ITEXT

pdf2Data

an iText 7 add-on

What is it?

pdf2Data allows you to extract data from PDF documents. The process is based on a framework that recognizes data inside PDF documents, based on areas that you have selected for extraction in your template. pdf2Data works best on documents that are based on the same template, such as an invoice coming from the same supplier. This makes it easier to automate document workflow, reducing human error and your processing time.

The data recognition is based on several rules, which need to be defined in advance per each template field.

TYPICAL RULES ARE:

- the same (horizontal / vertical) position on the page
- the same font size and style
- certain text pattern (numeric, currency sign, etc)
- certain keywords on the same as the required field
- certain cell(s) in the table

This means that you can create a fully automated solution for data recognition in a PDF document with basic set-up on the original sample template. The template relies on dynamic field selectors such as font, style, position and text patterns to find the required fields in your data. To ensure you have the best possible results, we leverage iText text extraction, which offers a high fidelity recognition process.

Setting up pdf2Data includes integration with pure Java API with CLI (commany line interface) and REST interfaces. You also can choose between the convenient web application software package included with pdf2Data that enables you to define selectors in a more intuitive way, by installing the software on your workstation or using a PDF commenting tool such as Adobe Reader, to define the selectors.

How does it work?

RECOGNITION IS BASED ON THE FOLLOWING STEPS:

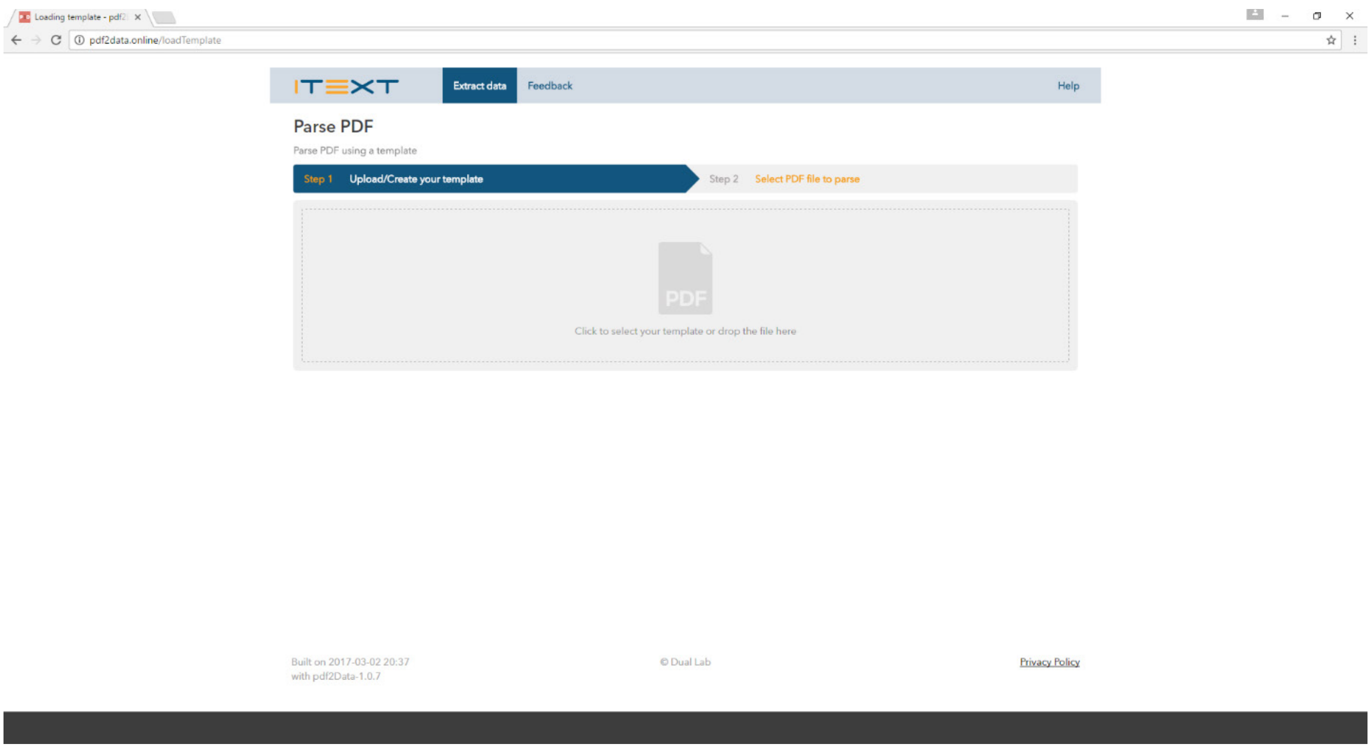
1. Select parts of the template that correspond to your data fields using the pdf2Data web application or any PDF Viewer with commenting functionality.
2. Define relevant rules for the correct data extraction in the comment attached to each selection.
3. Upload the template to the web site running the template engine, and see if it recognized your fields and data inside them.
4. Upload any other PDF document that is based on the same template and check if the software could recognize your data.

