Achim LINGOTT

Update inside

Einführung in Qt

2. Auflage

Entwicklung von GUIs für verschiedene Betriebssysteme

Quellcode unter
plus.hanser-fachbuch.de

HANSER

Lingott

Einführung in Qt

Update inside.

Mit unserem kostenlosen Update-Service zum Buch erhalten Sie aktuelle Infos zur GUI-Entwicklung mit Qt.

Und so funktioniert es:

1. Registrieren Sie sich unter:
   www.hanser-fachbuch.de/qt-update
2. Geben Sie diesen Code ein:

GKs2-pE4D-aKb5-45Hs

Der Update-Service läuft bis Februar 2025. Als registrierter Nutzer werden Sie in diesem Zeitraum persönlich per E-Mail informiert, sobald ein neues Buch-Update zum Download verfügbar ist.

Wenn Sie Fragen haben, wenden Sie sich gerne an: **update-inside@hanser.de**

Achim Lingott

Einführung in Qt

Entwicklung von GUIs für verschiedene Betriebssysteme

2., überarbeitete Auflage

HANSER

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Sylvia Hasselbach
Copy editing: Sandra Gottmann, Wasserburg
Bilder 4.12, 8.21, 8.24, 8.26: © Sven Lingott, Berlin
Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de
Umschlagrealisation: Max Kostopoulos
Titelmotiv: © gettyimages.de/a-r-t-i-s-t
Satz: Eberl & Koesel Studio, Kempten
Druck und Bindung: Hubert & Co. GmbH & Co. KG BuchPartner, Göttingen
Printed in Germany

Print-ISBN: 978-3-446-47610-3
E-Book-ISBN: 978-3-446-47784-1
E-Pub-ISBN: 978-3-446-47785-8

Inhalt

Vorwort

Dieses Buch vermittelt Einsteigern mit C++-Vorkenntnissen die Grundlagen der Qt-Programmierung. Mit der hervorragenden Qt-Bibliothek lassen sich grafische User Interfaces für die unterschiedlichsten Anwendungsfälle programmieren.

Die 1. Auflage dieses Buches beruhte auf der damals gerade erschienenen Version 6.0 der Qt-Bibliothek. Dort waren noch nicht alle Module der vorhergehenden Version 5 enthalten, und es musste bei einigen Beispielen auf die Version Qt 5 zurückgegriffen werden. Das ist heute nicht mehr so.

Das Manuskript zur 2. Auflage wurde auf der Grundlage der Bibliotheksversion Qt 6.4 erstellt.

Ebenso werden alle Programme mit CMake erstellt, anstelle des in der 1. Auflage verwendeten qmake.

Einige Quelltexte sind in der vorliegenden 2. Auflage nicht mehr (oder nur in Ausschnitten) im gedruckten Buch zu finden, sondern auf dem Download-Portal. Sehen Sie sich also auf alle Fälle die vollständigen Quelltexte an, um die Beispiele auch zu verstehen.

Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial.

Geben Sie auf *plus.hanser-fachbuch.de* einfach diesen Code ein:

```
plus-Gn4tp-r6L4g
```

Auf dieser Webseite sind alle Quelltexte für die Projekte zu finden, die im Buch beschrieben werden.

Updates und Aktualität

Damit Sie möglichst lange mit diesem Buch arbeiten können, haben Sie die Möglichkeit, sich für den kostenlosen Update-inside-Service zu registrieren: Geben Sie unter

www.hanser-fachbuch.de/qt-update

diesen Code ein:

`GKs2-pE4D-aKb5-45Hs`

Dann erhalten Sie bis Februar 2025 Aktualisierungen in Form zusätzlicher Kapitel als PDF-Datei. Darin stelle ich Ihnen wichtige Neuerungen vor und gehe auf Änderungen ein, die die Inhalte dieses Buches betreffen.

Danksagung

An dieser Stelle möchte ich allen Beteiligten einen großen Dank aussprechen. Er gilt allen beteiligten Mitarbeitern des Carl Hanser Verlages, insbesondere Frau Sylvia Hasselbach und Frau Kristin Rothe, die sich stets allen meinen Fragen zu Inhalt und Gestaltung gestellt haben und hilfreich mit Ideen zur Seite standen.

Diesen Dank möchte ich aber auch an diejenigen richten, die, vermutlich ohne es zu ahnen, zur einen oder anderen Idee beigetragen haben: Das sind die Teilnehmer meiner Seminare, deren Fragen mich dazu anregten, das eine oder andere Problem etwas umfassender oder überhaupt anzugehen.

Und nun viel Spaß beim Lesen und bei der Arbeit mit Qt!

Achim Lingott

Der Verlag und die Autoren haben sich mit der Problematik einer gendergerechten Sprache intensiv beschäftigt. Um eine optimale Lesbarkeit und Verständlichkeit sicherzustellen, wird in diesem Werk auf Gendersternchen und sonstige Varianten verzichtet; diese Entscheidung basiert auf der Empfehlung des Rates für deutsche Rechtschreibung. Grundsätzlich respektieren der Verlag und die Autoren alle Menschen unabhängig von ihrem Geschlecht, ihrer Sexualität, ihrer Hautfarbe, ihrer Herkunft und ihrer nationalen Zugehörigkeit.

1 Einführung und erste Schritte

Dieses Buch wurde hauptsächlich für Personen geschrieben, die Vorkenntnisse in der Programmiersprache C++ besitzen, aber bisher wenig oder nichts mit Qt zu tun hatten. Qt ist übrigens keine Abkürzung, sondern wird ausgesprochen wie das englische Wort *cute*.

Es soll Ihnen die Qt-Grundlagen nahebringen und Sie in die Lage versetzen sich selbstständig weiter mit Qt zu beschäftigen. Sie haben mit Qt eine ausgezeichnete Möglichkeit, grafische User Interfaces für verschiedene Programme zu erstellen. Vielleicht haben Sie schon C++-Programme, die jetzt statt einer Konsolenbedienung ein mit Qt erstelltes GUI erhalten sollen. Dieses Buch wäre ein Anfang dafür. Es soll zwar eine Einführung sein, wird aber an einigen Stellen doch weiter in die Tiefen der Qt-Programmierung, insbesondere die Qt-Bibliothek, einsteigen, um Ihnen wesentliche Zusammenhänge zu zeigen. Darauf können Sie dann später weiter aufbauen und Ihre eigenen Programme entwickeln.

