# Frequently Asked Questions

**Questions**



**Answers**

## 1. Why is reading a simple sheet taking so long?

You've probably enabled logging. Logging is intended only for autopsie style debugging. Having it enabled will reduce performance by a factor of at least 100. Logging is helpful for understanding why POI can't read some file or developing POI itself. Important errors are thrown as exceptions, which means you probably don't need logging.

## 2. What is the HSSF "eventmodel"?

The HSSF eventmodel package is a new API for reading XLS files more efficiently. It does require more knowledge on the part of the user, but reduces memory consumption by more



Copyright © 2002-2007 The Apache Software Foundation All rights reserved.

than tenfold. It is based on the AWT event model in combination with SAX. If you need read-only access to a given XLS file, this is the best way to do it.

## 3. Why can't read the document I created using Star Office 5.1?

Star Office 5.1 writes some records using the older BIFF standard. This causes some problems with POI which supports only BIFF8.

## 4. Why am I getting an exception each time I attempt to read my spreadsheet?

It's possible your spreadsheet contains a feature that is not currently supported by HSSF. For example - spreadsheets containing cells with rich text are not currently supported.

## 5. Does HSSF support protected spreadsheets?

Protecting a spreadsheet encrypts it. We wont touch encryption because we're not legally educated and don't understand the full implications of trying to implement this. If you wish to have a go at this feel free to add it as a plugin module. We wont be hosting it here however.

## 6. How do you tell if a field contains a date with HSSF?

Excel stores dates as numbers therefore the only way to determine if a cell is actually stored as a date is to look at the formatting. There is a helper method in HSSFDateUtil (since the 1.7.0-dev release) that checks for this. Thanks to Jason Hoffman for providing the solution.

```
case HSSFCell.CELL_TYPE_NUMERIC:
      double d = cell.getNumericCellValue();
      // test if a date!
      if (HSSFDateUtil.isCellDateFormatted(cell)) {
        // format in form of M/D/YY
        cal.setTime(HSSFDateUtil.getJavaDate(d));
        cellText =
          (String.valueOf(cal.get(Calendar.YEAR))).substring(2);
        cellText = cal.get(Calendar.MONTH)+1 + "/" +
                   cal.get(Calendar.DAY_OF_MONTH) + "/" +
                   cellText;
      }
```

## 7. I'm trying to stream an XLS file from a servlet and I'm having some trouble. What's the problem?

The problem usually manifests itself as the junk characters being shown on screen. The



Copyright © 2002-2007 The Apache Software Foundation All rights reserved.

problem persists even though you have set the correct mime type.

The short answer is, don't depend on IE to display a binary file type properly if you stream it via a servlet. Every minor version of IE has different bugs on this issue.

The problem in most versions of IE is that it does not use the mime type on the HTTP response to determine the file type; rather it uses the file extension on the request. Thus you might want to add a **.xls** to your request string. For example *http://yourserver.com/myServelet.xls?param1=xx*. This is easily accomplished through URL mapping in any servlet container. Sometimes a request like *http://yourserver.com/myServelet?param1=xx&dummy=file.xls* is also known to work.

To guarantee opening the file properly in Excel from IE, write out your file to a temporary file under your web root from your servelet. Then send an http response to the browser to do a client side redirection to your temp file. (Note that using a server side redirect using RequestDispatcher will not be effective in this case)

Note also that when you request a document that is opened with an external handler, IE sometimes makes two requests to the webserver. So if your generating process is heavy, it makes sense to write out to a temporary file, so that multiple requests happen for a static file.

None of this is particular to Excel. The same problem arises when you try to generate any binary file dynamically to an IE client. For example, if you generate pdf files using FOP, you will come across many of the same issues.

## 8. I want to set a cell format (Data format of a cell) of a excel sheet as ###,###,###.#### or ###,###,###.0000. Is it possible using POI ?

Yes. You first need to get a HSSFDataFormat object from the workbook and call getFormat with the desired format. Some examples are here.

## 9. I want to set a cell format (Data format of a cell) of a excel sheet as text. Is it possible using POI ?

Yes. This is a built-in format for excel that you can get from HSSFDataFormat object using the format string "@". Also, the string "text" will alias this format.

