QtMobilityYouPlayer

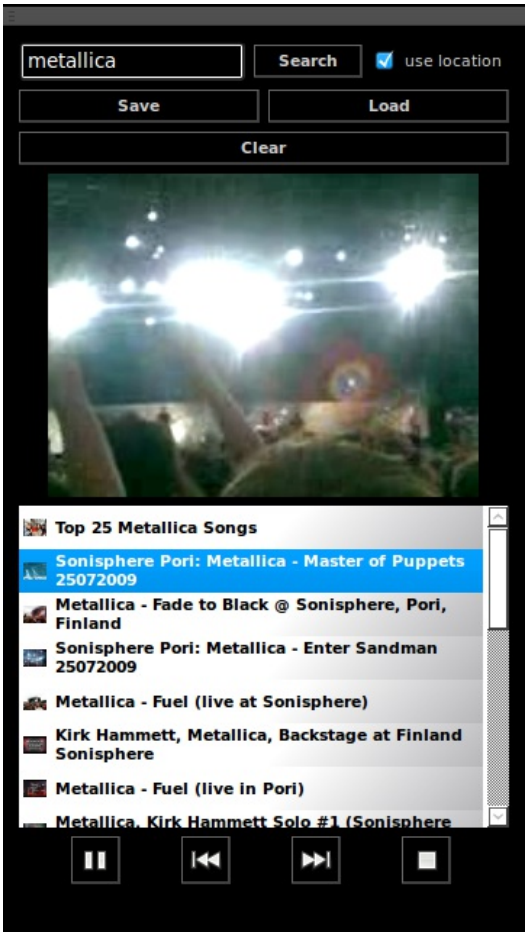
YouPlayer - QtMobility Api application

15 Aug 2010

Introduction

YouPlayer is small mediaplayer written in Qt. It serves a single purpose: use YouTube API to search, fetch music and stream media directly from YouTube. Similar idea like spotify client but use YouTube as a mediasource. It also includes interface which to extend to add other media source which to query playlists. Application uses three [QtMobility] API: multimedia, location and bearermanagement. When using location api you can limit your search results containing those near you (radius is used). Bearer management api is used to help connecting to the Internet with default connection.

Application has been developed using Ubuntu Linux using and QtCreator 1.3. Also some quick emulator test with Symbian 5th Edition SDK and Carbide.



How to use it

- Enter a search string and press search. If you want to limit search to a location near you check the "use location" checkbox.
- Save button enables one to save the playlist for a later purpose (to /home/username/youplayer_list.m3u)
- Load button loads previously saved list from filesystem.
- Clear buttons clears/empties playlist.
- To listen/watch playlist entries use playlist or use control buttons at the bottom.

The Code

Main object creation happens in youplayer.cpp's constructor. It is mainly shown below

```
// first some important includes
#include "qmediaplayer.h"
#include "qmediaplaylist.h"
#include "qgeopositioninfo.h"
#include "qgeopositioninfosource.h"
#include "youvideowidget.h"
```


(C) Copyright Nokia 2013. All rights reserved.

```
#include "gdataapi.h"

YouPlayer::YouPlayer(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::YouPlayer),
    locationsource(NULL),
    m_location("")
{
    ui->setupUi(this);
    // first create a QMediaPlayer that takes care of the media playing
    m_player = new QMediaPlayer(this);
    // also we want to use playlist with mediaplayer
    m_playlist = new QMediaPlaylist(this);
    m_playlist->setMediaObject(m_player);
    // class for using Google data api (gdata)
    m_mediaFeedApi = new GDataAPI(this);
     // we also pass a pointer to playlist so that we can append mediacontent into the list.
    m_mediaFeedApi->setQMediaPlayList(m_playlist);

    // create item model to store info about current media and its icon
    m_playlistModel = new YouPlaylistModel(this);
    // pass on QMediaPlaylist for model.
    // model is responsible for manipulating data from playlist
    m_playlistModel->setPlaylist(m_playlist);

    // we use own model with playlist data
    ui->playtableView->setModel(m_playlistModel);

    //videowidget for displaying video
    videoWidget = new YouVideoWidget(this);
    videoWidget->setMediaObject(m_player);
    ui->vidgetLayout->addWidget(videoWidget);
    // controls at the bottom
    m_controls = new YouPlayerControls(this);
    m_controls->setState(m_player->state());
    ui->controlLayout->addWidget(m_controls);

    // if we use GPS location in our queries
    locationsource = QGeoPositionInfoSource::createDefaultSource(this);
    // contect all signals
    connectSignals();
}
```