Qt entstand Anfang der 1990er-Jahre und ist in C++ geschrieben. Es ist mehrfachlizenziert und kann sowohl für die Open-Source-Programmierung als auch kommerziell genutzt werden.

Qt 6 baut vollständig auf die Standardversion C++17 auf. Deshalb ist zur Erstellung auch ein C++17-kompatibler Compiler nötig. Qt für Windows unterstützt die Betriebssysteme Windows 10 und Windows 11 mit der Architektur x86_64. Die unterstützten Compiler sind MSVC 2019, MSVC 2022 und MinGW 11.2. Ansonsten ist es auch für Linux, macOS, Android, iOS und weitere Plattformen, z. B. auch für WebAssembly, erhältlich (WebAssembly ermöglicht die Darstellung von Qt-Applikationen im Browser).

Ab der Version 2013 wird als Entwicklungsumgebung *Visual Studio* von Microsoft unterstützt (siehe Abschnitt 1.7). Die Hauptmerkmale von Plug-ins für Visual Studio sind:

- Assistenten zum Erstellen neuer Qt-Projekte und Qt-Klassen
- Verwendung des *Meta Object Compilers (moc)*, des *User Interface Compilers (uic)* und des *Resource Compilers (rcc)*
- Import und Export von Qt-Projektdateien
- Automatische Konvertierung eines Qt-Visual Studio-Projekts in ein *Qt Creator*-Projekt und umgekehrt
- Eingebautes Qt-Ressourcenmanagement
- Erstellen einer Übersetzungsdatei (.ts)

- Benutzung des *Qt Linguist*
- Qt-Dokumentation

Ebenso wie in Microsoft *Visual Studio* lassen sich Qt-Plug-ins auch in *Eclipse*, *Netbeans* oder in *Code::Blocks* verwenden.

Das Kompilieren und Linken von Qt-Quelltext wird wie üblich auf der Grundlage eines *Makefiles* durchgeführt, das von verschiedenen Programmen erstellt werden kann. In Qt existieren *qmake* und *CMake*. Es gibt Unterschiede. Das Programm *qmake* ist einfacher zu erlernen und bestens in Qt integriert, *CMake* ist schwieriger zu erlernen, bietet aber einige Vorteile, insbesondere beim Einbinden fremder Programme. Wir werden in den Buch-Beispielen grundsätzlich CMake verwenden.

Das Programm *CMake* benötigt eine Informationsdatei *CMakeLists.txt*.

Ein wichtiger Teil des Qt-Frameworks ist der *Meta-Object Compiler* (*moc*). Der *moc* durchsucht die Header-Dateien von Klassen nach C++-fremdem Quellcode und erstellt daraus C++-Quellcode, der gemeinsam mit den anderen C++-Inhalten kompiliert wird.

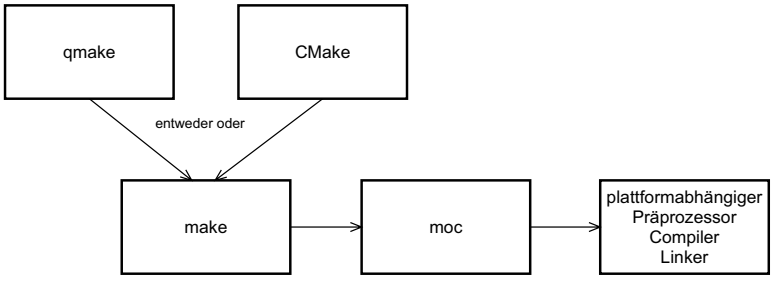

Bild 1.1 Die Erstellung eines Qt-Projekts

Aktuelle Informationen zu den gültigen Bibliotheken, aktuellen Anwendungen und auch Beispiele finden Sie unter

https://www.qt.io/

1.1 Download und Installation

Alle benötigten Programme und die Qt-Bibliothek erhalten Sie per Download von

https://www.qt.io/download

Wählen Sie *Downloads for open source users*. Auf den folgenden Seiten gelangen Sie zum Download des *Qt Online Installer*. Der Punkt *Benutzerdefinierte Installation* nach dem Download und dem Starten des *Qt Online Installer* ermöglicht Ihnen die Auswahl einzelner zu installierender Komponenten.

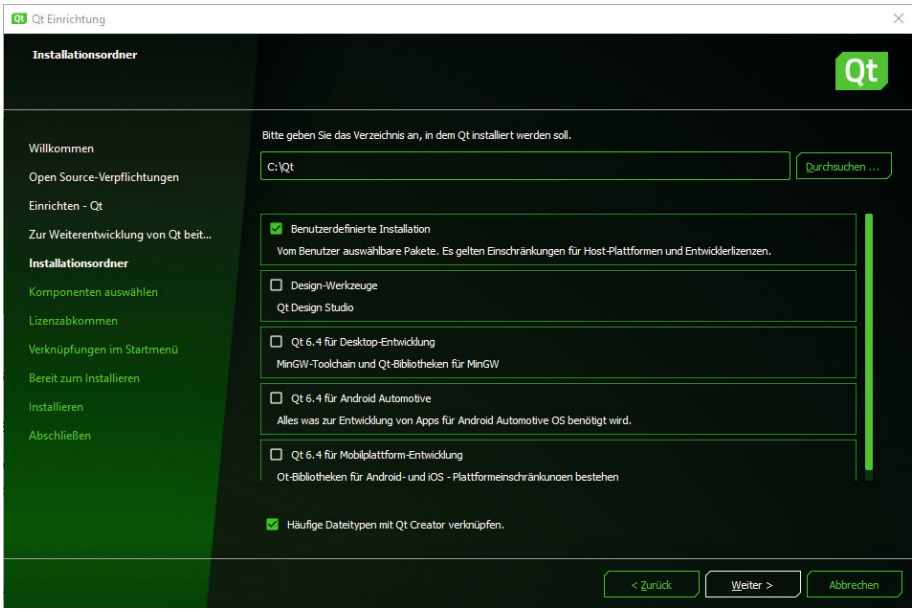

Bild 1.2 Qt Setup und Installer

Wählen Sie die gezeigten Komponenten aus (Bild 1.3).

