PDF::Xtract *version 0.08 Date: 2005/04/16*

NAME

PDF::Xtract - Extracting sub PDF documents from a multi page PDF document.

SYNOPSIS

```
use PDF::Xtract;
$pdf=new PDF::Xtract;
@pages=(500..600,1..3,20,40,700..1000);  # Put page numbers to be extracted in an array
$pages=\@pages;             # Get the reference to the above array.
$pdf->savePDFExtract( PDFDoc=>"c:/Docs/my.pdf", PDFSaveAs=>”out.pdf”, PDFPages=>$pages );

OR

use PDF::Xtract;
$pdf = new PDF::Xtract( PDFDoc=>'C:/my.pdf' );
$extract=$pdf->getPDFExtract( PDFPages=>$pages );
print "Content-Type text/plain\n\n<xmp>",  $pdf->getcwPDFExtract;

OR

# Extract and save, in the current directory,  all the pages in a PDF document with nice names.
use PDF::Xtract;
$pdf=new PDF::Xtract( PDFDoc=>"test.pdf" );
@tmp=$pdf->getPDFExtractVariables(PDFPageCountIn);
$PageCount=${$tmp[0]};
print STDERR "Total Pages = $PageCount\n";
$tmp=length($PageCount);
for ( $CurPage=1; $CurPage <= $PageCount; $CurPage++ ) {
        @CurPage=($CurPage); $CurRef=\@CurPage;
        $index=sprintf("%0${tmp}d",$CurPage); # This will hold the current output file name.
        $pdf->savePDFExtract( PDFPages=>$CurRef,PDFSaveAs=>"$index.pdf" );
}
```

DESCRIPTION

- PDF Xtract module is derived from Noel Sharrok's PDF::Extract module. It is a group of methods that allow the user to extract required pages as a new PDF document from a pre-existing PDF document. It is **much** more faster than PDF::Extarct.
- PDF::Xtract is published as a separate module, because of some significant differences with PDF::Extract in variables and functions implemented. While the code, for most part is a shameless copy of PDF::Extract, there are certain changes in the logic that allow this module to be much much faster with large PDF files.
- Notable differences between Xtract and Extract are also highlighted in this document for the benefit of users of PDF::Extract, who wish to use PDF::Xtract.
- With PDF::Xtract one can:-
  - Associate a PDF document to a PDF::Xtract object.
  - Get total number of pages in PDF document.
  - Extract required pages from a PDF document , as a new PDF document, in any specified page number order.
  - Specify name of file to save extracted PDF document.

OVERVIEW, RELEVANCE (and some Test Results!)

- PDF::Xtract modules reads an assigned PDF document and stores information required to reproduce sub pages of it in a set of hashes. When the user specify the page numbers to be extracted as an *array reference*, a new document is immediately generated which will contain pages of original document in the same order as requested. Generated document is written to a disk file. If the variable PDFSaveAs is specified, that will be used as the name of the disk file, otherwise a temporary file name is used.
- If document specified by PDFSaveAs exist, it will be over-written. So, one should assign PDFSaveAs before or along with a request to extract sub-pages.
- Methods available from this module can be called with or without arguments. However, they may not work unless they know the location of the original PDF document and the pages to extract. There are no default values.

This module has been tested with Active State Corp's Perl binary distribution 5.8.4 on MS windows XP. It is NOT tested on any other platform/environment (but I suppose it should work!)**.**

