Apache POI - the Java API for Microsoft Documents

by Andrew C. Oliver, Glen Stampoultzis, Avik Sengupta, Rainer Klute, David Fisher

1. 28 August 2011 - The Apache POI project is celebrating its 10th anniversary

The Apache POI project is celebrating its 10th anniversary. On the 28th of August 2001, the first 0.1 alpha version of POI was released. Read more...

2. 26 August 2011 - POI 3.8 beta 4 available

The Apache POI team is pleased to announce the release of 3.8 beta 4. This includes a large number of bug fixes and enhancements.

A full list of changes is available in the change log. People interested should also follow the dev mailing list to track further progress.

See the downloads page for more details.

3. 29 October 2010 - POI 3.7 available

The Apache POI team is pleased to announce the release of 3.7. This includes a large number of bug fixes, and some enhancements (especially text extraction). See the full release notes for more details.

A full list of changes is available in the change log. People interested should also follow the dev mailing list to track further progress.

See the downloads page for more details.

4. Mission Statement

The Apache POI Project's mission is to create and maintain Java APIs for manipulating various file formats based upon the Office Open XML standards (OOXML) and Microsoft's OLE 2 Compound Document format (OLE2). In short, you can read and write MS Excel files using Java. In addition, you can read and write MS Word and MS PowerPoint files using

Java. Apache POI is your Java Excel solution (for Excel 97-2008). We have a complete API for porting other OOXML and OLE2 formats and welcome others to participate.

OLE2 files include most Microsoft Office files such as XLS, DOC, and PPT as well as MFC serialization API based file formats. The project provides APIs for the OLE2 Filesystem (POIFS) and OLE2 Document Properties (HPSF).

Office OpenXML Format is the new standards based XML file format found in Microsoft Office 2007 and 2008. This includes XLSX, DOCX and PPTX. The project provides a low level API to support the Open Packaging Conventions using openxml4j.

For each MS Office application there exists a component module that attempts to provide a common high level Java api to both OLE2 and OOXML document formats. This is most developed for Excel workbooks (SS=HSSF+XSSF). Work is progressing for Word documents (HWPF+XWPF) and PowerPoint presentations (HSLF+XSLF).

The project has recently added support for Outlook (HSMF). Microsoft opened the specifications to this format in October 2007. We would welcome contributions.

There are also projects for Visio (HDGF), TNEF (HMEF), and Publisher (HPBF).

As a general policy we collaborate as much as possible with other projects to provide this functionality. Examples include: Cocoon for which there are serializers for HSSF; Open Office.org with whom we collaborate in documenting the XLS format; and Tika / Lucene, for which we provide format interpretors. When practical, we donate components directly to those projects for POI-enabling them.

4.1. Why should I use Apache POI?

A major use of the Apache POI api is for Text Extraction applications such as web spiders, index builders, and content management systems.

So why should you use POIFS, HSSF or XSSF?

You'd use POIFS if you had a document written in OLE 2 Compound Document Format, probably written using MFC, that you needed to read in Java. Alternatively, you'd use POIFS to write OLE 2 Compound Document Format if you needed to inter-operate with software running on the Windows platform. We are not just bragging when we say that POIFS is the most complete and correct implementation of this file format to date!

You'd use HSSF if you needed to read or write an Excel file using Java (XLS). You'd use XSSF if you need to read or write an OOXML Excel file using Java (XLSX). The combined SS interface allows you to easily read and write all kinds of Excel files (XLS and XLSX) using Java.

4.2. Components

The Apache POI Project provides several component modules some of which may not be of interest to you. Use the information on our Components page to determine which jar files to include in your classpath.

5. Contributing

So you'd like to contribute to the project? Great! We need enthusiastic, hard-working, talented folks to help us on the project. So if you're motivated, ready, and have the time time download the source from the Subversion Repository, build the code, join the mailing lists and we'll be happy to help you get started on the project!

Please read our Contribution Guidelines. When your contribution is ready submit a patch to our Bug Database.