Reading and writing files in QML

This article explains how to access local files using QML

Introduction

QML doesn't provide any API to access local files. Same story for Javascript, which is a language designed to run in web browsers and that natively doesn't provide any API, for security reasons, to access local files. The Qt documentation explains how to read/write files by using QML plugins. Anyway, accessing the code is a bit tricky and therefore I´ve written a small application which explains in few code lines how to achieve this target without using QML plugins. QML plugins are like C++ shared libraries (.dll or .so files) which are useful only if a developer wants to share the same code with different applications, thing which is usually not good for applications which will be uploaded to OVI store.

The Implementation: QML part

Here I show you how the QML file looks like. First of all we import FileIO and create a FileIO instance. We set an ID so that the FileIO can be accessed by other QML elements. For this example, I decided to call it "myFile". MyFile has only one property which is called "source" and it's a string which stores the full path name of the file we want to read/write. To keep this example simple, I created an onError slot which prints out possible errors. If needed the error signal could be connected to other slots/methods defined in QML, but this is something we will briefly see later on in this article. The write and read methods are able to read and write little files. These methods are called by the onCompleted slots only to minimize the amount of code of this example.

```
import QtQuick 1.1
import FileIO 1.0

Rectangle {
    width: 360
    height: 360
    Text {
        id: myText
        text: "Hello World"
        anchors.centerIn: parent
    }

    FileIO {
        id: myFile
        source: "my_file.txt"
        onError: console.log(msg)
    }

    Component.onCompleted: {
        console.log( "WRITE"+ myFile.write("TEST"));
        myText.text =  myFile.read();
    }
}
```

The Implementation: C++ part

As I said in the introduction, we are not going to use QML plugins. To make our object available to QML we have to register the FileIO class.

```
#include "fileio.h"

Q_DECL_EXPORT int main(int argc, char *argv[])
{
   ...
    qmlRegisterType<FileIO, 1>("FileIO", 1, 0, "FileIO");
    ...
}
```

The FileIO class is a simple class which can read and write a file on the disk using QTextStreams. As we have already seen in the previous section, these jobs are accomplished by the read and write methods which are exported by QML thanks to the Q_INVOKABLE macro. To make the source element property available to QML we had to define it in C++ as QObject property. The C++ class defines an error signal, which is emitted if something goes wrong while accessing the file. The signal can be cached in QML using nameOfMyFileIOInstance.connect(anotherQMLObject.method).

```
#ifndef FILEIO_H
#define FILEIO_H

#include <QObject>

class FileIO : public QObject
{
    Q_OBJECT

public:
    Q_PROPERTY(QString source
               READ source
               WRITE setSource
               NOTIFY sourceChanged)
    explicit FileIO(QObject *parent = 0);

    Q_INVOKABLE QString read();
    Q_INVOKABLE bool write(const QString& data);

    QString source() { return mSource; };

public slots:
    void setSource(const QString& source) { mSource = source; };

signals:
```


(C) Copyright Nokia 2013. All rights reserved.

```
    void sourceChanged(const QString& source);
    void error(const QString& msg);

private:
    QString mSource;
};

#endif // FILEIO_H
```

```
#include "fileio.h"
#include <QFile>

FileIO::FileIO(QObject *parent) :
    QObject(parent)
{

}

QString FileIO::read()
{
    if (mSource.isEmpty()){
        emit error("source is empty");
        return QString();
    }

    QFile file(mSource);
    QString fileContent;
    if ( file.open(QIODevice::ReadOnly) ) {
        QString line;
        QTextStream t( &file );
        do {
            line = t.readLine();
            fileContent += line;
         } while (!line.isNull());

        file.close();
    } else {
        emit error("Unable to open the file");
        return QString();
    }
    return fileContent;
}

bool FileIO::write(const QString& data)
{
    if (mSource.isEmpty())
        return false;

    QFile file(mSource);
    if (!file.open(QFile::WriteOnly | QFile::Truncate))
        return false;

    QTextStream out(&file);
    out << data;

    file.close();

    return true;
}
```

Related Links

- How to write data to a file in Qt
- How to read data from a file in Qt

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.