Bei *Developer and Designer Tools* benutzen Sie die Voreinstellungen.

Da sowohl die Bibliothek als auch die bereitgestellten Programme ständig weiterentwickelt werden, sehen die Installationsoberflächen und die Versionsangaben in Zukunft sicher anders aus. Arbeiten Sie also sinngemäß.

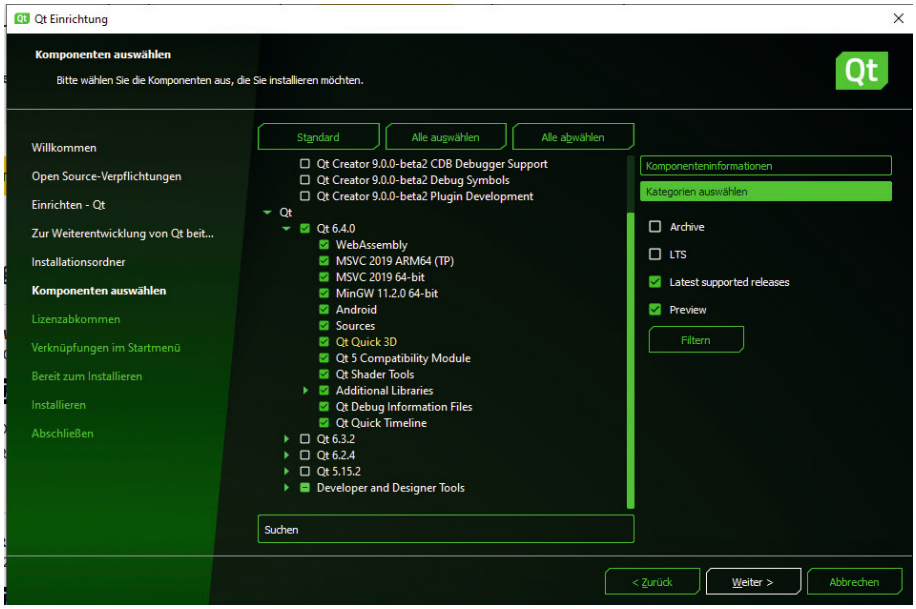

Bild 1.3 Auswahl im Qt Installer

Der Download- und Installationsprozess dauert sicher einige Zeit. Nach Fertigstellung starten Sie den installierten Qt Creator einmal und sehen sich das Ergebnis an.

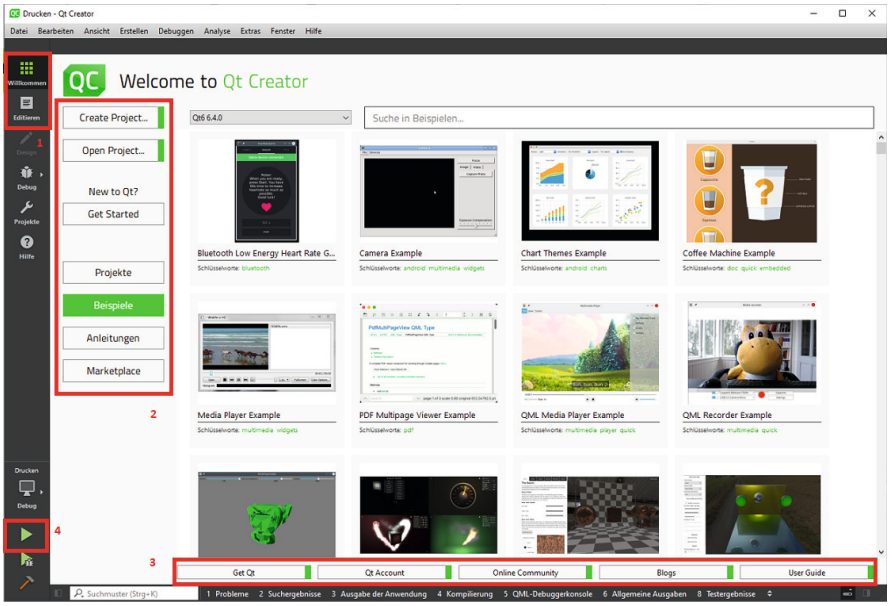

Bild 1.4 Qt Creator nach dem Start

Es fällt sofort auf, dass sehr viele Beispielprogramme zu verschiedenen Themen mit installiert wurden. Sie erreichen diese Beispiele durch Einschalten der Willkommen-Seite (1) und anschließende Betätigung des Buttons *Beispiele* (2). Weiterhin gibt es Zugänge zu sehr vielen Anleitungen und dem Marktplatz, über den Sie viele Zusatzprodukte beziehen können. Zugänge zur Online-Community und zum Handbuch des Qt Creators ergänzen diese Seite (3). Über den dreieckigen Button (4) wird ein Programm gestartet. Der darunter liegende Button startet den Debugger (siehe Kapitel 11).

1.2 Die Qt-Bibliothek

Die Bibliothek finden Sie unter

https://doc.qt.io/qt-6/

Unter *Reference/All Qt C++ Classes* sind die Informationen zu mehr als 1500 Klassen und unter *All Qt Modules* die Informationen zu den Modulen zu finden. Die Module unterteilen sich in *Qt Essentials* und *Qt Add-Ons*.

Die in den *Qt Essentials* enthaltenen Klassen sind grundlegend für alle Plattformen und werden für die meisten Qt-Anwendungen benötigt.

Die Add-on-Module werden eventuell zusätzlich für bestimmte Einsatzzwecke benötigt. Sie stehen aber möglicherweise nicht auf allen Plattformen zur Verfügung.

