POI-HSSF and POI-XSSF - Java API To Access Microsoft Excel Format Files

Overview

by Andrew C. Oliver, Nicola Ken Barozzi

1. Overview

HSSF is the POI Project's pure Java implementation of the Excel '97(-2007) file format. XSSF is the POI Project's pure Java implementation of the Excel 2007 OOXML (.xlsx) file format.

HSSF and XSSF provides ways to read spreadsheets create, modify, read and write XLS spreadsheets. They provide:

- low level structures for those with special needs
- an eventmodel api for efficient read-only access
- a full usermodel api for creating, reading and modifying XLS files

For people converting from pure HSSF usermodel, who wish to use the joint SS Usermodel for HSSF and XSSF support, then see the ss usermodel converting guide.

An alternate way of generating a spreadsheet is via the Cocoon serializer (yet you'll still be using HSSF indirectly). With Cocoon you can serialize any XML datasource (which might be a ESQL page outputting in SQL for instance) by simply applying the stylesheet and designating the serializer.

If you're merely reading spreadsheet data, then use the eventmodel api in either the org.apache.poi.hssf.eventusermodel package, or the org.apache.poi.xssf.eventusermodel package, depending on your file format.

If you're modifying spreadsheet data then use the usermodel api. You can also generate spreadsheets this way.

Note that the usermodel system has a higher memory footprint than the low level eventusermodel, but have the major advantage of being much simpler to work with. Also please be aware that as the new XSSF supported Excel 2007 OOXML (.xlsx) files are XML based, the memory footprint for processing them is higher than for the older HSSF supported

(.xls) binary files.

2. SXSSF (Since POI 3.8 beta3)

Since 3.8-beta3, POI provides a low-memory footprint SXSSF API built on top of XSSF.

SXSSF is an API-compatible streaming extension of XSSF to be used when very large spreadsheets have to be produced, and heap space is limited. SXSSF achieves its low memory footprint by limiting access to the rows that are within a sliding window, while XSSF gives access to all rows in the document. Older rows that are no longer in the window become inaccessible, as they are written to the disk.

In auto-flush mode the size of the access window can be specified, to hold a certain number of rows in memory. When that value is reached, the creation of an additional row causes the row with the lowest index to to be removed from the access window and written to disk. Or, the window size can be set to grow dynamically; it can be trimmed periodically by an explicit call to flushRows(int keepRows) as needed.

Due to the streaming nature of the implementation, there are the following limitations when compared to XSSF:

- Only a limited number of rows are accessible at a point in time.
- Sheet.clone() is not supported.
- Formula evaluation is not supported

See more details at SXSSF How-To

The table below synopsizes the comparative features of POI's Spreadsheet API:

Spreadsheet API Feature Summary

Spreadsheet API Feature Summary