Nelphi's Tool

Nelphindal

June 4, 2014

Contents

1 Introduction

This manual is like the toolbox itself under construction. Feel free to notify me if you find a bug or find a already written passage is unclear. On page 13 you can find some solutions for popular problems.

2 Prerequisites

To execute this toolbox you have any version of the java runtime environment, short form JRE, of at least version 7.x [1] installed. After custom installation you can execute this tool by double clicking on it (Launcher.jar).
The toolbox itself can be downloaded from my git-repository `https://github.com/Greonyral/someURL/Launcher.jar`. Currently supported operating system are Windows XP, Vista, 7 and 8. All distributions on current versions of Unix should be able to run the toolbox.

3 Overview

There are currently mainly 3 tools included into this toolbox:

1. AbcCreator: GUI for Bruzo's BruTE
2. SongbookUpdater: Fast update of the Songbook.pluginData for the Plugin by Chiran
3. VersionControl: A basic git client, used to distribute songs and keep them up-to-date

On every start the toolbox will ask you, which of these tools shall be executed. Every checked tool will be checked if available and up-to-date, and if not the newest version will be downloaded automatically. To start the main-tool you can either double-click on the Launcher.jar or invoke it manually by calling java.exe[2].

4 Features

What this toolbox can do:

1. Check for new version of this tool, download it and replace the currently used file by the new one
2. Create the local git-repository
3. Download changes from local git-repository
4. Upload commuted changes to local git-repository

[1]Download for example at `http://www.java.com`

[2]On windows you have to type the full path to the java runtime if it is not included in your PATH-variable. A valid command would be
'C:\ProgramFiles\java\jre7\java.exe -jar Launcher.jar'

5. Write for any known accounts the needed file for Songbook-Plugin by Chiran
6. A simple client to read and write messages
7. Reset the repository and delete created files by this tool. The repository band however will be kept.
8. Repair - delete created files by this tool including the band repository.
9. Modify song titles Disabled
10. Uniform songs - apply a name scheme to every selected song Disabled
11. Provide a GUI for the use of BruTE
12. Use SSH instead of HTTPS - normally only of interest on Unix

In the following each chapter will explain every feature. The basic settings can be changed in the config-file for the tool. It is named

.songbookUpdater

and can be found in the user home[3].

5 Getting new versions

There are basically two ways to get new versions. First one is to download it again with the same links like on page 4. Second way is to let the toolbox do this itself. The toolbox will contact the repository and download the new version of selected tools automatically.

6 VersionControl

Due to various songfiles we had, i wrote this tool originally to make it easier to maintain our songs. The original idea was to have a simple tool, hiding all unnecessary options of the used git-protocol. Because of this, all actions of this tool are reflecting options a git-client would offer, too. But however this tool has only a subset of the options of a full git-client. This allows you to use any git-client if you feel that this tool is not supporting the operation you want to do. But before this you should read at least one short introduction of git to get an idea what you are doing.

6.1 Basics of git - A short glossary

I will use some terminology of git. Therefore this will be a short introduction. Git is a version control. Normally it is used for programmers to distribute code, apply patches, build releases and roll back changes if necessary. Since source-code and the abc-files are more or less the same, i.e. text-files, it is a nice way to publish our songs, making it very simple to keep everyone up-to-date. So if you use git your actions will be (normally) always revocable. But enough of this let us start to get the basics done. So here are the basic concepts.

[3]%HOME% on Windows,   on Unix

6.1.1 Repository

A copy of a hierarchy within a directory with its files. To transfer data between two different repositories

```
git fetch
```

to get data from a repository to the local one and to transfer from it to remote one

```
git push
```

are used.

6.1.2 Commit

A commit is a snapshot of repository, also known as revision. Each commit has a short hash, commits with same hash are considered equal. Same files with same content can result in different hashes and therefore commits.