We use QAbstractTableModel derived class as a model for TableView. QMediaPlaylist provides nice methods for manipulating the models data via signals/slots

```
void YouPlaylistModel::setPlaylist(QMediaPlaylist *playlist)
{
    if (m_playlist) {
        disconnect(m_playlist, SIGNAL(mediaAboutToBeInserted(int,int)), this, SLOT(beginInsertItems(int,int)));
        disconnect(m_playlist, SIGNAL(mediaInserted(int,int)), this, SLOT(endInsertItems()));
        disconnect(m_playlist, SIGNAL(mediaAboutToBeRemoved(int,int)), this, SLOT(beginRemoveItems(int,int)));
        disconnect(m_playlist, SIGNAL(mediaRemoved(int,int)), this, SLOT(endRemoveItems()));
        disconnect(m_playlist, SIGNAL(mediaChanged(int,int)), this, SLOT(changeItems(int,int)));
    }

    m_playlist = playlist;

    if (m_playlist) {
        connect(m_playlist, SIGNAL(mediaAboutToBeInserted(int,int)), this, SLOT(beginInsertItems(int,int)));
        connect(m_playlist, SIGNAL(mediaInserted(int,int)), this, SLOT(endInsertItems()));
        connect(m_playlist, SIGNAL(mediaAboutToBeRemoved(int,int)), this, SLOT(beginRemoveItems(int,int)));
        connect(m_playlist, SIGNAL(mediaRemoved(int,int)), this, SLOT(endRemoveItems()));
        connect(m_playlist, SIGNAL(mediaChanged(int,int)), this, SLOT(changeItems(int,int)));
    }


    reset();
```

How use Location api to get GPS coordinates

```
if (locationsource) {
        connect(locationsource, SIGNAL(positionUpdated(QGeoPositionInfo)),
                this, SLOT(positionUpdated(QGeoPositionInfo)));
        locationsource->setUpdateInterval(120*1000); //120 sec
        locationsource->startUpdates();
    }
void YouPlayer::positionUpdated(const QGeoPositionInfo &info)
{

    if (info.isValid()) {
        m_location.clear();
        m_location = QString::number(info.coordinate().latitude()) + "," + QString::number(info.coordinate().longitude());
        qDebug() << "Position updated:" << m_location;
    }

}
```

How to setup network to use bearer management API and use default network configuration

```
void MediaFeedApi::createBearerSession()
{
    QNetworkConfigurationManager manager;
    cfg = manager.defaultConfiguration();
    session = new QNetworkSession(cfg,this);
    if (session)
        session->open();
}
void MediaFeedApi::stopBearerSession()
{
    if (session)
        session->close();
}
```


(C) Copyright Nokia 2013. All rights reserved.

How to downland a picture from web

```
void MediaFeedApi::downloadPicture(const QUrl &mediaUrl, const QString &location)
{
    QNetworkAccessManager *manager = new QNetworkAccessManager(this);

    connect(manager, SIGNAL(finished(QNetworkReply*)),
            this, SLOT(downloadFinished(QNetworkReply*)));

    QNetworkRequest request;
    request.setUrl(location);
    // we need to store this somewhere.
    m_downloadMap.insert(location,mediaUrl);
    manager->get(request);

}
```

We connected finished signal here, so we parse image here

```
void MediaFeedApi::downloadFinished(QNetworkReply* reply)
{
    const QByteArray &data = reply->readAll();
    const QImage &image = QImage::fromData(data);
    const QUrl &mediaUrl = m_downloadMap.value(reply->url());
    // we have loaded image, inform about it
    emit iconLoaded(mediaUrl, image);

    m_downloadMap.remove(reply->url());
    reply->deleteLater();
    reply->manager()->deleteLater();
}
```

We construct QUrl like this so that the encoding is done right

```
QUrl GDataAPI::construcURL()
{
    QUrl url;
    url.setPort(80);
    url.setScheme("http");
    url.setHost("gdata.youtube.com");
    url.setPath("feeds/mobile/videos");
    // add music category to the query, we want to search only music
    url.addQueryItem("category","Music");
    // use json format
    url.addQueryItem("alt", "json");
    // different formats
    url.addQueryItem("format", "1,6");
    // max results
    url.addQueryItem("max-results","20");

    return url;
}
```

Important part of the code is parsing of gdata api results (gdataapi.cpp)

```
void GDataAPI::parseAsJson(QByteArray &data)
{

    QJson::Parser parser;
    bool ok;
    QVariantMap result = parser.parse(data, &ok).toMap();
    if (!ok) {
        qWarning("An error occurred during parsing GoogleData");

    }

    QVariantMap feeds = result["feed"].toMap();
    QVariantList entries = feeds["entry"].toList();

    //loop through the different entries(actual media)
    foreach (QVariant entry, entries ) {
        // store entries to mediaresourcelist
        QMediaResourceList resourceList;
        QVariantMap entryData = entry.toMap();
        QVariantMap mediaGroup = entryData["media$group"].toMap();

        // check media url and add it to the resource
        foreach (QVariant mediaContent,  mediaGroup["media$content"].toList() ) {
            QVariantMap mediaContentMap = mediaContent.toMap();
            int format =  mediaContentMap["yt$format"].toInt();
            if (format == MEDIA_FORMAT) {
                const QUrl url = QUrl(mediaContentMap["url"].toString());
                QString mimeType = mediaContentMap["type"].toString();
                QMediaResource resource(url,mimeType);
                resourceList.append(resource);
            }
        }
        // download url for the pictures
        QString thumbNailSmallUrl;
        QString thumbNailLargeUrl;

        foreach (QVariant thumbNails,  mediaGroup["media$thumbnail"].toList() ) {
            if (thumbNailSmallUrl.isEmpty() || thumbNailLargeUrl.isNull()) {
                QVariantMap thumbNailsMap = thumbNails.toMap();
                // is this a good value for height?
                int height = thumbNailsMap["height"].toInt();
                if (height > 200)
                    thumbNailLargeUrl = thumbNailsMap["url"].toString();
                else
                    thumbNailSmallUrl = thumbNailsMap["url"].toString();
            }
        }
```


