Thema, Ziele: Qt IDE Qt-Creator, Layouts erstellen

Aufgabe 1: "Hello Qt-World" mit QLabel und HTML

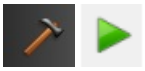
Im Unterverzeichnis *Vorlage\HtmlLabel* finden Sie ein "Hello Qt-World" GUI-Programm.

Öffnen Sie das entsprechende Qt Projekt und erweitern Sie die Vorlage gemäss den folgenden Schritten:

a) Überprüfen Sie mit Hilfe des File-Explorers welche Dateien das Verzeichnis "HtmlLabel" nun enthält. Führen Sie nun im Qt-Qreator den "Run qmake" aus dem "Build"-Menu aus. Überprüfen Sie nun welche Dateien entstanden sind. Welche Informationen enthalten diese?

   Lösung:

b) Kompilieren Sie dieses Projekt im Qt-Creator mit Hilfe des "Hammers" und führen Sie es anschliessend mit dem grünen Startknopf aus.



c) Untersuchen Sie, welche neuen Dateien und Verzeichnisse durch den "Build"-Vorgang entstanden sind.

   Lösung:

d) Studieren Sie das Source-Programm. Es läuft zwar, aber, ist es auch ein korrekt programmiertes C++ Programm?

   Lösung:

e) Modifizieren Sie das Programm so, dass anschliessend im Titelbalken Ihr Name steht.

f) Finden Sie heraus, von welchen Klassen *QLabel* abgeleitet (*derived*) ist. Beachten Sie auch Ober-Ober-Klassen etc. Dazu kann die in Qt integrierte Hilfe beigezogen werden. Klicken Sie irgendwo im Code wo *QLabel* steht darauf. Anschliessend drücken Sie F1. Nun sollte die Qt Dokumentation zu *QLabel* angezeigt werden.

Aufgabe 2: Manuelles Layout mit mehreren Widgets (ohne Layout Manager)

Im Unterverzeichnis *Vorlage\ManualLayout* finden Sie ein anderes "Hello Qt-World" GUI-Programm. Dieses besteht aus dem Hauptfenster 'wTop' und beinhaltet ein *QLabel* zur Anzeige des Textes "Hallo Qt-Welt".

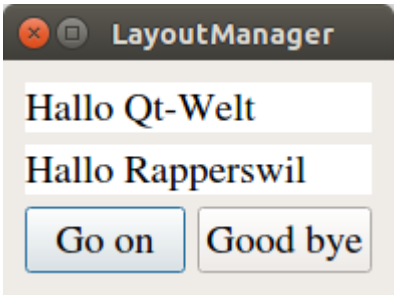
a) Erweitern Sie dieses Programm so, dass unterhalb des bestehenden QLabels ein neues Label mit dem Text "Hallo Rapperswil" zu stehen kommt.

b) Erweitern Sie Ihr Programm zusätzlich durch zwei "QPushButton"-Objekte, die zuunterst im Hauptfenster nebeneinanderliegen sollen. Der Button links soll mit "Go on", der rechts mit "Good bye" beschriftet sein.

Aufgabe 3: Layout Manager mit mehreren Widgets

Im Unterverzeichnis *Vorlage\LayoutManager* finden Sie ein weiteres "Hello Qt-World" GUI-Programm.

Erstellen Sie ein Layout mit den gleichen Widgets und derselben Anordnung wie bei Aufgabe 2 mit Hilfe von Layout Managern. Die Abbildung rechts zeigt wie das Layout schlussendlich aussehen sollte.



Aufgabe 4: Layout mit Qt GUI Creator erstellen

Erstellen Sie das gleiche Layout wie bei Aufgabe 3 mit Hilfe des QtCreators. Erstellen Sie dazu ein neues "Qt Widgets Application"-Projekt mit dem Namen QtCreator (Abbildung 2). Das Top-Widget soll ein QWidget sein und die dazugehörige Klasse soll `Widget` heissen (Abbildung 4) . Das fertige Layout könnte wie Abbildung 3 aussehen.
Nun sollte das Projektverzeichnis wie Abbildung 5 aussehen. Das Aussehen der grafischen Oberfläche kann nun im GUI Creator spezifiziert werden. Dazu kann ein Doppelklick auf Widget.ui durchgeführt werden, worauf sich der GUI Creator öffnet.
Das Hauptlayout des Top-Widgets kann mittels rechtsklick auf Widget (rot eingekreist) → Layout ausgewählt werden. Das funktioniert aber erst, wenn mindestens ein Child Widget bereits hinzugefügt wurde.

