

Beagleplayer

A lightweight, minimalist audio/video/media player and browser, utilizing QMPWidget. Beagleplayer is ideal, for quick, easy, browsing of your media library.

Written in Qt5 using c++, sqlite, qmpwidget, opengl, mplayer



Supported file types:

Audio: mp3, flac, wav

Video: avi, mp4, mkv

Dependencies:

```
$ sudo apt-get install mplayer libqt4-dev libqt4-opengl libqt4-opengl-dev libqt4-network libqt4-sql  
libqt4-sql-sqlite gcc g++ make qt4-qmake
```

Cache:

~/.cache/beagleplayer/

Classes:

beaglemain - Main Controller for GUI
fileObj - Primary memory object, for temporary storage of songs, songdirs, video, videodirs
cache - Sqlite query library for beagleplayer
browse - audio/video file browser widget
controls - audio/video track options/status widget
localsync - library for reading/writing to cache all media files+folders
volume - widget for track controls, handles volume, volume display
qmpwidget – library for utilizing mplayer, using opengl

beaglemain

beaglemain(QWidget *parent = 0)
Main Central Control for cache, GUI , Signals

beaglemain::addWidgets()
Initialize and add, all widgets to the GUI, in their correct layout position.

beaglemain::initCache()
Initialize the main cache once, read from the cache's text file, to determine if the database exists or not, if not create one. Do this once, then send the address of the cache object to each widget that needs to make queries.

beaglemain::connectSignals()
Connect all signals and slots from all widgets

cache

cache()
read cache location from text file(static), open it, Read/write/update/delete objects/rows, from sqlite
cache(dynamic)

table:

key | parid | albid | filename | filepath

The same column configuration exists for songs, song directories(songdirs), videos, video directories(viddirs).

cache::init()
Check if cache text file exists, if so read from it. If not, create a new database, fill the tables, create a new text file with the location of the db within it.

cache::openDB()
Open sqlite database, using the path read from cache textfile

`cache::closeDB()`

Close any open sqlite database.

`cache::addTables()`

Add the initial beagleplayer tables for songs, songdirs, videos, viddirs

`cache::initDBLocation()`

Create the initial cache text file, which holds the full path to the db cache's dynamic location.

`cache::readDBLocation()`

Read the initial cache text file, which holds the full path to the db cache's dynamic location.

localsync

`localsync()`

Constructs object to iterate through paths and songs, storing each within their respective sqlite table

`localsync::scanFiles(int scantype)`

Scans files within a directory recursively, storing each filename and path within a fileObj

`localsync::scanDir(QString dir, int scanType)`

Scans files within a directory recursively, storing each filename and path within a fileObj

`localsync::Sync(QDir usrDir, int syncType)`

Control for localsync, determines which type of files we're syncing(syncType), audio/video

browse

`browse(QWidget *parent = 0)`

Constructs widget to display files and paths, in a splitscreen, adjustable, window.

`browse::Sync(int mode)`

Open a dialog window to allow the user to select an import directory. The mode is = 1, 2, 3.

1: Sync artists, songs, video paths, videos

2: Open a dialog for importing audio

3: Open a dialog for importing video

`browse::initCache(cache *ini_cache)`

Set the initial cache object, to the address of the initial cache object we created and read from in beaglemain

signals:

`curListChanged(fileObj &filelist, int *itemList)` - The entire mode/list changed
`plItemChanged(string plName , string plPath, int plID, int plPar)` – A playlist item changed index
`selectionChanged(int)` - A track was single clicked, int is the index of the title
`FullSelection(int)` - A track was double clicked, int is the index of the title
`MenuSelection(int)` - a Menu item was selected

slots:

`updateTitle(int)` – Right ViewList(track/title list) has changed
`updateMenu()` - Left ViewList(path/artist list) has changed
`updateMode()` - Mode changed audio/video/playlist/radio

controls

`controls()`

Constructs object to control and display current track options and status. i.e. stop/play/fwd/rwd/etc

`controls::startLocal(char *finSong, char *finPath)`

`controls::startSelected(int selection)`

slots:

`setVol(int vol)`
`setSelection(int selection)`
`setSelectionAndPlay(int selection)`
`setCurList(fileObj &newlist, int * newIDlist)`
`changeCon(int mode)`

fileObj

`fileObj()`

Constructs a blank file Object, containing: fileName, filePath, fileID, filePar, arrays. In addition to a size.

`fileObj(fileObj &src)`

Copy constructor, duplicates and initializes arrays of object

`initFile(int initSize)`

initializes the arrays of object to initSize

`REinitFile(int oldsize, int newsize)`

reinitialize the array, scaling it for any new size, duplicating previous arrays

`display()`

Output debug entire object, all arrays.