

Anki-Pdf-Editor

Javafx + Spring

derMacon

October 10, 2019

- Introduction
- Installation / Requirements
- Usage
- Edit project properties
- Creating new cards
- Parsing user content
- Shortcuts - Programm specific
- Shortcuts - Vim in general

Introduction

Commandline tool to create anki flash cards via the vim editor. Once started the program will display a selected pdf document in which the user can navigate throughout vim itself. If a anki-card should contain a specific pdf page of the displayed document on either the front- or the backside of a note it can be passed in a simplified version where the pagenumber is written between tags.

All features can be used via shortcuts. For that the program opens a costum .vimrc.

Installation / Requirements

- Unix OS (due to filepaths)
- Vim
- Ankidroid 2.1 (or newer)
- Ankiconnect addon
- gnome-terminal (terminal emulation)

Usage

- To open run the program run the command 'java -jar *path/to/jar*'. Once run, spring will allocate the 8080 port on the machine and expose a rest api.
- The user will be confronted with the following menu:

```
silasUser@dimitri: ~/Documents/projects/c...  
File Edit View Search Terminal Help  
  
:: Spring Boot ::                (v2.1.9.RELEASE)  
  
Anki-Editor - version 1.0  
  
createDeckFile  
=====  
current project:  
*****  
* deck: TestDeck.anki *  
* pdf: manual.pdf *  
* page: 1 *  
*****  
=====
```

type

- a to write new cards
- e to edit project properties
- w to push to anki connect
- q to quit without pushing
- wq to push and exit

input: █

Edit project properties

- When selecting the project properties the user has the choice to either change the displayed pdf or the stack of cards to which the cards should be added

```
silasUser@dimitri: ~/Documents/projects/c...
File Edit View Search Terminal Help

Anki-Editor - version 1.0






createDeckFile
=====
current project:
*****
* deck: TestDeck.anki *
* pdf:  manual.pdf    *
* page: 1             *
*****
=====
type
- a to write new cards
- e to edit project properties
- w to push to anki connect
- q to quit without pushing
- wq to push and exit
- .....
input: e
=====
Type:
* 1: deck
* 2: pdf
- .....
input: 
```


Deck Selector

To: Send

Subject: Send

Open

Look In:     

 manual.pdf

File Name:

Files of Type: PDF

Open Cancel

Creating new cards

- a javafx gui will load containing a pdf viewer showing the selected pdf
- a second window containing a vim instance will open. This vim session is capable of communicating with the exposed rest api. By pressing either `z` (next page) or `shift + z` (previous page) the pdf viewer updates accordingly.
- to import / paste the displayed image as a png in the current card just press `p` and an image tag with the following syntax will be pasted into the document at the cursor position:
- if the user has overwritten this register the page is also accessible via the keykombination `,p`

Parsing user content

- to parse the user input to actual anki cards the user has to save his written content via `:wq` in the vim instance and type `alt + f4` to exit pdf viewer.
- in the top level menu either select `w` or `wq`
- in the background the java backend will generate the required json format for the ankidroid api (AnkiConnect)
- the json also contains html elements for the given images / linebreaks to ensure a readable anki card
- after pushing the changes to the anki rest api, the cards should be displayed in the appropriate deck in the anki desktop program itself.

Shortcuts - Programm specific

All program specific shortcuts in a nutshell:

- $z / \text{shift} + z$: turn next / previous page; Copy the current page tag to the default register (accessed via p)
- $, + c$: Append new card template to anki file
- $, + p$: Reload page tag, pastes the current page tag to cursor position
- $, + t / , + \text{shift} + t$: tab between fields

Shortcuts - Vim in general

version 1.1
April 1st, 06

vi / vim graphical cheat sheet

Esc
normal mode

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* next ident	(begin sentence) end sentence	"soft" bol down	+ next line
g goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= autoformat
Q ex mode	W next WORD	E end WORD	R replace mode	T back 'till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	}	end parag.
q record macro	w next word	e end word	r replace char	t 'till	y yank	u undo	i insert mode	o open below	p paste after	[misc]	misc
A append at eol	S subst line	D delete to eol	F "back" find ch	G eol/ goto ln	H screen top	J join lines	K help	L screen bottom	. ex cmd line	" reg. spec	bol/ goto col	
a append	s subst char	d delete	f find char	g extra cmds	h ←	j ↓	k ↑	l →	* repeat :t/T/t/F	' goto mk. bol	\ not used!	
Z quit	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un-indent	> indent	? find (rev.)			
Z extra cmds	X delete char	c change	v visual mode	b prev word	n next (find)	m set mark	reverse :t/T/t/F	repeat cmd	/ find			

motion	moves the cursor, or defines the range for an operator
command	direct action command, if red , it enters insert mode
operator	requires a motion afterwards, operates between cursor & destination
extra	special functions, requires extra input
q	commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line, mk = mark, yank = copy

words: **quux** **foo** **bar** **baz**
WORDS: **quux** **foo** **bar** **baz**

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:s/s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :mew (new file in vim),

Other important commands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

- use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z)* (e.g.: "ay\$ to copy rest of line to reg 'a')
- type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5l, d4j)
- duplicate operator to act on current line (dd = delete line, >> = indent line)
- ZZ to save & quit, ZQ to quit w/o saving
- zt: scroll cursor to top, zb: bottom, zz: center
- gg: top of file (vim only), gf: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio