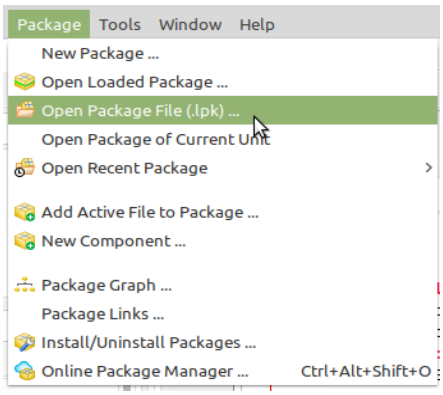
GitWizard

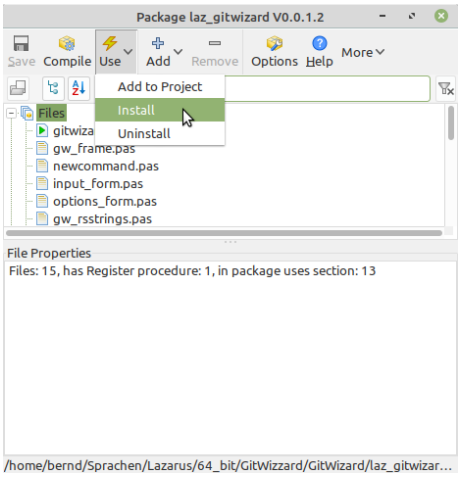
Description:

The GitWizard project is a kind of plugin for executing Git commands from the Lazarus IDE. The basic idea of the program is that the Lazarus user can create small script files with Git commands and then execute them with GitWizard.

Installation:

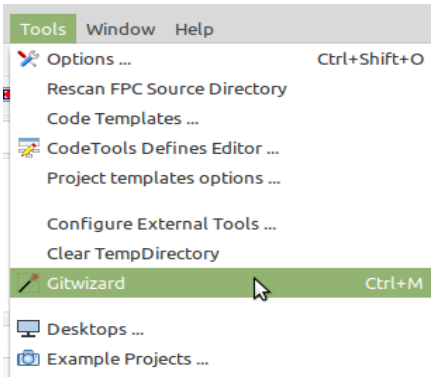
After GitWizard from here:
https://github.com/wennerer/Gitwizard.git
has been downloaded, navigate to laz_gitwizard.lpk in Lazarus and install the package.



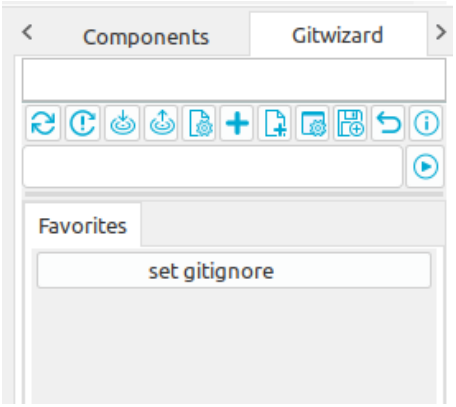


Settings

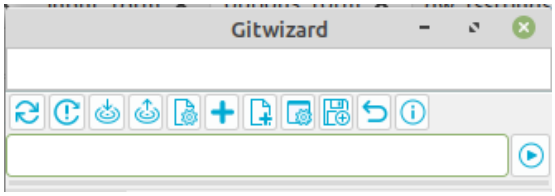
As soon as GitWizard has been successfully installed, there is a new entry in the Tools menu.

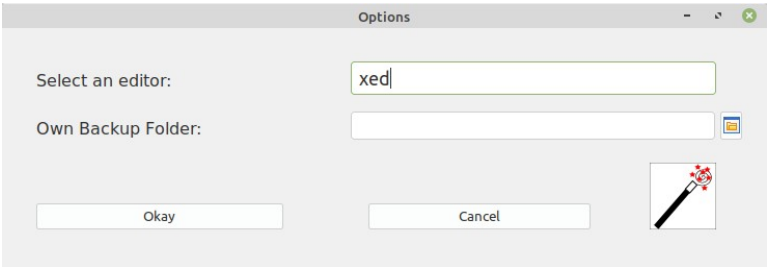


Click on this new entry to open the GitWizard.