Das Installationsprogramm bietet die Möglichkeit, weitere Module herunterzuladen und zu installieren. Das können Sie auch bei bereits erfolgter Installation über den Punkt *Qt Maintenance Tool* tun. Auch über *Uninstall Qt* erreichen Sie diese Möglichkeit (lassen Sie sich nicht vom Namen täuschen).

1.2.1 Die Klassen der Bibliothek

In der Bibliothek sind Klassen enthalten, die zur Erstellung und Gestaltung von GUIs benötigt werden, aber auch solche Klassen, die die Erstellung der dazugehörigen Logik ermöglichen. Das sind z. B. Klassen wie `QSqlDatabase`, die Sie zum Verbindungsaufbau zu einer Datenbank benötigen, oder `QXmlReader` zum Auswerten einer XML-Datei.

Viele Klassen sind abgeleitet von `QObject`. Diese Klasse stellt insbesondere Funktionen zur Verfügung, die von allen abgeleiteten Klassen benötigt werden, z. B. Funktionen wie `connect()` und `disconnect()`.

Aber aufgepasst, wer andere Bibliotheken wie z. B. die von .NET oder Java kennt, hat gelernt, dass alle im Programm erzeugten Klassen automatisch von der Klasse `Object` abgeleitet sind. Das ist in Qt nicht so! Es gibt zwar eine Klasse `QObject`. Aber von dieser Klasse erfolgt keine automatische Ableitung, sondern eine solche Ableitung muss programmiert werden. Viele Klassen der Bibliothek sind von `QObject` abgeleitet.

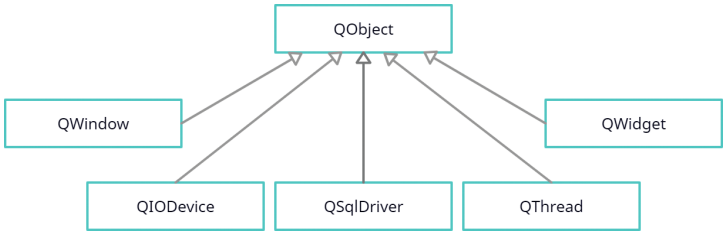

Bild 1.5
Ein sehr kleiner Ausschnitt der Qt-Bibliothek

Weitere Informationen können Sie in der Klassenbibliothek erhalten. Rufen Sie die Klasse `QObject` auf. Dort werden alle von dieser Klasse abgeleiteten Klassen angezeigt.

1.2.2 Die Module der Bibliothek

Während die Klassennamen in der Regel mit Q beginnen (z. B. `QWidget`), beginnen die Namen der Module meist mit Qt (z. B. `QtWidgets`). An dieser Stelle ein paar Informationen zu wichtigen Modulen.

QtCore

Das Modul enthält Klassen mit nichtgrafischer Funktionalität und ergänzt C++ durch einige Eigenschaften, wie z. B.

- einen neuen Mechanismus für die Objektkommunikation, die *signals* und *slots*,
- ein Makro `Q_OBJECT`, das die Kommunikation über *signals* und *slots* sowie einige andere Eigenschaften ermöglicht,

- den *Meta Object Compiler (MOC)*, der in jeder von `QObject` abgeleiteten Klasse C++-fremden Quellcode (wie *signals, slots*) in Code umwandelt, den jeder C++-Compiler verarbeiten kann,
- ein eigenes Property-System, das über ein Makro *Q_PROPERTY* ermöglicht, einen ähnlichen Zugriff auf Variablen zu erreichen, wie sie durch die Properties bei C# bekannt sind,
- Zeiger, die automatisch auf 0 gesetzt werden, wenn das referenzierte Objekt zerstört wird (solche Zeiger werden durch das Klassentemplate `QPointer` bereitgestellt),
- Unterstützung für die Erstellung benutzerdefinierter Typen.

Wichtige Klassen dieses Moduls sind z. B. `QObject`, `QPointer`, `QVariant`, `QMetaType`.

QtGUI

In diesem Modul sind die Basisklassen für grafische User Interfaces (GUI) enthalten, Klassen für die Ereignisbehandlung, OpenGL- und OpenGL ES-Integration, für Grafiken, Schriftarten und Text.

Für die weitergehende Entwicklung von Benutzeroberflächen bietet Qt mit dem Modul `QtQuick` eventuell besser geeignete Klassen und Funktionen. Das Modul `QtGUI` (und somit die darin enthaltenen Klassen) wird standardmäßig inkludiert.

Die wichtigsten Klassen dieses Moduls sind die Klassen `QGuiApplication` und `QWindow`.

`QGuiApplication` (abgeleitet von `QCoreApplication`) stellt die überschriebene statische Funktion `instance()` zur Verfügung. Diese gibt einen Zeiger zurück, der dem globalen Zeiger *qApp* entspricht.

QtQML

enthält alle Klassen für die Sprache *QML (Qt Meta Language)*, die von *Quick* benötigt wird (dazu mehr in Kapitel 8 dieses Buches).

Im Modul sind auch die QML-Typen definiert (siehe Kapitel 8) und folgende QML-Objekttypen:

```
Component, QtObject, Binding, Connections, Timer
```

Eine JavaScript-Umgebung ist ebenfalls integriert.

QtQuick

Dieses Modul enthält die Standardbibliothek für QML-Applikationen (eine weitere Möglichkeit zum Erstellen von UI, siehe Kapitel 8) und die erforderliche Engine. Es werden alle grundlegenden Typen zum Erstellen einer Benutzeroberfläche mit QML bereitgestellt sowie eine Zeichenfläche und Möglichkeiten zur Animation und Kontrollelemente für Benutzeroberflächen.

QtWidgets

Dieses Modul stellt eine Reihe von Elementen zur Gestaltung von GUIs zur Verfügung, wie die Klassen `QWidget`, `QPushButton`, `QFrame` und andere sowie Klassen für Styles, Layouts und die Model-View-Architektur.

