BSF4ooRexx/Java: Apache POI

Cookbook with Nutshell Examples

Overview

- Introduction
- Software components
- Microsoft Word
- Microsoft PowerPoint
- Microsoft Excel
- Conclusion

Introduction

- **Why using ooRexx?**

- Easy to learn und read for humans
- Learned it last semester in Business Programming 1 & 2

- **Apache POI-OpenXML4J**

- Java API for creating and manipulating Microsoft Office documents
- Components for Word (XWPF), PowerPoint (XSLF) and Excel (XSSF)

Software Components

- Open Object Rexx 5.0.0
- BSF4ooRexx850
- Azul Java Version Zulu 1.8
- Apache POI-OpenXML4J Java API binary version 5.2.3
- IntelliJ IDEA integrated development environment and ooRexx plugin

Open Object Rexx & BSF4ooRexx

- ooRexx download includes programming GUI
- Necessary to execute .rexx programs via the terminal

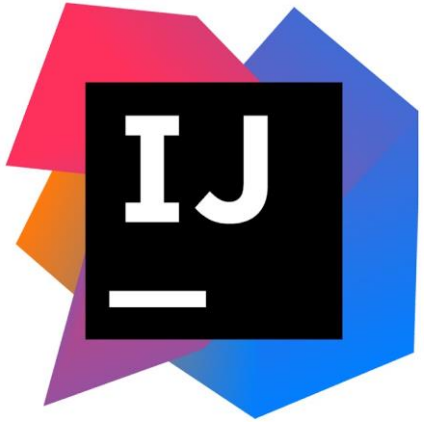
- BSF4ooRexx Bridge between ooRexx and Java
- Contains package to get access to Java classes and libraries

- Binary distributed version 5.2.3 was used to code the examples
- Java API contains all necessary .jar files to create Microsoft documents.
- Has to be connected with Java in the class path environment variable

Java

- Azul Zulu Java version 8.70.0.23 was used
- There are two restrictions when choosing a Java version:
  - Same bit rate as the used ooRexx version, 32- or 64-Bit version
  - To work correctly with Apache POI at least Java 8 is required

IntelliJ IDEA



- Additional not required software component
- Integrated development environment used for more convenient coding
- Requires an ooRexx plugin to create and execute .rexx programs

Microsoft Word

- Creating blank Word document
- Creating header and footer of a document
- Saving a created Word document
- Accessing existing Word document

Creating Blank Word Document

- Instancing main class

- Containing the most necessary methods and classes

```
15      document=.bsf~new("org.apache.poi.xwpf.usermodel.XWPFDocument")
```

Creating Header and Footer

- Requires additional class for headers and footers
- Creating default header and footer
- Setting individual text

```
16  .bsf~bsf.loadClass("org.apache.poi.wp.usermodel.HeaderFooterType", "HeaderFooterType")
17
18  -- Creating Header
19  header=document~createHeader(.HeaderFooterType~DEFAULT)
20  headerParagraph=header~createParagraph()
21  -- Add text to the header
22  headerText="This is the header of this document."
23  run=headerParagraph~createRun()
24  run~setText(headerText)
25
26  -- Creating Footer
27  footer=document~createFooter(.HeaderFooterType~DEFAULT)
28  footerParagraph=footer~createParagraph()
29  -- Add text to the footer
30  footerText="This is the footer of this document."
31  run1=footerParagraph~createRun()
32  run1~setText(footerText)
```

Saving Created Word Document

- Source of coded program is required
- Directory path is needed and concatenated with the created document
- Qualifying entire file
- Creating output document

```
49    -- Saving the output document
50    parse source . . a
51    call directory filespec('L', a)
52    filename=directory()"/First_Nutshell_Example.docx"
53    filename=qualify(filename)
54    output=.bsf~new("java.io.FileOutputStream", filename)
55    document~write(output)
```

Accessing Existing Word Document

- Source and directory of existing Word document is required
- Entire correct name has to be used to get access
- New file input stream for importing content
- Creating new document

```
15  parse source . . c
16  call directory filespec('L', c)
17  filename1=directory()"/First_Nutshell_Example.docx"
18  filename1=qualify(filename1)
19
20  olddocument=.bsf~new("java.io.FileInputStream", filename1)
21
22  -- Create a new XWPFDocument for the output
23  document=.bsf~new("org.apache.poi.xwpf.usermodel.XWPFDocument", olddocument)
```

Microsoft PowerPoint

- Creating blank PowerPoint slideshow
- Creating slides with standard layouts
- Embedding hyperlink in existing text
- Changing slideshow slide order

