Apache POI - HWPF - Java API to Handle Microsoft Word Files

Word File Format

by S. Ryan Ackley

1. The Word 97 File Format in semi-plain English

The purpose of this document is to give a brief high level overview of the HWPF document format. This document does not go into in-depth technical detail and is only meant as a supplement to the Microsoft Word 97-2007 Binary File Format freely available from Microsoft.

The OLE file format is not discussed in this document. It is assumed that the reader has a working knowledge of the POIFS API.

1.1. Word file structure

A Word file is made up of the document text and data structures containing formatting information about the text. Of course, this is a very simplified illustration. There are fields and macros and other things that have not been considered. At this stage, HWPF is mainly concerned with formatted text.

1.2. Reading Word files

The entry point for HWPF's reading of a Word file is the File Information Block (FIB). This structure is the entry point for the locations and size of a document's text and data structures. The FIB is located at the beginning of the main stream.

1.2.1. Text

The document's text is also located in the main stream. Its starting location is given as FIB.fcMin and its length is given in bytes by FIB.ccpText. These two values are not very useful in getting the text because of unicode. There may be unicode text intermingled with ASCII text. That brings us to the piece table.

Copyright © 2002-2009 The Apache Software Foundation All rights reserved.

The piece table is used to divide the text into non-unicode and unicode pieces. The size and offset are given in FIB.fcClx and FIB.lcbClx respectively. The piece table may contain Property Modifiers (prm). These are for complex(fast-saved) files and are skipped. Each text piece contains offsets in the main stream that contain text for that piece. If the piece uses unicode, the file offset is masked with a certain bit. Then you have to unmask the bit and divide by 2 to get the real file offset.

1.2.2. Text Formatting

1.2.2.1. Stylesheet

All text formatting is based on styles contained in the StyleSheet. The StyleSheet is a data structure containing among other things, style descriptions. Each style description can contain a paragraph style and a character style or simply a character style. Each style description is stored in a compressed version on file. Basically these are deltas from another style.

Eventually, you have to chain back to the nil style which is an imaginary style with certain implied values.

1.2.2.2. Paragraph and Character styles

Paragraph and Character formatting properties for a document's text are stored on file as deltas from some base style in the Stylesheet. The deltas are used to create a complete uncompressed style in memory.

Uncompressed paragraph styles are represented by the Pargraph Properties(PAP) data structure. Uncompressed character styles are represented by the Character Properties(CHP) data structure. The styles for the document text are stored in compressed format in the corresponding Formatted Disk Pages (FKP). A compressed PAP is referred to as a PAPX and a compressed CHP is a CHPX. The FKP locations are stored in the bin table. There are seperate bin tables for CHPXs and PAPXs. The bin tables' locations and sizes are stored in the FIB.

A FKP is a 512 byte OLE page. It contains the offsets of the beginning and end of each paragraph/character run in the main stream and the compressed properties for that interval. The compessed PAPX is based on its base style in the StyleSheet. The compressed CHPX is based on the enclosing paragraph's base style in the Stylesheet.

1.2.2.3. Uncompressing styles and other data structures

All compressed properties(CHPX, PAPX, SEPX) contain a grpprl. A grpprl is an array of

Copyright © 2002-2009 The Apache Software Foundation All rights reserved.

sprms. A sprm defines a delta from some base property. There is a table of possible sprms in the Word 97 spec. Each sprm is a two byte operand followed by a parameter. The parameter size depends on the sprm. Each sprm describes an operation that should be performed on the base style. After every sprm in the grpprl is performed on the base style you will have the style for the paragraph, character run, section, etc.

Copyright © 2002-2009 The Apache Software Foundation All rights reserved.