Steps 1 to 3 need to be done only once per template. Step 4 can be repeated for as many documents as needed. But they all need to be based on the same template.

Please note that although this example makes use of our own pdf2Data server, it is of course possible to use your own. Or to simply forego the web interface altogether and define your template using the Adobe Reader commenting facilities.

Simple example

Figure 1: landing page of pdf2Data web application



Loading template - pdf2

pdf2data.online/loadTemplate

Extract data | Feedback | Help

Parse PDF

Parse PDF using a template

Step 1 Upload/Create your template

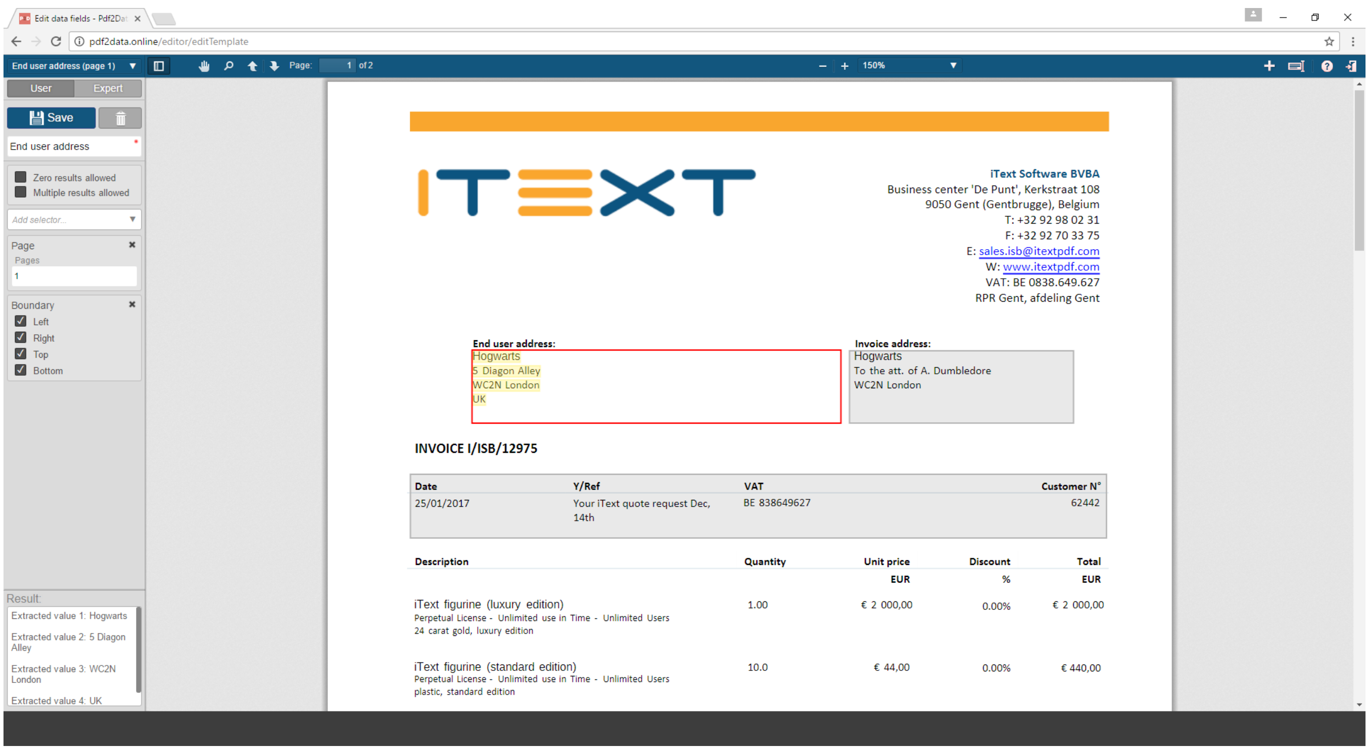
Step 2 Select PDF file to parse

Click to select your template or drop the file here

Built on 2017-03-02 20:37
with pdf2Data-1.0.7