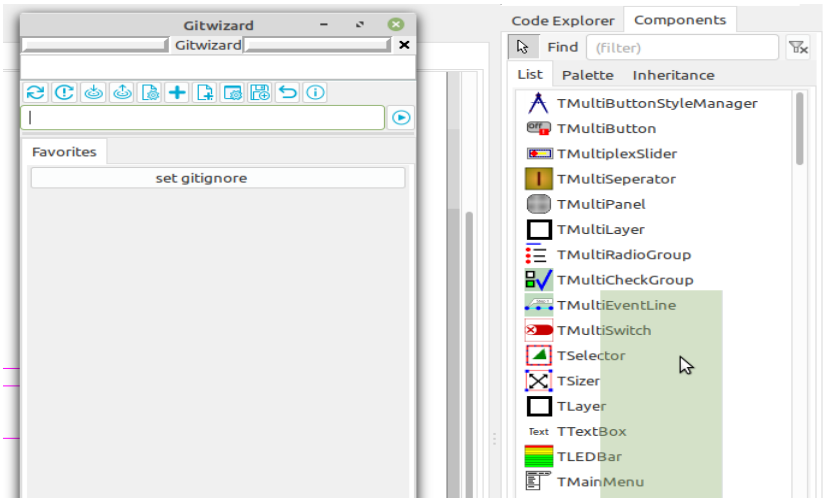
Before you start working with it, please click the"Options" button and enter an editor of your choice. This will then open the script files or gitignore.

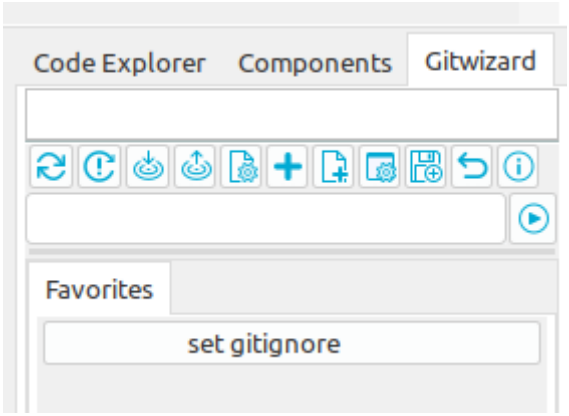




At Own Backup Folder you can enter the folder who saved your own commands. This folder is used as the initial directory in the backup dialogue.

If you want to dock GitWizard in the IDE, this works in the same way as with all other windows. Make the header visible and drag the window to the desired position.

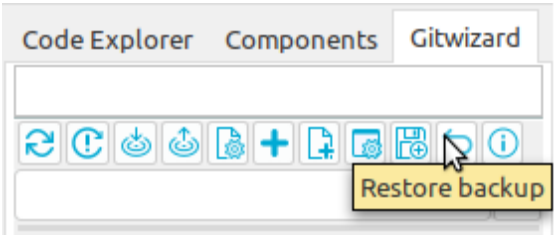




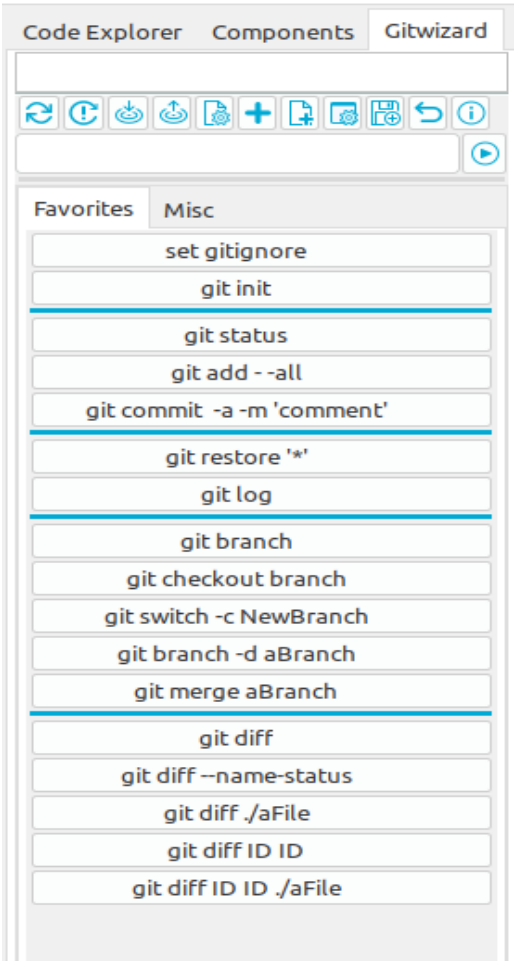
First steps

Use commands provided

If you want to use the provided commands to learn how to use GitWizard, you can use the "Restore backup" button.



