Apache PDFBox - Working with pdfs for Dummies

Andreas Lehmkühler

ApacheCon

Leading the Wave of Open Source

The Speaker

- Andreas Lehmkühler
- Degree in electrical engineering
- Senior Developer
- Committer since December 2008
- PMC Chair since November 2009
- ASF member since April 2010

Agenda

- History
- **P**ortable **D**ocument **F**ormat
- Key features
- Using PDFBox inside applications
- Future prospects

History of Apache PDFBox

Yet another library dealing with pdfs?

- serves only one purpose
  - rendering
  - creating
  - editing
- can‘t be easily embedded in your own application
- wrong language
- license not suitable

The Beginning

- Ben Litchfield started development in 2002
- initial purpose: extract text content to be indexed by the Lucene search engine
- hosted on Sourceforge
- BSD license
- Daniel Wilson and Philip Koch joined PDFBox

Becoming an ASF Project

- several ASF projects showed their interest in a pdf-library
- software granted
- entering Apache Incubator in 2008
- first incubation release 0.8 in May 2009
- graduation as top level project in November 2009
- 10 committers
- current release 1.3.1

Portable Document Format

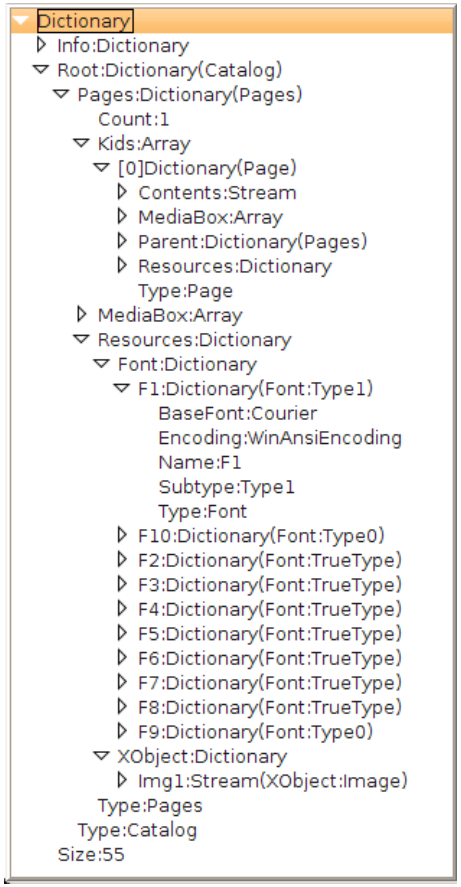
Portable Document Format (1)

- created by Abobe 1993
- open standard for document interchange
- ISO/IEC 32000-1:2008
- uses a subset of postscript
- includes used fonts
- single pages are independent

Portable Document Format (2)

- dictionaries
- streams (command streams or binary data)
- streams are mostly compressed
- encryption support
- consists of several kinds of objects
  - boolean, numeric, string, name, array
  - stream, dictionary

Portable Document Format (3)