(C) Copyright Nokia 2013. All rights reserved.

```
        QVariantMap title = entryData["title"].toMap();
        QVariantMap content = entryData["content"].toMap();

        QMediaContent media(resourceList);

        // store entries to own data object
        YouPlayerMediaContent mediaData;
        mediaData.setPoster(thumbNailLargeUrl);
        mediaData.setContentText(content["$t"].toString());
        mediaData.setContentTitle(title["$t"].toString());
        mediaData.setThumbnailLarge(thumbNailLargeUrl);
        mediaData.setThumbnailSmall(thumbNailSmallUrl);
        // add this so we can compare them on model
        mediaData.setMediaUrl(media.canonicalUrl());

        emit mediaAdded(mediaData);

        // download pictures in background (uses QueueConnection)
        emit loadIcon(media.canonicalUrl(),thumbNailSmallUrl);

        // give ready media to use
        m_playlist->addMedia(media);

    }
}
```

When all data is ready we can return it from the model to the view

```
QVariant YouPlaylistModel::data(const QModelIndex &index, int role) const
{
    QVariant value = m_data[index];
    if (!value.isNull())
        return QVariant();

    if (index.isValid() && role == Qt::DisplayRole) {

        QString title = QString();
        if (!value.isValid() && index.column() == Title) {
            if (m_mediaData.length() != 0 && m_mediaData.length() >= index.row()) {
                YouPlayerMediaContent data = m_mediaData.at(index.row());
                title = data.contentTitle();
            } else {
                // we can't use google info for this anymore..
                title = QLatin1String("Loaded media - ") + QString::number(index.row()+1);
            }
        }
        return title;
    }
    else if (index.isValid() && role == Qt::DecorationRole) {

        if (m_mediaData.length() != 0
            && m_mediaData.length() >= index.row()
            && index.column() == Title ) {
            // should contain url for image/icon
            YouPlayerMediaContent data = m_mediaData.at(index.row());
            //const QPixmap &pixmap = QPixmap::fromImage(data.mediaImage());
            const QIcon &icon = QIcon(QPixmap::fromImage(data.mediaImage()));
            return icon;
        }

    }
    else if (index.isValid() && role == Qt::BackgroundRole) {
        QLinearGradient linearGrad(QPointF(100, 100), QPointF(200, 200));
        linearGrad.setColorAt(0, Qt::white);
        linearGrad.setColorAt(1, Qt::gray);
        const QBrush &brush(linearGrad);
        return brush;
    }

    return QVariant();
}
```

How to add your own Media feed provider (not YouTube)

- extend class MediaFeedApi
- take at look what methods are currently implemented for YouTube in GDataApi.cpp (this class implements MediaFeedApi)
- tweak to your own needs

How to compile and test it

- First Dependencies (=needed libraries for successfull compilation):
  - **Qt (4.6), QtMobility** modules: **Multimedia, Bearer Management and Location** (corrected for QtMobility-Beta release). If new to Qt and QtMobility see Getting started with Qt Mobility APIs
  - **Qjson** library that converts Google data api to Maps and Lists. Source is available at http://gitorious.org/qjson. Main site is at http://qjson.sourceforge.net/. Installation instructions for symbian/linux/windows are available in http://qjson.sourceforge.net/get_it/. Note the capabilities for Symbian below. For linux remember to run make install so that libs gets installed (also ldconfig might be needed).
- **Needed capabilities for Symbian**, add to qjson/src/src.pro file before compilation and remember to run qmake after updating pro file

```
    TARGET.CAPABILITY = Location \
    NetworkServices \
    ReadUserData \
    WriteUserData \
    UserEnvironment \
    ReadDeviceData \
    WriteDeviceData \
    SwEvent
```

- After dependencies and capabilitys get the File:Youplayer against beta.zip file and unzip it somewhere. Then cd to that dir and build it with following commands:

(C) Copyright Nokia 2013. All rights reserved.

- - qmake
  - make #for linux this is sufficient and for symbian emulator build
  - make release-gcce #for symbian device
  - make sis # for symbian package
- For installing sis Youplayer.sis to Symbian device sis-package needs to be signed. Easiest way to get it done is to use open signed online service from SymbianSigned (https://www.symbiansigned.com/app/page/public/openSignedOnline.do). Just enter your phones IMEI, email address and YouPlayer.sis to the reserver fields. After a moment you will have the signed sis in email. Install this sis to the phone.
- **Note!** New zip package File:Youplayer against beta.zip uploaded which contains unsigned sis files for YOUplayer and qjson under **sis** folder.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.