Im Bereich der Add-on-Module ist vielleicht das Modul **`QtPrintSupport`** zu erwähnen. Es enthält Klassen, die ein Drucken auf jeder Plattform ermöglichen. Ebenso lassen sich damit PDF-Dateien erzeugen (siehe Kapitel 7).

Auch wichtig kann das Modul **`QtSQL`** sein. Darin sind Klassen enthalten, die eine Integration von Datenbanken ermöglichen. Man könnte die Klassen grob in die Bereiche Datenbanktreiber, SQL-Zuständigkeit und GUI-Zuständigkeit einteilen (dazu mehr in Kapitel 6).

1.3 Die installierten Programme

Qt Creator

Ein Bild davon ist weiter vorn zu finden (Bild 1.4). Qt Creator stellt eine Arbeitsumgebung für den Programmierer zur Verfügung, sowohl für Qt- als auch Quick-Applikationen. Seine Benutzung wird in Abschnitt 1.4 genauer beschrieben. Mit dem Ct Creator lassen sich auch reine C- und C++-Anwendungen erstellen, ebenso Qt-Projekte mit Python.

Außer dem Qt Creator gibt es inzwischen Plug-ins für andere C++-Entwicklungsumgebungen wie Visual Studio von Microsoft und Eclipse, Netbeans und Code::Blocks (siehe Abschnitt 1.7).

Alle Programme außer dem Qt Creator werden für jeden installierten Compiler und jede installierte Bibliotheksversion separat installiert. Deshalb finden Sie auch im Windows-Start-Verzeichnis jedes der folgenden Programme entsprechend oft.

Qt Assistant

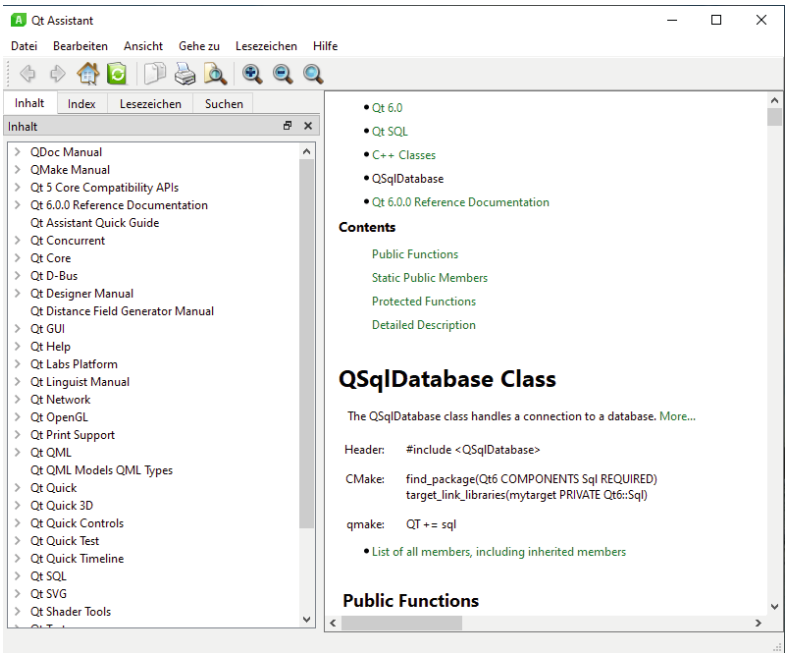

Bild 1.6 Qt Assistant

Mithilfe dieses Programms können Sie sich die Dokumentationen zu den in der Bibliothek vorhandenen Modulen und den dazugehörigen Klassen ansehen. Diese Informationen sind auch im Qt Creator erhältlich.

Qt Designer

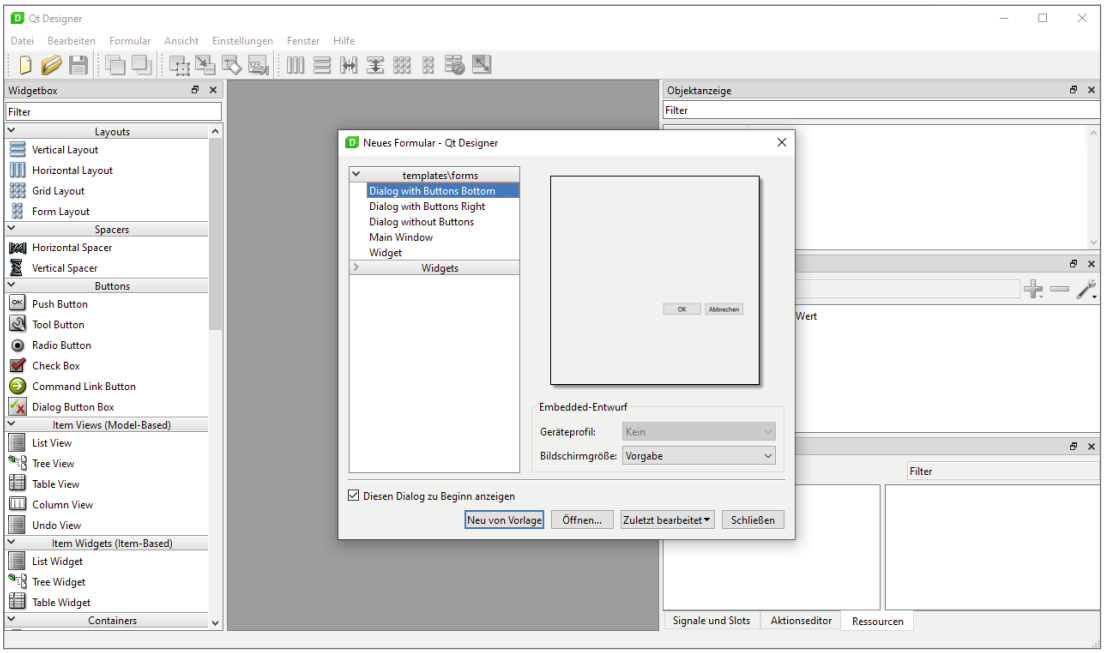

Bild 1.7 Qt Designer

Damit lassen sich GUIs auf grafischem Wege erstellen, indem gewünschte Widgets auf einem Fenster (dem Formular) nach der Methode *What You See Is What You Get* (WYSIWYG) platziert werden. Dieser Designer ist auch in den Qt Creator integriert und kann von ihm aus benutzt werden. In Plug-ins für andere Entwicklungsumgebungen steht er ebenfalls zur Verfügung. Der Qt Designer erstellt bei seiner Benutzung eine XML-Datei, in der die Einträge des Designers gespeichert werden. Mithilfe dieser XML-Datei können bereits erstellte Formulare auch in anderen Projekten wiederverwendet werden.