This module significantly outperforms PDF::Extract in speed. I have a 7,781,888 bytes document "test.pdf" of 238 pages. I ran the following code on my Desktop (Intel P-IV 2.8 GHz, 760MB RAM, winXP Pro, SP-1):

```
use Time::HiRes qw(gettimeofday);
use PDF::Xtract;
use PDF::Extract;

my $start=&lt;
$pdf=new PDF::Xtract( PDFDoc=>"test.pdf", PDFErrorLevel=>2, PDFVerbose=>0 );

@pages=(1..119); $pages=\@pages;
$pdf->setVars(PDFSaveAs=>"output-01.pdf", PDFPages=>$pages );
print STDERR "Timer says : PDF::Xtract : ",&lt-$start,"\n";

my $start=&lt;
@pages=(120..238); $pages=\@pages;
$pdf->setVars(PDFSaveAs=>"output-02.pdf", PDFPages=>$pages );
print STDERR "Timer says : PDF::Xtract : ",&lt-$start,"\n";

my $start=&lt;
$pdf1=new PDF::Extract( PDFDoc=>"test.pdf" );
$pdf1->setVars(PDFPages=>"1..119"); $pdf1->savePDFExtract(PDFSaveAs=>"output-11.pdf");
print STDERR "Timer says : PDF::Extract : ",&lt-$start,"\n";
my $start=&lt;
$pdf1->setVars(PDFPages=>"120..238" );  $pdf1->savePDFExtract(PDFSaveAs=>"output-12.pdf");
print STDERR "Timer says : PDF::Extract : ",&lt-$start,"\n";

sub lt{
        # Stuff used while debugging and performance checks
        my @timer=gettimeofday();
        my $timeNow=$timer[0]+$timer[1]/1000000;
        return $timeNow;
}
```

Here is the Result : (format edited)

```
Timer says : PDF::Xtract :   0.33763313293457
Timer says : PDF::Xtract :   0.06195712089538
Timer says : PDF::Extract : 41.299989938736
Timer says : PDF::Extract : 26.859375
```

First and the 3rd lines in the result includes time taken by respective modules to read the assigned PDF document apart from the time taken to extract first 119 pages. In this case, Xtract was about 125 times faster than Extract.

Second and fourth line is the result of just extracting pages 120-238, after already having read the input PDF. If compared, we can see that Xtract was about **450 <u>times faster</u>** than Extract!! (makes me happy!).

When tested with larger PDF files (100s of MB), PDF::Extract did not give any result till my patience ran out (Hours!!), but Xtract always finished in seconds.

VARIABLES

PDFDoc (set and get)

```
$file=$pdf->getVars("PDFDoc");
```

This variable contains the path to the original PDF document PDF::Xtract object refers to. This assignment makes PDF::Xtract read and store information from the document to recreate any page within it as a PDF document.

PDFSaveAs (set and get) *[note : differences with PDF::Extract]*

This variable is set to path-file name to which savePDFExtract will save extracted document. (Note: If you do not change the value of PDFSave as before calling savePDFExtract, you might be over-writing a possibly existing file. For example, suppose you are trying to burst a PDF file to separate files – each containing a single page of original PDF. If you miss out changing the file name to save in PDFSaveAs when you change PDFPages in a loop, you may end up with just one file containing only the last page of original file.)

PDFPages (set and get) *[note : differences with PDF::Extract]*

This is set to an array reference of page numbers (to be) extracted from PDF::Xtract object. It should not be a string or an array – only an array reference.

```
@pages=(reverse(1..55),10,reverse(150..200)); $ref_pages=\@pages;
@tmp=$pdf->setVars( PDFPages =>$ref_pages); # Returns an array

print "Pages are", join(",",@{$tmp[0]});     # Value for PDFPages is an array reference.
```

Notes:

- When you set PDFPages, module will immediately extract those pages in the same order and place in the file specified by PDFSaveAs. If it is not set, it will use a temporary file name. You can use getPDFExtract/savePDFExtract to retrieve the document from the temporary file. However, it more efficient to just set the PDFSaveAs beforehand.
- Xtract retrieves pages in the same order as specified in the request PDFPages, not in their natural order. For example specifying page numbers as (1,2,3) is different from specifying as (3,2,1).
- If user specifies a set of pages of which *only some of the pages are present* in the original document, then those pages which are present in the original document will get extracted. Those page numbers which failed to get extracted will be in array PDFPagesNotFound.

PDFCache (set and get) *[note : differences with PDF::Extract]* This variable is not used by the Xtract module.

PDFErrorPage (set and get) *[note : differences with PDF::Extract]* This variable is not used by the Xtract module.

PDFExtract (get only) *[note : differences with PDF::Extract]* This variable is not used by the Xtract module.

PDFPagesFound (get only)

This variable contains an array reference of the page numbers that were selected and found within the original PDF document. PDFPagesFound will be a undefined if no page got selected.

```
$pagesFound=@{($pdf->getVars("PDFPagesFound"))[0]};   # getVars returns an array and PDFPagesFound is an array reference.
or
@pages = @{($pdf->getVars("PDFPagesFound"))[0]};
```

- **PDFPageCount** (get only) *[note : differences with PDF::Extract]* This variable is not used by the Xtract module.

