```cpp
#include "databaseoperationsaudio.h"

DatabaseOperationsAudio::DatabaseOperationsAudio(QObject *parent) :
    DatabaseOperations(parent)
{
}

void DatabaseOperationsAudio::getSource()
{
    mCursor=mDBConnection.query("MediaTake.SourceAudio",mongo::Query());

    while(mCursor->more())
    {
        mBSONObj = mCursor->next();

        emit(updatePath(mBSONObj.getStringField("SourcePath")));
    }
    mDBConnection.killCursor(mCursor->getCursorId());
}

void DatabaseOperationsAudio::updateDB(QString vSource,QFileInfo vParent, QFileInfoList
vChildren)
{
    //    mBSONObjBuilder = new mongo::BSONObjBuilder;
    //    mBSONObjBuilder->append("Parent",vParent.absoluteFilePath().toStdString());
    //    mBSONObj = mBSONObjBuilder->obj();

    //    delete mBSONObjBuilder;

    //    mBSONObjBuilder = new mongo::BSONObjBuilder;
    //    mBSONObjBuilder->append("Parent","1");


    //    mCursor = mDBConnection.query("MediaTake.FileListAudio",mongo::Query(mBSONObj),
0,0,&(mongo::BSONObj()=mBSONObjBuilder->obj()));

    //    delete mBSONObjBuilder;

    //    if(!(mCursor->more()))
    //    {

    int vSeconds;
    int vMinutes;
    char vSecondString[3];



    for(QFileInfoList::size_type i; i < vChildren.size(); ++i)
    {

        mBSONObjBuilder = new mongo::BSONObjBuilder;
        mBSONObjBuilder->append("FilePath",vChildren[i].filePath().toStdString());

        mBSONObj = mBSONObjBuilder->obj();

        delete mBSONObjBuilder;

        mBSONObjBuilder = new mongo::BSONObjBuilder;
        mBSONObjBuilder->append("FilePath","1");

        mDBConnection.killCursor(mCursor->getCursorId());


        mCursor = mDBConnection.query("MediaTake.FileListAudio",mongo::Query(mBSONObj),
0,0,&(mongo::BSONObj()=mBSONObjBuilder->obj()));

        delete mBSONObjBuilder;
        if(!(mCursor->more()))
        {
```

```
            mFile = new
TagLib::FileRef(vChildren[i].absoluteFilePath().toStdString().c_str());

            if(!mFile->isNull())
            {

                mTag = mFile->tag();
                mAudioProp = mFile->audioProperties();


                vSeconds = mAudioProp->length() % 60;
                vMinutes = (mAudioProp->length() - vSeconds) / 60;
                std::sprintf(vSecondString, "%02i", vSeconds);

                mBSONObjBuilder = new mongo::BSONObjBuilder;
                mBSONObjBuilder->append("Source",vSource.toStdString());
                mBSONObjBuilder-
>append("Parent",vParent.absoluteFilePath().toStdString());
                mBSONObjBuilder-
>append("FileName",vChildren[i].fileName().toStdString());
                mBSONObjBuilder-
>append("FilePath",vChildren[i].absoluteFilePath().toStdString());
                mBSONObjBuilder->append("Album",mTag->album().toCString());
                mBSONObjBuilder->append("Track",std::to_string(mTag->track()));
                mBSONObjBuilder->append("Title",mTag->title().toCString());
                //                    mBSONObjBuilder->append("AlbumArtist",mFile-
>file()->properties()["ALBUMARTIST"].toString().toCString());
                mBSONObjBuilder->append("Artist",mTag->artist().toCString());
                mBSONObjBuilder-
>append("Length",std::string().append(std::to_string(vMinutes)).append(":").append(vSecon
dString));
                mBSONObjBuilder->append("Bitrate",std::to_string(mAudioProp->bitrate()));
                mBSONObjBuilder->append("Composer",mFile->file()->properties()
["COMPOSER"].toString().toCString());
                mBSONObjBuilder->append("Genre",mTag->genre().toCString());
                mBSONObjBuilder->append("Year",std::to_string(mTag->year()));

                mBSONObj = mBSONObjBuilder->obj();

                try {
                    mDBConnection.insert("MediaTake.FileListAudio",mBSONObj);
                } catch( const mongo::DBException &e ) {
                    QMessageBox::information(NULL,"Database Insertion
Error",QString(e.what()));
                }

            }
            else