Answer the first query with yes, then load the commands from the providedCommands/linuxCommands or providedCommands\winCommands directory.
There is now a selection of commands in the GitWizard.
ATTENTION: The previously selected editor will be overwritten with the editors xed or notepad used by me. So please change it again if necessary!!!



An additional tab has also been created on which also contains commands

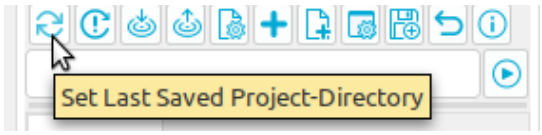
Tip: if you want to empty everything afterwards and create your own commands, you must delete the files gw_command.xml and gw_options.xml in the LazarusConfig folder. You should also empty the folder linuxCommands or winCommands.

Directory panel

The directory panel shows which directory is currently set as the Git project directory.

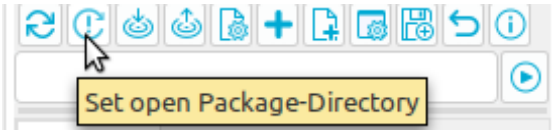
The toolbar buttons

The first button

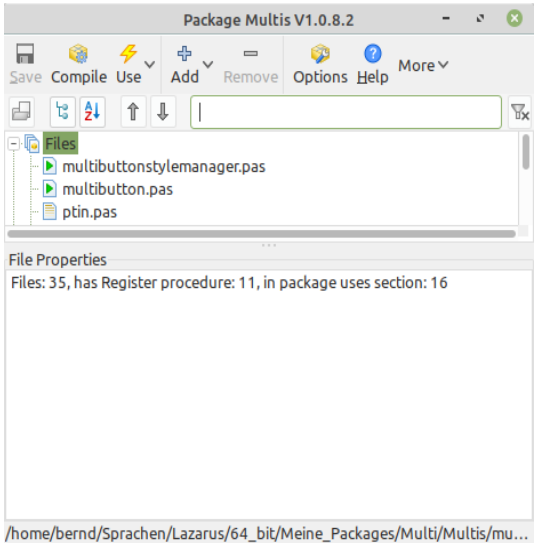


is used to set the last project saved in Lazarus as the Git project directory.

The second button



is used to set an open package as a git project directory.

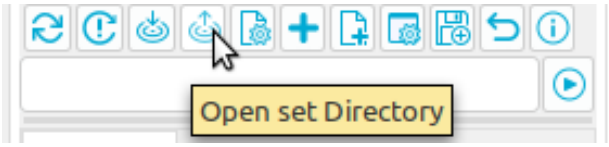


For this to work, a package window such as must be open. But beware, if several such windows are open, the command will fail.

The third button

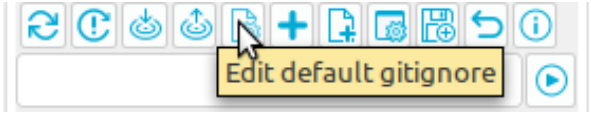
allows you to set any directory as the git project directory using a dialog.

With the fourth button



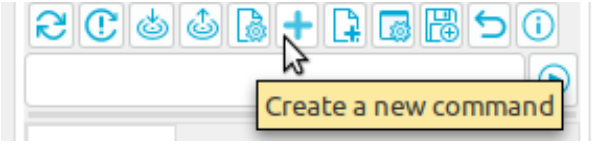
you can open the set git project directory in the standard explorer.

Fifth button



There is a standard Gitignore file in the GitWizard project directory, which can be copied to the git project directory using the first command button. This file can be opened for editing with the fifth toolbar button. Attention: an editor must be set under Options!

