Last Published: 01/17/2021 22:54:35

Apache POI - Component Overview

Apache POI Project Components

The Apache POI project is the master project for developing pure Java ports of file formats based on Microsoft's OLE 2 Compound Document Format. OLE 2 Compound Document Format is used by Microsoft Office Documents, as well as by programs using MFC property sets to serialize their document objects.

Apache POI is also the master project for developing pure Java ports of file formats based on Office Open XML (ooxml). OOXML is part of an ECMA / ISO standardisation effort. This documentation is quite large, but you can normally find the bit you need without too much effort! ECMA-376 standard is here, and is also under the Microsoft OSP.

POIFS for OLE 2 Documents

POIFS is the oldest and most stable part of POI. 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 for the binary (non-XML) Microsoft Office formats ultimately rely on it by definition. Please see the POIFS project page for more information.

HSSF and XSSF for Excel Documents

HSSF is our port of the Microsoft Excel 97 (-2003) file format (BIFF8) to pure Java. XSSF is our port of the Microsoft Excel XML (2007+) file format (OOXML) to pure Java. SS is a package that provides common support for both formats with a common API. They both support read and write capability. Please see the HSSF+XSSF project page for more information.

HWPF and XWPF for Word Documents

HWPF is our port of the Microsoft Word 97 (-2003) file format to pure Java. It supports read, and limited write capabilities. It also provides simple text extraction support for the older Word 6 and Word 95 formats. Please see the HWPF project page for more information. This component remains in early stages of development. It can already read and write simple files.

We are also working on the XWPF for the WordprocessingML (2007+) format from the OOXML specification. This provides read and write support for simpler files, along with text extraction capabilities.

HSLF and XSLF 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.

We are also working on the XSLF for the PresentationML (2007+) format from the OOXML specification.

HPSF for OLE 2 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 both reading and writing of properties.

Please see the HPSF project page for more information.

HDGF and XDGF for Visio Documents

HDGF is our port of the Microsoft Visio 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 / Diagram project page for more information.

XDGF is our port of the Microsoft Visio XML (.vsdx) file format to pure Java. It has slightly more support than HDGF. Please see the XDGF / Diagram project page for more information.

HPBF for Publisher Documents

HPBF is our port of the Microsoft Publisher 98(-2007) file format to pure Java. It currently only supports reading at a low level for around half of the file parts, and simple text extraction. Please see the HPBF project page for more information.

HMEF for TNEF (winmail.dat) Outlook Attachments

HMEF is our port of the Microsoft TNEF (Transport Neutral Encoding Format) file format to pure Java. TNEF is sometimes used by Outlook for encoding the message, and will typically come through as winmail.dat. HMEF currently only supports reading at a low level, but we hope to add text and attachment extraction. Please see the HMEF project page for more information.

HSMF for Outlook Messages

HSMF is our port of the Microsoft Outlook message file format to pure Java. It currently only some of the textual content of MSG files, and some attachments. Further support and documentation is coming in slowly. For now, users are advised to consult the unit tests for example use. Please see the HSMF project page for more information.

Microsoft has recently added the Outlook file format to its OSP. More information is now available making implementing this API an easier task.

Component Map

The Apache POI distribution consists of support for many document file formats. This support is provided in several Jar files. Not all of the Jars are needed for every format. The following tables show the relationships between POI components, Maven repository tags, and the project's Jar files.

Component	Application type	Maven artifactId	Notes
POIFS	OLE2 Filesystem	*poi*	Required to work with OLE2 / POIFS based files
HPSF	OLE2 Property Sets	*poi*	
HSSF	Excel XLS	*poi*	For HSSF only, if common SS is needed see below
HSLF	PowerPoint PPT	*poi-scratchpad*	
HWPF	Word DOC	*poi-scratchpad*	
HDGF	Visio VSD	*poi-scratchpad*	
HPBF	Publisher PUB	*poi-scratchpad*	
HSMF	Outlook MSG	*poi-scratchpad*	
DDF	Escher common drawings	*poi*	
HWMF	WMF drawings	*poi-scratchpad*	
OpenXML4J	OOXML	*poi-ooxml* plus either *poi-ooxml-lite* or *poi-ooxml-full*	See notes below for differences between these options
XSSF	Excel XLSX	*poi-ooxml*	
XSLF	PowerPoint PPTX	*poi-ooxml*	
XWPF	Word DOCX	*poi-ooxml*	
XDGF	Visio VSDX	*poi-ooxml*	
Common SL	PowerPoint PPT and PPTX	*poi-scratchpad* and *poi-ooxml*	SL code is in the core POI jar, but implementations are in poi-scratchpad and poi-ooxml.