Courier: Toto je pokusný text s češtinou – ěščřžýáíé

Times: Toto je pokusný text s češtinou – ěščřžýáíé

Tahoma: Toto je pokusný text s češtinou – ěščřžýáíé

Arial: Toto je pokusný text s češtinou – ěščřžýáíé

Arial Narrow: Toto je pokusný text s češtinou – ěščřžýáíé

Arial Black: Toto je pokusný text s češtinou – ěščřžýáíé

Arial Rounded: Toto je pokusný text s češtinou – ěščřžýáíé

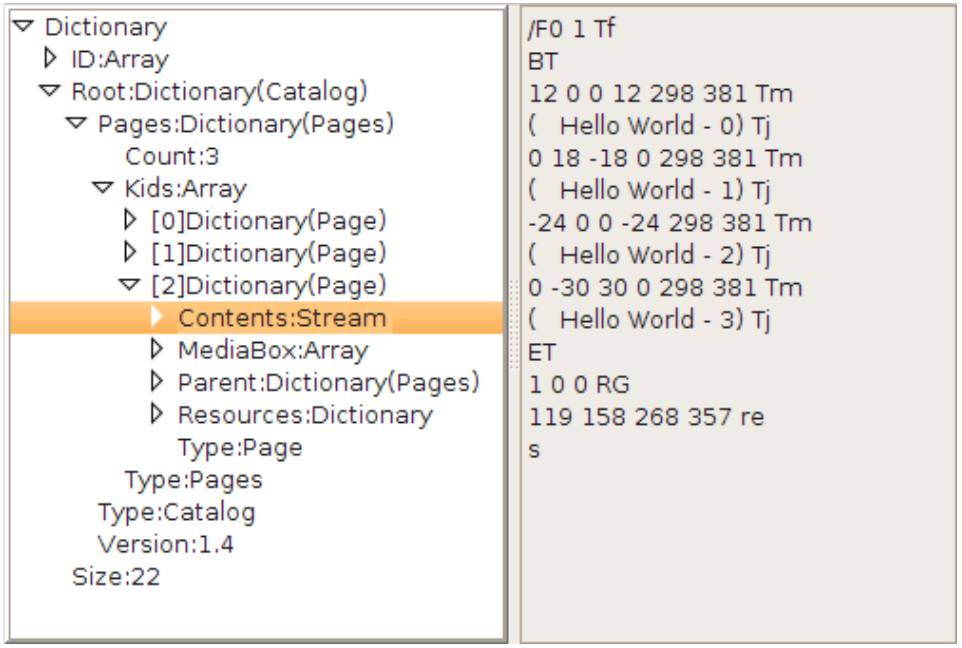
Verdana: Toto je pokusný text s češtinou – ěščřžýáíé

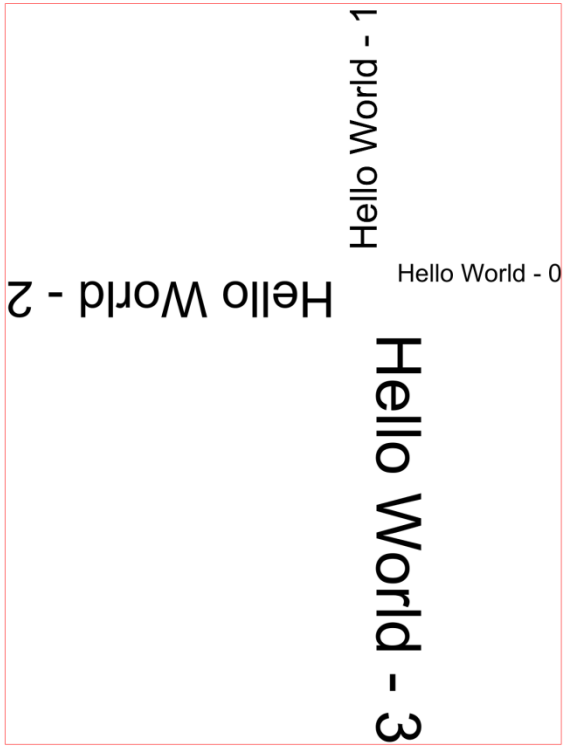
Sans serif: Toto je pokusný text s češtinou – ěščřžýáíé

Portable Document Format (4)

- low level commands to create content
  - draw text
  - create and draw a path
  - draw an image
- no high level concepts
  - high level text formatting like paragraphs, justification, automatic word-wrapping
  - tables

Portable Document Format (5)





Key Features

Text Extraction

- commandline tool „ExtractText“
- possible options
  - automatic sorting of text chunks
  - different encodings
  - optional selection of start and/or end page
  - simple HTML-output

Merging/Splitting pdfs

- commandline tool „PDFSplit“
  - splits a pdf at the given page
  - known issue: PDFBox doesn‘t split the used resources -> results are too large
- commandline tool „PDFMerge“
  - merges two given pdfs into a new one

Creating Images

- commandline tool „PDFToImage“
  - converts a page to an image
  - supports several output formats
  - supports several color models
  - optional selection of start and/or end page
  - different resolutions
  - supports a different crop box