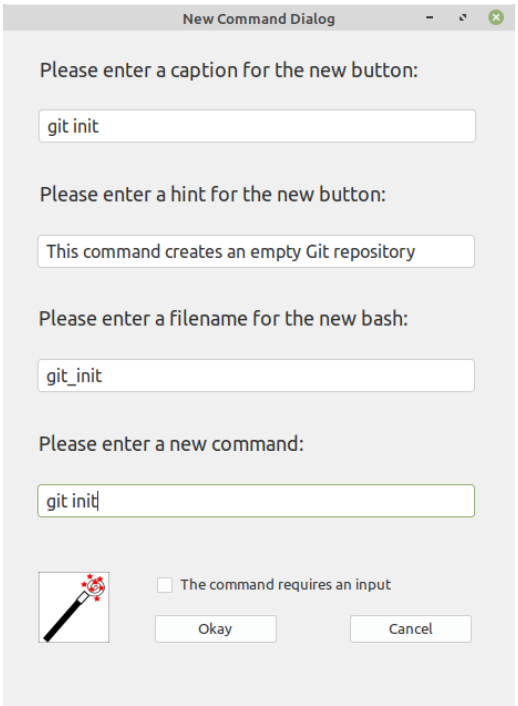
Sixth button



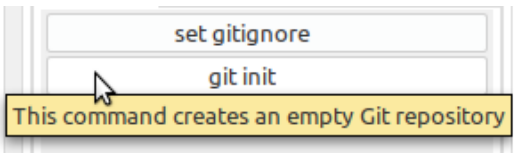
The sixth button can be used to create new commands. Here are two examples. One is a command for which no further input is required and the other is a command for which input is required for execution.

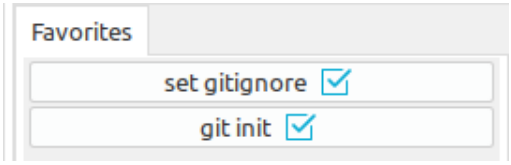
Create a simple command:

Pressing the sixth button (+) opens the dialog for creating a new command:



The git init command is created here, for example. After everything has been entered, confirm with OK. The result should look like this:

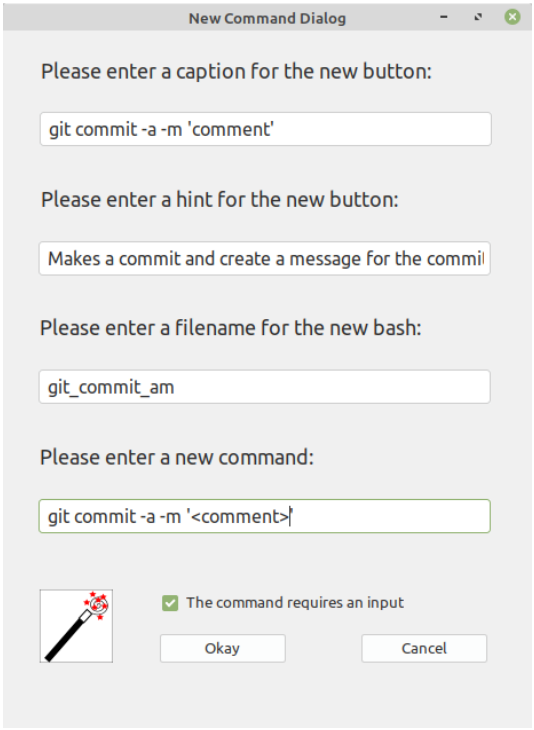




The set gitignore command button is always available and cannot be deleted. GitWizard recognizes whether in the set git project directory a .gitignore file and a .git file are available and indicates this with a blue tick.

