

KDE Tutorial, Teil 2: Ein erstes Programm mit Hauptfenster

In diesem Teil werden wir unser erstes vollständiges KDE Programm entwickeln. Dieses Programm wird ein Hauptfenster samt Menü besitzen.

Das Projekt besteht dabei aus zwei Quelltext-Dateien:

1. main.cpp
Das obligatorische Hauptprogramm einer C++ Anwendung
2. mainwindow.cpp
Das KDE Hauptfenster

Das KDE Hauptprogramm

```
#include <KAboutData>
#include <KCmdLineArgs>
#include <KApplication>
#include "mainwindow.h"

int main ( int argc, char **argv )
{
    KAboutData aboutData (
        "Tutorial",
        NULL,
        ki18n ( "Tutorial" ), // Name des Programms
        "0.2",
        ki18n ( "KDE Tutorial, Teil 2" ),
        KAboutData::License_GPL_V3,
        ki18n ( "Copyright Andreas Blochberger, 2009" ),
        ki18n ( "Ein erstes KDE Programm mit Hauptfenster" ),
        "http://argeleb.wordpress.com",
        "" ); // E-Mail Adresse zum Melden von Bugs
    KCmdLineArgs::init ( argc, argv, &aboutData );
    KApplication app;

    MainWindow* mainWindow = new MainWindow();
    mainWindow->show();

    return app.exec();
}
```

Jedes KDE Programm sollte sich ähnlich bedienen lassen und auch ähnlich aussehen, das ist ja eine der Stärken einer einheitlichen Desktopumgebung. Mit der Klasse [KAboutData](#) werden allgemeine Informationen zum Programm zusammengefasst. Diese Daten sind dann im About-Dialog der Anwendung einzusehen.

Als erstes wird mit der Funktion [KCmdLinArgs::init](#) sichergestellt, dass die Kommandozeilen-Argumente, die an das Programm übergeben werden, korrekt interpretiert werden. Jedes KDE Programm versteht nämlich verschiedene KDE/Qt-Optionen. Ruft man das so erzeugte Programm später auf der Kommandozeile mit der Option `--help` auf, erscheint folgender Hilfetext:

```
Usage: tutorial [Qt-options] [KDE-options]

KDE Tutorial, Teil 2

Generic options:
  --help           Show help about options
  --help-qt       Show Qt specific options
  --help-kde      Show KDE specific options
```

```
--help-all      Show all options
--author        Show author information
-v, --version   Show version information
--license       Show license information
--              End of options
```

Noch bevor das Hauptfenster instantiiert wird, muss das Applikations--Objekt der Klasse [KApplication](#) erzeugt werden.

Die nächsten beiden Zeilen erzeugen das Hauptfenster der Anwendung. Dabei handelt es sich um eine Klasse, die wir uns im folgenden näher ansehen werden. Mit `mainwindow->show()` wird das Fenster zur Anzeige gebracht.

Schließlich sorgt ein Aufruf der Funktion [app.exec\(\)](#) dafür, dass das KDE-Programm am Laufen gehalten wird.

Das Hauptfenster

Die Klasse `MainWindows` des Projekts stellt das Hauptfenster der Anwendung dar. Die Klasse besteht, wie üblich, aus zwei Dateien, die Deklaration im Header-File und die Definition in der C++ Datei.

MainWindow Deklaration

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <KXmlGuiWindow>

class MainWindow : public KXmlGuiWindow
{
public:

    MainWindow ( QWidget *parent=0 );
};

#endif // MAINWINDOW_H
```

MainWindow Definition

```
#include "mainwindow.h"

MainWindow::MainWindow ( QWidget *parent ) : KXmlGuiWindow ( parent )
{
    setupGUI\(\);
}
```

Das Hauptfenster einer KDE Anwendung verfügt über ein Standard-Menü (ein Einstellungsmenü und das Hilfe-Menü). Diese Funktionalität wird von der Klasse [KXmlGuiWindow](#) bereitgestellt.

Mit der Methode [setupGUI\(\)](#) wird die Standard-Oberfläche des Fensters erzeugt, also in diesem Fall das Hauptmenü.

Ruft man nun das Programm auf, so erscheint folgendes Fenster:



Abbildung 1: Hauptfenster

Ruft man das Hilfemenü auf, so kann man bereits einige Menüpunkte sehen, die standardmäßig erzeugt werden. Wählt man „About Tutorial“ auf, erscheint folgender Dialog:



Abbildung 2: Über das Programm

Erstellen des Programms

Die Verzeichnis-Struktur des ersten Programms sieht wie folgt aus:

- <Projektverzeichnis>
 - build
 - src

Um das Programm mit cmake erstellen zu können, muss eine **CMakeLists.txt** Datei im Projektverzeichnis erstellt werden:

```
project (tutorial)
find_package(KDE4 REQUIRED)
include (KDE4Defaults)
include_directories(${KDE4_INCLUDES})
set(tutorial_SRCS
  src/mainwindow.cpp
  src/main.cpp)
kde4_add_executable(tutorial ${tutorial_SRCS})
target_link_libraries(tutorial ${KDE4_KDEUI_LIBS})
install(TARGETS tutorial  ${INSTALL_TARGETS_DEFAULT_ARGS})
```

Um das Programm in der Konsole zu erzeugen, wechselt man in das Verzeichnis **build** des Projekts

und gibt folgende Befehle ein:

```
cmake ..  
make
```