            {
                mBSONObjBuilder = new mongo::BSONObjBuilder;
                mBSONObjBuilder->append("Source",vSource.toStdString());
                mBSONObjBuilder-
>append("Parent",vParent.absoluteFilePath().toStdString());
                mBSONObjBuilder-
>append("FileName",vChildren[i].fileName().toStdString());
                mBSONObjBuilder-
>append("FilePath",vChildren[i].absoluteFilePath().toStdString());
                mBSONObjBuilder->append("Album","");
                mBSONObjBuilder->append("Track","");
                mBSONObjBuilder->append("Title","");
                //                    mBSONObjBuilder->append("AlbumArtist",mFile-
>file()->properties()["ALBUMARTIST"].toString().toCString());
                mBSONObjBuilder->append("Artist","");
                mBSONObjBuilder->append("Length","");
                mBSONObjBuilder->append("Bitrate","");
                mBSONObjBuilder->append("Composer","");
                mBSONObjBuilder->append("Genre","");
                mBSONObjBuilder->append("Year","");
```

```
                mBSONObj = mBSONObjBuilder->obj();

                try {
                    mDBConnection.insert("MediaTake.FileListAudio",mBSONObj);
                } catch( const mongo::DBException &e ) {
                    QMessageBox::information(NULL,"Database Insertion
Error",QString(e.what()));
                }

            }
            delete mFile;
        }

        //mFile->~FileRef();


    }

    //    }

    mDBConnection.killCursor(mCursor->getCursorId());
    //    mCursor.release();
}

void DatabaseOperationsAudio::updateTreeView()
{
    mCursor = mDBConnection.query("MediaTake.FileListAudio", mongo::BSONObj());

    vector <QString> vTrackProp;

    mFileProp.clear();
    while(mCursor->more())
    {
        mBSONObj = mCursor->next();

        vTrackProp.push_back(mBSONObj.getStringField("Parent"));
        vTrackProp.push_back(mBSONObj.getStringField("FileName"));
        vTrackProp.push_back(mBSONObj.getStringField("FilePath"));
        vTrackProp.push_back(mBSONObj.getStringField("Album"));
        vTrackProp.push_back(mBSONObj.getStringField("Track"));
        vTrackProp.push_back(mBSONObj.getStringField("Title"));
        vTrackProp.push_back(mBSONObj.getStringField("Artist"));
        vTrackProp.push_back(mBSONObj.getStringField("Length"));
        vTrackProp.push_back(mBSONObj.getStringField("Bitrate"));
        vTrackProp.push_back(mBSONObj.getStringField("Composer"));
        vTrackProp.push_back(mBSONObj.getStringField("Genre"));
        vTrackProp.push_back(mBSONObj.getStringField("Year"));


        mFileProp.push_back(vTrackProp);
        vTrackProp.clear();
    }

    emit(updateTreeWidgetLibraryDisplay(mFileProp));

    mDBConnection.killCursor(mCursor->getCursorId());
    //    mCursor.release();
}

void DatabaseOperationsAudio::updateSource(QString vPath)
{

    mBSONObjBuilder = new mongo::BSONObjBuilder;
    mBSONObjBuilder->append("SourcePath",vPath.toStdString());

    mBSONObj = mBSONObjBuilder->obj();
    delete mBSONObjBuilder;

    mCursor=mDBConnection.query("MediaTake.SourceAudio",mongo::Query(mBSONObj));
```

```
    if(!(mCursor->more()))
    {
        try {
            mDBConnection.insert("MediaTake.SourceAudio",mBSONObj);
        } catch( const mongo::DBException &e ) {
            QMessageBox::information(NULL,"Database Insertion Error",QString(e.what()));
        }
    }

}
void DatabaseOperationsAudio::removeFromDB(QString vPath)
{
    mBSONObjBuilder = new mongo::BSONObjBuilder;
    mBSONObjBuilder->append("SourcePath",vPath.toStdString());
    mBSONObj = mBSONObjBuilder->obj();

    mDBConnection.remove("MediaTake.SourceAudio",mongo::Query(mBSONObj));

    delete mBSONObjBuilder;

    mBSONObjBuilder = new mongo::BSONObjBuilder;
    mBSONObjBuilder->append("Source",vPath.toStdString());
    mBSONObj = mBSONObjBuilder->obj();
    mDBConnection.remove("MediaTake.FileListAudio",mongo::Query(mBSONObj));

}
```