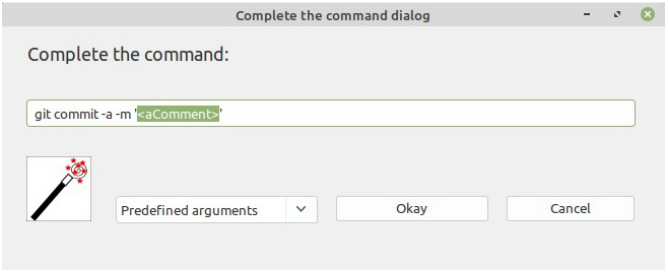
Create a command with input:

Pressing the sixth button (+) opens the dialog for creating a new command:



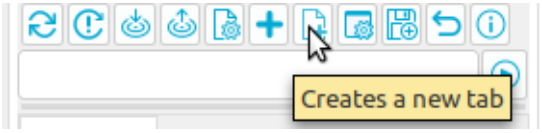
The command git commit -a -m 'comment' is created here. When executing the command, comment must be replaced with the desired comment. For this to be possible, the checkbox at "The command requires an input"must be set in the dialog.
Tip: if you set the part to be edited in < >, is selected when it is called up

It should then look like this:

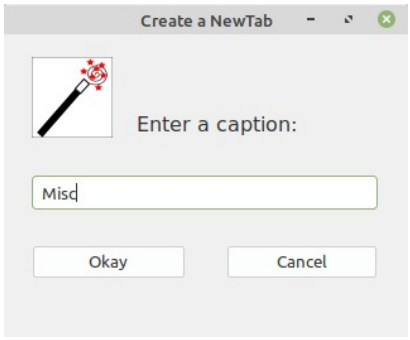


Tip: By double-clicking on Predefined arguments you can add new arguments.

With the seventh button

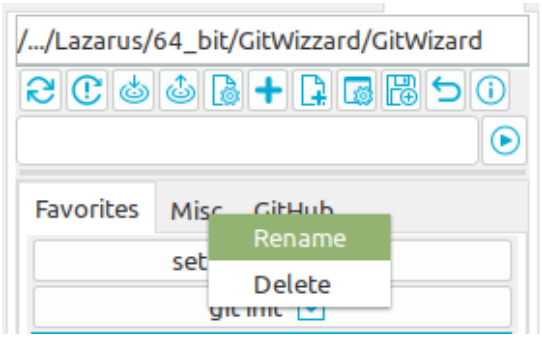


an additional tab is created. If you have created a series of commands after some time, it can be useful to distribute them across different tabs.



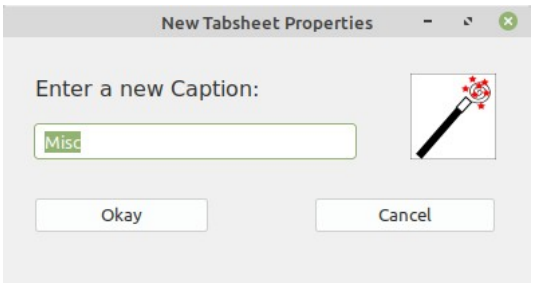
Edit Tab‘s

Tip: If you right-click on a tab, a pop-up menu appears:



Attention: the Favourites tab cannot be edit!

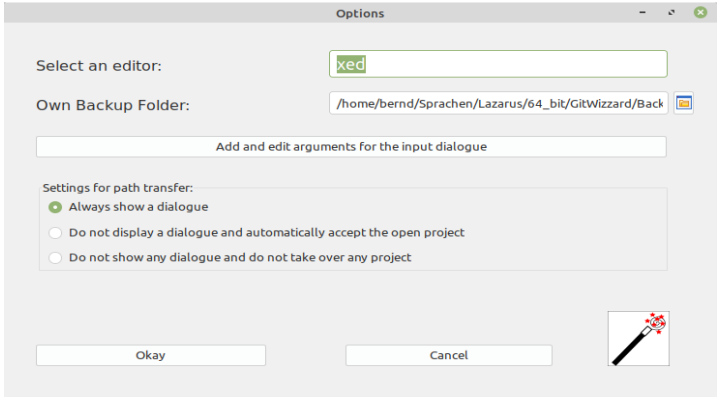
If you click on rename, you can enter a new caption in a dialogue:



A tab can only be deleted if there are no command buttons and/or separators on it.
All command buttons and/or separators must therefore be moved to another tab
or deleted.
If the tab is empty, simply click on Delete.

