org.apache.tika.parser.pdf

Class PDFParser

java.lang.Object
  org.apache.tika.parser.AbstractParser
    org.apache.tika.parser.pdf.PDFParser

All Implemented Interfaces:

`Serializable`, `Initializable`, `Parser`

Direct Known Subclasses:

`PDFPreflightParser`

```
public class PDFParser
extends AbstractParser
implements Initializable
```

PDF parser.

This parser can process also encrypted PDF documents if the required password is given as a part of the input metadata associated with a document. If no password is given, then this parser will try decrypting the document using the empty password that's often used with PDFs. If the PDF contains any embedded documents (for example as part of a PDF package) then this parser will use the `EmbeddedDocumentExtractor` to handle them.

As of Tika 1.6, it is possible to extract inline images with the `EmbeddedDocumentExtractor` as if they were regular attachments. By default, this feature is turned off because of the potentially enormous number and size of inline images. To turn this feature on, see `PDFParserConfig.setExtractInlineImages(boolean)`.

Please note that tables are not stored as entities within PDFs. It takes significant computation to identify and then correctly extract tables from PDFs. As of this writing, the `PDFParser` extracts text within tables, but it does not compute table cell boundaries or table row boundaries. Please see tabula for one project that tries to maintain the structure of tables represented in PDFs.

See Also:

`Serialized Form`

Field Summary

Fields

Modifier and Type	Field and Description
`static String`	`PASSWORD` **Deprecated.** Supply a `PasswordProvider` on the `ParseContext` instead

Constructor Summary

Constructors

Constructor and Description
`PDFParser()`

Method Summary

All Methods **Instance Methods** **Concrete Methods** **Deprecated Methods**

Modifier and Type	Method and Description
`void`	`checkInitialization(InitializableProblemHandler handler)`
`boolean`	`getEnableAutoSpace()`
`boolean`	`getExtractAnnotationText()` **Deprecated.** use `getPDFParserConfig()`
`protected org.apache.pdfbox.pdmodel.PDDocument`	`getPDDocument(InputStream inputStream, String password, org.apache.pdfbox.io.MemoryUsageSetting memoryUsageSetting, Metadata metadata, ParseContext parseContext)`
`protected org.apache.pdfbox.pdmodel.PDDocument`	`getPDDocument(Path path, String password, org.apache.pdfbox.io.MemoryUsageSetting memoryUsageSetting, Metadata metadata, ParseContext parseContext)`
`PDFParserConfig`	`getPDFParserConfig()`

boolean	**getSortByPosition**() **Deprecated.** use **getPDFParserConfig()**
Set<MediaType>	**getSupportedTypes**(ParseContext context) Returns the set of media types supported by this parser when used with the given parse context.
boolean	**getSuppressDuplicateOverlappingText**() **Deprecated.** use **getPDFParserConfig()**
void	**initialize**(Map<String,Param> params) This is a no-op.
void	**parse**(InputStream stream, ContentHandler handler, Metadata metadata, ParseContext context) Parses a document stream into a sequence of XHTML SAX events.
void	**setDropThreshold**(float dropThreshold)
void	**setEnableAutoSpace**(boolean v) If true (the default), the parser should estimate where spaces should be inserted between words.
void	**setExtractAnnotationText**(boolean v) If true (the default), text in annotations will be extracted.
void	**setInitializableProblemHandler**(InitializableProblemHandler initializableProblemHandler)
void	**setMaxMainMemoryBytes**(long maxMainMemoryBytes)
void	**setOcrImageType**(String imageType)
void	**setOcrStrategy**(String ocrStrategyString)
void	**setPDFParserConfig**(PDFParserConfig config)
void	**setSortByPosition**(boolean v) If true, sort text tokens by their x/y position before extracting text.
void	**setSuppressDuplicateOverlappingText**(boolean v) If true, the parser should try to remove duplicated text over the same region.

Methods inherited from class org.apache.tika.parser.AbstractParser

parse

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

PASSWORD

```
public static final String PASSWORD
```

Deprecated. *Supply a `PasswordProvider` on the `ParseContext` instead*
Metadata key for giving the document password to the parser.

Since:
Apache Tika 0.5

See Also:
Constant Field Values

Constructor Detail

PDFParser

```
public PDFParser()
```

Method Detail

getSupportedTypes

```
public Set<MediaType> getSupportedTypes(ParseContext context)
```

Description copied from interface: Parser
Returns the set of media types supported by this parser when used with the given parse context.