Creating a Lucene Index

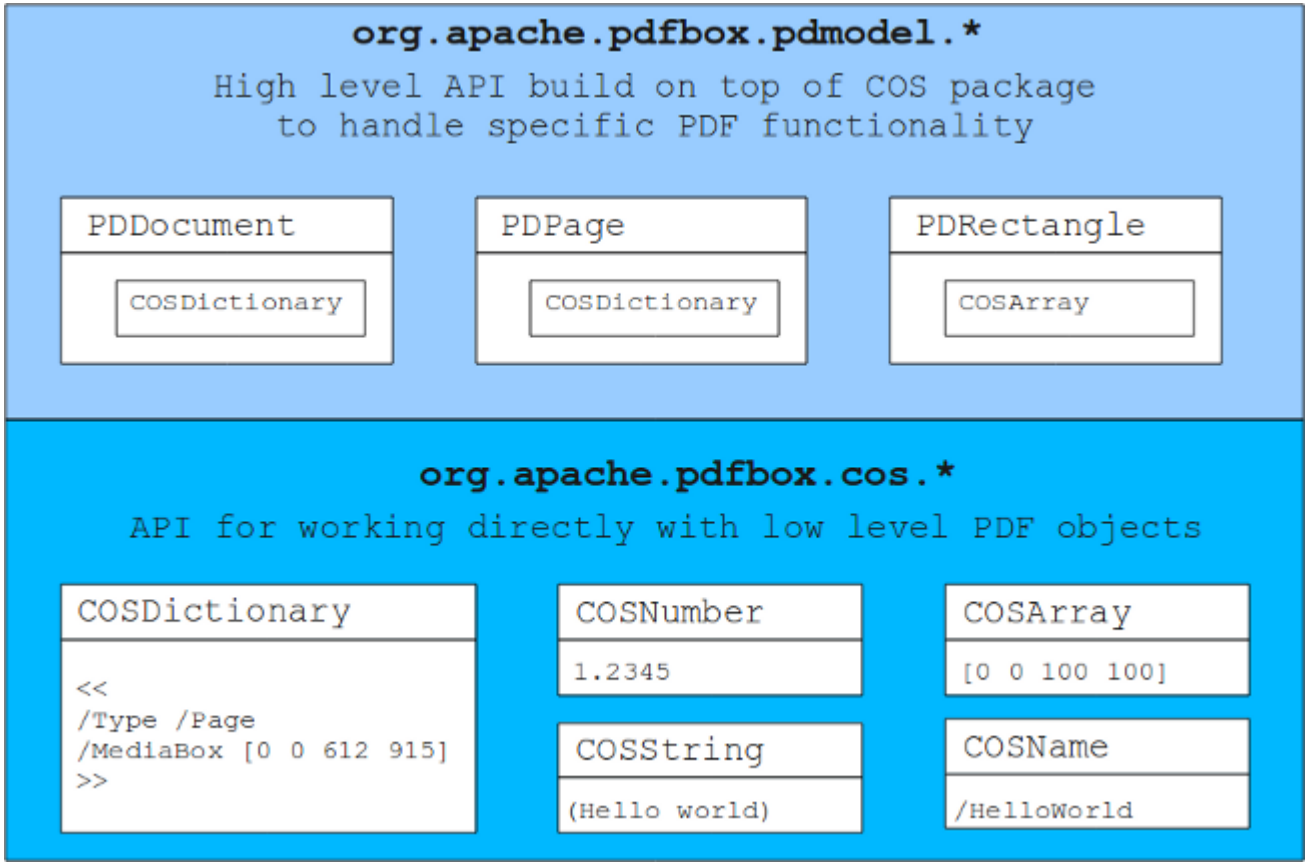
- commandline tool „IndexFiles“
  - part of the lucene component
  - works on all pdf files in the given directory
  - creates a new index or updates an existing index

Printing

- commandline tool „PrintPDF“
  - prints the given pdf
  - printing with/without dialog
  - known limitations: only properly rendered pdfs can be printed

Using PDFBox Inside Applications

Memory Presentation Model



Creating an Empty pdf

```
PDDocument doc = null;
try {
        doc = new PDDocument();
        doc.addPage(new PDPage());
        doc.save(„blank.pdf“);
}
catch(IOException exception) {
        doc.close();
}
```