© Dual Lab

Privacy Policy



Edit data fields - Pdf2Dat

pdf2data.online/editor/editTemplate

End user address (page 1) Page: 1 of 2 150%

User | Expert

Save

End user address

Zero results allowed
Multiple results allowed

Add selector...

Page
Pages
1

Boundary
✓ Left
✓ Right
✓ Top
✓ Bottom

Result:
Extracted value 1: Hogwarts
Extracted value 2: 5 Diagon Alley
Extracted value 3: WC2N London
Extracted value 4: UK

iText Software BVBA
Business center 'De Punt', Kerkstraat 108
9050 Gent (Gentbrugge), Belgium
T: +32 92 98 02 31
F: +32 92 70 33 75
E: sales.isb@itextpdf.com
W: www.itextpdf.com
VAT: BE 0838.649.627
RPR Gent, afdeling Gent

End user address:
Hogwarts
5 Diagon Alley
WC2N London
UK

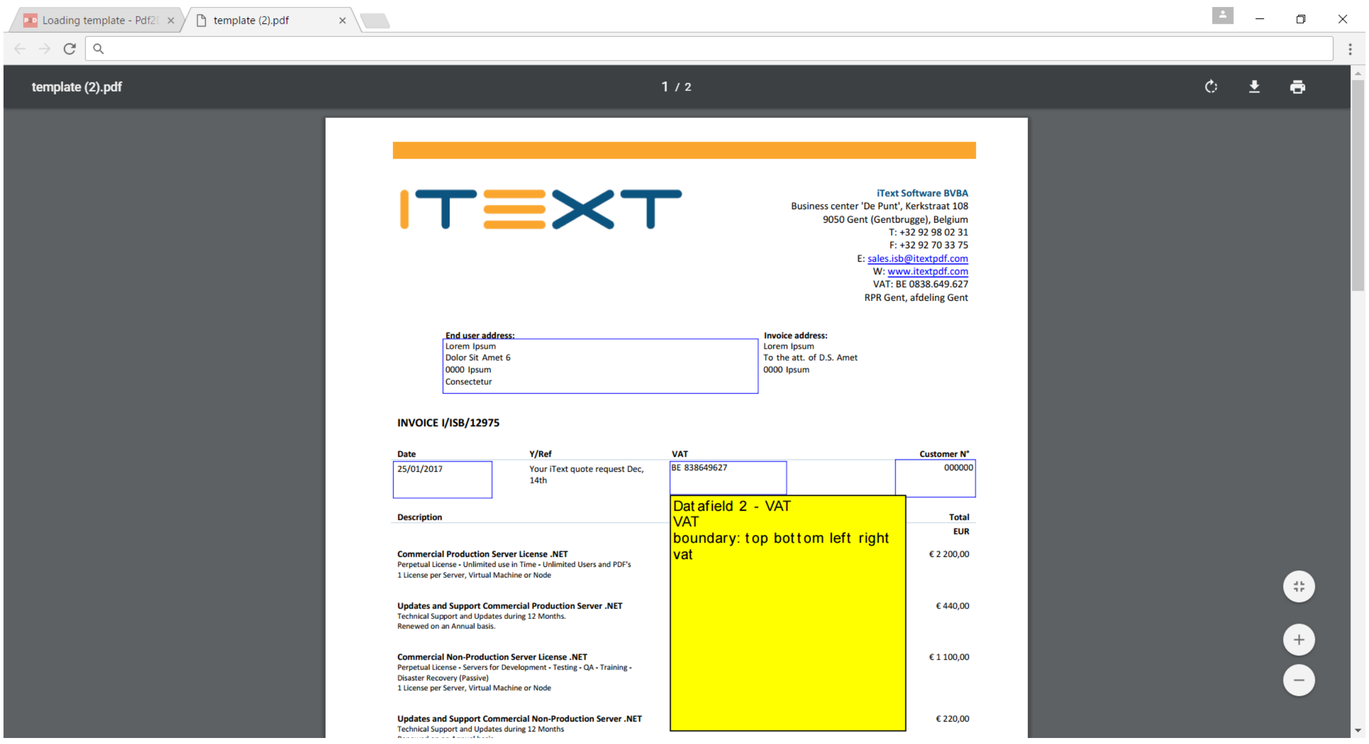
Invoice address:
Hogwarts
To the att. of A. Dumbledore
WC2N London