Creating Blank PowerPoint Slideshow

- PowerPoint presentations are called slide show in this context
- Instancing main class
- Containing the most necessary methods and classes

```
15    presentation=.bsf~new("org.apache.poi.xslf.usermodel.XMLSlideShow")
```

Creating Slides with Standard Layouts

- PowerPoint contains a slide master for standard layout slides
- Slide layout class is required
- Slide layouts are retrieved from slide master
- Creating slides with chosen layout

```
19  -- Creating slide layouts
20  template=presentation~getSlideMasters()~get(0) -- retrieves all master slides
21  layout=template~getLayout("title slide") -- Set Layout
22  layout1=template~getLayout("title only")
23  layout2=template~getLayout("title and content")
24
25  -- Creating title slide
26  titleslide=presentation~createSlide(layout)
```

Embedding Hyperlink in Existing Text

- Accessing placeholder by inserting index
- Clearing text before setting new one
- Creating hyperlink text
- Embedding link with URL

```
29    Textbox1=firstslide~getPlaceholder(1)
30    Textbox1~clearText()
31    textRun=Textbox1~addNewTextParagraph()~addNewTextRun()
32    textRun~setText("Apache POI Javadocs")
33    link=textRun~createHyperlink()
34    link~setAddress("https://poi.apache.org/apidocs/5.0/")
```

Changing Slideshow Slide Order

- PowerPoint memorizes how many slides are created and the order of the slides
- Individually addressing any slide by their index
- Changing slide order by setting new index

```
60    -- Changing slide order of the presentation
61    slides=presentation~getSlides()
62    selectedslide=slides~get(3)
63    presentation~setSlideOrder(selectedslide, 0)
```

Microsoft Excel

- Creating blank workbook and spreadsheet
- Setting individual cell format for dates
- Rotating cell content
- Creating cell content using formulars

Creating blank Workbook and Spreadsheet

- Instancing main class of workbooks
- Spreadsheet class is additionally required
- Allows to create spreadsheets with individual names

```
15  workbook=.bsf~new("org.apache.poi.xssf.usermodel.XSSFWorkbook")
16
17  spreadsheet=.bsf~new("org.apache.poi.xssf.usermodel.XSSFSheet")
18
19  -- Create a new sheet
20  spreadsheet=workbook~createSheet("Cell Types")
```

Setting Individual Cell Format for Dates

- Created cells are automatically standard cells
- Setting cell format individually
- Creating style with new cell format
- Setting format of the cell

```
62    row = spreadsheet~createRow(3)
63    cell=row~createCell(0)
64    cell~setCellValue("Set cell value DATE")
65    cell~setCellStyle(style1)
66    cell = row~createCell(1)
67    cell~setCellValue(Date())
68    style2=workbook~createCellStyle
69    helper=workbook~getCreationHelper()
70    style2~setDataFormat(helper~createDataFormat()~getFormat("m/d/yy"))
71    cell~setCellStyle(style2)
```

Rotating Cell Content

- Creating new style which contains rotation
- Any rotation angle from 0 to 360 degrees is possible
- Setting content to cells
- Applying the style to a cell

```
20    -- 90 degrees angle
21    myStyle=workbook~createCellStyle()
22    myStyle~setRotation(90)
23    row=spreadsheet~createRow(0)
24    cell=row~createCell(3)
25    cell~setCellValue("90 degrees Angle")
26    cell~setCellStyle(myStyle)
```

Creating Cell Content Using Formulars

- Possibility to create cell values via Excel built in formular rand between
- Creating 100 random numbers in column A from 0 to 1000
- Values created in cell A:3 to A:102 with a loop

```
30    -- Creating a formula to generate random numbers in column A
31    do i=3 to 102
32        row=spreadsheet~createRow(i - 1)
33        cell=row~createCell(0)
34        cell~setCellFormula("Randbetween(0, 1000)")
35    end
```

Evaluation Using Formulars

- Creation of Excel formular to sum up all created values
- Counting how many values are generated

- Counting cells with a specific value

```
37  -- Summing up the values in column A
38  row1=spreadsheet~createRow(1)
39  formulaCellSum=row1~createCell(1)
40  formulaCellSum~setCellFormula("Sum(A3:A102)")
41
42  -- Creating formula for counting non-empty cells in column A
43  formulaCellCount=row1~createCell(2)
44  formulaCellCount~setCellFormula("Count(A3:A102)")
45
46  -- Creating formula for counting values in column A at least 500
47  formulaCellCountAtLeast500=row1~createCell(3)
48  formulaCellCountAtLeast500~setCellFormula("Countif(A3:A102, ""=>500"")")
```

Conclusion

- Includes only a small part of what is possible with Apache POI
- Apache Software Foundation is constantly developing the API
- Source packages for Apache POI version 5.2.4 & 5.2.5 are currently available on the homepage