Record Generator HOWTO

by Glen Stampoultzis, Andrew C. Oliver

1. How to Use the Record Generator

1.1. History

The record generator was born from frustration with translating the Excel records to Java classes. Doing this manually is a time consuming process. It's also very easy to make mistakes.

A utility was needed to take the defintition of what a record looked like and do all the boring and repetitive work.

1.2. Capabilities

The record generator takes XML as input and produces the following output:

- A Java file capabile of decoding and encoding the record.
- A test class that provides a fill-in-the-blanks implementation of a test case for ensuring the record operates as designed.

1.3. Usage

The record generator is invoked as an Ant target (generate-records). It goes through looking for all files in `src/records/defintitions` ending with _record.xml. It then creates two files; the Java record definition and the Java test case template.

The records themselves have the following general layout:

```
<record id="0x1032" name="Frame" package="org.apache.poi.hssf.record"
        excel-record-id="FRAME">
    <description>The frame record indicates whether there is a border
                 around the displayed text of a chart.</description>
    <author>Glen Stampoultzis (glens at apache.org)</author>
    <fields>
        <field type="int" size="2" name="border type">
            <const name="regular" value="0" description="regular rectangle or no border
            <const name="shadow" value="1" description="rectangle with shadow"/>
```

```
        </field>
        <field type="int" size="2" name="options">
            <bit number="0" name="auto size"
               description="excel calculates the size automatically if true"/>
            <bit number="1" name="auto position"
               description="excel calculates the position automatically"/>
        </field>
    </fields>
</record>
```

The following table details the allowable types and sizes for the fields.

Type	Size	Java Type
int	1	byte
int	2	short
int	4	int
int	8	long
int	varword	array of shorts
bits	1	A byte comprising of a bits (defined by the bit element)
bits	2	An short comprising of a bits
bits	4	A int comprising of a bits
float	8	double
hbstring	java expression	String

The Java records are regenerated each time the record generator is run, however the test stubs are only created if the test stub does not already exist. What this means is that you may change test stubs but not the generated records.

1.4. Custom Field Types

Occationally the builtin types are not enough. More control over the encoding and decoding of the streams is required. This can be achieved using a custom type.

A custom type lets you escape to java to define the way in which the field encodes and decodes. To code a custom type you declare your field like this:

```
    <field type="custom:org.apache.poi.hssf.record.LinkedDataFormulaField"
        size="var" name="formula of link" description="formula"/>
```

Where the class name specified after `custom:` is a class implementing the interface `CustomField`.

You can then implement the encoding yourself.

1.5. How it Works

The record generation works by taking an XML file and styling it using XLST. Given that XSLT is a little limited in some ways it was necessary to add a little Java code to the mix.

See record.xsl, record_test.xsl, FieldIterator.java, RecordUtil.java, RecordGenerator.java

There is a corresponding "type" generator for HWPF. See the HWPF documentation for details.

1.6. Limitations

The record generator does not handle all possible record types and goes not intend to perform this function. When dealing with a non-standard record sometimes the cost-benifit of coding the record by hand will be greater than attempting modify the generator. The main point of the record generator is to save time, so keep that in mind.

Currently the the XSL file that generates the record calls out to Java objects. The Java code for the record generation is currently quite messy with minimal comments.