6.1.3 Tree

Series of commits are contained of one or more branches. The older commit is the parent, the younger one the child. It can be happen that there are multiple parents or children. In this case there are multiple trees of one branch. Git has no problems with multiple children. But in our case, the existence of multiple children will cause sooner or later the merging of both trees to one commit with multiple parents. This will be done every time contacting the remote repository. In case the program can not merge changes automatically you will get an alert and you will not be able to upload any commits until you fixed the merge-conflict. Conflicts can be avoided as long everyone is modifying only the own directory. After a successful merge both branches are equal, all following commits are contained in every previously merged tree.

6.1.4 Branch

But you can define also two different trees which you will never want to merge, or at least not for the moment. In this case every not merged tree has to be checked in a branch. Technically it is not more then a named pointer to a top of a tree. By default there is always a branch, called master. This tool uses two branches. The branch

> master

is containing the band directory and the branch

> bboard

the branch for the messages.

6.1.5 Head

The head is the newest commit of a tree.

6.1.6 State

Git knows several states of file depending on the difference to the latest commit of current branch. Some are acquired passively, other states can be reached only by invoking certain git commands. The first states are the passive ones.

New is every file not been in the repository before

Changed is every file with different content compared to last commit

Missing is every missing file compared to last commit

The states of commits are as follows.

Added is every new file to be committed

Modified is every changed file to be committed

Deleted is every file to be removed on next commit, or on foreign commit the file which has been removed.

6.2 Install git

There are various implementations of git. Search for

> git windows

will find some results. There is not the best solution but a nice overview over good GUIs for git is `http://git-scm.com/downloads/guis`. Get the one looking good for your needs, the included installers should set everything up to work. If you can not decide what you need, get Git Extensions[4], the chance is high that it will cover everything you will ever need.

6.3 Create local repository

By default the local repository is Music/band within the base path you selected. The tool will create the band directory if it does not exist and the option to

> Download songs

is set, initialize the repository and start the download. It can take several minutes depending on your downstream-speed.
You can do it manually by invoking

```
git clone https://github.com/Greonyral/lotro-songs.git
```

This needs a git installation, you can find a guide to set it up on page 7

[4]`http://sourceforge.net/projects/gitextensions/`

6.4 Download to local repository

This option is executed each time you select the VersionControl. The tool will get the head from the remote repository and will compare it with the one in the local repository. Any changes between the remote and local commit will be displayed if the option to show the changes is active. In any case the remote commit is not equal the local branch will be set to be equal to the remote branch. Local not yet uploaded commits will be based on the new head. An error message will be displayed if the attempt failed.

6.5 Upload from local repository

Uploading changes consists of two parts. First one is to create a snapshot, this is making a commit in terms of git. Second part is to upload this snapshot to a remote repository. Both parts will be done by enabling the option to

> Commit

for you. All you need is a log-in on the platform providing the remote repository[5]. The registration is for free. After this the owner of the repository (this is me) has to add your account to have write access. For this a simple mail with your github login is enough. The upload will be done by the tool-box after every successful commit. You can enable the option to commit, select no file and let the upload to be done in case the commit is successful but the upload fails.

6.6 Commit changes

This option requires the user-name and password entered. The tool will show you each song changed or missing and will ask you if you want to add it. This means it will be as it is now included into the commit to be made. You can abort this progress at any point, the added files will be remembered and included next time you finish a commit. To remove a uncommitted file you can see page 14 for details.

6.7 The Bulletin Board

On enabling this option the tool will check for new messages and display them. Only the newest per author will be displayed, if there are multiple messages of the same person. After this you may select if you want to write a message yourself.
You can have a look on the older messages again. All messages are stored within

> SongbookUpdater

located in

> PluginData

The messages are simple files without any ending but you can view them with any text editor.

[5]www.github.com/greonyral/lotro-songs

6.8 Reset

This will undo several actions of the local repository. All tracked files will be rolled back to last commit. Any files added, or deleted, for next commit will be unstaged. This option should unblock any failing download.

7 Repair

Deletes every file, created by this tool - besides created abc-files by BruTE. To prevent data-loss The deletion of the directory hosting the local repository has to be confirmed. The directory created for the bulletin board will be deleted.

