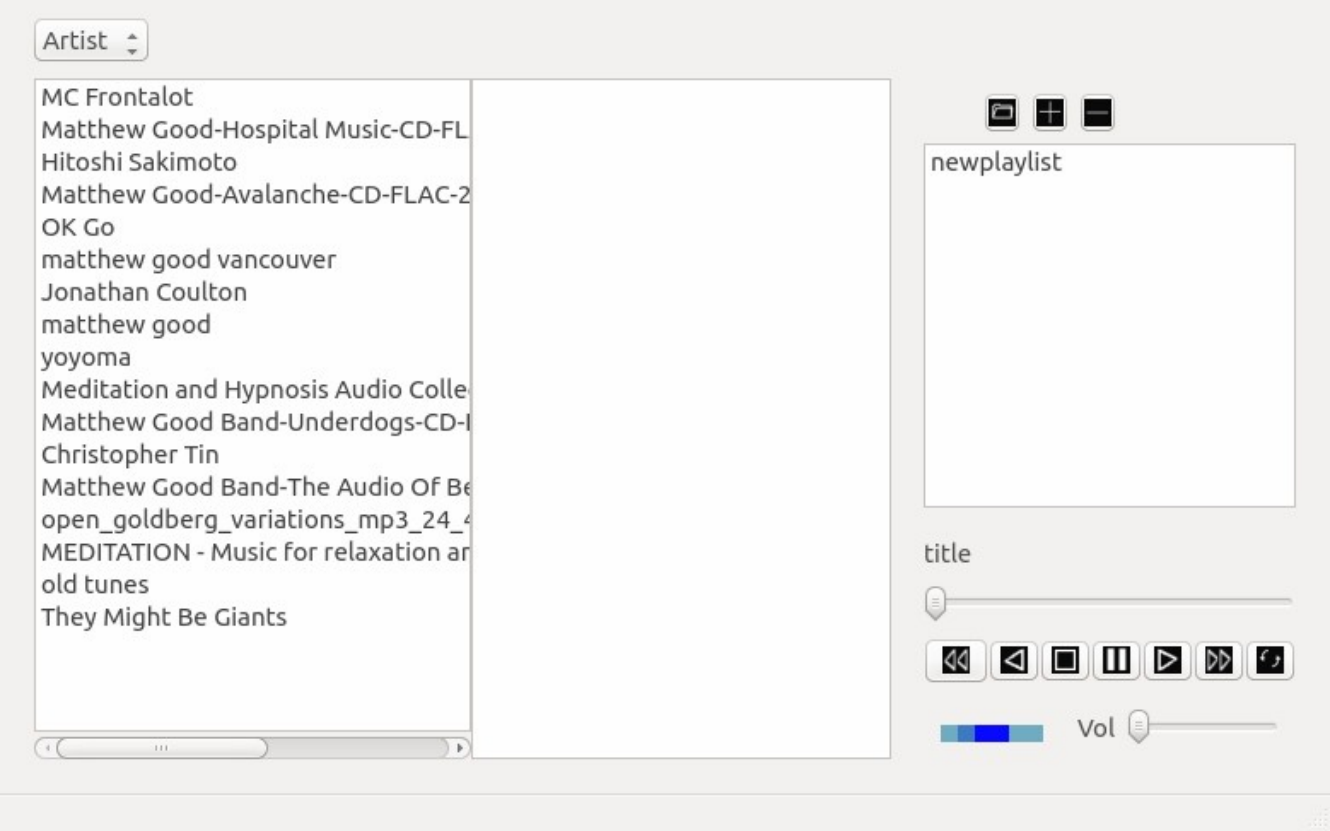
Mar 11, 2015
CheckoutCrypto Canada

Beagleplayer

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

Written in Qt5 using c++, sqlite, html5, javascript



Supported file types:

Audio: mp3, flac, wav
Video: avi, mp4, mkv

Dependencies:

# Recommended with Qt5
$ sudo apt-get install qt5-default libqt5webkit5 libqt5webkit5-dev

# optionally with Qt4
$ sudo apt-get install libqt4-dev libqt4-network libqt4-sql libqt4-sql-sqlite gcc g++ make qt4-qmake libqt4-webkit

Mar 11, 2015
CheckoutCrypto Canada

Cache
~/.cache/beagleplayer2/beagle.db

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
detached – inherits html5applicationviewer to open a webkit window, controls all signals to/from
appearance – widget for browsing and selecting detached player theme

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 inital 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

Mar 11, 2015
CheckoutCrypto Canada

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.

Detached

Important Slots:

```
/* Remote Commands to signal player controls from javascript
 * connect = 1  - play
 * connect = 2 - pause
 * connect = 3 - stop
 * connect = 4 - next
 * connect = 5 - prev
 */
remoteCmd(int cmd)
emits: remConFile(cmd);

remoteVol(int vol)
Set: volume = vol;
emits: remConVol(vol);

remoteSeek(int pos)
Set: min = pos;
emits: remConSeek(pos);

remoteRange(int max)
emits: remConRange(max);

remotePage(string page) // change the window url

remoteScreen(bool type)  // change the window size: true = fullscreen
Set: screenMode = type;
emits: remScreenToggle(type);
```

Object Setters – Set these from base controls

```
setTrack(string track, string path, int mode){
Set: songChange = true;
Set: trackName = track;
Set: trackPath = path;
Set: mode = mode;

setSeekPos(int pos)
Set: min = pos;

setVolume(int vol)
Set: volume = vol;

setRange(int end)
Set: max = end;

setState(int mediaState)
Set: state = mediaState;

setScreenMode(bool fullscreen)
 Set: screenMode = fullscreen;
```

Object Getters Get these from javascript or controls

```
QString getPath()
return trackPath;

QString getTrack()
return trackName;

int getMode()
return: mode;

bool getPlaylistMove()
Set: songChange = false;  // if SongChange was previously true
return true/false;

int getVolume()
return: volume;

int getRange()
return: max;

int getSeekPos()
return min;

int getState()
return state;

bool getScreenMode()
return:screenMode;
```

Important signals

```
remConRange(int);
purpose:  inform controls widget that our detached player opened a new media
  file and detected a new range;

remConSeek(int);
purpose:  inform controls widget that our detached player's timer was seeked

remConFile(int);
purpose:  inform controls widget that our detached player closed, so we can
  open it again

remConVol(int);
purpose:  inform controls widget that our detached player made a player control
  command such as: play, pause, prev, next

remScreenToggle(bool);
purpose:  inform controls widget that our detached player window size has
  changed to fullscreen/normal

detachClose();
purpose:  inform controls widget that our detached player closed, so we can
  open it again
```