Qt Linguist

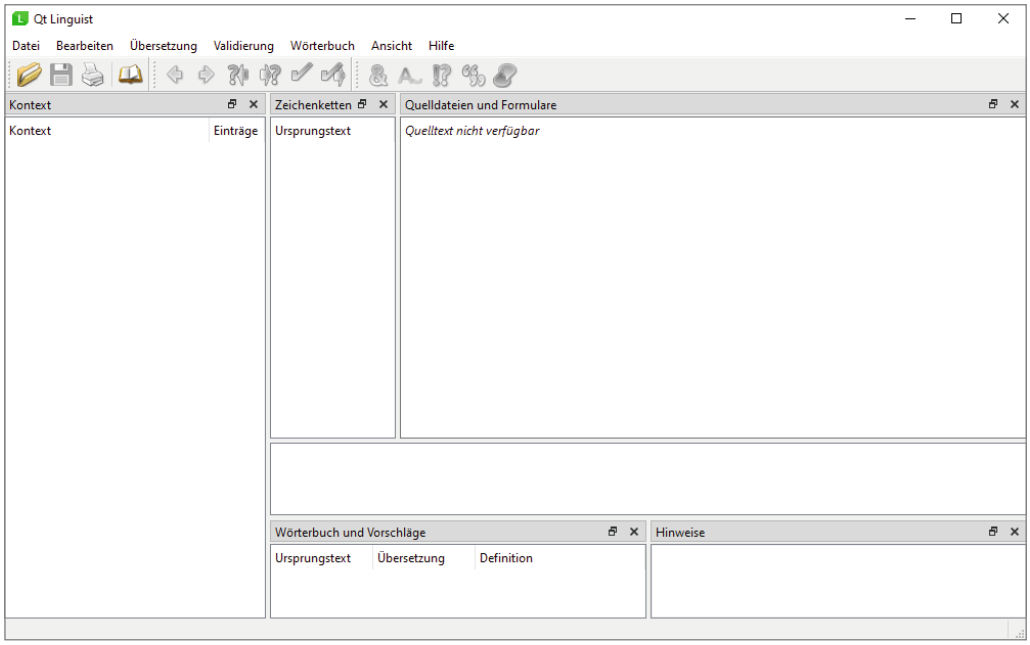

Bild 1.8 Qt Linguist

Mithilfe dieses Programms können Sie entsprechend vorbereitete Zeichenketten (z. B. den Titel eines Fensters, Button-Beschriftungen oder die Inhalte eines Labels) in verschiedene Sprachen übersetzen lassen. Es entstehen Übersetzungsdateien, die beim Erstellen des Programms mit eingebunden werden können und somit beim Start automatisch die Zeichenketten in der gewünschten Sprache erscheinen lassen (siehe Abschnitt 5.4).

Konsole

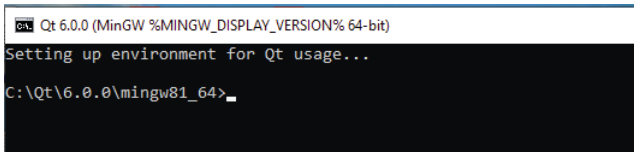

Bild 1.9
Konsole für MinGW mit Qt 6

Da die Qt-Programme bei der Installation nicht in die PATH-Variable des Betriebssystems eingetragen werden, ersparen Sie es sich, das Installationsverzeichnis zu suchen und in der Konsole zum entsprechenden Verzeichnis zu wechseln, wenn Sie die Konsole aus dem Qt-Ordner des Windows-Start-Verzeichnisses benutzen.

1.4 Im Qt Creator erstellte Dateien und ihre Bedeutung

Beim Neuanlegen eines Projekts wird ein Projektverzeichnis erstellt, das alle Dateien enthält, die Sie auch im *Projektexplorer* des Qt Creator sehen. Weiterhin entsteht beim ersten Erstellen des Projekts ein weiterer Ordner, dessen Name sich nach Projektname, Bibliothek und Compiler richtet.

1.4.1 Datei CMakeLists.txt

Diese Datei wird angelegt, wenn Sie bei der Projekterstellung als Build-System *CMake* auswählen. *CMake* erstellt eine Konfigurationsdatei – *CMakeLists.txt*. Mit dieser Datei können Sie ein solches Projekt auch aus dem Projektverzeichnis aufrufen. Genauso gibt es dort auch eine Datei *CMakeLists.txt.user*. Sie enthält alle notwendigen Informationen zu Rechner, Betriebssystem etc.

Der Projektexplorer im Qt Creator könnte jetzt so aussehen.