INVOICE I/ISB/12975

Date	Y/Ref	VAT	Customer N°
25/01/2017	Your iText quote request Dec, 14th	BE 838649627	62442

Description	Quantity	Unit price	Discount	Total
		EUR	%	EUR
iText figurine (luxury edition) Perpetual License - Unlimited use in Time - Unlimited Users 24 carat gold, luxury edition	1.00	€ 2 000,00	0.00%	€ 2 000,00
iText figurine (standard edition) Perpetual License - Unlimited use in Time - Unlimited Users plastic, standard edition	10.0	€ 44,00	0.00%	€ 440,00

Figure 2: an example selector to extract the customer address



iText Software BVBA
Business center 'De Punt', Kerkstraat 108
9050 Gent (Gentbrugge), Belgium
T: +32 92 98 02 31
F: +32 92 70 33 75
E: sales.isb@itextpdf.com
W: www.itextpdf.com
VAT: BE 0838.649.627
RPR Gent, afdeling Gent

End user address:
Lorem Ipsum
Dolor Sit Amet 6
0000 Ipsum
Consectetur

Invoice address:
Lorem Ipsum
To the att. of D.S. Amet
0000 Ipsum

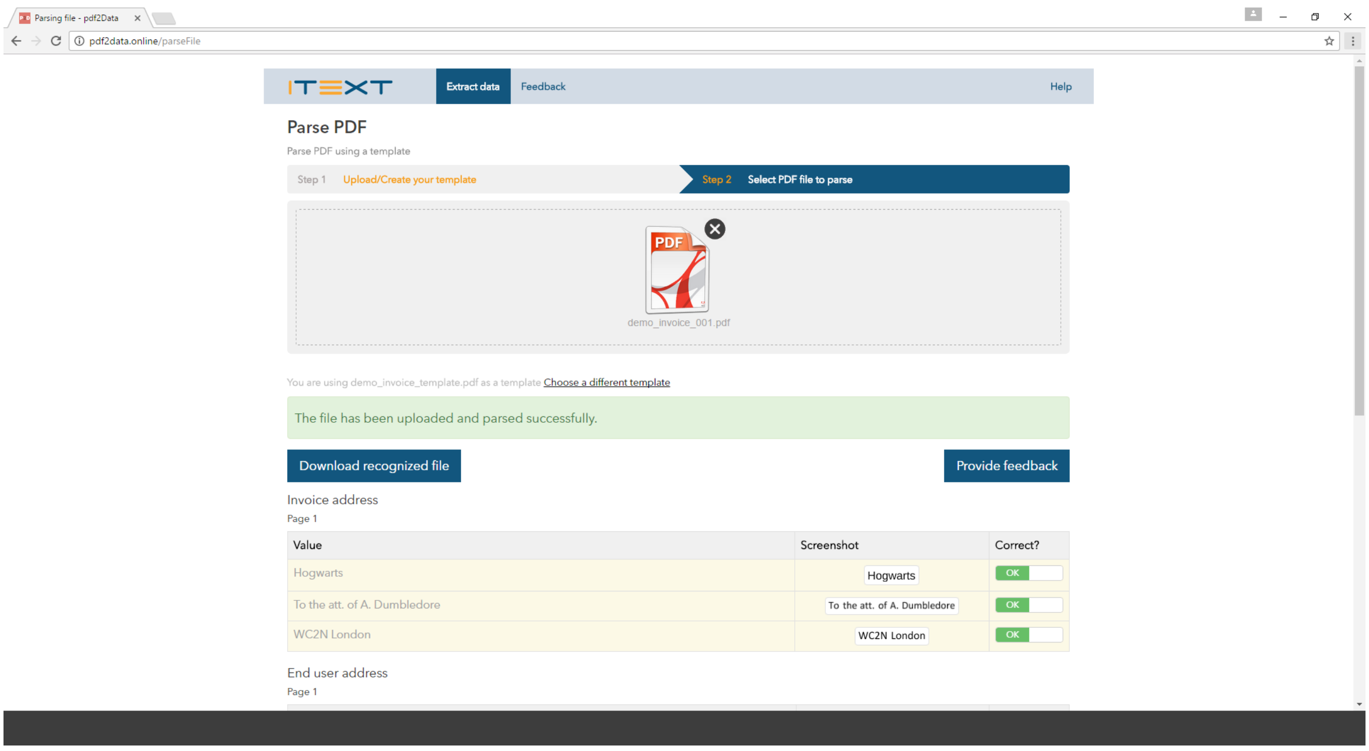
INVOICE I/ISB/12975

Date	Y/Ref	VAT	Customer N°
25/01/2017	Your iText quote request Dec, 14th	BE 838649627	000000

Datafield 2 - VAT
VAT
boundary: top bottom left right
vat

Description	Total EUR
Commercial Production Server License .NET Perpetual License - Unlimited use in Time - Unlimited Users and PDF's 1 License per Server, Virtual Machine or Node	€ 2 200,00
Updates and Support Commercial Production Server .NET Technical Support and Updates during 12 Months. Renewed on an Annual basis.	€ 440,00
Commercial Non-Production Server License .NET Perpetual License - Servers for Development - Testing - QA - Training - Disaster Recovery (Passive) 1 License per Server, Virtual Machine or Node	€ 1 100,00
Updates and Support Commercial Non-Production Server .NET Technical Support and Updates during 12 Months	€ 220,00

Figure 3: the template, opened in Adobe Acrobat



Parse PDF

Parse PDF using a template

You are using demo_invoice_template.pdf as a template Choose a different template

The file has been uploaded and parsed successfully.

Invoice address

Page 1

Value	Screenshot	Correct?
Hogwarts	Hogwarts	OK
To the att. of A. Dumbledore	To the att. of A. Dumbledore	OK
WC2N London	WC2N London	OK

End user address

Page 1

Figure 4: sample extraction

A more in-depth look

ABOUT THE SELECTORS

Important: Keep in mind this is not intended as a replacement for the full documentation. Rather as an overview of the possibilities for data-extraction.