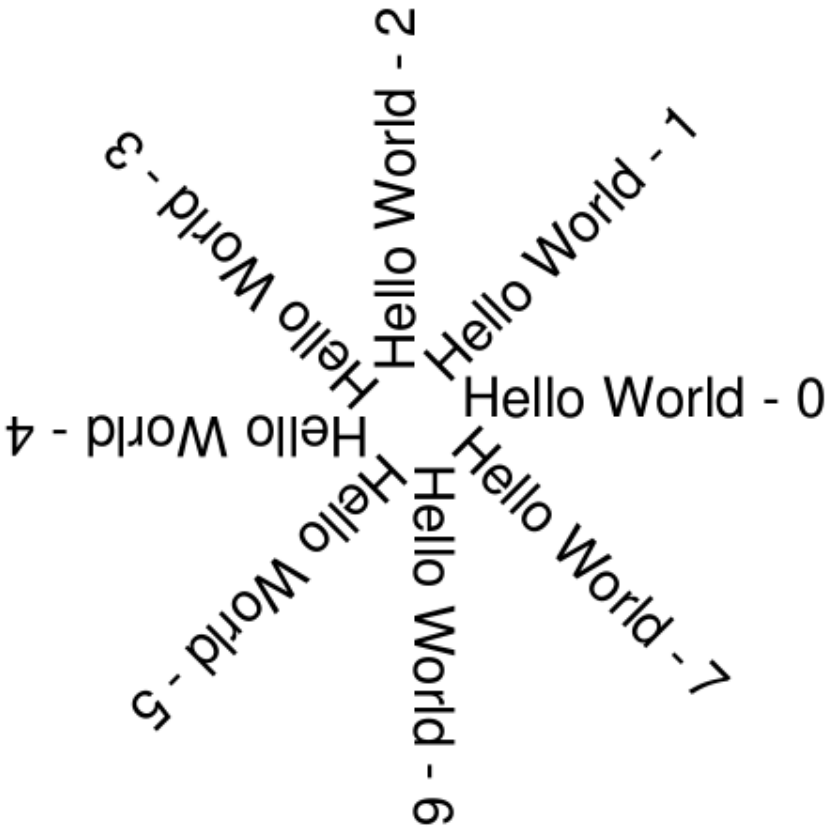
Adding Some Simple Text

```
// create a font
PDFont font = PDType1Font.HELVETICA;
// create a content stream
PDContentStream stream = new PDContentStream(doc,page);
// set font and font size
stream.setFont(font, 12.0f);
// add text
stream.beginText();
stream.moveTextPositionByAmount(100,100);
stream.drawText(„Hello world!!“);
stream.endText();
stream.close();
```

Using the Text Matrix

```
// create a content stream
PDContentStream stream = new PDContentStream(doc,page);
// set font and font size
stream.setFont(font, 12.0f);
// add text
stream.beginText();
for (int i=0;i<8;i++) {
    stream.setTextRotation(-i*Math.PI*0.25,100,100);
    stream.drawText(„  Hello world -“ + i);
}
stream.endText();
stream.close();
```

Using the Text Matrix

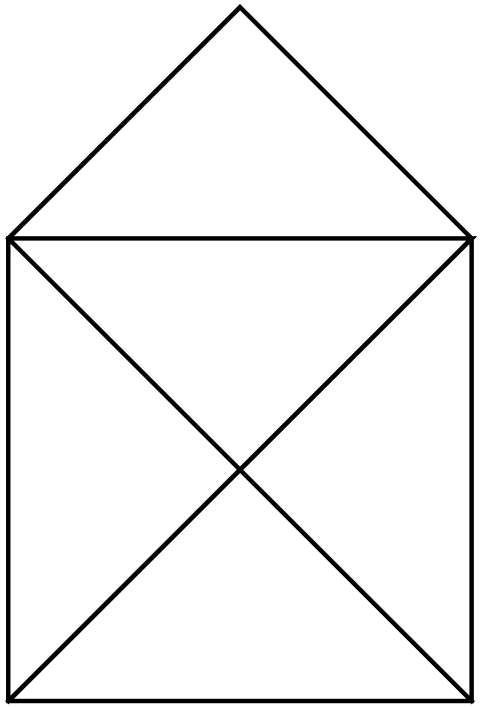


Text Related Methods

```
public void setFont(PDFont font, float fontSize)

public void beginText()
public void endText()

public void moveTextPositionByAmount(float x, float y)

public void setTextMatrix(double a, double b, double c, double d,
        double e, double f);
public void setTextScaling(double sx, double sy,
        double tx, double ty);
public void setTextTranslation(double tx, double ty)
public void setTextRotation(double angle,
        double tx, double ty);
```