Specified by:
`getSupportedTypes` in interface `Parser`

Parameters:
`context` – parse context

Returns:
immutable set of media types

parse

```
public void parse(InputStream stream,
                  ContentHandler handler,
                  Metadata metadata,
                  ParseContext context)
           throws IOException,
                  SAXException,
                  TikaException
```

Description copied from interface: Parser
Parses a document stream into a sequence of XHTML SAX events. Fills in related document metadata in the given metadata object.

The given document stream is consumed but not closed by this method. The responsibility to close the stream remains on the caller.

Information about the parsing context can be passed in the context parameter. See the parser implementations for the kinds of context information they expect.

Specified by:
`parse` in interface `Parser`

Parameters:
`stream` – the document stream (input)

`handler` – handler for the XHTML SAX events (output)

`metadata` – document metadata (input and output)

`context` – parse context

Throws:
`IOException` – if the document stream could not be read

`SAXException` – if the SAX events could not be processed

`TikaException` – if the document could not be parsed

getPDDocument

```
protected org.apache.pdfbox.pdmodel.PDDocument getPDDocument(InputStream inputStream,
                                                             String password,
                                                             org.apache.pdfbox.io.MemoryUsageSetting memoryUsageSetting,
                                                             Metadata metadata,
                                                             ParseContext parseContext)
                                                      throws IOException
```

Throws:
`IOException`

getPDDocument

```
protected org.apache.pdfbox.pdmodel.PDDocument getPDDocument(Path path,
                                                             String password,
                                                             org.apache.pdfbox.io.MemoryUsageSetting memoryUsageSetting,
                                                             Metadata metadata,
                                                             ParseContext parseContext)
                                                      throws IOException
```

Throws:
`IOException`

getPDFParserConfig

```
public PDFParserConfig getPDFParserConfig()
```

setPDFParserConfig

```
public void setPDFParserConfig(PDFParserConfig config)
```

getEnableAutoSpace

```
public boolean getEnableAutoSpace()
```

See Also:
`setEnableAutoSpace(boolean)`

setEnableAutoSpace

```
@Field
public void setEnableAutoSpace(boolean v)
```

If true (the default), the parser should estimate where spaces should be inserted between words. For many PDFs this is necessary as they do not include explicit whitespace characters.

getExtractAnnotationText

```
public boolean getExtractAnnotationText()
```

Deprecated. *use* *getPDFParserConfig()*
If true, text in annotations will be extracted.

setExtractAnnotationText

```
public void setExtractAnnotationText(boolean v)
```

If true (the default), text in annotations will be extracted.

getSuppressDuplicateOverlappingText

```
public boolean getSuppressDuplicateOverlappingText()
```

Deprecated. *use* *getPDFParserConfig()*

See Also:
`setSuppressDuplicateOverlappingText(boolean)`

setSuppressDuplicateOverlappingText

```
@Field
public void setSuppressDuplicateOverlappingText(boolean v)
```

If true, the parser should try to remove duplicated text over the same region. This is needed for some PDFs that achieve bolding by re-writing the same text in the same area. Note that this can slow down extraction substantially (PDFBOX-956) and sometimes remove characters that were not in fact duplicated (PDFBOX-1155). By default this is disabled.

getSortByPosition

```
public boolean getSortByPosition()
```

Deprecated. *use* *getPDFParserConfig()*

See Also:
`setSortByPosition(boolean)`

setSortByPosition

```
@Field
public void setSortByPosition(boolean v)
```

If true, sort text tokens by their x/y position before extracting text. This may be necessary for some PDFs (if the text tokens are not rendered "in order"), while for othe6 -- Add FileProfilerr PDFs it can produce the wrong result (for example if there are 2 columns, the text will be interleaved). Default is false.

setOcrStrategy

```
@Field
public void setOcrStrategy(String ocrStrategyString)
```

setOcrImageType

```
@Field
public void setOcrImageType(String imageType)
```

setInitializableProblemHandler

```
public void setInitializableProblemHandler(InitializableProblemHandler initializableProblemHandler)
```

setDropThreshold

```
@Field
public void setDropThreshold(float dropThreshold)
```

setMaxMainMemoryBytes

```
@Field
public void setMaxMainMemoryBytes(long maxMainMemoryBytes)
```

initialize

```
public void initialize(Map<String,Param> params)
               throws TikaConfigException
```

This is a no-op. There is no need to initialize multiple fields. The regular field loading should happen without this.

Specified by:

`initialize` in interface `Initializable`

Parameters:

`params` – params to use for initialization

Throws:

`TikaConfigException`

checkInitialization

```
public void checkInitialization(InitializableProblemHandler handler)
                        throws TikaConfigException
```

Specified by:

`checkInitialization` in interface `Initializable`

Parameters:

`handler` – if there is a problem and no custom initializableProblemHandler has been configured via Initializable parameters, this is called to respond.

Throws:

`TikaConfigException`