Font-based

name	functionality
fontFamily	This selector extracts the font name of the annotated text region and then uses this font name to filter glyphs with that font name. It is assumed that all the annotated text of the region has the same font family.
fontSize	This selector extracts the font size of the annotated text region and then uses this font size to filter glyphs with that font size. It is assumed that all the annotated text of the region has the same font size.
fontStyle	This selector extracts the font style of the annotated text region, e.g. bold, italic, and then uses this font style to filter glyphs with this font style. It is assumed that all the annotated text of the region has the same font style.
font	The font selector identifies the font used for the text in the selector region and extracts all symbols with the same font from the PDF document. If the text in the selector region uses several fonts, only the first one is used. The font is considered as a combination of the font family, font size and font style. If only some of these properties should be honoured when extracting text, please use selectors fontFamily, fontSize, fontStyle

Position-based

name	functionality
boundary	This selector restricts the area on the page where the data is located. The selector boundary without any additional properties means that only the data inside the selector region will be extracted. However, it is often useful to honor only some of the region borders. In this case these borders can be specified as additional properties of the boundary selector. For example, boundary:left selector means that only the symbols positioned to the right of the left boundary are extracted. Similarly, - boundary:top means that only the symbols positioned below the top boundary are extracted, - boundary: left right means that symbols between the left and right boundaries are selected, while the vertical position is ignored. Note. The selector boundary without arguments is equivalent to boundary: left right bottom top.
align	For “align:left” this selector uses the left bound of annotation for selecting only those lines that begin near that boundary. For “align:right” this selector uses the right bound of annotation for selecting only those lines that end near that boundary.
page	This selector restricts the area to a given page nr. Use 1, 2, etc or -1 (for the last page), -2, etc

Picker

name	functionality
picker	This selector drills down on an extracted group that was recognized on the previous step. The parameter is an integer number greater or less than 0 (it depends from direction: first-to-last for positive index and last-to-first for negative index). If it exceeds the number of input groups, the result will be empty. The typical usecase is that you've already selected a column of a table (using the table selector), and that you'd now like to drill down on the last row in the column. This is typically the case when the table lists purchased items, and the last row contains the total price.

Formats

name	functionality
price	recognizes a decimal number, possibly with decimal separator, group separator and preceding or following currency sign. Supported: dollar (sign, USD, CAD, HKD, AUD), euro (sign, EUR), yen (sign, JPY), pound (sign, GBP, EBP)
date	recognizes date string in a number of common formats. Supported: All permutations of MM (or M), DD (or D), YYYY (or YY) with separators like '.,/-: ', literal representations of month are caught by '[A-Za-z]+' or '' regexp
iban	recognizes IBAN string
VAT	Supports the VAT numbers of all 27 EU countries. (as of time of writing Feb 2017)

Text

name	functionality
paragraph	This selector combines data into paragraphs with specified (NORMAL, LARGE, HUGE) line spacing and return them. By default, if line spacing isn't specified selector will use normal line spacing.
line	This selector combines data into lines with specified (NORMAL, LARGE, HUGE) character spacing and returns them. By default, if char spacing isn't specified selector will use normal char spacing.

Pattern-based

name	functionality
pattern	This selector is usual regular expression selector. It allows convenience selection of a piece of data to be extracted. You can specify a prefix, type of data to be extracted, and a suffix. E.g. (prefix) "total price" (suffix) "EUR"
regexp	The regExp selector implements the standard regular expression search with a few additional options.

Table

name	functionality
tableCluster	The tableCluster selector attempts to process symbols into a logical table. This selector has many parameters (including the number of columns, rows, the columns to be selected, the row to be selected, the headers, etc). Consult the full documentation for a thorough understanding.