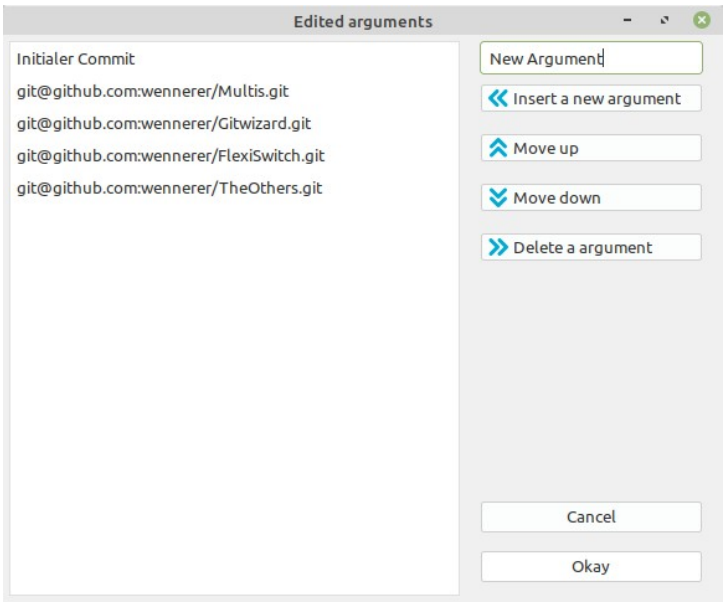
The eighth button

opens the options dialog. Here you only need to select an editor to open the script files or the gitignore file.



For example: xed, gedit, notpad etc.
If you have your own backup directory you can store the directory in which you save your commands would like. It is then called as Initial directory when opening the Backup dialogs used.

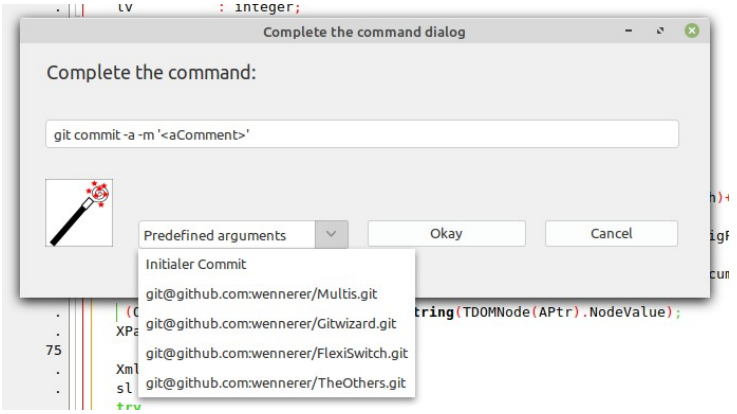
If you click on Add and edit arguments for the input dialogue, the following window opens the following window opens:



Arguments can be stored here which can then be selected in the command input window.
New arguments are entered in the edit field and added with Insert a new argument.

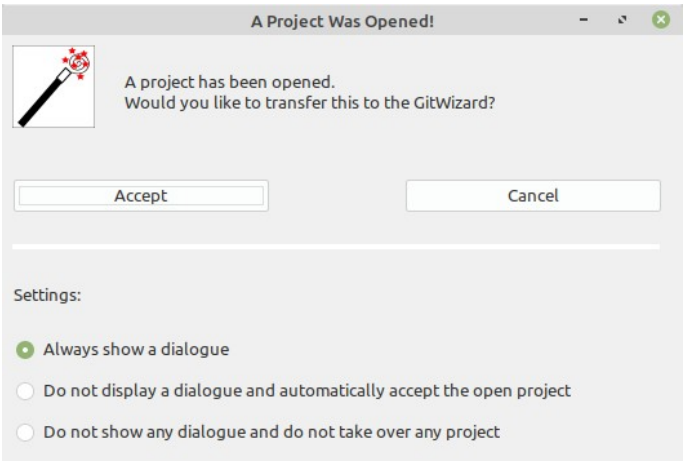
With Move up or Move down the sequence can be changed

Delete a argument removes it from the list.



