SXSSFWorkbook：生成大Excel的低内存实现

背景

最近项目中出现一个线上bug，用户反馈Excel文件无法导出。查看日志后发现`java.lang.OutOfMemoryError: Java heap space`。excel占用如此大内存，是因为将图片也下载到了excel中方便用户查看。组内讨论后决定做一些定制化的修改，将excel行保存到文件中。中途发现已经有现成的解决方案：`SXSSFWorkbook`。将`XSSFWorkbook`替换成`SXSSFWorkbook`后，快速解决了问题

如何使用

将`XSSFWorkbook`替换为`SXSSFWorkbook`即可，也可指定参数

```
/**
 * rowAccessWindowSize：保留在内存中的行数
 * compressTmpFiles：是否压缩临时文件，压缩后为.gz，如果临时文件过大可以进行压缩
 * useSharedStringsTable：是否使用共享字符串，开启后，所有不同的字符串都会留一份在内存，可能会占用更多内存，如果值有大量重复，比如性别为男女，可以开启
 */
public SXSSFWorkbook(XSSFWorkbook workbook,
                     int rowAccessWindowSize,
                     boolean compressTmpFiles,
                     boolean useSharedStringsTable);
```

使用完记得使用`workbook.dispose()`删除临时数据文件

```
try {
```

```
    response.setHeader("Content-disposition", "attachment;filename=" + fileName + ".xlsx");
    OutputStream os = response.getOutputStream();
    workbook.write(out);
} catch (IOException e) {
    log.error(e);
} finally {
    try {
        out.close();
        workbook.dispose(); // 删除临时文件
    } catch (Exception e){
        log.error(e);
    }
}
```

Workbook对比

Workbook	Excel版本	扩展名	文件格式	备注
HSSFWorkbook	Excel 97-2003	.xls	二进制	最多65535行，256列
XSSFWorkbook	Excel 2007起	.xlsx	openXML	最多1048576行，16384列
SXSSFWorkbook	Excel 2007起	.xlsx	openXML	低内存占用的`XSSFWorkbook`，需`dispose`
SXSSFWorkbookWithCustomZipEntrySource	Excel 2007起	.xlsx	openXML	指定zip压缩方式的`SXSSFWorkbook`,大大减少磁盘使用空间，但会使性能变差

原理
--

SXSSF是XSSF的低内存占用版，只会在内存中保存最新的x条rows，前面的rows已被写入磁盘，不可访问

内存中保留的行数`_randomAccessWindowSize`默认为100，创建`SXSSFWorkbook`时可以指定大小，只能为-1或正数

```
private int _randomAccessWindowSize = DEFAULT_WINDOW_SIZE; //
100

rivate void setRandomAccessWindowSize(int rowAccessWindowSize) {
    if(rowAccessWindowSize == 0 || rowAccessWindowSize < -1) {
        throw new IllegalArgumentException("rowAccessWindowSize must
be greater than 0 or -1");
    }
    _randomAccessWindowSize = rowAccessWindowSize;
}
```

创建row时会检查内存中的行数，超过指定数量则写入临时数据文件。`_randomAccessWindowSize=-1`则全留在内存

```
if(_randomAccessWindowSize >= 0 && _rows.size() >
_randomAccessWindowSize) {
    try {
        flushRows(_randomAccessWindowSize);
    } catch (IOException ioe) {
        throw new RuntimeException(ioe);
    }
}
```

合并单元格

如果需要合并单元格，正常使用`addMergedRegion`即可，即使row已经写入文件也没关系。实际上合并单元格只需要记录行列， 并不会把相关所有row都放到内存计算。例如合并80000行，也只需记录为`<mergeCell ref="C2:D80001"/>`

```
sheet.addMergedRegion(new CellRangeAddress(1, 80000, 2, 3));
```

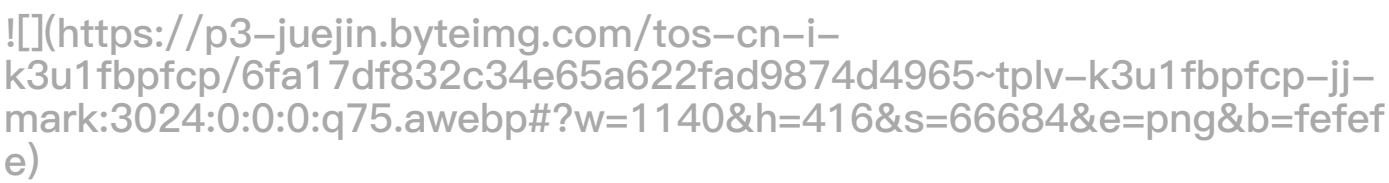
临时sheet数据文件

`createSheet()`时会在`{java.io.tmpdir}/poifiles`下生成`poi-sxssf-sheet{randomLong}.xml`，`createRow()`时若满足条件会持续写入文件，最后`wb.dispose()`时删除

