EXT: PDF Generator 2

Extension Key: **pdf_generator2**

Table of Contents

Introduction

What does it do?

If the user wants to print a document document served by typo3 he is usually offered a HTML page that contains only the relevant content of the page while the images are served as separate files. This works fine for printing, but if the user wants to save the page for archiving or offline printing it would be nice to embed the images within only one document. Adobe Acrobat PDF offers a suitable file format for that purpose which is widely and os-independently supported.

This extension is based on the pdf_generator but uses a different html to pdf converter.

How does it work?

The extension generates a “printer-friendly” HTML-page internally. This page is passed through html2pdf to generate a PDF file. The PDF file is then served to the user while being cached by the usual caching mechanism of typo3.

The PDF page can be linked to from the Typoscript template using the stdWrap.postUserFunc parameter. Either dynamic Filenames (i.e. Index.php?id=12&type=123) or static (<alias>.PDF of 12.pdf) filenames are supported.

What is html2pdf?

Html2pdf s a 100% php based script that converts html including css to pdf (and alternatively to postscript which is not supported here). Unlike the htmldoc based solution css is supported and no external binaries are required.

It uses either the fpdf or the pdflib library to perform the rendering. Both methods give slightly different result both regarding rendering speed and rendering results. If you want to use pdflib you'll have to install additional fonts.

Supported content
Theoretically the PDF_generator 2 extension supports all content that can be rendered in HTML with css (i.e. all). It is however restricted by the html2pdf script it is using.

TemplaVoila Support
The extension supports templavoila by using the extension tv_pdfgen. See this extensions manual for a description on how to use templavoila with the pdf_generator2.

File Based Caching
If you do not want to store large generated files in the database, you can use the additional extension file_based_cache which will cache large files in the filesystem instead.

Users manual
The user should include an link to http://www.adobe.com/products/acrobat/readstep.html somewhere on the page to allow users to download the free acrobat reader if the do not have it already.

Not much else has to be done from a Users point of view. Simply click the link to the PDF File. The setup will determine whether the file will be opened in the browser window (inline) or in an external adobe acrobat viewer window (works on Internet explorer, not on Netscape. Netscape will always open the PDF inline).

Administration

Install the Extension
1. Download the newest Extension using the Extension Manager/Extension Repository
2. Install the Extension using the Extension Manager
3. Configure the Extension.
4. install additional fonts for pdflib (sometime they are also needed for fpdf, i haven't found out when exactly yet. If you see an error message that a font can't be found, install them)
5. Update your .htaccess if you want to use “static” PDF filenames
6. You **don't** need to install the fpdf extension, the fpdf library is already included in this one.

Install additional fonts for PDFlib
If you want to use PDFlib you'll have to install additional fonts. This is most easily done by installing the pdf_generator2_fonts extension. They are packaged separately from the rest to keep the extension sizes manageable.

Update .htaccess
You will need to update your .htaccess if you want the links to your PDF files be static (i.e. <alias.pdf>). This change is necessary to forward all requests to /<...>.PDF files to typo3's index.php.

```
ErrorDocument 404 /error.html
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule   ^typo3$  /typo3/index_re.php
RewriteRule   ^[^/]*\.html$  index.php
RewriteRule   ^[^/]*\.pdf$  index.php
```

Available Options

simulateStaticPdf
This parameter will define whether the link to the PDF files will be a static one (i.e. <alias>.pdf) or dynamic ones (index.php?id=12&type=123). If you enable this you will need to update your .htaccess.

Disable gzip compression
If this parameter is set and you have gzip-compression turned on for the Frontend ([FE][compressionLevel] in the installation tool) the generated PDF's will not be handled by the additional gzip-compression. It is highly recomended to leave this parameter at default (on) since PDF-files incorporate their own compression and you will only gain little additional compression. Additionally you may get problems with browsers that pass the stream to acrobat without decompressing it (e.g. Netscape 4.08).

typeID
This parameter defines the typeId that is used for the extension. Normally you will not need to change it.

