID  
)  
  
val sheet = Sheet(name = "new sheet")  
    .withRows (  
        Row(index = 1, style = style).withCellValues ("SPOIWO", 2014)  
    )
```



# Key Features (1)

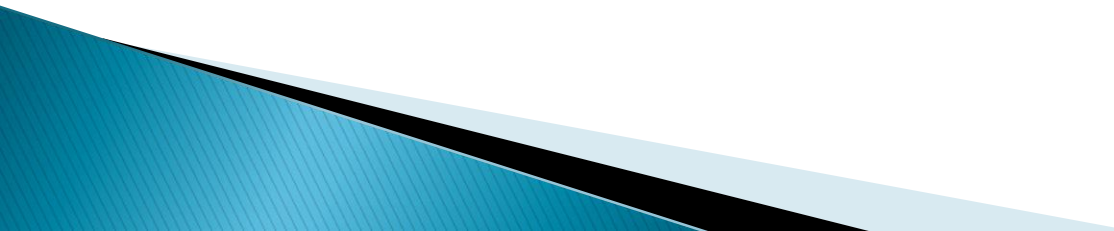
- ▶ Immutable workbook model
  - Case classes matching their mutable versions in POI
  - Broad coverage of POI functionality
  - All parameters can be set in constructor or via `with/without` functions

# Key Features (2)

- ▶ Model decoupled from conversions
  - Once spreadsheet model is defined it can be converted to any supported format (XLSX, CSV, HTML)
- ▶ Conversions for all model relevant elements
  - Easy adoption

# Example 1 – POI

```
def workingWithDifferentTypesOfCells() {  
    val wb = new XSSFWorkbook  
    val sheet = wb.createSheet("new sheet")  
    val row = sheet.createRow(2.toShort)  
    row.createCell(0).setCellValue(1.1)  
    row.createCell(1).setCellValue(new Date)  
    row.createCell(2).setCellValue(Calendar.getInstance)  
    row.createCell(3).setCellValue("a string")  
    row.createCell(4).setCellValue(true)  
  
    val fileOut = new FileOutputStream("workbook.xls")  
    wb.write(fileOut)  
    fileOut.close  
}
```

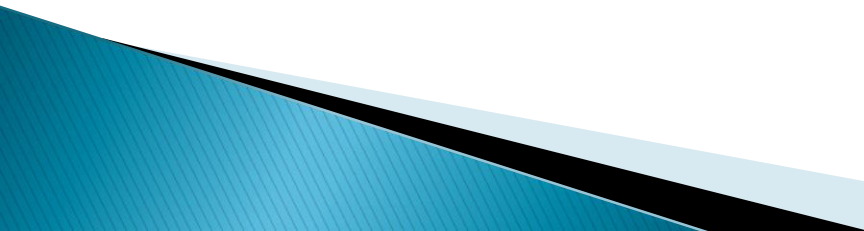


# Example 1 – Spoiwo

```
def workingWithDifferentTypesOfCells() = Sheet(  
    name = "new sheet",  
    row = Row(index = 2).withCellValues(  
        1.1,  
        new Date(),  
        Calendar.getInstance(),  
        "a string",  
        true  
    )  
).saveAsXlsx("workbook.xlsx")
```

# Example 2 – POI

```
def mergingCells() throws IOException {  
    val wb = new XSSFWorkbook  
    val sheet = wb.createSheet("new sheet")  
    val row = sheet.createRow(1.toShort)  
    val cell = row.createCell(1.toShort)  
    cell.setCellValue("This is a test of merging")  
  
    sheet.addMergedRegion(new CellRangeAddress(  
        1, //first row (0-based)  
        1, //last row (0-based)  
        1, //first column (0-based)  
        2  //last column (0-based)  
    ))  
  
    val fileOut = new FileOutputStream("workbook.xls")  
    wb.write(fileOut)  
    fileOut.close  
}
```





# Example 2 – Spoiwo

```
def mergingCells() = Sheet(  
    name = "new sheet"  
).withRows(  
    Row(index = 1, Cell("This is a test of merging", index=1))  
).withMergedRegions(  
    CellRange(rowRange = 1 -> 1, columnRange = 1 -> 2)  
).saveAsXlsx("workbook.xls")
```

# Roadmap

- ▶ SPOIWO/1.0: **Currently available**
  - Model => XLSX
  - Model => CSV
- ▶ SPOIWO/1.1 (Jul'14):
  - Model => HTML 4.1
- ▶ SPOIWO/1.2 (Sep'14):
  - CSV => Model
- ▶ SPOIWO/1.3 (TBD):
  - XSLX => Model

So, give it a go...

...use it...

<http://www.norbitltd.com/spoiwo>

...contribute...

<http://github.com/norbert-radyk/spoiwo>

...and enjoy reporting!



Q & A

# Thank you!

Should you have any questions or need help:

**Norbert Radyk**

E-mail: [norbert.radyk@norbitltd.com](mailto:norbert.radyk@norbitltd.com)

Mobile: +447473777787