- **PDFFileName** (get only) *[note : differences with PDF::Extract]* This variable is not used by the Xtract module.

- **PDFError** (get only) *[note : differences with PDF::Extract]*

    - @error=@{($pdf->getVars("PDFError"))[0]};
    - This variable contains an array reference; each element of the array holding a string describing the errors if any in processing the original PDF file. More recent errors will have higher index. In case of PDF::Extract, this varable used to hold a single string. PDFErrorSize fixes maximum size of the array.

- **PDFErrorLevel** (get/set)

  Module will log those stuff of greater sensitivity than PDFErrorLevel. Default is 3 (errors). Other possible levels are 0(silly), 1(Info), 2 (warning).

- **PDFErrorSize** (get/set)

  Holds the size of the PDFError array. Default is 20 elements.

- **PDFDebug** (get/set) Not very significant....

- **PDFDocStat** (get) Internally used for checking whether there is a re-assignment of PDFDoc to same input file!

- **PDFPageCountIn** (get)

  Stores number of pages in the original PDF document.
  $InputPages=$pdf->(getVars(PDFPageCountIn))[0];

- **PDFPageCountOut** (get) Number of pages successfully extracted in the last extraction operation.

- **PDFPageCountErr** (get) Number of pages that *failed* to get extracted.

- **PDFPagesFound** (get) Reference to the array containing successfully extracted pages.

- **PDFPagesNotFound** (get) Reference to array of pages that could not be extracted.

- **PDFVerbose** (get/set)

  Make the module talkative! Module will print all that goes to PDFError array to STDERR too. Usable as a debugging feature or to know that something is going on when you are working with very lare PDF files.

- **PDFReadSize** (get/set)

  Modules reads in the PDF file in chunks, PDFReadSize specify the chunk size in bytes. Default is 1024000 bytes. Setting larger values should be OK if you have a lot of RAM, but setting it too high may cause the module to fail without any message. Earlier versions of Xtract read the complete input PDF in one shot, but this approach was failing in case of PDF files larger than 450MB. Version 0.07 was tested successfully, with a 1200MB PDF and PDFReadSize of 102,400,000 bytes.

- **PDFWriteSize** (get/set)

  Module writes extracted pages to disk file after accumulating PDFWriteSize bytes of data in memory. Default is

1024000. This does not seem to have any significant impact on performance. It is just there!

- **PDFClean**

If set to a positive number, PDF::Xtract will fail if any error occur (such as a requested page not found) during extraction. This behavior is PDF::Extract.

METHODS

$pdf=**new** PDF::Xtract;
new PDF::Xtract(PDFDoc=>"c:/Docs/my.pdf", PDFPages=>$array_ref , PDFSaveAs="c:/out.pdf")

This will create a new PDF::Xtract object and assign specified module variables. However, it can also be called without any argument.

$output=$pdf->**getPDFExtract; #** Returns the last extracted PDF sub document. This sub document is created *immediately* on user specifying PDFPages.

$pdf->**savePDFExtract;** # This method saves its output to the file defined for PDFSaveAs.
# Not setting PDFSaveAs results in an error.

$pdf->**getVars OR** $pdf->**getPDFExtractVariables;**
$pdf->**setVars OR** $pdf->**setPDFExtractVariables;**

Above methods, obviously, retrieves OR sets values of public variables associated with th PDF::Xtract object. GetVars returns an array who's individual elements might be references to arrays (for example, in case of PDFPagesFound it will be reference to an array, for PDFPagesCountIn it will be a scalar.) See below:

```
unless ($pdf->getPDFExtract ) {
        # Suppose there was an error
        @error=@{($pdf->getVars("PDFError"))[0]};
}
```

Arguments for setVars is an array of key-value pairs for public variables, which will be set.

If you set a variable not specified in this documentation but name starting with "My", it will be set as a public variable and can subsequently be get from the Xtract object. But, there is no guarantee that these may be available to future versions. *[note : differences with PDF::Extract]*

AUTHOR

SunilS <mailto:sunils_AT_hpcl_DOT_co_DOT_in>

LICENSE

DISCLAIMER