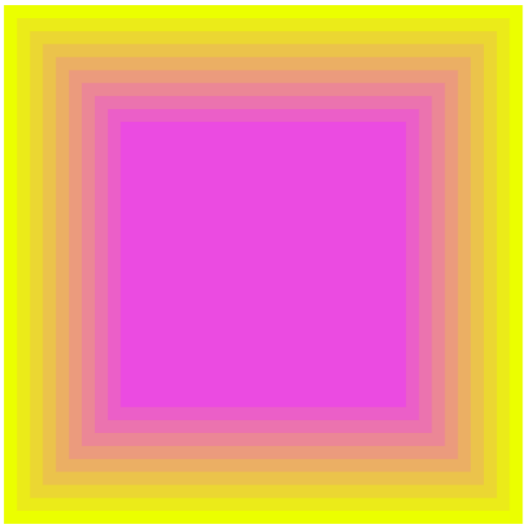
Creating Graphical Content

```
PDPageContentStream stream = new PDPageContentStream(doc, page);
stream.setLineWidth(3);
stream.moveTo(100, 100);
stream.lineTo(400, 400);
stream.lineTo(100, 400);
stream.lineTo(100, 100);
stream.lineTo(400, 100);
stream.lineTo(400, 400);
stream.lineTo(250, 550);
stream.lineTo(100, 400);
stream.lineTo(400, 100);
stream.stroke();
stream.close();
```



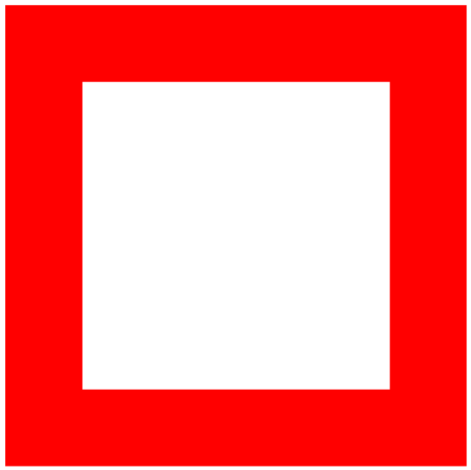
Working with Colors

```
PDPageContentStream stream = new PDPageContentStream(doc, page)

for (int i=0;i<10; i++) {
        stream.setNonStrokingColor(255-20*1, 255-20*i, 25*i);
        stream.fillRect( 10+10*i, 10+10*i, 400-20*i, 400-20*i);
}
stream.close();
```



Path Filling

```
PDPageContentStream stream = new PDPageContentStream(doc, page);
stream.setStrokingColor( Color.BLACK);
stream.setNonStrokingColor( Color.RED);
stream.addRect(100,100,300,300);
stream.addRect(150,150,200,200);
stream.fill(PathIterator.WIND_EVEN_ODD);
stream.close();
```



Graphics Related Methods

```
public void setNonStrokingColorSpace(PDColorSpace colorSpace)
public void setStrokingColorSpace( PDColorSpace colorSpace )

public void drawImage(PDXObjectImage image, float x, float y)

public void concatenate2CTM(double a, double b, double c,
        double d, double e, double f)

public void addRect(float x, float y, float width, float height)
public void fillRect(float x, float y, float width, float height)
public void addBezier312(float x1, float y1, float x2, float y2,
        float x3, float y3)
public void addBezier32(float x2, float y2, float x3, float y3)
public void addBezier31(float x1, float y1, float x3, float y3)
public void drawPolygon(float[] x, float[] y)
public void fillPolygon(float[] x, float[] y)
```

Printing a pdf

```
PDDocument doc = PDDocument.load(„test.pdf“);
PrinterJob job = PrinterJob.getPrinterJob();

// silent print
doc.silentPrint(job);

// open a print dialog
doc.print(jobs)
```

