MSOfficeParsers

Tika's MSOffice Parsers (Apache POI)

Beta SAX Parsers for .docx and .pptx

As of Tika 1.15, there are experimental/beta SAX parsers for .docx files. On very large files (e.g. "War and Peace"), this parser appears to be 4x faster and require far less memory than our traditional DOM based parsers. For smaller files, the gain is not nearly as great. For the 386MB pptx submitted on TIKA-2201, it would have taken ~60GB to load the file in memory.

These parsers are still in their early stages and don't have all of the features of the DOM parsers. However, the .docx parser does offer parameterization to include or exclude deleted text.

To select it programmatically, set `setUseSAXDocxExtractor` or `setUseSAXPptxExtractor` to `true` on an OfficeParserConfig and put that in the ParseContext: `context.set(OfficeParserConfig.class, officeParserConfig);`.

To set it via the config file, try:

```
<properties>
    <parsers>
        <parser class="org.apache.tika.parser.DefaultParser"/>
        <parser class="org.apache.tika.parser.microsoft.ooxml.OOXMLParser">
            <params>
                <param name="useSAXDocxExtractor" type="bool">true</param>
                <param name="useSAXPptxExtractor" type="bool">true</param>
            </params>
        </parser>
    </parsers>
</properties>
```

See TIKA-1321 for the parser and TIKA-2180 and TIKA-2201 for some symptoms that the current DOM parser might be slowing you down.

How to build Tika with POI's trunk

You'll need to have the following build tools installed: Ant, Forrest and Maven. You'll also need the source code for both projects via svn, git or download of src.

1. Build POI – "`gradlew clean build jar`". Optionally, to run the integration tests: "`ant test-integration`". Note, you can also grab a nightly build from Jenkins. As of this writing, you still have to generate the poms. So, the nightly build saves you from building POI, but it doesn't yet save you from having to download the source code (install Ant, etc.) to build the poms. 2. Generate the poms – "`ant maven-poms`". 3. From `build/dist/maven`, install this new build of POI into your local Maven repo – e.g.:

```
mvn install:install-file -Dfile=poi-4.0.0-SNAPSHOT.jar -DpomFile=poi-4.0.0-SNAPSHOT.pom

mvn install:install-file -Dfile=poi-ooxml-schemas-4.0.0-SNAPSHOT.jar -DpomFile=poi-ooxml-schemas-4.0.0-SNAPSHOT.pom

mvn install:install-file -Dfile=poi-ooxml-4.0.0-SNAPSHOT.jar -DpomFile=poi-ooxml-4.0.0-SNAPSHOT.pom

mvn install:install-file -Dfile=poi-scratchpad-4.0.0-SNAPSHOT.jar -DpomFile=poi-scratchpad-4.0.0-SNAPSHOT.pom
```

4. Update the version of POI in Tika's `tika-parsers/pom.xml`. 5. Make any necessary modifications to Tika (if there are mods to POI's API) 6. Build Tika – "`mvn clean install`"

And there you have a shiny, sparkling, new Tika with the dev version of POI!