Common SS	Excel XLS and XLSX	*poi-ooxml*	WorkbookFactory and friends all require poi-ooxml, not just core poi

This table maps artifacts into the jar file name. "version-yyyymmdd" is the POI version stamp. You can see what the latest stamp is on the downloads page.

Maven artifactId	Prerequisites	JAR
poi	jcl-over-slf4j (commons-logging replacement), commons-codec, commons-collections, commons-math	poi-version-yyyymmdd.jar
poi-scratchpad	poi	poi-scratchpad-version-yyyymmdd.jar
poi-ooxml	poi, poi-ooxml-lite, commons-compress, SparseBitSet For SVG support: batik-all, xml-apis-ext, xmlgraphics-commons For PDF support: pdfbox, fontbox, rototor graphics2d	poi-ooxml-version-yyyymmdd.jar
poi-ooxml-lite	xmlbeans	poi-ooxml-lite-version-yyyymmdd.jar
poi-examples	poi, poi-scratchpad, poi-ooxml	poi-examples-version-yyyymmdd.jar
poi-ooxml-full (known as ooxml-schemas)	xmlbeans For signing: bcpkix-jdk15on, bcprov-jdk15on, xmlsec, slf4j-api	poi-ooxml-full-version-yyyymmdd.jar

> **Note**
>
> Apache commons-math3 and commons-compress were added as a dependency in POI 4.0.0.
> Zaxxer SparseBitSet was added as a dependency in POI 4.1.2

poi-ooxml requires poi-ooxml-lite. This is a substantially smaller version of the poi-ooxml-full jar (ooxml-schemas-1.4.jar for POI 4.0.0, ooxml-schemas-1.3.jar for POI 3.14 or to POI 3.17, ooxml-schemas-1.1.jar for POI 3.7 up to POI 3.13, ooxml-schemas-1.0.jar for POI 3.5 and 3.6). The larger ooxml-schemas jar is normally only required for development. Similarly, the ooxml-security jar, which contains all of the classes relating to encryption and signing, is normally only required for development. A subset of its contents are in poi-ooxml-schemas. This JAR is ooxml-security-1.1.jar for POI 3.14 onwards and ooxml-security-1.0.jar prior to that.

The OOXML jars require a stax implementation, but now that Apache POI requires Java 8, that dependency is provided by the JRE and no additional stax jars are required. The OOXML jars used to require

DOM4J, but the code has now been changed to use JAXP and no additional dom4j jars are required. By the way, look at this FAQ if you have problems when using a non-Oracle JDK.

The ooxml schemas jars are compiled with Apache XMLBeans 2.3, and so can be used at runtime with any version of XMLBeans from 3.0.0 or newer. Wherever possible though, we recommend that you use XMLBeans 3.1.0 with Apache POI, and that is the version now shipped in the binary release packages.

Examples

Small sample programs using the POI API are available in the src/examples (viewvc) directory of the source distribution.

All of the examples are included in POI distributions as a poi-examples artifact.

Running POI on other JVM languages

POI can be run on most languages that run on the JVM. For code examples, see Running POI on other JVM languages

Contributed Software

Besides the "official" components outlined above there is some further software distributed with POI. This is called "contributed" software. It is not explicitly recommended or even maintained by the POI team, but it might still be useful to you.

See POI Ruby Bindings and other code in the poi-contrib module

by Andrew C. Oliver, Rainer Klute, David Fisher

Last Published: 01/17/2021 22:54:35