Welcome to POI

by Andrew C. Oliver, Glen Stampoultzis, Avik Sengupta

1. News

All POI news can now be found at the poi news weblog.

2. 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 Word Excel 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'll soon have file format interpretors. When practical, we donate components directly to those projects for POI-enabling them.

2.1. 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 port 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.

2.2. What does POI stand for?

POI stands for Poor Obfuscation Implementation. Why would we name our project such a derogatory name? Well, Microsoft's OLE 2 Compound Document Format is a poorly conceived thing. It is essentially an archive structured much like the old DOS FAT filesystem. Redmond chose, instead of using tar, gzip, zip or arc, to invent their own archive format that does not provide any standard encryption or compression, is not very appendable and is prone to fragmentation.

Poi is also a Hawaiian delicacy that Merriam Webster's dictionary defines as: "A Hawaiian food of taro root cooked, pounded, and kneaded to a paste and often allowed to ferment." This seemed strangely descriptive of the file format.

So if you like acronyms, then POI is an acronym. If you hate them, then we just used the name of the food for our project. If you wish to signify your love or hate for acronyms, use POI or Poi to refer to the project respectively.

3. Components To Date

3.1. Overview

A common misconception is that POI writes Excel files. POI is the name of the project. POI contains several components, one of which, HSSF, writes Excel files. The following are components of the entire POI project and a brief summary of their purpose.

3.2. POIFS (POI Filesystem)

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.

3.3. HSSF (Horrible Spreadsheet Format)

HSSF is our port of the Microsoft Excel 97(-2002) file format (BIFF8) to pure Java. It supports read and write capability. Please see the HSSF project page for more information.

3.4. HWPF

HWPF is our port of the Microsoft Word 97 file format to pure Java. It supports read and write capability. 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. Jump in!. (Please note that the HWPF codebase is NOT included in the 2.0 releases. Please use CVS to access this code.)

3.5. HPSF (Horrible Property Set Format)

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. Currently HPSF supports read functionality only. Please see the HPSF project page for more information.

4. What happened to the HSSF Serializer?

The HSSF Serializer, which was part of our 1.0 release and last builds on Sourceforge, has been donated to the Cocoon project, and is available starting from version 2.0.2.

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 convolution 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!