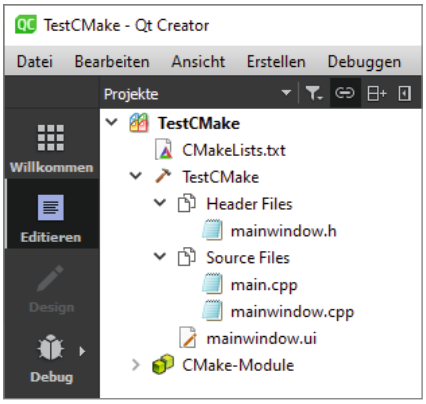

Bild 1.10
Projektexplorer des Qt Creators bei Verwendung von CMake

Die Datei *CMakeLists.txt* könnte im Beispielprojekt folgenden Inhalt haben:

```
01  cmake_minimum_required(VERSION 3.5)
02
03  project(TestCMake LANGUAGES CXX)
04
05  set(CMAKE_INCLUDE_CURRENT_DIR ON)
06
07  set(CMAKE_AUTOUIC ON)
08  set(CMAKE_AUTOMOC ON)
09  set(CMAKE_AUTORCC ON)
10
11  set(CMAKE_CXX_STANDARD 11)
12  set(CMAKE_CXX_STANDARD_REQUIRED ON)
13
14  find_package(QT NAMES Qt6 Qt5 COMPONENTS Widgets REQUIRED)
15
```

```
16  set(PROJECT_SOURCES
17          main.cpp
18          mainwindow.cpp
19          mainwindow.h
20          mainwindow.ui
21  )
22
23  add_executable(TestCMake  ${PROJECT_SOURCES})
24  find_package(Qt${QT_VERSION_MAJOR} COMPONENTS Widgets REQUIRED)
25  target_link_libraries(TestCMake PRIVATE Qt6::Widgets)
```

Die Zeilen enthalten einzelne Funktionen und haben folgende Bedeutung:

- **Zeile 1:** Es wird die Mindestversion von *CMake* festgelegt.
- **Zeile 3:** Festlegen des Projektnamens und der Programmiersprache (CXX bedeutet C++)
- **Zeilen 7 – 9** gestatten *CMake*, den *Meta-Object Compiler* (*moc*), *Ressource Compiler (rcc)* und den *User Interface Compiler* selbstständig bei Bedarf zu benutzen.
- **Zeile 11** legt die Mindestversion des C++-Compilers fest.
- **Zeilen 24, 25** enthalten die in der Bibliothek angegebenen Zeilen (Bild 1.11).

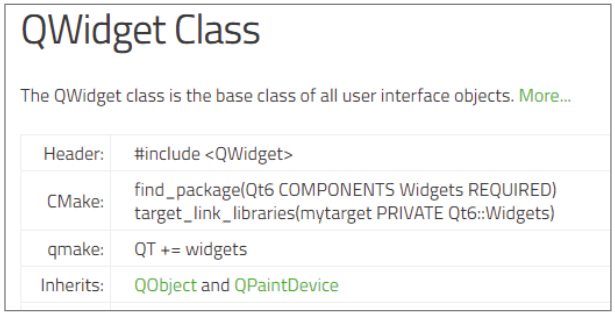

Bild 1.11
Ausschnitt aus der Qt-Bibliothek

Weitere Informationen über *CMake* finden Sie hier:

https://doc.qt.io/qt-6/cmake-manual.html

1.4.2 Dateien .pro und .pri

Sollten Sie als Build-System *qmake* wählen, erhalten Sie eine Datei mit der Endung .pro. Sie hat als Inhalte hauptsächlich Variablen, die in der Regel Listen von Zeichenketten enthalten (Informationen für *qmake*).

Wichtige Variablen sind z. B.:

QT

Sie erhält eine Liste der Module, die im Projekt verwendet werden. Damit ist es möglich, beim Inkludieren der Klassen durch den Präprozessor im Quelltext nur den Klassennamen anzugeben und nicht das Modul, in dem sich die Klasse befindet (z. B. `#include <QSqlDatabase>` und nicht `#include <QtSql/QSqlDatabase>`). Die dieser Variablen zu übergebenden Namen finden Sie in der Klassenbibliothek unter den jeweiligen Klassennamen (siehe Bild 1.11).

SOURCES

Sie enthält eine Liste aller Quelltextdateien des Projekts, z. B. *main.cpp* und *mainwindow.cpp.*

HEADERS

Hier sind alle Dateinamen der Headerdateien (.h) aufgeführt.

FORMS

Darin ist die vom Qt Designer verwendete XML-Datei enthalten. Bei mehreren Formularen im Projekt (z. B. weil ein Formular aus einem anderen aufgerufen wird) sind hier eventuell auch mehrere Dateien eingetragen.

RESOURCES

Dieser Variablen wird die benutzte Ressourcendatei zugewiesen (siehe Kapitel 5). Der Eintrag könnte z. B. so aussehen:

```
RESOURCES += res.qrc
```

Weitere Informationen finden Sie auf der Webseite

https://doc.qt.io/qt-6/qmake-project-files.html

In manchen Projekten können Sie auch Dateien mit der Endung .pri finden. Diese Dateien enthalten Teilinhalte der Projektdateien und werden von den .pro-Dateien inkludiert: `include(teil.pri)`

1.4.3 Datei .ui

Diese Dateien, z. B. eine Datei *mainwindow.ui*, sind XML-Dateien, die vom Qt Designer erstellt und in das Projekt eingebunden werden. Wenn Sie also ein GUI mit dem Qt Designer erstellen, entsteht automatisch eine Datei .ui. Vom Projekt aus können Sie den Designer einfach durch Doppelklick auf diese Datei aufrufen.

Von solchen Dateien können sich auch mehrere in einem Projekt befinden, z. B. weil Sie ein Projekt mit mehreren Formularen benötigen und jedes Formular mit dem Designer erstellt wurde.

1.4.4 Datei ui_mainwindow.h

Diese Datei (ihr Name richtet sich nach dem gewählten Namen der für das GUI zuständigen Klasse, z. B. *ui_[Klassenname].h*) wird normalerweise vom Programmierer nicht bearbeitet. Sie befindet sich deshalb in einem speziellen Ordner. Bei Anlegen eines Projekts wird erst einmal ein Ordner mit dem Namen des Projekts angelegt. Darin befinden sich die Quelltextdateien, die Projektdatei, die .ui-Datei und eine Datei *CMakeLists.txt.user*.

Ein weiterer Ordner trägt bei unserem Beispiel den Namen *build-Test-Desktop_Qt_6_4_0_MinGW_64_bit-Debug* und enthält die ausführbare Datei (hier *Test.exe*).

In diesem build-Ordner ist in einem Unterordner *Test_autogen/include* auch die Datei *ui_mainwindow.h* enthalten.