Settings for path transfer

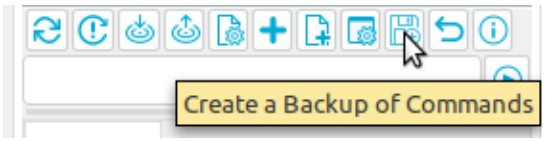
In the options dialogue, you have the option of setting rules for how the path of the project can be automatically transferred to the GitWizard when an existing project is opened. If you select with dialogue, a dialog appears as soon as an existing project is opened.



It is also possible to make a selection in this dialogue.

Important! If a new project is created, the path is not automatically adopted.

The ninth button

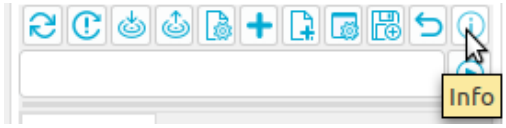


opens a directory dialog. The initial directory is set to ..../GitWizard/providedCommands/ linuxCommands or winCommands. The supplied commands are located in this directory. It is recommended that you select a separate folder for your own backup. This will prevent conflicts when updating GitWizard. Attention: If you make a backup of your own commands, the contents of the backup directory will first be deleted and then the new commands will be copied in!

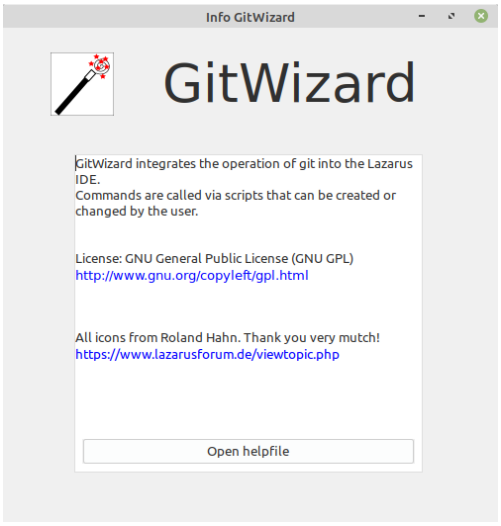
The tenth button

opens a directory dialog. The initial directory is ..../GitWizard/providedCommands/ linuxCommands or winCommands. The supplied commands are located in this directory. Attention: If you restore the last saved state, the content of the folder ..../GitWizard/ linuxCommands or winCommands is deleted first and then the backup files are copied into it.

The last button



opens a small info window.

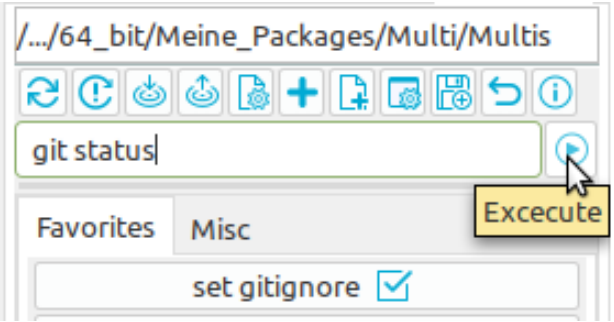


A click on the button opens the help-file

Many thanks to Roland Hahn for the images!
https://www.lazarusforum.de/viewtopic.php?f=1&t=14263&p=128092&hilit=hahn#p128092

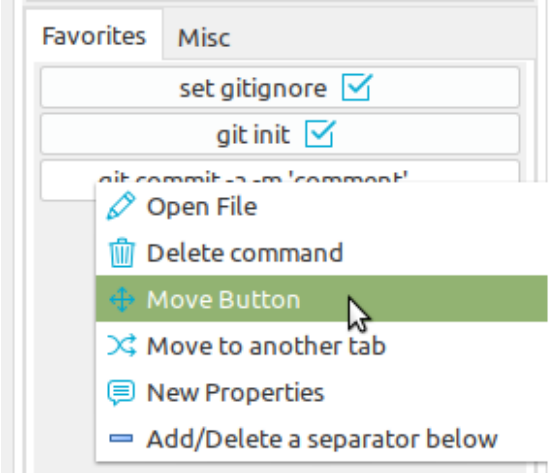
Executing individual commands

To execute (test) individual commands, there is an input line. Simply enter the desired command there (here git status) and either press Enter or use the Execute button.