8 GUI-Wrapper of BruTE

Of the included tools this one is the largest. It includes a complete runtime of Bruzo's BruTE and offers a - hopefully - easy to use GUI to adjust the settings.

8.1 Style

This option passes its value to the underlying BruTE-runtime. It influences the naming of the parts. For now valid styles are

`Rocks`

the default style and

`TSO`

the style used for Starlight-Orchestra.

8.2 Settings on start-up

Most of the settings offered by this tool are optional.

8.2.1 Test with Abc-Player

You can provide the path to a runtime of Abc-Player to test your created abc. For this select one of

```
 AbcPlayer.exe AbcPlayer.jar
```

The abc will start to play automatically hitting

`Test`

as soon it has been created.

8.2.2 Specify additional drum-maps

To import custom drum-maps all maps have to be placed in the same directory and fit into the name scheme. This is

```
DRUM ID SUFFIX
DRUM := drum
ID := [0-9][0-9]*
SUFFIX := .drummap.txt
```

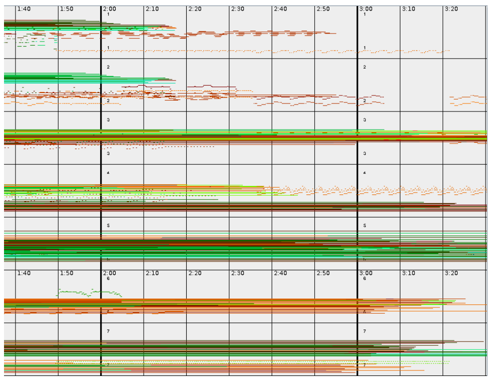

Figure 1: Tracks of a midi-file displayed

8.2.3 Specify custom midi to abc mapping

You can change the default mapping of the midi-instruments to their counterparts in the abc. The map representing the default mapping can be found in the repository next to the Launcher.jar.

8.3 Settings in the loop

After the initialization is done, the tool will enter a loop asking for the midi-file to transcribe. The created abc-file will created as the midi-file-ending replaced by abc. Hitting abort will abort the tool and terminate it, even if other tools have been selected for execution.

8.3.1 Visualization of the midi

As soon the tool read the selected midi-file a second window will appear, visualizing the parsed midi-file.

The colors are representing the volume as encoded in the midi-file. It goes from blue for very quiet over green and orange to red for very loud. The lines are showing that there is a note active played at this time.

8.3.2 Setting for this abc

The main window will display every non empty track to left side, every instrument of LoTRO to right, i will call them categories, and some instruments in the center.
The bottom-most category is special, it is indicating that the tracks linked with it shall not be played, or that the instrument of the midi-file was not mapped to a category by default. The instruments in center represent the later band setting. On pointing with your mouse on tracks every instrument playing this track will be marked yellow, instruments and tracks playing the same instrument type will be marked orange. The color scheme is same while pointing on the instruments and categories.

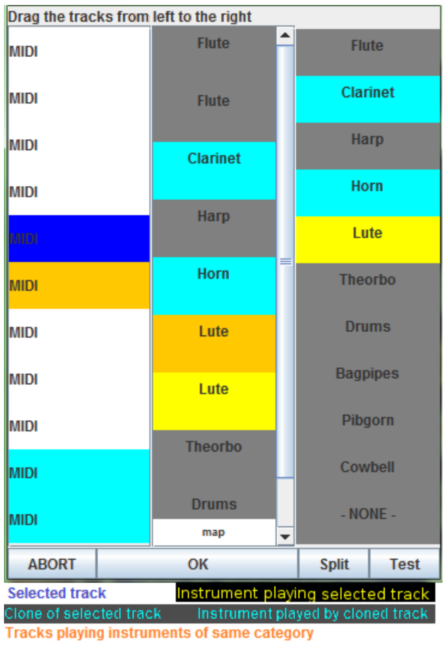

Figure 2: Selected track with existing clone

8.3.3 Set instrument to play a midi track

It will be the mainly setting you will change. You can create new instruments by dragging the midi-track to the category you want a instrument from. Empty instruments will be removed. You can track multiple tracks to same instrument if you want to have a instrument to play multiple tracks.
To have a midi-track played by more than one instrument of same category, the button

```
Split
```

