Санкт-Петербургский государственный университет

кафедра системного программирования

Свитков Сергей Андреевич

Реализация библиотеки для потоковой обработки .xlsx файлов

Курсовая работа

Научный руководитель:
ст. преп к.т.н. Литвинов Ю. В.

Санкт-Петербург
2018

Оглавление

Введение **3**

1. Обзор существующих решений **4**

1.1. Apache POI 4

1.2. SJXLSX 4

1.3. Итоги обзора 5

2. Постановка задачи **6**

3. Анализ формата XLSX **7**

3.1. Workbook 7

3.2. Worksheet 8

4. Реализация **9**

4.1. Алгоритм 9

4.2. Архитектура 9

4.3. Реализация библиотеки 9

5. Апробация **10**

Заключение **11**

Листинги **12**

Список литературы **13**

Введение

В современном мире большой популярностью пользуются многопользовательские веб-приложения. Приложения такого рода могут использоваться для самых разнообразных целей — от совместного редактирования документов несколькими пользователями до анализа различной статистики операторами связи.

Одну из категорий веб-приложений представляют проекты в сфере телекоммуникаций и биллинга. Такие приложения используются операторами связи для анализа различной статистики по действиям абонентов: перемещения между зонами роуминга, количество входящих/исходящих вызовов, и т.д.

Для формирования отчётов требуется формат представления данных, предоставляющий строгое структурирование. К таким можно отнести JSON, XML, XLSX. Однако, следует принять во внимание, что отчетность или статистика, представленная в таком формате, может использоваться как при взаимодействии различных компонент приложения или различных приложений, так и для анализа человеком. Преимущество формата XLSX в том, что для открытия таких файлов существуют общеизвестные решения (Microsoft Excel, OpenOffice Calc, Google Sheets).

Таким образом, возникает необходимость в библиотеке, которая позволила бы формировать документы формата XLSX. Следует отметить, что, поскольку объем данных не ограничен, а веб-приложения являются многопользовательскими, то необходимо формировать файл в потоковом режиме, то есть, держать в оперативной памяти только ограниченное количество данных.

Исходя из сформулированных требований было принято решение проанализировать существующие решения в данной области и, в случае отстутствия подходящей реализации, создать свою библиотеку для решения подобных задач.

1. Обзор существующих решений

Задача формирования документов формата XLSX не является новой, имеется ряд существующих библиотек.

1.1. Apache POI

Библиотека Apache POI (далее — просто POI)предоставляет средства как для формирования, так и для чтения файлов формата .xlsx . До версии 3.8 в библиотеке отсутствовала поддержка потоковой обработки файлов. Несмотря на то, что, начиная с версии 3.8, появилась поддержка потоковой обработки, некоторые проблемы с использованием оперативной памяти остались. Так, некоторые операции над документами всё равно можно проводить только храня целый документ в памяти. К недостаткам библиотеки можно отнести отсутствие возможности задать условия автоматического создания новых страниц в документе, а так же отсутствие полной документации и примеров использования.

Было проведено тестирование библиотеки на количество памяти, требуемое для формирования XLSX файла размером в 1 млн рядов из 50 колонок. Результаты на [рис. 1] //построить график. Из эксперимента можно сделать выводы о том, что, несмотря на наличие возможности потокового формирования файла, с памятью всё равно есть проблемы.

На момент начала данной работы не было известно о том, что в POI реализована поддержка потоковой обработки документов, что во многом и послужило мотивацией для данной задачи. После анализа последней версии библиотеки было решено реализовать собственный алгоритм потокового формирования документов и сравнить полученную реализацию с POI по производительности.

1.2. SJXLSX

Библиотека SJXLSX — разработка коммьюнити. Документации в проекте крайне мало, к тому же последнее обновление в репозитории

было сделано в 2015м году. Был проведен эксперимент по проверке на количество затрачиваемой памяти, результаты эксперимента [рис. 2] // ещё один график. Из эксперимента можно сделать выводы о том, что по количеству используемой операционной памяти при генерации файла данная реализация является крайне неэффективной.

1.3. Итоги обзора

Исходя из результатов обзора было принято решение о реализации библиотеки для потоковой генерации файлов и последующем сравнении её с двумя существующими решениями.

2. Постановка задачи

Целью данной работы является разработка библиотеки для потоковой обработки файлов формата xlsx и сравнение её с существующими реализациями. Для достижения этой цели были поставлены следующие задачи:

- сформулировать подход, который будет использоваться для реализации библиотеки;
- реализовать библиотеку;
- провести апробацию полученной реализации;
- сравнить полученную реализацию с существующими по метрикам:
  - потребление RAM при создании документа;
  - скорость работы;
- разместить исходный код и примеры использования библиотеки на github;

3. Анализ формата XLSX

Перед началом реализации библиотеки было необходимо изучить структуру формата XLSX Формат XLSX был создан в декабре 2006 года при участии Microsoft, Ecma, ISO/IEC. К сожалению, документации по стандарту MS-XLSX крайне мало, в открытом доступе можно найти только два стандарта: ISO/IEC 29500 (2008) и ECMA-376 1st edition (2006). Эти документы имеют крайне большой объем (порядка 7 тысяч страниц), и на их изучение ушло бы большое количество времени. Для формата XLSX так же существует другой стандарт, Open Office XML. В отличие от стандарта MS-XLSX, данный стандарт довольно хорошо документирован, поэтому было принято решение формировать документы в соответствии с данным стандартом.