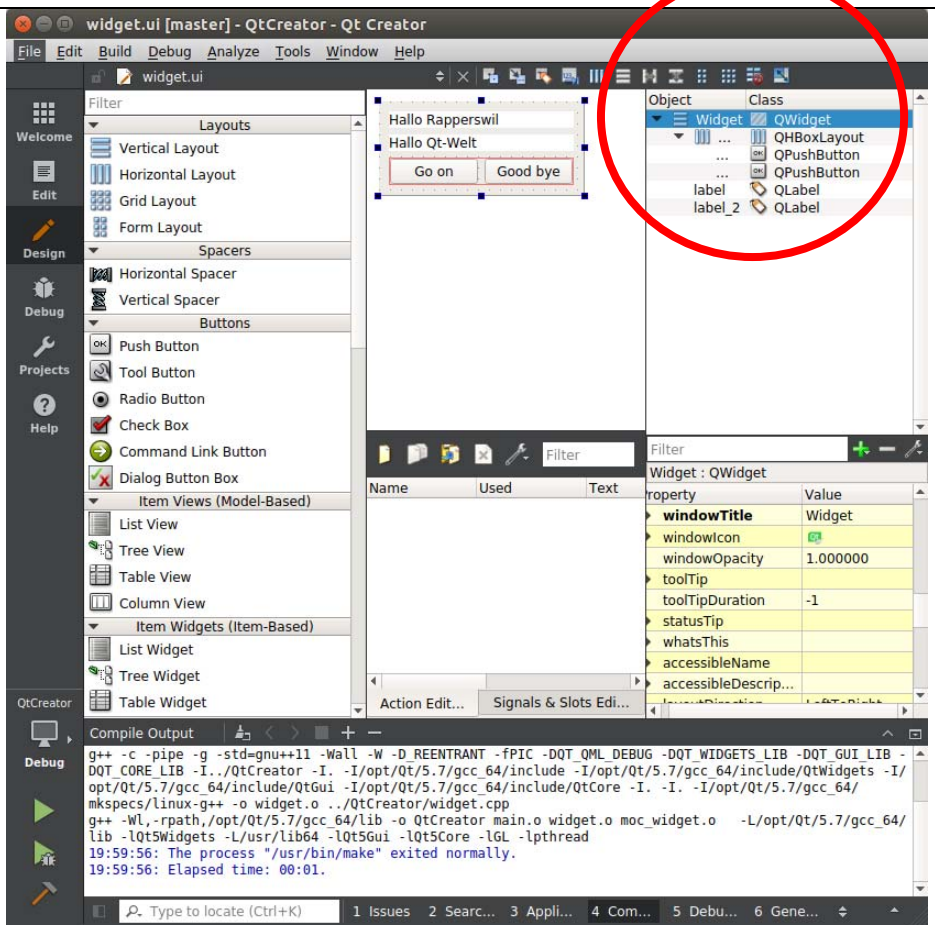


Abbildung 1 Einfügen eines Hauptlayouts

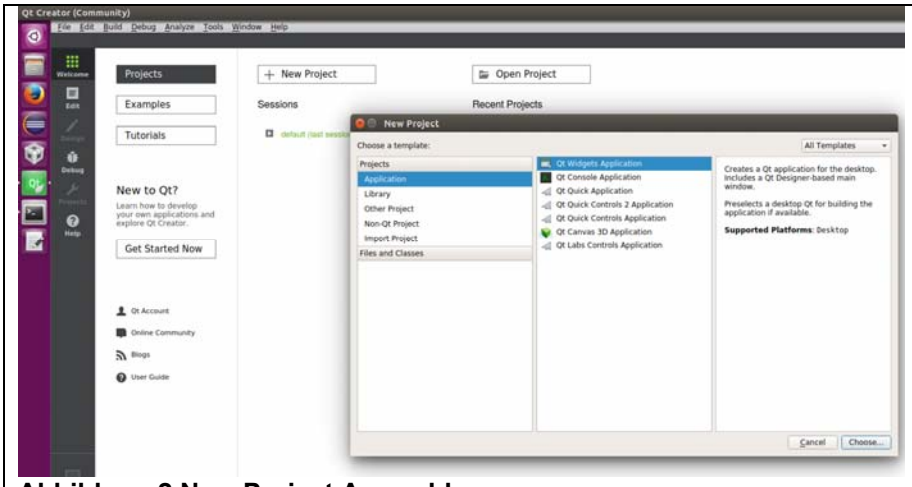


Abbildung 2 New Project Auswahl

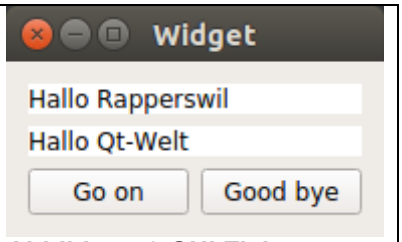


Abbildung 3 GUI Ziel

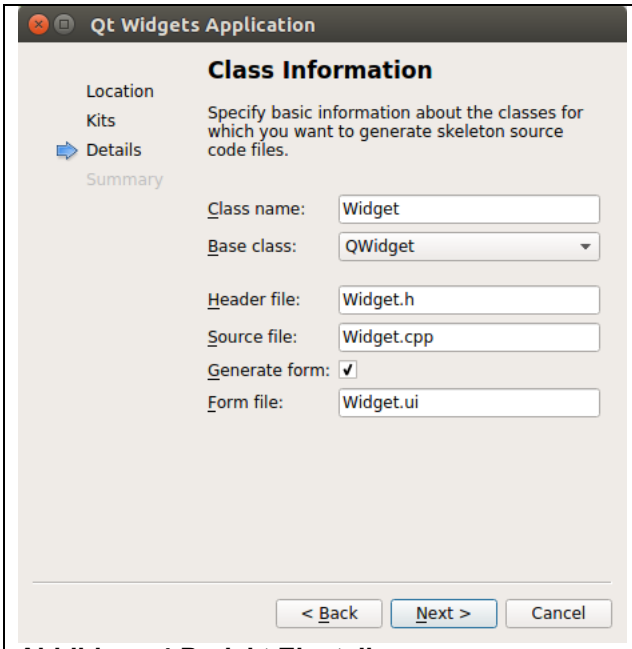


Abbildung 4 Projekt Einstellungen

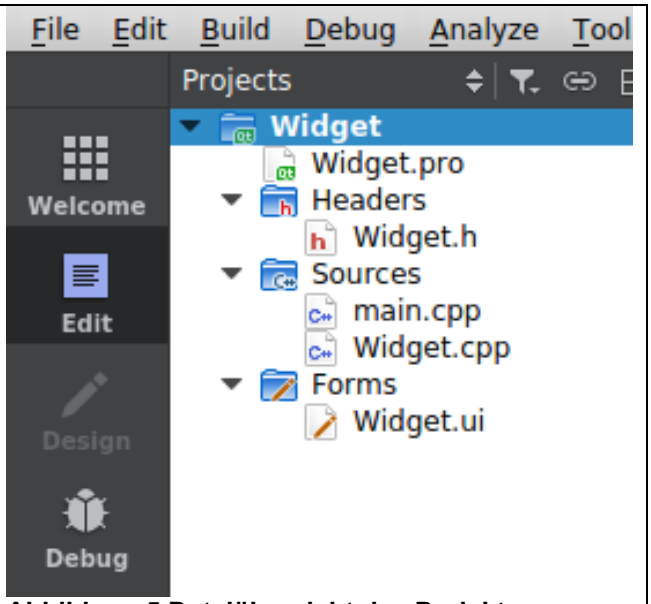


Abbildung 5 Dateiübersicht des Projekts

Zusätzliche Hinweise zur Qt-Programmierumgebung (Qt Creator)

Verwendete Projektorganisation (Shadow-Building)

Mit Hilfe der Qt-Library ist es bekanntlich möglich, ausgehend von den gleichen Source-Files, ausführbare Programme (sogenannte "Builds") für unterschiedliche Konfigurationen (Plattformen, Compiler etc.) zu erstellen.

Dazu legt der Qt-Creator für jede Konfiguration (Plattform, Compiler etc.) ein eigenes Build-Verzeichnis an, das die konfigurationsspezifischen Daten wie makefiles, object-files etc. enthält. Diese Aufteilung hat den Vorteil, dass die plattformspezifischen Dateien völlig getrennt sind von den (gemeinsamen) Source-Dateien. Ausserdem werden dadurch die verschiedenen "Builds" (Plattformen, Compiler etc.) sauber voneinander getrennt. (Dieser Mechanismus wird als "Shadow-Building" bezeichnet.)

Der Name eines Build-Verzeichnisses wird dabei aus dem Namen der Projektdatei (Bsp. "Counter1.pro") und der verwendeten Konfiguration (Plattform, Compiler etc.) gebildet. Das führt dann zum Beispiel zu einem Namen wie "build-Counter1-Desktop-Qt_5_4_1_ MinGW_32bit-Debug" für das entsprechende Build-Verzeichnis.