has to be pushed down. No association to an instrument will be deleted as long the bottom is pushed down and all instruments belong to same category.
In case you want one midi-track played by multiple instruments of different categories, you have to create a clone of the midi-track for each different category. Right click on the track to clone will create a new track to link to another category than the original track. Linking the clone to the empty category will remove it again.

8.3.4 Volume

Left click on the track will display the volume slider for each associated instrument. Another left click will hide the volume-slider(s) again. You can not change the volume of tracks not being mapped to any instrument.

8.3.5 Change the drum-map

Every instrument of the type drum is showing a button. Clicking on it will show you a dialogue to change. The default map 0 will be used, as long no drum-map is highlighted.

8.3.6 Example

The tool created with the default mapping a lute for a part. In the end you want to have two flutes playing it. On of them is already there, the other one is missing. The following two steps may be swapped. First you drag the track to the already existing flute. The now empty lute will be removed from the center and on hovering with the mouse over the flute or track will show you that the track is played now by the flute. On testing you notice that this track is to fast for only one flute and it would be better to have only every second note be played by this flute and the other half of notes to be played on the second flute. So you enable the split mode and drag the track to the flute of the category side. A new flute will appear in the center and on hovering over the track will mark now both flutes.

9 SSH

SSH is a protocol for cryptographically secured internet connections (instead of HTTPS). It allows login at github without password and user-name. Since the support is not that well on Windows this options should be used only on Unix systems. You can specify your own URL, if you do not want to use the default URL for SSH. The tool will look in the config[6] and replace the key URL by the value of the key SSH if there is any specified. Both keys have to be in the section

> remote ”origin”

[6].git/config

10 Troubleshooting

10.1 You found a bug

Send me a report, please. Try to describe it as good as you can to make it easy for me to find it. I will try to fix it, if you did it well enough. Mail it to greonyral@arcor.de.

10.2 The tool is always asking to add one or more files, which you do not want to add

Open the directory containing the local repository. You are in correct directoy if you can see a directory called

```
.git
```

. If you can find a file called

```
.gitignore
```

, open it with any text editor and append a line with the file(s) you want to ignore in future. The paths have to be relative to the directory you are in.
In case the file does not exist it is getting a bit harder since Windows normally do not know files with no file ending and a leading dot. Create a new file, write the file(s) with their relative path(s) into it, one file per line and safe it to make it easier for you on the Desktop for example 'ignore.txt'. Now open the command line, navigate to the Desktop with

```
cd Desktop
```

and enter

```
move ignore.txt .gitignore
```

This will rename the 'ignore.txt' to '.gitignore'. Only thing to do now is to cut-and-paste it into the directory.

10.3 You get an error on merging

It should never happen, but if it happened to you try to reset the local repository into a valid state. For more information see the paragraph about how to reset9.

10.4 You want to create a shadow copy of your repository

You can tell the tool that the repository somewhere else by changing the base of the path. It is saved in your home-directory called

10.5 You uploaded a file you do not share under any circumstances

In this case you have to wipe your commit. Its basically descriped in the next paragraph. To prevent uploading again you can put this file in to your ignore-list. For more information on how to do this see page13

10.6 You uploaded a subset of changes by accident

No panic. Even deleted songs are normally not lost. A benefit of git is to save all previous commited states of your repository. You can try to fix it yourself on your own risk or ask me for help to recover it. This operation is a

```
git reset
```

command. There are easy how-tos in the web.

10.7 Purge a commit

This operation is rewriting the history and should be never done. But there might be reasons to do so. If you do it alone, inform EVERYONE what happened, that we have a change to wipe our history if we had the contaminated commit.