How to use filebrowser.py as a file open dialog.

Please see filebrowsertest.py (included in the source) as an example program.

filebrowser.py can be run from the command-line or as a python library.

Running from the Command-Line:

Please call “python filebrowser.py --daemon start &” at the beginning of your program and “python filebrowser.py --daemon stop” at the end. The “--daemon start” command will end when “--daemon stop” is called, so in order to prevent blocking your program the “--daemon start” command needs to be run in parallel. None of the other commands need to be run in parallel.

To open the file dialog, call “python filebrowser --daemon communicate --mode open”. The mode can be set to “open” to allow the user to select a currently existing file, “save” to select a location to put an output file to, or “normal” to open it as a file browser. “open” and “save” modes will return the selected locations separated by line breaks.

If the user selects a file on a remote filesystem, filebrowser.py will return a path in a temporary location. Please call “python filebrowser --daemon flush” any time your program makes a change to the filesystem. This will sync the temporary path with the remote path. All temporary paths will be removed when “--daemon stop” is called.

Arguments:

- “**--daemon {start|stop|communicate|flush}**”: 'start' starts the daemon. 'stop' stops the daemon. 'communicate' will launch the file browser/dialog. 'flush' syncs the local and remote filesystems.
- “**--multi false**” forces the user to only select one file.
- “**--workspacedir PATH**” opens up in the given path. It's recommended to put a path specific to your tool so the user has a place to put their files.
- “**--text TEXT**” displays text above the file list.

When in open mode:

- “**--openremote {temp|workspace|none}**” what to do if the user selects a remote file. 'temp' will download the file into a temporary directory. 'workspace' will save it to the path from --workspacedir. 'none' will tell the user they can't select that file.
- “**--filemode {auto|both|files|folders}**” 'files' only allows the user to select files to open in “open” mode. 'folders' only allows the user to select folders to open. 'both' allows the user to select either. 'auto' defaults to 'files'

when in save mode:

- “**--defaultfilename FILENAME**” is the default file name when using save mode. This shows up on the text box at the bottom of the dialog, and the user can change it.
- “**--saveremote {true|false}**” if set to false, prevents the user from selecting a destination on a remote filesystem.

Running as a python library:

assuming it is imported as “import filebrowser”:
There are four main functions:

- filebrowser.**do_open**(): Allows the user to select a currently existing file or files. Returns the files as a list of paths.
- filebrowser.**do_save**(): Allows the user to select a location to put an output file to. Returns the destinations as a list of paths.
- filebrowser.**do_browse**(): Opens the file browser.
- filebrowser.**flush**(): Syncs local and remote filesystem

If the user selects a file on a remote filesystem, filebrowser.py will return a path in a temporary location. Please call flush() any time your program makes a change to the filesystem. This will sync the temporary path with the remote path.

Arguments:

- **workspacedir**: opens up in the given path. It's recommended to put a path specific to your tool so the user has a place to put their files.
- **text**: displays text above the file list.
- Used with filebrowser.do_open():
    - **opentomode**: What to do when the user selects a file on a remote filesystem. OpenToModes.Temp downloads the file into a temporary directory and returns the temporary path. OpenToModes.Workspace downloads the file into the workspace directory (as given by the workspacedir argument). OpenToModes.None will tell the user they can't select that file.
    - **filemode**: FileModes.Files only allows the user to select files. FileModes.Folders only allows the user to select folders. FileModes.Both allows the user to select files or folders.
    - **multi**: if set to False, only allows the user to select one file.
- Used with filebrowser.do_save:
    - **defaultfilename**: The default filename. This shows up on the text box at the bottom of the dialog, and the user can change it.
    - **saveremote**: if set to False, doesn't allow the user to select a remote destination to save to.
    - **multi**: if set to True, the user can select multiple destinations.