Формат XLSX представляет собой ZIP-архив с XML файлами. Его структура (рис. 1) представляет собой следующий набор файлов и директорий:

- `Content_Types`.xml — типы контента в архиве и пути к ним;
- `_rels` — зависимости между файлами внутри архива;
- docProps — метаданные: имя автора, дата создания, ...;
- xl — директория с основными файлами архива: workbook, страницы, стили, таблицы;

Рассмотрим более подробно основные компоненты архива: Workbook и Worksheet.

3.1. Workbook

Workbook представляет собой XML-файл, который не содержит данных файла, но содержит следующие мета-данные: ссылки на отдельные Worksheet и их свойства. Пример файла workbook.xml: листинг 1. Контент документа содержится непосредственно в Worksheets.

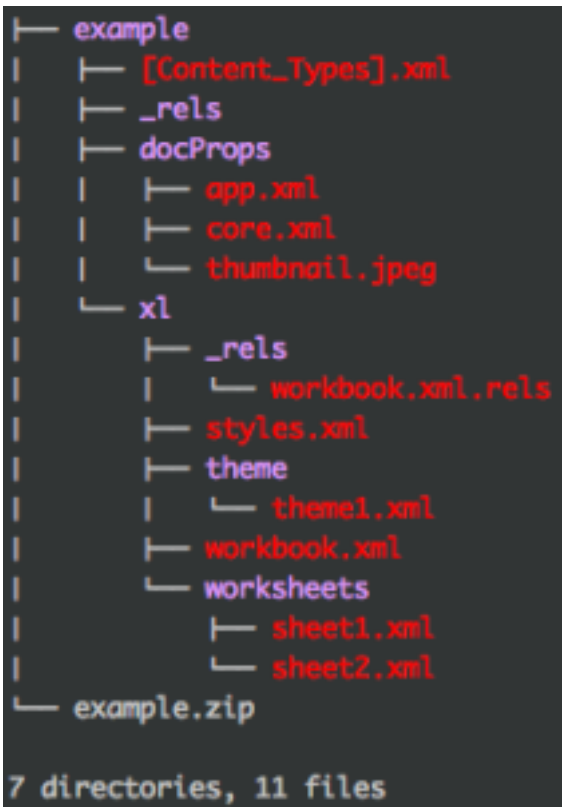

Рис. 1: Структура формата XLSX

3.2. Worksheet

Worksheet содержат данные, из которых и состоит документ. Worksheet может иметь один из следующих форматов: grid, chart, dialog sheet. Наиболее популярным и хорошо задокументированным является grid, рассмотрим его более подробно.

Grid представляет собой ”сетку” из ”клеток” (cells) с данными. Каждая клетка может содержать какой-то определенный тип данных: числа, булевские переменные, формулы, и т.д.. Для оптимизации использования памяти строковые значения хранятся не в теле самой клетки, а в отдельной части документа. Это позволяет минимизировать дупликацию строк. Пример файла worksheet.xml (листинг 2)

Закончив анализ формата XLSX можно приступить к реализации библиотеки.

4. Реализация

В данной секции будут описаны алгоритм работы и архитектура библиотеки. Исходный код реализации библиотеки опубликован на Github. Работа велась под учётной записью likeanowl.

4.1. Алгоритм

4.2. Архитектура

4.3. Реализация библиотеки

5. Апробация

Заключение

В ходе данной работы были достигнуты следующие результаты:

Листинги

```
<workbook . . .>
    . . .
    <workbookPr . . ./>
    <sheets>
        <sheet name=''sheet1'' r:id=''rId1''>
        <sheet name=''sheet2'' r:id=''rId2''>
        <sheet name=''sheet3'' r:id=''rId3''>
    </sheets>
    . . .
</workbook>
```

Листинг 1: Пример файла workbook.xml

```
<worksheet . . .>
. . .
    <cols>
        <col min=''1'' max=''1'' width=''26.140625'' customWidth=''1''/>
            . . .
        </cols>

        <sheetData>
            <row r=''1''>
            <c r=''A1'' s=''1'' t=''s''>
            <v>0</v>
            . . .
            </c>
            </row>
            . . .
        </sheetData>
        . . .

        <mergeCells count=''1''>
        <mergeCell ref=''B12:J16''/>
        </mergeCells>

        <pageMargins . . ./>
        <pageSetup . . ./>

        <tableParts ccount=''1''>
        <tableParts count=''1''>
    </tablePart r:id=''rId2''/>
</worksheet>
```

Листинг 2: Пример файла worksheet.xml

Список литературы