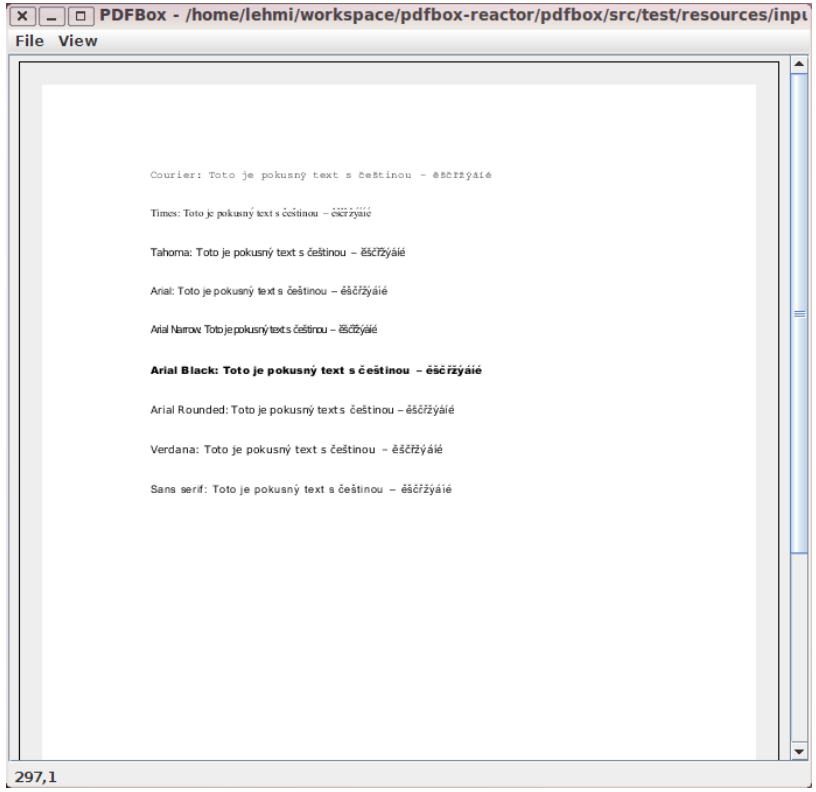
Converting a pdf into Images

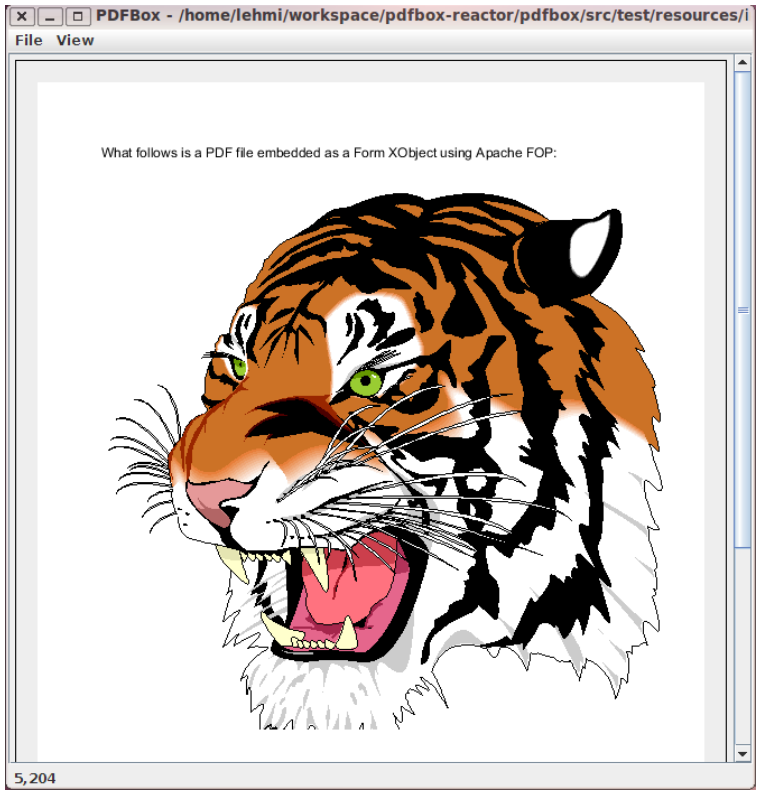
```
// load pdf
PDDocument doc = PDDocument.load(„test.pdf“);
// get all images
List<PDPage> pages = doc.getDocuumentCatalog().getAllPages();
// iterate through all pages
Iterator<PDPage> iterator = pages.iterator();
while (iterator.hasNext())
{
        // create a buffered image
        BufferedImage image = iterator.next().convertToImage();
        // save the buffered image, e.g. using ImageIO
        ...
}
```

Debugging pdfs (1)

- PDFReader bundled with PDFBox
- GUI based tool
  - +/- flips pages
  - print
  - no zoom feature (yet)

Debugging pdfs (2)

