Apache POI - Java API To Access Microsoft Format Files

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

1. POI 3.1-BETA2 Released (2008-05-28)

The POI team is pleased to announce the release of 3.1 BETA2 which is one of the final steps before 3.1 FINAL. The status of this release is a beta, meaning that we encourage users to try it out. If you find any bugs, please report them to the POI bug database or to the POI Developer List.

A full list of changes is available in the changelog, and download the source and binaries from your local mirror.

The release is also available from the central Maven repository under Group ID "org.apache.poi" and Version "3.1-beta2".

2. POI 3.0.2 Released

The POI team is pleased to announce POI 3.0.2, the latest release of Apache POI. There have been many important bug fixes since the 3.0.1 release and a lot of new features. A full list of changes is available in the changelog, and download the source and binaries from your local mirror.

The release is also available from the central Maven repository under Group ID "org.apache.poi" and Version "3.0.2-FINAL".

We would also like to confirm that versions 3.0.1 and 3.0.2 of Apache POI do *not* contain any viruses. Users of broken virus checkers which do detect a 94 byte file, sci_cec.db, as containing one are advised to contact their vendor for a fix.

3. ApacheCon Europe Coming Soon

ApacheCon Europe 2008 banner

ApacheCon Europe 2008 will once again be held at the Mövenpick Hotel in Amsterdam, April 7-11. This year, there will be a number of POI sessions, including a tutorial covering the new Office Open XML support.

For further information, see the ApacheCon Europe Web site at www.eu.apachecon.com

4. Office Open XML Support

We are currently working to support the new Office Open XML file formats, such as XLSX and PPTX, which were introduced in Office 2007.

Support for these is currently only available in an svn branch, but we hope to have a full release including it by the summer. People interested should follow the dev list to track progress.

5. Purpose

The POI project consists of APIs for manipulating various file formats based upon Microsoft's OLE 2 Compound Document format using pure Java. In short, you can read and write MS Excel files using Java. Soon, you'll be able to read and write Word, PowerPoint and Visio files using Java. POI is your Java Excel solution as well as your Java Word solution. However, we have a complete API for porting other OLE 2 Compound Document formats, and welcome others to participate.

OLE 2 Compound Document Format based files include most Microsoft Office files such as XLS and DOC as well as MFC serialization API based file formats.

At this time, none of our releases support the new Office Open XML file formats, such as .xlsx or .docx. Work to support these is in progress, and people interested should follow the dev list. We expect this support to make it into a full release by the summer.

As a general policy, we try to 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 Lucene for which we provide format interpretors. When practical, we donate components directly to those projects for POI-enabling them.

5.1. Why/when would I use POI?

We'll tackle this on a component level. POI refers to the whole project.

So why should you use POIFS or HSSF?

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, write or modify an Excel file using Java (XLS).

6. Components To Date

6.1. Overview

The following are components of the entire POI project and a brief summary of their purpose.

6.2. POIFS for OLE 2 Documents

POIFS is the oldest and most stable part of the project. It is our port of the OLE 2 Compound Document Format to pure Java. It supports both read and write functionality. All of our components ultimately rely on it by definition. Please see the POIFS project page for more information.

6.3. HSSF for Excel Documents

HSSF is our port of the Microsoft Excel 97(-2003) file format (BIFF8) to pure Java. It supports read and write capability. (Support for Excel 2007 .xlsx files is in progress). Please see the HSSF project page for more information.

6.4. HWPF for Word Documents

HWPF is our port of the Microsoft Word 97 file format to pure Java. It supports read, and limited write capabilities. Please see the HWPF project page for more information. This component is in the early stages of development. It can already read and write simple files.

Presently we are looking for a contributor to foster the HWPF development. Jump in!

6.5. HSLF for PowerPoint Documents

HSLF is our port of the Microsoft PowerPoint 97(-2003) file format to pure Java. It supports read and write capabilities. Please see the HSLF project page for more information.

6.6. HDGF for Visio Documents

HDGF is our port of the Microsoft Viso 97(-2003) file format to pure Java. It currently only supports reading at a very low level, and simple text extraction. Please see the HDGF project page for more information.

6.7. HPSF for Document Properties

HPSF is our port of the OLE 2 property set format to pure Java. Property sets are mostly use to store a document's properties (title, author, date of last modification etc.), but they can be used for application-specific purposes as well.

HPSF supports reading and writing of properties. However, you will need to be using version 3.0 of POI to utilise the write support.

Please see the HPSF project page for more information.

7. Contributing

So you'd like to contribute to the project? Great! We need enthusiastic, hard-working, talented folks to help us on the project in several areas. The first is bug reports and feature requests! The second is documentation - we'll be at your every beck and call if you've got a critique or you'd like to contribute or otherwise improve the documentation. We could especially use some help documenting the HSSF file format! Last, but not least, we could use some binary crunching Java coders to chew through the complexity that characterizes Microsoft's file formats and help us port new ones to a superior Java platform!

So if you're motivated, ready, and have the time, join the mail lists and we'll be happy to help you get started on the project!