Apache POI - Java API To Access Microsoft Format Files

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

1. POI 3.0.2 BETA2 Release

The latest release of Apache POI is 3.0.2 BETA2 which was promoted to "Beta" on 12 January 2008. It contains a mixture of new features and bug fixes, compared to 3.0.1. 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".

2. POI 3.0.1 Release

The latest release of Apache POI (formerly Apache Jakarta POI), version 3.0.1, has now been released. It contains a mixture of new features and bug fixes, compared to 3.0. A full list of changes is available in the changelog, and download the source and binaries from your local mirror.

We would also like to confirm that verions 3.0 and 3.0.1 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. 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 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.

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.

3.1. 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 or write an Excel file using Java (XLS). You can also read and modify spreadsheets using this API, although right now writing is more mature.

4. Components To Date

4.1. Overview

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

4.2. 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.

4.3. 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. Please see the HSSF project page for more information.

4.4. 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!

4.5. 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 of some, but not yet all of the core records. Please see the HSLF project page for more information.

4.6. 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.

4.7. 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.

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 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!

Copyright (c) @year@ The Apache Software Foundation. All rights reserved. $Revision: 615249 $ $Date: 2008-01-25 18:48:14 +0300 (Fri, 25 Jan 2008) $