Add a link in your Typoscript Template using makePdfLink

To add the a link to the PDF page you can wrap any content element in a link using stdwrap.postUserFunc. The following code gives some examples:

```
[...]
100 = IMAGE
100.file = fileadmin/pdf_link.gif
100.stdWrap.postUserFunc = tx_pdfgenerator2->makePdfLink
[...]

[...]
110 = TEXT
110.value = printable version
110.postUserFunc = tx_pdfgenerator2->makePdfLink
110.postUserFunc.target = _blank
[...]

[...]
temp.PDF = TEXT
temp.PDF.value = pdf
temp.PDF.postUserFunc = tx_pdfgenerator2->makePdfLink
temp.PDF.postUserFunc.include_post_vars = 1

page.120 < temp.PRINT
[...]
```

If you use the Template Auto-Parser extension and you want to put the link somewhere on your page, you may need to put it in a COA with another element. If you want to place it below the content for example you would change

```
  subparts.content < styles.content.get
```

to something like

```
  subparts.content = COA
  subparts.content {
    10 = CONTENT
    10 < styles.content.get
    20 < temp.PDF
  }
```

and define temp.PDF like above.

Please note that the link will be generated as a USER_INT object by default, i.e. that the link will be generated after the page has been pulled from the cache. If you want to go for maximum performance you can disable that with the parameter postUserFunc.no_user_int = 1 which will generate and cache the link. This is only recommended if you are sure that there are no USER_INT objects on the page that set POST or GET parameters. If you don't understand what i'm writing about leave it at default.

Add a link in your Typoscript Template using openPdfLink

Altenatively you can also use the function tx_pdfgenerator2->openPdfLink which will generate the opening tag for the link. You will need to generate the closing tag yourself´. Example:

```
page.36= USER_INT
page.36.userFunc = tx_pdfgenerator2->openPdfLink

page.37=TEXT
page.37.value = Link to PDF

page.38=HTML
page.38.value = </a>
```

Add a link in your Typoscript Template using getPdfTarget

If you just need the target of the pdf link you can use the function getPdfTarget. Amongst other things this can be useful with templavoila. Example:

```
lib.pdflink = USER_INT
lib.pdflink.userFunc = tx_pdfgenerator2->getPdfTarget
lib.pdflink.userFunc.include_post_vars = 1
```

Customizing the PDF-Page via Typoscript

The PDF is rendered based on a very simple template. You can easily update the pdf_generator page object to contain additional information or content elements on every PDF page. One especially interesting use is to add special HTML comments to control the behavior of the html2pdf conversion. See the html2pdf documentation (on http://www.tufat.com/docs/html2ps/index.html) for details. A lot of parameters can be set via the Constant Editor (see next chapter).

You don't have to put any of the following lines into your template, they are already there. They are noted here just for reference purposes.

```
pdf_generator = PAGE
pdf_generator {
```

```
  typeNum = {$extension.pdf_generator.typeNum}
  config.pageGenScript = EXT:pdf_generator/gen_pdf.php
  config.admPanel = 0
  config.xhtml_cleaning = 0
  config.USERNAME_substToken =
  config.ftu = 0
  config.disableCharsetHeader = 1
  config.prefixLocalAnchors = 0

  50 = CONTENT
  50 < styles.content.get
}
```

Customizing the PDF-Page via the Constant Editor

You can set several options on a per-template basis with the Constant Editor. These options will result in parameters used for the html2pdf conversion. The following parameters can be set:

Property:	Data type:	Description:	Default:
browserwidth	string	If set controlls the width of the page in pixels. Set this to control scaling of tables and pictures. It defaults to 700 which will give roughly the same resolution as pages printed from Firefox or Internet Explorer.	700
size	string	Sets the size of the generated PDF. The size can be Letter, Legal, Executive, B5, Folio, A0Oversize and A0 through A10 Custom sizes are specified by the page width and length in mm separated by the letter "x" like 210x297	A4
landscape	boolean	Set this to true if you want to generate the page in landscape mode	False
top	string	Specifies the top margin in mm.	13
bottom	string	Specifies the bottom margin in mm.	13
left	string	Specifies the left margin in mm.	13
right	string	Specifies the right margin in mm.	13
pdfversion	string	Specifies the PDF version to render to.	1.3
use_pdflib	boolean	If this parameter is set the rendereing will be done with pdflib instead of fpdf. This will result in slightly different rendering results and times. You'll need to install additional fonts if you want to use this.	false
cssmedia	String	Css can contain several different medias. Here you can select which one to use.	screen
renderlinks	boolean	Define whether links are rendered clickable.	true
renderfields	boolean	Define whether fields are rendered editable. May not be supported with dlib depending on the pdflib version.	true
Rederforms	boolean	Define whether forms are rendered as dynamic pdf forms. Only supported with fpdf.	false
string_search[1-4]	string	Using this parameter together with string_replace you can remove/replace certain parts of your html-file prior to passing it to the html2pdf conversion. One example is to remove the “Back” link in the recipe extension. You can do this (when using english language) with string_search1 = “<A href="javascript:history.go(-1);">Back</A>” and string_replace1 = “” You can actually use string_search[5-n] as long as they are all set	
string_replace[1-4]	string	See string_search[1-4]	
regexp_search[1-4]	string	The same as string_search, but for regular expressions. The search and replace parameters must be valid regular expressions that will be passed to preg_replace. You can actually use regexp_search[5-n] as long as they are all set	
regexp_replace[1-4]	string	See regexp_search[1-4]	

Reference for tx_pdfgenerator2->makePdfLink, tx_pdfgenerator2->openPdfLink and tx_pdfgenerator2->getPdfTarget

Note that not all properties make sense for all functions. For example: the getPdfTarget function doesn't support the Target property.

Property:	Data type:	Description:	Default:

attachment	int	This controls the setting for the Content-Disposition header. Normally a
		
		`Content-Type: application/pdf`
		`Content-Disposition: inline`
		
		header is generated. If this parameter is set to 1
		
		`Content-Type: application/pdf`
		`Content-Disposition: attachment`
		
		and if it is set to 2
		
		`Content-Type: application/octet-stream`
		`Content-Disposition: attachment`
		
		is sent instead.
		Option 1 will force Internet Explorer to open the PDF file in a new Acrobat Reader window. Netscape will open it inline (same as normal). If it is set to 2 both IE and Netscape will query the user what to do with the file. (Tested on win2k with IE6 and Netscape 7)
		The change is created by attaching a “attachment=<...>” parameter to the URL.
			0

filename	string	Sets the a filename for the Content-Disposition header. Since most browsers seem to ignore this it is of limited use.	
include_post_vars	boolean	If set to 1 the received POST variables will be appended as GET variables to the URL. This is useful if you want to generate dynamic data (like search result for example). IMPORTANT NOTE: if you set this parameter it is possible that passwords and other sensible information will get encoded in the URL and thus be logged in proxy-logs for example.	false
no_blur	boolean	If this is set, no onFocus="blurLink(this); code is added to the link. You can also use noBlur instead.	
no_user_int	boolean	If set the generated link will not be a non-cached USER_INT object. Use this for slightly better performance if you have no frontend plugins that act as USER_INTobjects and set GET or POST variables. If you don't understand what i'm writing leave it at default. This parameter is only supported for the makePdfLink function.	false
simulateStaticPdf	boolean	Overrides the global setting of the extension.	
Target	string	Sets the target for the link	
ATagParams	string	Ads additional parameters to the <a..> tag.	

Special Tags

You can force a page break by inserting any one of the following commands into the HTML page:

```
<!--NewPage-->
<pagebreak/>
<?page-break>
```

It is highly NOT RECOMMENDED to use these directives inside table cells, as you can get unpredictable results.