Diese Datei wird auf der Grundlage der .ui-Datei vom *User Interface Compiler* (*uic*) generiert. Es ist sinnlos, in dieser Datei als Programmierer Änderungen vornehmen zu wollen, denn sie wird beim nächsten Aufruf dieses Compilers, also bei der nächsten Benutzung des Qt Designers, wieder überschrieben.

Sollten Sie in diese Datei einmal hineinschauen, ist zu sehen, dass auch Instanzen von `QMenuBar` und `QStatusBar` erstellt werden, obwohl Sie in unserem Projekt nicht verwendetet werden. Man könnte also in solchen Projekten (die beim Anlegen mit der Oberklasse `QMainWindow` erstellt wurden) sofort ein Menü erstellen oder die Statuszeile nutzen.

1.4.5 Datei main.cpp

Ein Beispiel zu unserem Test-Projekt sehen Sie hier:

```
01  #include "mainwindow.h"
02  #include <QApplication>
03
04  int main(int argc, char *argv[])
05  {
06      QApplication a(argc, argv);
07      MainWindow w;
08      w.show();
09      return a.exec();
10  }
```

Beim Start des Programms wird die main()-Funktion aufgerufen. Sie erstellt als Erstes eine Instanz der Klasse `QApplication` (Zeile 6). Damit werden die Grundlagen eines künftigen Programms gelegt. Als nächster Schritt folgt die Erstellung einer Instanz der Klasse `MainWindow`, also der von uns programmierten Klasse, die abgeleitet ist von `QMainWindow` (die wiederum über `QWidget` abgeleitet ist von `QObject` und uns somit auch die Funktionen `connect()` zur Verfügung stellt, die wir später häufig benutzen werden). Die Funktion `show()` aus `QWidget` (der Oberklasse von `QMainWindow`) zeigt das gewünschte Formular, und die Funktion `exec()` der Klasse `QApplication` startet einen *event loop*. Diese Funktion wird erst beendet, wenn das Formular geschlossen wird. Solange wird auch die main-Funktion nicht beendet, und das Formular wird angezeigt.

Eigentlich könnten Sie in dieser `main()`-Funktion durchaus Qt- und C++-Quelltext mischen, beispielsweise so:

```
01  #include <QApplication>
02  #include <iostream>
03  using namespace std;
04
05  int main(int argc, char *argv[])
06  {
07      QApplication a(argc, argv);
08
09      cout << "Hallo Qt-Buch...!";
10
11      return a.exec();
12  }
```

Das können Sie auch in einer bestehenden Qt-Widgets-Anwendung einmal ausprobieren, indem Sie die bestehende *main.cpp* durch diesen Quelltext ersetzen. Die Zeile 9 erzeugt jetzt eine Ausgabe im Qt Creator. Würden Sie eine reine Konsolenanwendung erstellen, ist diese Ausgabe auf der Konsole zu sehen.

Kommentarzeichen in Qt-Programmen

In allen Qt-Quelltexten gelten folgende Kommentarzeichen:

`//`	Kommentar bis Zeilenende
`/* ... */`	mehrzeiliger Kommentar
`/** ... */`	mehrzeiliger Dokumentationskommentar für Dokumentationstools wie Doxygen
`/*! ... */`	Dokumentationskommentar für *QDoc*, ein Qt-eigenes Dokumentationstool

In der Datei *CMakeLists.txt* gilt das Doppelkreuz # als Kommentarzeichen bis Zeilenende.

1.5 Der Qt Designer

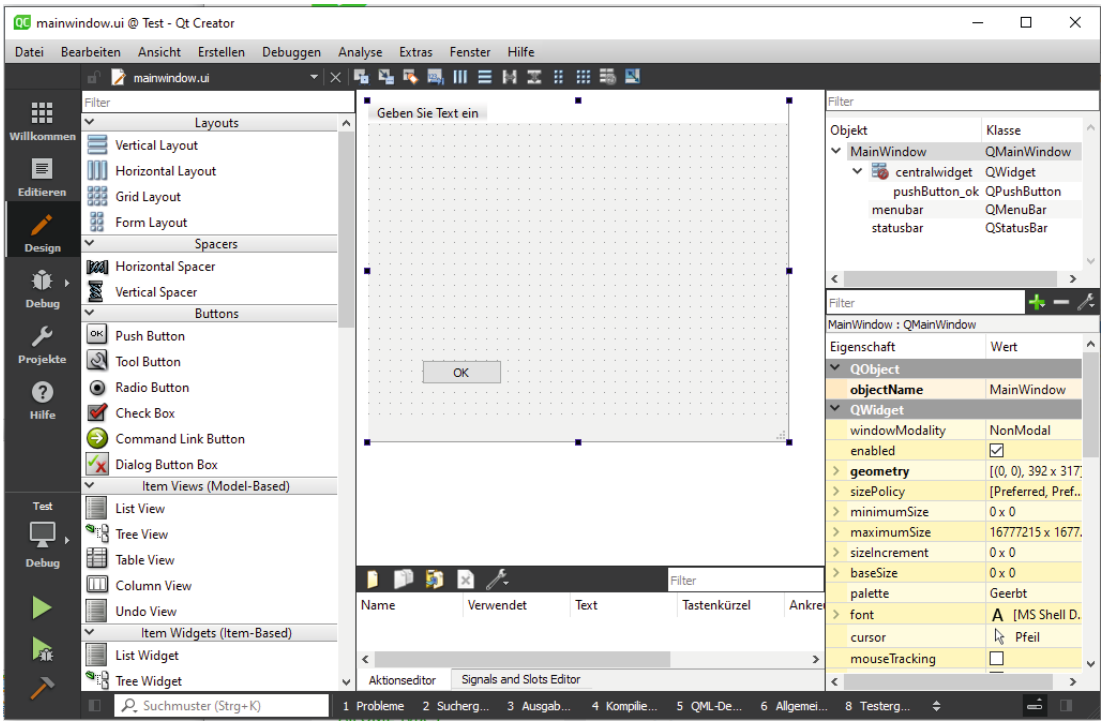

Bild 1.12 Qt Designer