## 10. How do I add a border around a merged cell?

Add blank cells around where the cells normally would have been and set the borders individually for each cell. We will probably enhance HSSF in the future to make this process easier.



Copyright © 2002-2007 The Apache Software Foundation All rights reserved.

## 11. I tried to set cell values and Excel sheet name on my native language, but I failed to do it. :(

By default HSSF uses cell values and sheet names as compressed unicode, so to support localization you should use Unicode. To do it you should set it manually:

```
                // for sheet name
                HSSFWorkbook wb = new HSSFWorkbook();
                HSSFSheet s = wb.createSheet();
                wb.setSheetName( 0, "SomeUnicodeName", HSSFWorkbook.ENCODING_UTF_16 );

                // for cell value
                HSSFRow r = s.createRow( 0 );
                HSSFCell c = r.createCell( (short)0 );
                c.setCellType( HSSFCell.CELL_TYPE_STRING );
                c.setEncoding( HSSFCell.ENCODING_UTF_16 );
                c.setCellValue( "\u0422\u0435\u0441\u0442\u043E\u0432\u0430\u044F" );
```

Make sure you make the call to setEncoding() before calling setCellValue(), otherwise what you pass in won't be interpreted properly.

## 12. I'm having trouble creating a spreadsheet using POI using websphere 3.5.

Make sure you have fix pack 4 installed.

## 13. I am using styles when creating a workbook in POI, but Excel refuses to open the file, complaining about "Too Many Styles".

You just create the styles OUTSIDE of the loop in which you create cells.

GOOD:

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("new sheet");
HSSFRow row = null;

// Aqua background
HSSFCellStyle style = wb.createCellStyle();
style.setFillBackgroundColor(HSSFColor.AQUA.index);
style.setFillPattern(HSSFCellStyle.BIG_SPOTS);
HSSFCell cell = row.createCell((short) 1);
cell.setCellValue("X");
cell.setCellStyle(style);

// Orange "foreground", foreground being the fill foreground not the font color.
style = wb.createCellStyle();
```



Copyright © 2002-2007 The Apache Software Foundation All rights reserved.

```
style.setFillForegroundColor(HSSFColor.ORANGE.index);
style.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);

for (int x = 0; x < 1000; x++) {

// Create a row and put some cells in it. Rows are 0 based.
        row = sheet.createRow((short) k);

        for (int y = 0; y < 100; y++) {
               cell = row.createCell((short) k);
                cell.setCellValue("X");
               cell.setCellStyle(style);
        }
}

// Write the output to a file
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```

BAD:

```
HSSFWorkbook wb = new HSSFWorkbook();
HSSFSheet sheet = wb.createSheet("new sheet");
HSSFRow row = null;

for (int x = 0; x < 1000; x++) {
        // Aqua background
        HSSFCellStyle style = wb.createCellStyle();
        style.setFillBackgroundColor(HSSFColor.AQUA.index);
        style.setFillPattern(HSSFCellStyle.BIG_SPOTS);
        HSSFCell cell = row.createCell((short) 1);
        cell.setCellValue("X");
        cell.setCellStyle(style);

        // Orange "foreground", foreground being the fill foreground not the font colo
        style = wb.createCellStyle();
        style.setFillForegroundColor(HSSFColor.ORANGE.index);
        style.setFillPattern(HSSFCellStyle.SOLID_FOREGROUND);

        // Create a row and put some cells in it. Rows are 0 based.
        row = sheet.createRow((short) k);

        for (int y = 0; y < 100; y++) {
                cell = row.createCell((short) k);
                cell.setCellValue("X");
                cell.setCellStyle(style);
        }
}

// Write the output to a file
FileOutputStream fileOut = new FileOutputStream("workbook.xls");
wb.write(fileOut);
fileOut.close();
```



Copyright © 2002-2007 The Apache Software Foundation All rights reserved.

## 14. Will POI read any spreadsheet and rewrite it with modifications.

POI is not guanteed to read the contents of any spreadsheet. Certain features may cause spreadsheets to fail to read. More problematic is rewriting spreadsheets. POI tried hard to preserve the records of the original spreadsheet but some features may cause problems. We advise that you limit the formatting of spreadsheets you process so as to not be unpleasantly suprised at a later stage.



*Copyright © 2002-2007 The Apache Software Foundation All rights reserved.*