



UWP



Extraction



WINDOWS > GUIDES > EXTRACTION

Simplify PDF text extraction on Windows

Extracting text from a PDF can be a complex task, so we offer several abstractions to make this simpler. In a PDF, text usually consists of glyphs that are absolutely positioned. Nutrient heuristically splits these glyphs up into words and blocks of text. Our `PdfView` leverages this information to allow users to select and annotate text.

Text parser

`TextParser` offers a simple API to get the `text`, `Glyph`s, `Word`s, and `TextBlock`s from a given PDF page. Every page of a PDF has a text parser that returns information about the text on a page:

```
1 var textParser = await doc.GetTextParserAsync(0);
2 var text = await textParser.GetTextForRectsAsync(rects);
3 var glyphs = await textParser.GetGlyphsAsync();
4 var words = TextParser.WordsFromGlyphs(glyphs);
5 var textsBlocks = await textParser.GetTextAsync();
```



`TextParser` also ensures that the text it extracts from a PDF follows any logical structure defined in the PDF (see section 14.7 of the [PDF specification](#)), thereby enabling support for correctly handling different text-writing directions.

Glyphs

`Glyph` is the building block for all text extraction in Nutrient. It represents a single glyph on a page. Its `Rect` property specifies, in PDF coordinates, where it's located on the page, and its `Content` property contains the text of the glyph.



ASK AI

property returns the text it contains. The `Index` property specifies the index of the glyph on the page in reading order. Consider a page with the following text:

```
1 The quick brown fox jumps over the lazy dog.
2 -----^
```

The `Glyph` that represents the o in over will have an `Index` of 26. This index is unique to this glyph, and it can be used to directly access it from the `Glyph`s returned from a `TextParser`:

```
1 var textParser = await doc.GetTextParserAsync(0);
2 var glyphs = await textParser.GetGlyphsAsync();
3
4 // Guaranteed to be `true`.
5 var indexEqualTo26 = glyphs[26].Index == 26
```

This makes getting a particular glyph (and glyphs near it) much faster, as you already know the index. Ordering glyphs correctly is important if, for example, you wish to combine multiple glyphs and display something to the user.

Text blocks

A `TextBlock` returned from the `TextParser` represents a contiguous group of glyphs, usually in a single line. For PDFs with multiple columns of text, a text block is a single line in a given column.

Words

A `Word`, as the name suggests, represents a single word in a PDF. `TextParser` can calculate these words from glyphs using the `WordsFromGlyphs` method:

```
1 var textParser = await doc.GetTextParserAsync(0);
2 var glyphs = await textParser.GetGlyphsAsync();
3 var words = TextParser.WordsFromGlyphs(glyphs);
```

A `Word` has a `Frame` that describes the area the word covers on the page. It also has a `Range` that describes the range within the provided glyphs that make up the word, and a `Contents`, which is the string of text that `Word` represents.

Was this helpful?

 YES

 NO

Questions? [Contact us](#)