TYPICAL USECASE : PROCESSING AN INVOICE

Boilerplate code

```
// build a new Pdf2DataExtractor based on a template
Pdf2DataExtractor extractor = new Pdf2DataExtractor(template);

// sampleFile: the file you wish to process
// targetPdf: the path where you wish to store the annotated pdf (for visual inspection)
// targetXML: the path where you wish to store the extracted data (in xml format)
extractor.parsePdf(sampleFile, targetPDF, targetXML);
```

Practical guidelines

1. Use the boundary selector with all four borders enabled to check that the text can be extracted.
   pdf2Data relies on internal PDF structure rather than visual presentation to extract the text. Not all text visible in the document can be reliably extracted from PDF documents. The boundary selector triggers one of the simplest low level text extraction algorithms, and it is always a good idea to try first or if other selectors don't work for your document.
2. Do not use the boundary selector with all four sides enabled in the final version of the template, unless the text has a fixed position on the page and can not grow depending on the variable data.
3. Try the table selector in the automatic mode if your data is located inside a table cell. Table selector in automatic mode is the best choice for tables with clear borders around the cells (or at least between columns). It also tries a number of smart heuristic algorithms even if your table does not have any borders at all.
4. Use the pattern selector if your data is surrounded by boilerplate text (i.e. text that is unlikely to change and always precedes or follows after the important data). e.g. "total price: xxxx EUR"
5. Use the paragraph selector to combine several lines into a single paragraph or to select a paragraph with a predefined first (title) line. Sometimes the extracted text comes out as a sequence of distinct lines. You can always try combining them into a single paragraph using Paragraph selector at the end. Paragraph selectors can also be used very efficiently to allocate the blocks of text with a predefined first (title) line.

DEPLOYING YOUR OWN PDF2DATA WEB APPLICATION

1. Download a Java SE Development Kit 8 and install it.
2. Download a Apache Tomcat 8.x software and install it.
3. Download the pdf2Data web application war file
4. Deploy the application on the installed Tomcat server In most cases it is sufficient to copy a war file into subdirectory webapps in Tomcat directory
5. Create the file “web.properties” as follows

```
dir.temp=your_folder_for_resources
mail.to=pdf2data@duallab.com
mail.smtp.host=smtp.duallab.com
mail.smtp.port=25
mail.ssl.smtp.port=567
mail.ssl.enable=false
mail.smtp.starttls.enable=false
mail.from=your email address
user.name=your email address
user.password=your email password
```

USING THE COMMAND LINE INTERFACE (CLI)

Using the CLI is done in a two-step process. First the template pdf is pre-processed. The annotations are extracted, along with their corresponding selectors, and stored in an xml file. The parser then uses the template information in the xml file alongside the document to be processed to produce both an xml document (containing the data) and a pdf document containing information about the recognition (for quality control purposes.)

Preprocessor

```
java -jar preprocess.jar --template=Tempate.pdf --xml=Template.xml
Arguments:
-t=Template.pdf, --template=Template.pdf
This argument defines a template PDF file that contains annotations with rules for each
region.

-x=Template.xml, --xml=Template.xml
This argument defines an XML file that will contain rules for each region we want to recognize
after preprocessing of the corresponding PDF file.
```

Parser

```
java -jar parse.jar --template=Template.xml --pdf=Test.pdf --outPdf=pdfFile.pdf
--outXml=xmlFile.xml
Arguments:
-t=Template.xml, --template=Template.xml
This argument defines an XML file with rules for recognizing regions of data.

-p=Test.pdf, --pdf=Test.pdf
This argument defines a PDF file with data we want to recognize.

-r=pdfFile.pdf, --outPdf=pdfFile.pdf
This argument defines a PDF file that will contain visual representation of recognized
elements in the form of annotations.

-x=xmlFile.xml, --outXml=xmlFile.xml
This argument defines an XML file that will contain a recognized data from the corresponding
PDF file.
```

CONCLUSION

In this whitepaper, we've briefly presented our new add-on pdf2Data. Pdf2Data allows you to seamlessly integrate data extraction in your existing workflow. A template document has to be defined, either via Adobe Reader comments or through the available web-interface software package. Afterwards, this template can be applied effortlessly to ensure high-volume, high-throughput data transformations. The extracted data is put into an .XML file, which can then be read by most libraries.

Learn more at www.itextpdf.com