File based logging in Qt for debugging

ID		**Creation date**	28th Mar 2010
Platform	S60 5th Edition	**Tested on devices**	Nokia N97 Mini
Category	Qt	**Subcategory**	

Keywords (APIs, classes, methods, functions): qInstallMsgHandler

Overview

File Based Logging in Qt For Debugging.

I agree that there are ways to debug application using IDE, but in long run every developers requires a way to print debugging info in to the file. In case of Qt there are standard ways to use file logging as shown below.

- Using `Log4Qt`
- Using `QxtLogger`

But the above 2 requires developers either to include a heavier static lib or have to convert to a specific data format (like `QVariants` or `QVariantList`) before logging .

Here we will see a simple file logging methods which is very easy to use and requires not more then 25 lines. Before you start logging still it provides same level of flexibility to log debug/info/warning etc.

Step 1.

We will create a logging handler.

```
#include <fstream>

using namespace std;
ofstream logfile;

void SimpleLoggingHandler(QtMsgType type, const char *msg) {
    switch (type) {
        case QtDebugMsg:
            logfile << QTime::currentTime().toString().toAscii().data() << " Debug: " << msg << "\n";
            break;
        case QtCriticalMsg:
            logfile << QTime::currentTime().toString().toAscii().data() << " Critical: " << msg << "\n";
            break;
        case QtWarningMsg:
            logfile << QTime::currentTime().toString().toAscii().data() << " Warning: " << msg << "\n";
            break;
        case QtFatalMsg:
            logfile << QTime::currentTime().toString().toAscii().data() <<  " Fatal: " << msg << "\n";
            abort();
        }
    }
```

Step 2.

Open A file to be where application will be logging for example

```
    logfile.open("c:\\data\\logfile.txt", ios::app);
```

Step 3.

Install our handler as a qInstallMsgHandler.

```
    qInstallMsgHandler(SimpleLoggingHandler);
```

That is it now we can use the normal **qCritical()**, **qDebug()**, **qFatal()** and **qWarning()** to direct messages to file. for example

```
#include <QTime>

#define PREFIX "[myDebugPrint]" << QTime::currentTime().toString("hh:mm:ss.zzz")

void MyClass::myFunction(int variable)
{
    qDebug() << PREFIX << "MyClass::myFunction() begin";
    qDebug("code is not activated");
    qDebug() << variable;
    qDebug() << PREFIX << "MyClass::myFunction() end";
}
```

Reference links

(C) Copyright Nokia 2013. All rights reserved.

- Qt - cross-platform application and UI framework
- SDK help

--skumar_rao 16:52, 29 March 2010 (UTC)

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.