Extracting Text From PDF

Version 1.0
Date 04/11/10

Revision history

Doc No:	**Ver sion no.**	**Change reference no.**	**Author**	**Published date**	**Sections changed**	**Description of changes**
	1.0			04/11/10		Base Document

CONTENTS

Purpose of the document

This document describes the following

- How to extract text from a PDF file using the open source "jPod" library. http://opensource.intarsys.de/home/en/index.php?n=JPod.HomePage

Scope of the document

This document only explains how to extract text from a PDF file using the open source "jPod" library. It however doesn't explains other uses of this library like

- Manipulating PDF files
- Creating images from PDF files
- Manipulating and formatting the extracted text for output. *(Interested readers are encouraged to refer to the "Extract2" project which can be found in the SVN)*
- Details on PDF Specifications.

Desired outcome

- Use of the open source "jPod" library to extract text from a PDF file.

Targeted recipient

- Developers

Prerequisite

Recipients are expected to have knowledge on the following which are covered in the presentation "Introduction To SVN".

- Advance developers who have good knowledge on core java and web applications.

Required Software

- jPod.jar
- iscwt.jar

- isrt.jar
- jBig2.jar

All these jar files can be found inside "D:/Softwares/JPOD" in CCS7 machine.

jPod Library

jPod is a Open Source project of the Intarsys group that can be used to read, manipulate and write, along with the basic frameworks to build higher level PDF logic.

Using jPod to Extract Text From a PDF File

To start with you need to add the following libraries as mentioned in the *Required Software* section in your project (or they should be present in your class path)

- jPod.jar
- iscwt.jar
- isrt.jar
- jBig2.jar

Step 1 : - Create the following class "**CommonPdf.java"** which contains the common functionalities

Listing 1 – CommonPdf.java

```
import java.io.IOException;
import de.intarsys.pdf.parser.COSLoadException;//jpod extract pdf method
import de.intarsys.pdf.pd.PDDocument;
import de.intarsys.tools.locator.FileLocator;

/**
 * Common superclass for PDF functionalities
 *
 * <p><strong>Use Cases Supported -</strong>
 * <ol>
 * <li> Opening a document
 * <li> Saving a document
 * <li> Closing a document
 * <li> Creating a new document
 * </ol>
 *
 * </p>
 */
public class CommonPdf {
    private PDDocument doc;//For internal representation of a pdf document

    /**
     * Opens a PDF document
     * @param pathname
     *          The path name to the document.
     * @throws IOException
     * @return PDDocument
     *
     */
    protected PDDocument basicOpen(String pathname) throws IOException,
            COSLoadException {
        FileLocator locator = new FileLocator(pathname);//Initializes a new FileLocator object from the
given file path
        return PDDocument.createFromLocator(locator);
    }

    /**
     * Saves a PDF document
     * @param doc
     *          The PDocument and the name of the output file.
     * @throws IOException
     *
     */
    protected void basicSave(PDDocument doc, String outputFileName)
            throws IOException {
        FileLocator locator = new FileLocator(outputFileName);
        doc.save(locator, null);
    }

    /**
     * Close the current document.
     *
     * @throws IOException
     */
    public void close() throws IOException {
        if (getDoc() != null) {
            getDoc().close();
        }
    }

    /**
     * Create a new document.
     */
    public void create() {
        // First create a new document.
        setDoc(PDDocument.createNew());
        // You could add more information about the environment:
        getDoc().setAuthor("CCS "); //$NON-NLS-1$
        getDoc().setCreator("CCS PDF API"); //$NON-NLS-1$
    }
    /**
     * The current document.
     * @return The current document.
     */
    public PDDocument getDoc() {
        return doc;
    }

    /**
     * Open a document.
     *
     * @param pathname
     *            The path name to the document.
     * @throws COSLoadException
     * @throws IOException
     */
    public void open(String pathname) throws IOException, COSLoadException {
        setDoc(basicOpen(pathname));
    }

    /**
     * Save current document to path.
     *
     * @param outputFileName
     *            The destination path for the document.
     * @throws IOException
     */
    public void save(String outputFileName) throws IOException {
        basicSave(getDoc(), outputFileName);
    }

    /**
     * Set the current document.
     *
     * @param doc
     *            The new current document.
     */
    protected void setDoc(PDDocument doc) {
        this.doc = doc;
    }
}
```

Step 2 - Next Create a class “ExtractText.java” that extents the above class i.e “CommonPdf.java”

Listing 2 – ExtractText.java

```
public class ExtractText extends CommonPdf {


    protected String extractText(PDPageTree pageTree, StringBuilder sb) {
        for (Iterator it = pageTree.getKids().iterator(); it.hasNext();) {
            PDPageNode node = (PDPageNode) it.next();
            if (node.isPage()) {
                try {
                    CSTextExtractor extractor = new CSTextExtractor();
                    PDPage page = (PDPage) node;
                    AffineTransform pageTx = new AffineTransform();
                    PDFGeometryTools.adjustTransform(pageTx, page);
                    extractor.setDeviceTransform(pageTx);
                    CSDeviceBasedInterpreter interpreter = new CSDeviceBasedInterpreter(
                            null, extractor);
                    interpreter.process(page.getContentStream(), page.getResources());
                    sb.append(extractor.getContent());
                } catch (CSException e) {
                    e.printStackTrace();
                }
            } else {
                extractText((PDPageTree) node, sb);
            }
        }
        return sb.toString();
    }


    protected String extractText(String filename) throws COSVisitorException,
            IOException {
        PDDocument doc = getDoc();
        StringBuilder sb = new StringBuilder();
        extractText(doc.getPageTree(), sb);
        return sb.toString();
    }

    public String run(String args) throws Exception {

        String s;
        try {
            String inputFileName = args;
            open(inputFileName);
            s = extractText(inputFileName);
        } finally {
            close();
        }
        return s;
}}
```

Step 3 :

The method highlighted in red color in step 2 i.e "run" is to be called from the client code for extracting. This method accepts as a string the path of a file and returns a string containing the contents of the PDF in text form.

An example client code is given below

Listing 3 : TestPdfExtraction.java

```
public class TestPdfExtraction {

    public static void main(String[] args) throws Exception {
        ExtractText et = new ExtractText();
        String s1 = et.run("C:/pdf1.pdf");

    }
}
```