> `java.io.tmpdir`默认值
>
>
> * Windows：C:\Users\xxx\AppData\Local\Temp\poifiles
> * Linux：/tmp

```
private void createPOIFilesDirectory() throws IOException {
    if (this.dir == null) {
        String tmpDir = System.getProperty("java.io.tmpdir");
        if (tmpDir == null) {
            throw new IOException("Systems temporary directory not defined - set the -Djava.io.tmpdir jvm property!");
        }
        this.dir = new File(tmpDir, "poifiles");
    }
    this.createTempDirectory(this.dir);
}

public File createTempFile() throws IOException {
    return TempFile.createTempFile("poi-sxssf-sheet", ".xml");
}
```



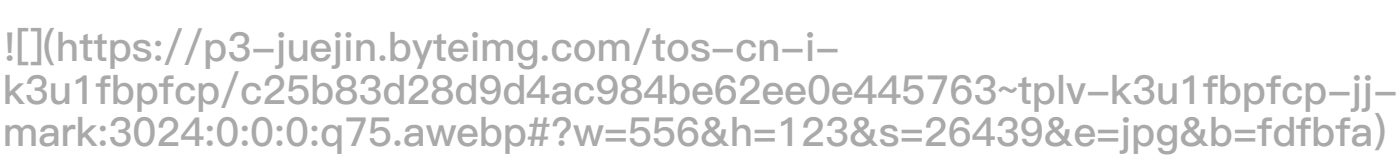
> 格式openXML可参考:
>
>
> [learn.microsoft.com/en-us/dotne...](http://cxyroad.com/"https://learn.microsoft.com/en-us/dotnet/api/documentformat.openxml.spreadsheet.cell?view=openxml-3.0.1")

临时excel模板文件

调用`workbook.write()`时，首先把所有row刷到磁盘，然后生成模板文件`poi-sxssf-template{randomLong}.xlsx`。此时文件夹下同时存在临时数据文件和模板文件。模板文件包含了excel的基础信息，比如合并单元格信息，共享字符串信息，样式信息等。无需手动调用`dispose`删除

```
public void write(OutputStream stream) throws IOException {
    flushSheets();

    File tmplFile = TempFile.createTempFile("poi-sxssf-template",
".xlsx");
    boolean deleted;
    try {
        try (FileOutputStream os = new FileOutputStream(tmplFile)) {
            _wb.write(os);
        }
        try (
            ZipSecureFile zf = new ZipSecureFile(tmplFile);
            ZipFileZipEntrySource source = new ZipFileZipEntrySource(zf)
        ) {
             injectData(source, stream);
        }
    } finally {
        deleted = tmplFile.delete(); // 删除模板
    }
    if (!deleted) {
        throw new IOException("Could not delete temporary file after
processing: " + tmplFile);
    }
}
```



如果直接打开模版excel会发现是空的excel，有合并单元格等样式效果，可是共享字符串存放在哪里，xlsx的文件格式是什么样的呢？出于好奇，将模版文件从`.xlsx`改为`.zip`再解压，发现了新世界。xlsx的文件目录如下

```
|    [Content_Types].xml
|
| ———————_rels
|        .rels
|
|———————docProps
|        app.xml
|        core.xml
|
|———————xl
|    |    sharedStrings.xml
|    |    styles.xml
|    |    workbook.xml
|    |
|    |———————_rels
|    |        workbook.xml.rels
|    |
|    |———————worksheets
|         sheet1.xml
```

打开`xl/sharedStrings.xml`后确实发现了共享字符串

```
<?xml version="1.0" encoding="UTF-8"?>
<sst count="100000" uniqueCount="1"
xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
>
  <si><t>共享字符串1</t></si>
</sst>
```

合并单元格信息在`xl/worksheets/sheet1.xml中`，可以看到 `<mergeCell ref="C2:D80001"/>`，表示从`C1`合并至`D80001`

```
<?xml version="1.0" encoding="UTF-8"?>
<worksheet
```

```
xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
>
    <dimension ref="A1:CV1000"/>
    <sheetViews>
        <sheetView workbookViewId="0" tabSelected="true"/>
    </sheetViews>
    <sheetFormatPr defaultRowHeight="15.0"/>
    <sheetData/>
    <mergeCells count="1">
        <mergeCell ref="C2:D80001"/>
    </mergeCells>
    <pageMargins bottom="0.75" footer="0.3" header="0.3" left="0.7"
right="0.7" top="0.75"/>
</worksheet>
```

共享字符串

`useSharedStringsTable`设置为true后，cell中的字符串会放在内存中，变量为`_sharedStringSource`，临时数据文件中的`<v></v>`保存的是共享字符串index，`t`从`inlineStr`变为`s`

删除临时sheet数据文件

使用`SXSSFWorkbook`结束后一定要调用`workbook.dispose()`，会遍历excel所有sheet的临时数据文件进行删除

```
public boolean dispose() {
    boolean success = true;
    for (SXSSFSheet sheet : _sxFromXHash.keySet()){
        try {
            success = sheet.dispose() && success;
        } catch (IOException e) {
```

```
            LOG.atWarn().withThrowable(e).log("Failed to dispose sheet");
            success = false;
        }
    }
    return success;
}
```

原文链接: https://juejin.cn/post/7372734627164110858