Frequently Asked Questions

Questions

Answers

1. My code uses some new feature, compiles fine but fails when live with a "MethodNotFoundException" or "IncompatibleClassChangeError"

You almost certainly have an older version of POI on your classpath. Quite a few runtimes and other packages will ship an older version of POI, so this is an easy problem to hit without

your realising.

The best way to identify the offending earlier jar file is with a few lines of java. These will load one of the core POI classes, and report where it came from.

```
ClassLoader classloader =
   org.apache.poi.poifs.filesystem.POIFSFileSystem.class.getClassLoader();
URL res = classloader.getResource(
            "org/apache/poi/poifs/filesystem/POIFSFileSystem.class");
String path = res.getPath();
System.out.println("Core POI came from " + path);
```

2. My code uses the scratchpad, compiles fine but fails to run with a "MethodNotFoundException"

You almost certainly have an older version earlier on your classpath. See the prior answer.

3. I'm using the poi-ooxml-schemas jar, but my code is failing with "java.lang.NoClassDefFoundError: org/openxmlformats/schemas/*something*"

To use the new OOXML file formats, POI requires a jar containing the file format XSDs, as compiled by XMLBeans. These XSDs, once compiled into Java classes, live in the *org.openxmlformats.schemas* namespace.

There are two jar files available, as described in the components overview section. The *full jar of all of the schemas is ooxml-schemas-1.1.jar*, and it is currently around 15mb. The *smaller poi-ooxml-schemas jar* is only about 4mb. This latter jar file only contains the typically used parts though.

Many users choose to use the smaller poi-ooxml-schemas jar to save space. However, the poi-ooxml-schemas jar only contains the XSDs and classes that are typically used, as identified by the unit tests. Every so often, you may try to use part of the file format which isn't included in the minimal poi-ooxml-schemas jar. In this case, you should switch to the full ooxml-schemas-1.1.jar. Longer term, you may also wish to submit a new unit test which uses the extra parts of the XSDs, so that a future poi-ooxml-schemas jar will include them.

There are a number of ways to get the full ooxml-schemas-1.1.jar. If you are a maven user, see the the components overview section for the artifact details to have maven download it for you. If you download the source release of POI, and/or checkout the source code from subversion, then you can run the ant task "compile-ooxml-xsds" to have the OOXML schemas downloaded and compiled for you (This will also give you the XMLBeans

generated source code, in case you wish to look at this). Finally, you can download the jar by hand from the POI Maven Repository.

Note that for POI 3.5 and 3.6, the full ooxml schemas jar was named ooxml-schemas-1.0.jar. For POI 3.7, the filename was bumped to ooxml-schemas-1.1.jar when generics support was added. You can use ooxml-schemas-1.1.jar with POI 3.5 and 3.6 if you wish, but POI 3.7 won't wokr with ooxml-schemas-1.0.jar (it needs thew newer one).

4. Why is reading a simple sheet taking so long?

You've probably enabled logging. Logging is intended only for autopsy 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.

5. What is the HSSF "eventmodel"?

The SS eventmodel package is an API for reading Excel files without loading the whole spreadsheet into memory. It does require more knowledge on the part of the user, but reduces memory consumption by more than tenfold. It is based on the AWT event model in combination with SAX. If you need read-only access, this is the best way to do it.

6. 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.

7. 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 POI. If you encounter this then please create the simplest file that demonstrates the trouble and submit it to Bugzilla.

8. How do you tell if a spreadsheet cell contains a date?

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 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;
        }
```

9. 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 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.

10. 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 DataFormat object from the workbook and call getFormat with the desired format. Some examples are here.

11. 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 DataFormat object using the format string "@". Also, the string "text" will alias this format.

12. 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.

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();
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 color.
   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();
```

14. An OLE2 ("binary") file is giving me problems, but I can't share it. How can I investigate the problem on my own?

The first thing to try is running the Binary File Format Validator from Microsoft against the

file, which will report if the file complies with the specification. If your input file doesn't, then this may well explain why POI isn't able to process it correctly. You should probably in this case speak to whoever is generating the file, and have them fix it there. If your POI generated file is identified as having an issue, and you're on the latest codebase, report a new POI bug and include the details of the validation failure.

Another thing to try, especially if the file is valid but POI isn't behaving as expected, are the POI Dev Tools for the component you're using. For example, HSSF has *org.apache.poi.hssf.dev.BiffViewer* which will allow you to view the file as POI does. This will often allow you to check that things are being read as you expect, and narrow in on problem records and structures.

15. An OOXML ("xml") file is giving me problems, but I can't share it. How can I investigate the problem on my own?

There's not currently a simple validator tool as there is for the OLE2 based (binary) file formats, but checking the basics of a file is generally much easier.

Files such as .xlsx, .docx and .pptx are actually a zip file of XML files, with a special structure. Your first step in diagnosing the issues with the input or output file will likely be to unzip the file, and look at the XML of it. Newer versions of Office will normally tell you which area of the file is problematic, so narrow in on there. Looking at the XML, does it look correct?

When reporting bugs, ideally include the whole file, but if you're unable to then include the snippet of XML for the problem area, and reference the OOXML standard for what it should contain.