PDFParser (Apache PDFBox)

PDFParser

Configuration options

There are three ways of configuring the PDFParser.

1. Programmatically via setter methods on the PDFParser.
2. Programmatically via the PDFParserConfig object submitted through the `ParseContext`.
3. Via the tika-config.xml file (many thanks to Thamme Gowda and Chris Mattmann's work on TIKA-1508).

The first two are fairly self-explanatory through the javadocs.

In the following, we document all of the parameters for the PDFParser in Tika 2.x. You only need to specify the parameters you want to change.

PDFParser Configuration

```
<properties>
  <parsers>
    <parser class="org.apache.tika.parser.DefaultParser">
      <!-- this is not formally necessary, but prevents loading of unnecessary parser -->
      <parser-exclude class="org.apache.tika.parser.pdf.PDFParser"/>
    </parser>
    <parser class="org.apache.tika.parser.pdf.PDFParser">
      <params>
        <!-- these are the defaults; you only need to specify the ones you want
            to modify -->
        <!-- if you want to extract content, whether or not the PDF allows extraction
             at all, do not set this parameter.  If you want to extract content and
             the PDF allows for extraction for accessibility, set this to true.
             If you do not want to extract content when the PDF does not allow extraction
             but does allow extraction for accessibility, set this to false -->
        <param name="allowExtractionForAccessibility" type="bool">true</param>
        <param name="averageCharTolerance" type="float">0.3</param>
        <param name="detectAngles" type="bool">false</param>
        <param name="extractAcroFormContent" type="bool">true</param>
        <param name="extractActions" type="bool">false</param>
        <!-- as of 2.8.0 -->
        <param name="extractIncrementalUpdateInfo" type="bool">false</param>
        <param name="catchIntermediateIOExceptions" type="bool">true</param>
        <param name="dropThreshold" type="float">2.5</param>
        <param name="enableAutoSpace" type="bool">true</param>
        <param name="extractAnnotationText" type="bool">false</param>
        <param name="extractBookmarksText" type="bool">true</param>
        <param name="extractFontNames" type="bool">false</param>
        <param name="extractInlineImages" type="bool">false</param>
        <param name="extractMarkedContent" type="bool">false</param>
        <param name="extractUniqueInlineImagesOnly" type="bool">true</param>
        <param name="ifXFAExtractOnlyXFA" type="bool">false</param>
        <param name="maxMainMemoryBytes" type="long">-1</param>
        <!-- as of 2.8.0 -->
        <param name="maxIncrementalUpdates" type="int">10000</param>
        <param name="ocrDPI" type="int">300</param>
        <param name="ocrImageFormatName" type="string">png</param>
        <param name="ocrImageQuality" type="float">1.0</param>
        <param name="ocrRenderingStrategy" type="string">ALL</param>
        <param name="ocrStrategy" type="string">auto</param>
        <param name="ocrStrategyAuto" type="string">better</param>
        <param name="ocrImageType" type="string">gray</param>
         <!-- as of 2.8.0 -->
        <param name="parseIncrementalUpdates" type="bool">false</param>
        <param name="setKCMS" type="bool">false</param>
        <param name="sortByPosition" type="bool">false</param>
        <param name="spacingTolerance" type="float">0.5</param>
        <param name="suppressDuplicateOverlappingText" type="bool">false</param>
        <!-- as of versions after 2.8.0 -->
        <param name="throwOnEncryptedPayload" type="bool">false</param>
      </params>
    </parser>
  </parsers>
</properties>
```

Optional Dependencies

If you need to process TIFF or JPEG2000 images within PDFs (either for inline image extraction or OCR), please consider adding the optional dependencies specified by PDFBox. These dependencies are not compatible with ASL 2.0; please make sure that any third party licenses are suitable for your project.

Finally, M. Caruana Galizia alerted us to the need to use maven-shade's `ServicesResourceTransformer` because the third-party dependencies' services file will be overwritten unless you transform the services. See an example: here.

OCR

Note: the configuration of some of these features via the config file requires a nightly build of Tika after 11/8/2016 or Tika version >= 1.15.

Start with the instructions on TikaOCR. In short, you need to have Tesseract installed.

There are two ways of running OCR on PDFs:

1. Extracting the inline images and letting Tesseract run on each inline image.
2. Rendering each PDF page as a single image and running Tesseract on that single image.

We have not carried out evaluations to determine which strategy is better. We suspect that the tried and true *It Depends(TM)* is operative here. We added OCR'ing of the single image option because some PDFs can contain hundreds of images per page where each image is a tiny part of the overall page, and OCR would be useless. However, we recognize, that if the page is logically broken into sections, running OCR on the individual inline images might yield better results.

Note: These two options are independent. If you set `extractInlineImages` to true and select an `OcrStrategy` that includes OCR on the rendered page, Tika will run OCR on the extracted inline images *and* the rendered page.