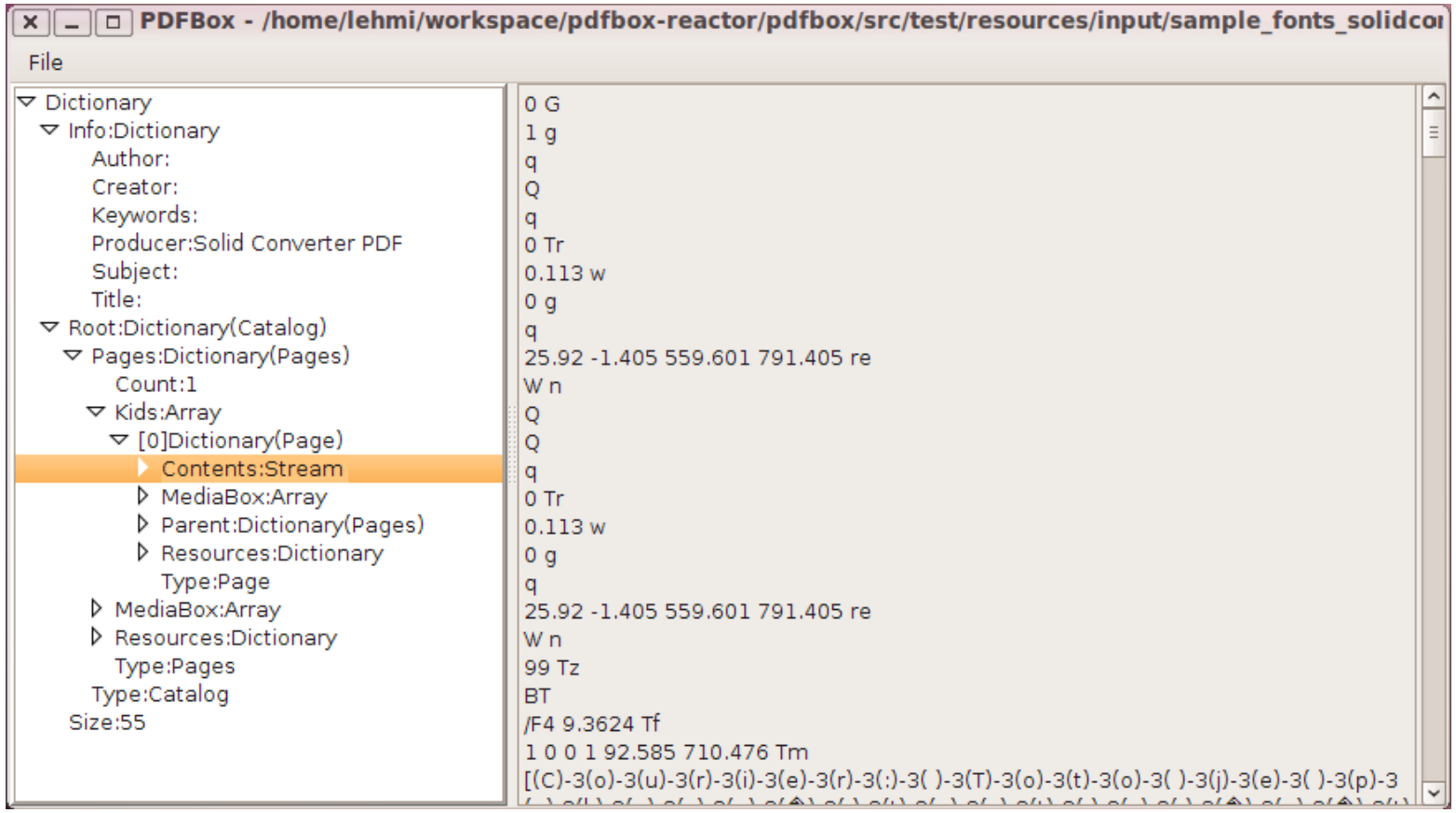




Debugging pdfs (3)

- PDFDebugger bundled with PDFBox
- GUI based tool
- Split-screen
  - left side: treeview of all objects
  - right side: content of the selected object

