CompositeParserDiscussion

Composite Parser Discussion

A given mime type may be supported by several parsers. Work on TIKA-1445 (adding metadata back into OCR'd text) raised the prominence of this issue. Currently, the CompositeParser picks the first parser that supports a given mime type. In discussion on TIKA-1445 other potential use cases were identified.

The purpose of this page is to track a unified vision of the strategies that we'll implement in Tika.

The JIRA issue for this is TIKA-1509.

This page is just a start. Please contribute

Strategies

Classic

Sort the parsers by non-tika vs tika and then alphabetically by class name. Pick the first parser that will handle a given mime type.

Supplementary/Additive

Concatenate the results (metadata and content) for several parsers

For Metadata, this merging should be configurable if multiple parsers output the same Metadata Key, between:

- First/earliest parser's value(s) win
- Last/latest parser's value(s) win
- Capture all and return multiple values

TODO For Content, decide how we could support appending or resetting of the SAX stream

We need a better name for this!

Back-off

Try one parser and if the output doesn't meet some criterion, apply another. One use case for this might be: if a file is identified as XML, try the XMLParser and if that throws an exception, try the HTMLParser.

Pick the Best Output

One use case for this: the charset detector identifies two equally likely charsets. Apply both and use the wished-for junk detector (TIKA-1443) to determine which output is more likely to be not junk.

Fastest

If there are two parsers, use the faster one even if it might mean lower quality (eg avoid OCR)

Mime type hierarchies

Consider a case like:

- application/vnd.ms-excel
  - application/x-tika-msoffice

Or

- application/dita+xml;format=concept
  - application/dita+xml;format=topic
    - application/dita+xml

If there were two parsers available for application/vnd.ms-excel, and another for application/x-tika-msoffice, should it be possible to specify in a strategy that a parser for the parent type also be used? Should it be possible to set a strategy like "use the dita concept, then the general dita, then the dita topic", hopping around up and down the hierarchy?

Or do we keep the current behaviour where once a point in the hierachy with a parser is found, it is parsed at that point?

Allowing the User to select a strategy

The right strategy for one user may not be the right for another. The right strategy for one file may not be the right one for another. We therefore need to allow users to pick their strategy, on an overall basis, and on a per-file basis

From TikaConfig

Currently, a great many Tika users just call `TikaConfig.getDefaultConfig()` and go with that.

It might be nice if they could also do things like `TikaConfig.getMaxiumMetadataConfig()` or `TikaConfig.getTryEachInTurnConfig()` to pick a different strategy

(Naming TBC, align with above)

With a Tika Configuration file

Users may wish to have full control over what parsers are used, what strategies are used for which mime types etc

For example, they might want default behaviour for most types, but to send XML through a fallback parser, and combine Image + GDAL + OCR for jpeg. The configuration file needs to support this

```
<parsers>
  <!-- Most things can use the default -->
  <parser class="org.apache.tika.parser.DefaultParser">
    <!-- Don't use DefaultParser for these mimetypes, alternate config below -->
    <mime-exclude>image/jpeg</mime-exclude>
    <mime-exclude>application/xml</mime-exclude>
    <mime-exclude>application/pdf</mime-exclude>
    <!-- Exclude (blacklist) these parsers, have them ignored by DefaultParser -->
    <parser-exclude class="org.apache.tika.parser.jdbc.SQLite3Parser"/>
    <parser-exclude class="org.apache.tika.parser.executable.ExecutableParser"/>
  </parser>

  <!-- No PDF, thank you! -->
  <parser class="org.apache.tika.parser.EmptyParser">
    <mime>application/pdf</mime>
  </parser>

  <!-- JPEG needs special handling - try+combine everything -->
  <parser class="org.apache.tika.parser.(suppliment)">
     <params>
        <!-- If several parsers output the same metadata key, first parser to do so wins -->
        <param name="metadataPolicy" value="FIRST_WINS" />
        <!-- If several parsers output the same metadata key, last parser to do so wins -->
        <!--
        <param name="metadataPolicy" value="LAST_WINS" />
         -->
        <!-- If several parsers output the same metadata key, store all their values -->
        <!--
        <param name="metadataPolicy" value="KEEP_ALL" />
         -->
     </params>
     <parser class="org.apache.tika.parser.ocr.TesseractOCRParser" />
     <parser class="org.apache.tika.parser.image.ImageParser" />
     <parser class="org.apache.tika.parser.jpeg.JpegParser" />
     <parser class="org.apache.tika.parser.gdal.GDALParser" />
     <!-- TODO DO we need to give mimetypes here too? Or can we get implicitly? -->
  </parser>

  <!-- XML needs special handling - use fallbacks to get something -->
  <parser class="org.apache.tika.parser.(fallback)">
     <params>
        <!-- If we move onto a second/third/etc parser, discard metadata from previous parsers -->
        <param name="metadataPolicy" value="DISCARD" />
        <!-- If we move onto a second parser, and both output the same metadata key, earlier parser wins -->
        <!--
        <param name="metadataPolicy" value="FIRST_WINS" />
         -->
        <!-- If we move onto a second parser, and both output the same metadata key, later parser wins -->
        <!--
        <param name="metadataPolicy" value="LAST_WINS" />
         -->
        <!-- If move onto a second parser, and both output the same metadata key, store all their values -->
        <!--
        <param name="metadataPolicy" value="KEEP_ALL" />
         -->
     </params>
     <parser class="my.custom.xml.parser" />
     <parser class="org.apache.tika.parser.xml.XMLParser" />
     <parser class="org.apache.tika.parser.html.HTMLParser" />
     <parser class="org.apache.tika.parser.txt.TXTParser" />
     <mime>application/xml</mime>
  </parser>
</parsers>
```

In Code

Whatever we do, this must be available from code too, much as how today people can create custom `CompositeParser` instances, or wrap things up with custom `ParserDecorator` instances

We also need examples for all of these, not only in unit tests, but also in the examples package.