Option 1: Configuring OCR on Inline Images

This will extract inline images as if they were attachments, and then, if Tesseract is correctly configured, it should run against the images. Note: by default, extracting inline images is turned off because some rare PDFs contain thousands of inline images per page, and it has a big hit on performance, both memory usage and time.

```
...
        <parser class="org.apache.tika.parser.pdf.PDFParser">
            <params>
                <param name="extractInlineImages" type="bool">true</param>
            </params>
        </parser
...
```

Option 2: Configuring OCR on Rendered Pages

This will render each PDF page and then run OCR on that image. This method of OCR is triggered by the `ocrStrategy` parameter, but users can manipulate other parameters, including the `image type` (see `org.apache.pdfbox.rendering.ImageType` for options) and the dots per inch `dpi`. The defaults are: `gray` and `300` respectively. For `ocrStrategy`, we currently have: `no_ocr` (rely on regular text extraction only), `ocr_only` (don't bother extracting text, just run OCR on each page), `ocr_and_text` (both extract text and run OCR) and (as of Tika 1.21) `auto` (try to extract text, but run OCR if fewer than 10 characters were extracted of if there are more than 10 characters with unmapped Unicode values).

```
...
        <parser class="org.apache.tika.parser.pdf.PDFParser">
            <params>
                <param name="ocrStrategy" type="string">ocr_only</param>
                <param name="ocrImageType" type="string">rgb</param>
                <param name="ocrDPI" type="int">100</param>
            </params>
        </parser>
...
```

Setting Parse Time/Per File configurations via tika-server

See: Configuring Parsers At Parse Time in tika-server.

Optional Dependencies

Note, you should include the following dependency to process JBIG2 images:

```
    <dependency>
        <groupId>org.apache.pdfbox</groupId>
        <artifactId>jbig2-imageio</artifactId>
        <version>3.0.2</version>
    </dependency>
```

Note, **if their licenses are compatible with your application**, you may want to include the following jai libraries in your classpath to handle jp2, jpeg2000 and tiff files. **The licenses are not Apache 2.0 compatible!**

```
    <dependency>
        <groupId>com.github.jai-imageio</groupId>
        <artifactId>jai-imageio-core</artifactId>
        <version>1.4.0</version>
    </dependency>
    <dependency>
        <groupId>com.github.jai-imageio</groupId>
        <artifactId>jai-imageio-jpeg2000</artifactId>
        <version>1.3.0</version>
        <scope>test</scope>
    </dependency>
```

If you are using Java 8, make sure to see pdf-rendering for JVM settings that may improve the speed of processing.

Common Text Extraction Challenges with PDFs

This is mostly a stub. The focus of this section is on extracting electronic text from the PDF with no OCR.

One could write several volumes on how text extraction from PDFs could go wrong. It would only be poetic justice for said author to print out those volumes, pour coffee on the paper, scan them in as PDFs on different scanners, some with OCR, some without, at different angles of rotation with user-defined fonts randomly deleted.

High level preliminaries:

0. Your matrix algebra (or, your tool's matrix algebra) has to be moderately advanced to do text extraction well.

1. The PDF format is display-based not text-based
    a. One major goal is to display the same content on different devices b. A PDF may be image-only and contain no actual electronic text c. When there is electronic text, there may be no space characters stored in the text, rather spaces may appear in the rendering of the image via specific coordinates for the characters.
    2. The PDF format is page-based

Tables Aren't Extracted as Tables

Right. In PDF/UA (Universal Accessibility) tables can be stored with structural markup. As of Tika 1.24, you can set "extractMarkedContent" = "true" via the PDFParserConfig, and Tika will extracted marked content, including tables, if the PDF was generated with marked content.

In many PDFs, tables are often not stored as tables. A human is easily able to see tables, but all that is stored in the PDF is text chunks and coordinates on a page (if there's any text at all). One needs to apply some advanced computation to extract table structure from a PDF. Tika does not currently do this. Please see TabulaPDF as one open source project that extracts tables from PDFs and maintains their structure.

Note that as of 2023, we still see papers at research conferences on extracting structural elements (including tables) from PDFs – THIS IS NOT A SOLVED PROBLEM. And, humorously, this kind of task is sometimes called "PDF remediation".

No Text

Mildly Garbled Text

Completely Garbled Text

No spaces/Extra spaces

See 1c. above. Depending on how the PDF was generated, it is possible that it doesn't store actual space characters. Rather software has to use coordinates on the page plus matrix algebra plus font information about the width of characters to "impute" where spaces would be. The math is the simple part; sometimes there can be missing or wrong font information that can lead to no spaces or extra spaces.

Word/Line breaks in the middle of my text ?!

Character Encoding/Unicode Mappings

See also diagnosing PDF text problems.

See also

Upgrading to PDFBox 2.x