Assist PDF Production Code

Aptean Ltd

Contents

1 Assist PDF Production Code

The extension adds the ability to export Wiki pages as PDF, as well as combining pages in categories into a single PDF.

The intention is that, when extracting a PDF document, the application will:

- Prompt for the download of the PDF. This will include a version number if one is provided.
- User saves and the download will appear in the downloads list in the browser.

The process allows for:

- Downloading any single page as a PDF.
- Downloading any category as a book, where the first page in the category is seen as the title page.

Assist also implements the ability to directly access and download a PDF version of a category or page through the use of the DocLink template.

1.1 Extension:PDFBook

The extension used in Aptean Assist is a heavily-modified version of the extension publicly available.

Note that the version on MediaWiki is out of date and doesn't work properly. The latest version should be downloaded from the git repository, and then amended.

1.2 Versions of the extension

Versions exist in extensions/Old Extensions for MediaWiki versions 1.16 and 1.34.

The latest version in extensions works for 1.39+

1.3 Enabling the extension

- Copy the extension from an existing wiki or codebase.
- Enable in LocalSettings.php

```
<code>
wfLoadExtension( 'PdfBook' );
</code>
```

- Configure in LocalSettings.php

```
<code>
$wgPdfBookTab   = true; # Whether or not an action tab is wanted for printing to PDF
$wgPdfBookLeftMargin    = "1cm";        # Left page margin
$wgPdfBookRightMargin   = "1cm";        # Right page margin
$wgPdfBookTopMargin     = "1cm";        # Top page margin
$wgPdfBookBottomMargin  = "1.5cm";      # Bottom page margin
$wgPdfBookFont  = "Arial";      # Default font to use if unspecified in content
$wgPdfBookFontSize      = 12;   # Point size of default font
$wgPdfBookLinkColour    = "217A28";     # Colour to use when rendering hyperlinks in text
$wgPdfBookTocLevels     = 2;    # Number of outline levels to use when building the table of contents
#$wgPdfBookExclude      empty   # List of article titles which should not be included in the book
$wgPdfBookFormat = "single";
$wgPdfBookWidth = "1000";
$wgPdfBookDebug = false; // writes extra debugging statements
$wgPdfBookExtDebug = false; // Extra debugging - Leaves produced HTML files behind so they can be viewed
<code>
```


1.4 Structure of code

There are 3 main files:

- extension.json - the declaration of the extension and the dependencies.
- PdfBookAction.php - the main code
- PdfBookHooks.php - adds hooks to the skins i.e. the links for download/print as PDF.

1.4.1 extension.json

Modified PHP requirement to 7.4 - works fine.

1.4.2 PdfBookHooks.php

Modified function actionLink to call the default as singlebook rather than single, as the changes below format the book better.

1.4.3 PdfBookAction.php

All important code and changes are in public function show.

1.4.3.1 Structure

- Sets up environment and settings
- Extracts all articles
- Creates a cache from the articles, code and querystring
- If cache does not exist,
    - ♦ create content of cache from pages, sorted in category sort sequence if part of a category
    - ♦ Replaces are made for various purposes.
    - ♦ Version of the document is extracted from the pages if present
    - ♦ Splits into title page and content pages
        - ◊ If a single page, splits as the TOC comment
        - ◊ If multiple pages from a category, first page is title, every other page is content
        - ◊ Adds HTML Headers
        - ◊ Builds the HTMLDoc commend.
        - ◊ Converts the articles into a PDF using the HTMLDoc command into the stored cache file.
- If cache does exist,
    - ♦ Version of the document is extracted from the pages if present
- Output the cache file

1.4.3.2 Changes

Added Debug control. Also adds lots of debugging statements. You can view the debug statements in the database - see Assist Support Guide for some information on how to do that. Also, you can just click Page Logs from the page to see the logs associated to the extraction.

Fixed some bugs in how it refers to $title in some sections, to ensure that the debugging is correct (now uses $doctitle instead in debugging and throughout the code where the original document is referenced).

Added formatting checks based on start of title of page being produced. If they start with standard formatting e.g. FS, SDD, OV, UG, then changes format to singlebook. Note this is now the default anyway, so largely redundant.

Adds body and header/footer images, which must be present in the upload directory - they should be anyway:

<code>

```
                $bodyimagefile = "$wgUploadDirectory/ApteanPageBk$width.png";
                $bodyimagefile = "$wgUploadDirectory/ApteanPageBk$width.png";
```


```
                                $hfimagefile1 = "$wgUploadDirectory/ApteanHF1.png";
                                $hfimagefile2 = "$wgUploadDirectory/ApteanHF2.png";
                                $hfimagefile3 = "$wgUploadDirectory/ApteanHF.png";
        </code>
```

Resizes fonts and declares whether there is a title based on the start of the title of the page being produced, like above.

Added code to make DISPLAYTITLE work, although this is commented out in latest version.

Removed reference to HTMLDoc application path - in my tests, this didn't work properly, so removed and hardcoded.

Added code to remove borders around images - doesn't look great in PDF.

Added support for NEW_PAGE span tags - these are added through using NewPage template in pages.

Added attempt to remove smart quotes from produced test - doesn't work.

Added code to remove DocLink templates

Added version to the produced file served back tot he user, if the version is in the documents.

Changed lots about how the HTMLDoc command is built.

Added ability to keep produced files for debugging.

1.5 Support

1.5.1 Logging

As mentioned above, when PDF code is run, debug statements will be added to the page logs. These can be accessed from the database directly (see Assist Support Guide) and can also be accessed on the page itself (through the *Page logs* link in the *More* section of the toolbox.

You can extend the debug logging by editing the Assist system's LocalSettings.php or LocalSettingsAdditional.php file and setting extended debugging. This will ensure that the PDF production code saves any files it produces (logging the location), and logs the command used to perform the PDF conversion, which can then be moved and saved locally and tested to find the issue.

1.5.2 Zero Bytes PDF

Problems typically display as the PDF being downloaded is zero bytes or cannot be opened.

The most common issues are with single page PDFs. The normal solutions are:

- For full documents, ensure that there is a heading 1 element in the file being downloaded.
- If you are transcluding pages into a document, there is typically a limit of around 15 transcluded pages before the document will not produce. Instead, consider grouping the pages into a category and extracting that instead of creating a single document.

1.5.3 '?' characters

Occasionally, PDFs will be produced with '?' characters in unusual places. This represents and unprintable character in the PDF. Typically, this is because of smart quotes or hyphen characters, usually because text has been pasted in from a Word document that has "helpfully" automatically changes all hyphens, quotes and single quotes into extended characters. The PDF conversion routine does not support these characters at this time. To solve this, edit the page in question and replace the quotes and hyphens with plain equivalents.

1.5.4 Unexpected Pages in PDF Books

When PDF books are created from Categories, sometimes there are pages included that are not expected.

Most of the time this is caused by inheritance of categories. If a page has a category and this is transcluded into another page, the default is that the page transcluding will inherit the categories listed in the transcluded page.

Check the category list of pages in Assist - usually you will see the offending page, and then the categories can be amended to resolve the issue.

Either remove the categories of the page if it is in the wrong category, or ensure that the categories are surrounded by NOINCLUDE tags. You should use the source editor to do this, as the Visual Editor does not handle this kind of meta data very well.