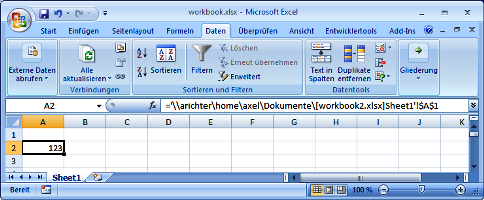
**ADVANTAGE OF APACHE POI OVER JEXCEL API**



**Advantage of Apache POI over jExcel API**

Here are the Advantage of Apache POI over jExcel API. The following article will shown my valid reasons

After further research, I found that [Apache POI](https://poi.apache.org/) is the way to go for following main reasons. There are some other reasons related to advanced features but let’s not go into that much detail.

* Backing of Apache foundation.
* [JExcel](http://jexcelapi.sourceforge.net/) doesn’t support xlsx format whereas POI supports both xls and xlsx formats.
* Apache POI provides stream-based processing, that is suitable for large files and requires less memory.
* Apache POI provides excellent support for working with Microsoft Excel documents and it’s able to handle both XLS and XLSX formats of spreadsheets.

Some important points about Apache POI API are:

* + Apache POI contains HSSF implementation for Excel ’97(-2007) file format i.e XLS.
  + Apache POI XSSF implementation should be used for Excel 2007 OOXML (.xlsx) file format.
  + Apache POI HSSF and XSSF API provides mechanisms to read, write or modify excel spreadsheets.
  + Apache POI also provides SXSSF API that is an extension of XSSF to work with very large excel sheets.
  + SXSSF API requires less memory and is suitable when working with very large spreadsheets and heap memory is limited.

There are two models to choose from – event model and user model. Event model requires less memory because the excel file is read in tokens and requires processing them. User model is more object oriented and easy to use and we will use this in our examples.  
Apache POI provides excellent support for additional excel features such as working with Formulas, creating cell styles by filling colors and borders, fonts, headers and footers, data validations, images, hyperlinks etc.

**Stack overflow:**

First, here are the things where both APIs have the same end functionality:

* Both are free
* Cell styling: alignment, backgrounds (colors and patterns), borders (types and colors), font support (font names, colors, size, bold, italic, strikeout, underline)
* Formulas
* Hyperlinks
* Merged cell regions
* Size of rows and columns
* Data formatting: Numbers and Dates
* Text wrapping within cells
* Freeze Panes
* Header/Footer support
* Read/Write existing and new spreadsheets
* Both attempt to keep existing objects in spreadsheets they read in intact as far as possible.

However, there are many differences:

* Perhaps the most significant difference is that Java JXL does not support the Excel 2007+ ".xlsx" format; it only supports the old BIFF (binary) ".xls" format. Apache POI supports both with a common design.
* Additionally, the Java portion of the JXL API was last updated in 2009 (3 years, 4 months ago as I write this), although it looks like there is a C# API. Apache POI is actively maintained.
* JXL doesn't support Conditional Formatting, Apache POI does, although this is not that significant, because you can conditionally format cells with your own code.
* JXL doesn't support *rich* **text** formatting, i.e. different formatting within a text string; Apache POI does support it.
* JXL only supports certain text rotations: horizontal/vertical, +/- 45 degrees, and stacked; Apache POI supports any integer number of degrees plus stacked.
* JXL doesn't support drawing shapes; Apache POI does.
* JXL supports most Page Setup settings such as Landscape/Portrait, Margins, Paper size, and Zoom. Apache POI supports all of that plus Repeating Rows and Columns.
* JXL doesn't support Split Panes; Apache POI does.
* JXL doesn't support Chart creation or manipulation; that support isn't there yet in Apache POI, but an API is slowly starting to form.
* Apache POI has a more extensive set of documentation and examples available than JXL.

Additionally, POI contains not just the main "usermodel" API, but also an event-based API if all you want to do is read the spreadsheet content.

In conclusion, because of the better documentation, more features, active development, and Excel 2007+ format support, I use Apache POI.

**Apache POI Quick Guide –TutorilasPoint**

**Interview Question**

Differences between jxl and poi....

* The most significant difference is that Java JXL does not support the Excel 2007+ ".xlsx" format; it only supports the old BIFF (binary) ".xls" format. Apache POI supports both with a common design.
* JXL only supports certain text rotations: horizontal/vertical, +/- 45 degrees, and stacked; Apache POI supports any integer number of degrees plus stacked.
* Additionally, the Java portion of the JXL API was last updated in 2009 (3 years, 4 months ago as I write this), although it looks like there is a C# API. Apache POI is actively maintained
* Google---create excel workbook using java
* Google--how to freeze the first row using java
* Google-- apache poi javatpoint
* <http://viralpatel.net/blogs/java-read-write-excel-file-apache-poi/>
* <https://www.mkyong.com/java/apache-poi-reading-and-writing-excel-file-in-java/>
* <https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFWorkbook.html>
* <http://www.codejava.net/coding/how-to-write-excel-files-in-java-using-apache-poi>
* <https://poi.apache.org/spreadsheet/index.html>
* <https://poi.apache.org/spreadsheet/quick-guide.html>
* <http://www.javavillage.in/splits-and-freeze-pane-in-excel-using-apache-poi.php>
* <https://poi.apache.org/spreadsheet/>
* <https://en.wikipedia.org/wiki/Apache_POI>
* <http://javabeat.net/apache-poi-reading-excel-sheet-using-java/>
* <http://javabeginnerstutorial.com/code-base/write-excel-file/>
* <http://www.avajava.com/tutorials/lessons/how-do-i-read-from-an-excel-file-using-poi.html>
* <http://www.codejava.net/coding/how-to-read-excel-files-in-java-using-apache-poi>
* <http://thinktibits.blogspot.in/2014/08/apache-poi-freeze-pane-excel-java.html>
* <http://www.journaldev.com/2562/apache-poi-tutorial>