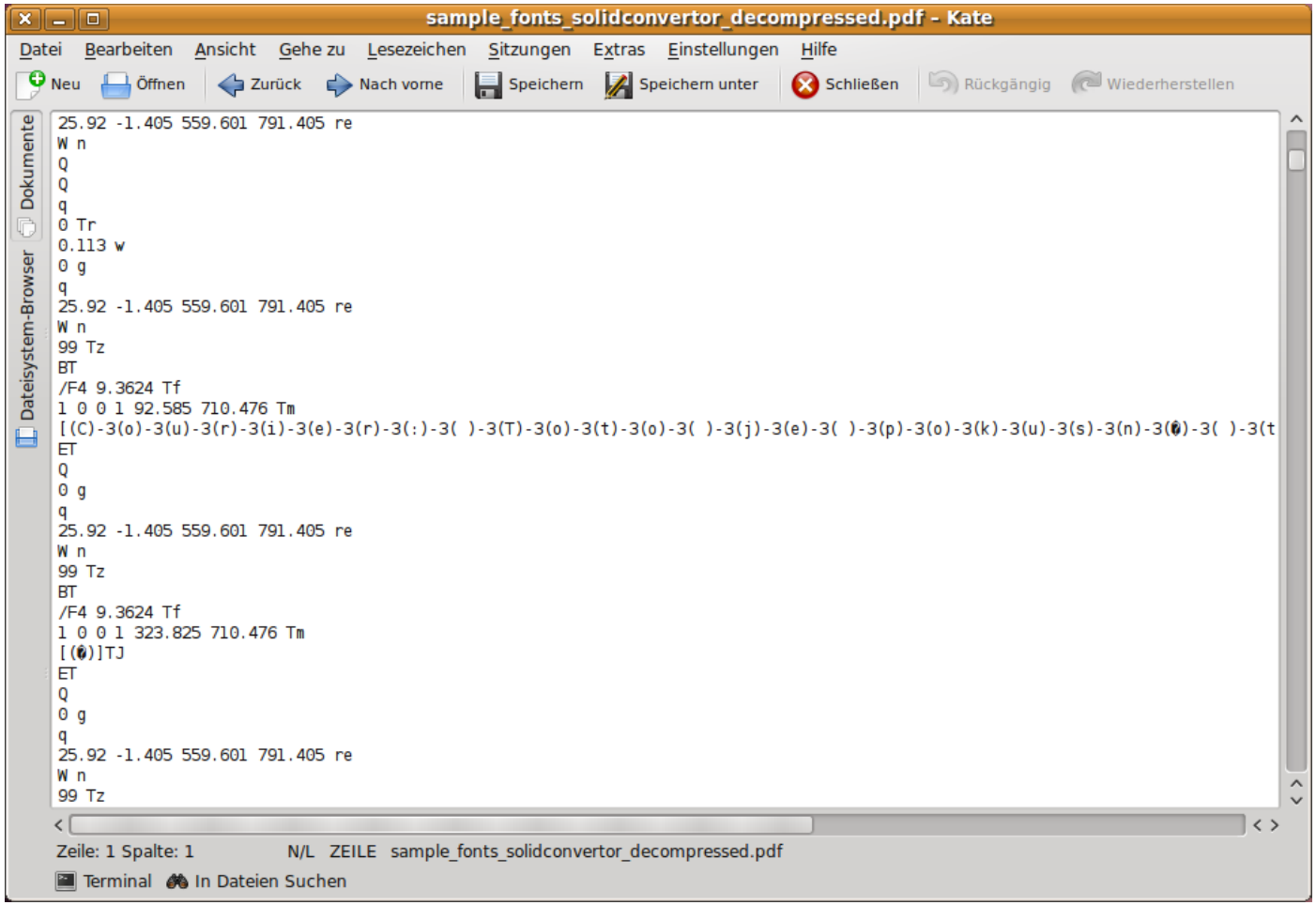
Debugging pdfs (4)



Debugging pdfs (5)

- WriteDecodedDoc bundled with PDFBox
- command line tool to decompress pdfs
- decompressed pdf can be easily opened in any editor

Debugging pdfs (6)



Future Prospects

- Improve font support
  - Type3, CID, encoding …
- Improve rendering
  - Tiling + shading patterns, colorspaces
- Split to several components
  - Core: parser, text extraction, no gui
    -> GAE, Android…
  - Rendering
  - Examples
- GUI handling all commandline tools

Get Involved

http://pdfbox.apache.org

users@pdfbox.apache.org
dev@pdfbox.apache.org

http://www.apache.org/~lehmi/apachecon

http://www.apache.org/foundation/getinvolved.html