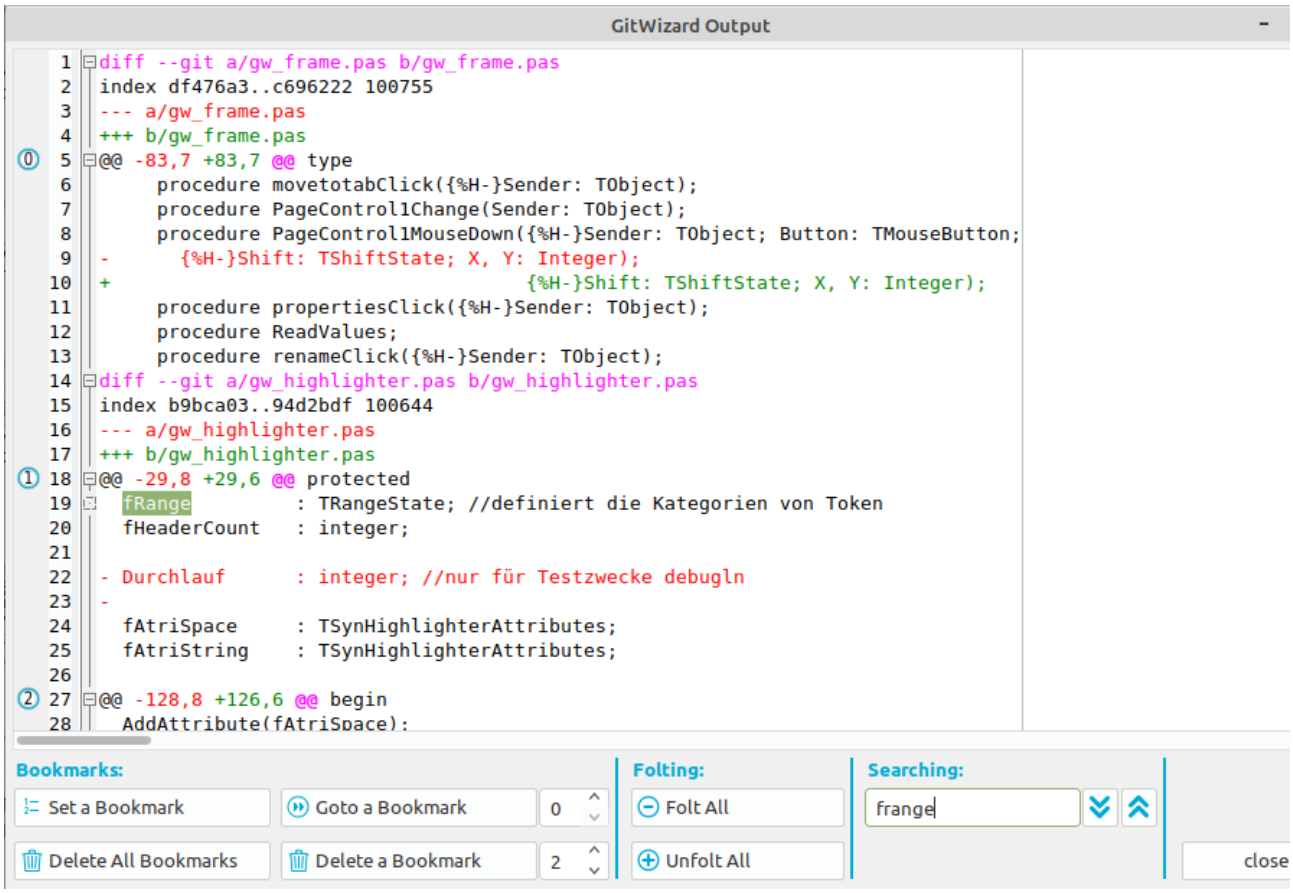
Edit command buttons

All command buttons have a pop-up menu that opens by right-clicking on the button.



Open file, opens the script file for editing.
Delete command, deletes the command from the gw_commands.xml and the script file from the command folder.
Move button, moves the button within the same tab.
Move to another tab, moves the command to another tab.
New properties, allows to change the caption and the hint.
Add/Delete a separator below, adds or removes a separator below the command button.

The Output Window



The highlighter and the folding options in the output window are „optimised“ for a git diff. It is possible to set up to 10 bookmarks and search for characters.