The following command performs a “section” break”.

```
<sectionbreak/>
```

Elements that are positioned fixed (i.e. on all pages), will only show up in the section they are declared in. Basically the body tag is split at the points where this command is found, the split up parts are then inserted in the original body tag again, thereby forming several new html-documents. If you keep this in mind, you'll also understand that tis command should only show up in places that are at the level below the body tag, otherwise you'll get unpredictable results

Special Fields

The following fields are rendered if the render fields checkbox in the constant editor is checked (default):

Name	Replaced with
##PAGE##	Number of current page
##PAGES##	Total number of pages in the generated PDF file
##FILENAME##	Filename (URL) of the source HTML
##FILESIZE##	Size of the source file (without included files)
##TIMESTAMP##	Date/time the PDF file have been generated

Internals

The extension consists of two php classes and one php script (plus a whole lot of classses that perform the actual html to pff conversion)

gen_pdf.php

This is the main script that generates the PDF file. This is done by calling the original pagegen script and then passing it to the html2pdf conversion script. The generated PDF is then returned to typo3 to be cached for subsequent requests. This script also “fixes” the following issues with the generated HTML:

1. Sometimes the RTE generates <P> tags within <PRE> tags. They are removed
2. When using buletted lists the output is theoretically correct but looks quite ugly (the dots are quite a bit above the text line. By changing the alignment of the images to left they look much better.

class.tx_pdfgenerator2.php

This class contains the makePdfLink function that is used to generate the link to the PDF file. It takes the current page id and all attached GET parameters and creates a new link using these values. If the simulateStaticPdf option is set these links will be named <alias>.pdf?<additional parameters>. if not they will be called index.php?id=<id>&type=<pdf_type>&<additional parameters>.

class.ux_tslib_fe.php

This class extends/overrides the core tslib_fe class. Two functions are overridden in typo3 <3.7. In typo3 3.7 and above callbacks are used instead:

checkAlternativeIdMethods

This change will check for a “.pdf” file ending and automatically generate the pageType for the PDF page (default:123)

processOutput

This change will check if a PDF file is currently generated and will add the additionally needed headers then. These include

- Content-Type and Content-Disposition
  see Reference for postUserFunc = tx_pdfgenerator2->makePdfLink
- Cache-control: private, Connection: Keep-Alive
  These are need for Internet Explorer to allow downloading of the file and to avoid IE requesting it twice.
- Content-Length: <calculated length>

  This is also needed to avoid the need to press Refresh on IE when working with small PDF files.

Additionally the gzip compression is turned off here by setting the ['FE']['compressionlevel'] to ''.

Known problems

- No proper check is done on parameters

FAQ

- **Q:** I see errors starting with:
- ```
  MySQL server has gone away
  Warning: Cannot add header information - headers already sent by (output started at /local/typo3/
  httpd/htdocs/tslib/class.tslib_fe.php:1312)
  ```
- **A:** This may happen when large pdf files are generated. The MySql Documentation recomends to set the max_allowed_packet=# value to a larger value.
  You can also use the extension file_based_cache in order to store large files in the file system instead of the database.

Troubleshooting

Things to do if anything fails:

- Check the output for any clues
- Check the html2pdf documentation for any clues: http://www.tufat.com/docs/html2ps/index.html
- enable writing of the interim html data to a file by editing the pdfgen.php script. Search for the line
  // write the html for debugging puposes
  and replace the 'if(0)' with 'if(1)' in the line below.

To-Do list

- Performance can probably be optimized a bit
  ```

- Better parameter checking
- Better debugging

Changelog

- 0.0.1: - Initial Version, based on the pdf_generator 3.3.0
- 0.1.0: -fixed usage of /tmp path
- 0.1.1: - fixed some small bugs with the constant editor defaults
  - fixed another bug where images were not rendered properly
  - support fonts for pdflib in an additional package
- 0.1.2: - fixed internal links
- 0.2.0: - new version based on html2ps 1.9
- 0.3.0: - new version based on html2ps 1.9.2, fix “getfontascender” bug
- 0.3.1: - fixed bug in html2ps
- 0.4.0: - fixed a bug concerning remotely linked (and “pseudo-remotely” linked via a <base..:> tag) files.
- 0.5.0: - fields (pagenumbers etc. are now supported), - support for <sectionbreak/>