Getting Started with Qt 5

Introduction to programming Qt 5 for cross-platform application development

Packt
www.packt.com

Benjamin Baka

Getting Started with Qt 5

Introduction to programming Qt 5 for cross-platform application development

Benjamin Baka

Packt

BIRMINGHAM - MUMBAI

Getting Started with Qt 5

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Pavan Ramchandani
Acquisition Editor: Siddharth Mandal
Content Development Editor: Mohammed Yusuf Imaratwale
Technical Editor: Surabhi Kulkarni
Copy Editor: Safis Editing
Project Coordinator: Pragati Shukla
Proofreader: Safis Editing
Indexer: Tejal Daruwale Soni
Graphics: Alishon Mendonsa
Production Coordinator: Jisha Chirayil

First published: February 2019

Production reference: 1280219

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78995-603-0

`www.packtpub.com`

Contributors

About the author

Benjamin Baka is a full-stack software developer and is passionate about cutting-edge technologies and elegant programming techniques. He has 10 years in different technologies, from C++, Java, Ruby, Python to Qt. Some of the projects he's working on can be found on his GitHub page. He is currently working on exciting technologies all from the camp of mPedigree Network.

> *I'd like to thank the Baka family for all of their support in my many endeavors. Another thanks to Samuel Afari for encouraging me to go beyond my limits.*
>
> *To the entire Packt team, it's been a pleasure working with you.*
>
> *And I continue to stay forever indebted to Guido Sohne and Lorenzo Cabrini for their amazing guidance and input in my life.*

About the reviewer

Nibedit Dey is a techno-entrepreneur and innovator with over 8 years of experience in building complex software-based products using Qt and C++. Before starting his entrepreneurial journey, he worked for L&T and Tektronix in different research and development roles. Additionally, he has reviewed *The Modern C++ Challenge*, *Hands-on GUI programming with C++ and Qt5*, and *Hands-On High Performance Programming with Qt 5 books* for Packt.

> *I would like to thank the online programming communities, bloggers, and my peers from earlier organizations, from whom I have learned a lot over the years.*

Packt is searching for authors like you

Table of Contents

Preface

There are many buzzwords in computing today, most of which revolve around various software technologies and concepts. Browsers have become the preferred means of accessing information and consuming all manner of data. But there is still a void that can only be filled by standalone applications that must be installed and run on an operating system. The browser itself as an application cannot be accessed through a browser and bears witness to this assertion.

Applications such as VLC, Adobe Photoshop, Google Earth, and QGIS are a few examples of applications that run directly on an operating system. Interestingly enough, these well-known software brands are built with Qt.

Qt (pronounced "cute") is a cross-platform application framework and widget toolkit that is used in creating graphical user interface applications that run on a number of different hardware and operating systems. The aforementioned applications were written using this same toolkit.

The main aim of this book is to introduce Qt to the reader. Through the use of simple and easy-to-understand examples, it will walk the user from one concept to the next without focusing too much on theory. The size of the book requires us to be concise in our presentation of materials. Coupled with the ample examples presented, we hope to shorten the path to understanding and learning how to use Qt.

Who this book is for

Anyone looking to embark on the development of graphical user interface applications will find this book useful. No prior exposure to other toolkits is required in order to understand this book. However, having such skills will prove useful.

The book does, however, presume that you have a working knowledge in the use of C++. If you can express your thoughts in developing algorithms and the use of object-oriented programming, you will find the content easy to consume.

Expert or intermediate persons with Qt knowledge should seek more detailed materials that are available out there. This book is not a reference guide, and should only be used as introductory material.

What this book covers

`Chapter 1`, *Introducing Qt 5*, walks you through the process of getting your machine ready to starting writing and running Qt programs. The chapter ends by introducing the hello world program in Qt, exploring the general structure and compilation process of a Qt program.

`Chapter 2`, *Creating Widgets and Layouts*, covers GUI components that are usually present in most applications and how they are created in Qt. The chapter then ends by detailing how to use layouts with widgets.

`Chapter 3`, *Working with Signals and Slots*, introduces one of the most important concepts to grasp in Qt, signals and slots. It demonstrates to the reader how to make an application trigger and respond to actions.

`Chapter 4`, *Implementing Windows and Dialog*, brings the reader closer to how a real-world Qt program should be written. It illustrates how to use and organize a program using classes, windows, and dialog boxes.

`Chapter 5`, *Managing Events, Custom Signals, and Slots*, explores the writing of custom signals and slots and introduces the topic of Events in Qt.

`Chapter 6`, *Connecting Qt with Databases*, deals with how to write applications that connect to a database and how to also present the data in visual form.

To get the most out of this book

The beginning of each chapter will begin with a little theory that should help consolidate your understanding. Thereafter, a series of examples are used to explain the concepts and to help the reader grasp the topic better.

This book also avoids continuing with examples from previous chapters. Each chapter's examples are short and do not require the reader to have knowledge of previous chapters. That way, you can pick any chapter you link and work through it.

Appropriate links to set up the environment on Windows have been provided. Linux and macOS platforms have been catered for directly in this book.

Download the example code files

You can download the example code files for this book from your account at `www.packt.com`. If you purchased this book elsewhere, you can visit `www.packt.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packt.com`.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Getting-Started-with-Qt-5`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `https://www.packtpub.com/sites/default/files/downloads/9781789956030_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "To set the password to the `connection` parameter, the code fragment, `db_conn.setPassword("")`, is issued."

A block of code is set as follows:

```
QSqlDatabase db_conn =
        QSqlDatabase::addDatabase("QMYSQL", "contact_db");

db_conn.setHostName("127.0.0.1");
db_conn.setDatabaseName("contact_db");
db_conn.setUserName("root");
db_conn.setPassword("");
db_conn.setPort(3306);
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
[default]
exten => s,1,Dial(Zap/1|30)
exten => s,2,Voicemail(u100)
exten => s,102,Voicemail(b100)
exten => i,1,Voicemail(s0)
```

Any command-line input or output is written as follows:

```
% mkdir helloWorld
% ./run_executable
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "It displays the text **Hello world !** in a label."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packt.com/submit-errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

1
Introducing Qt 5

Qt gives developers a great toolbox with which to create fantastic and practical applications with minimal stress, as you will soon discover. In this chapter, we will introduce Qt and describe how to set it up on a machine. By the end of the chapter, you should be able to do the following:

- Install Qt
- Write a simple program in Qt
- Compile and run a Qt program

The objectives have been kept simple and straightforward. So let's get started!

Installing Qt on Linux

The Ubuntu operating system makes it reasonably easy to install Qt 5. Issue the following commands to set up your box:

```
sudo apt-get install qt5-default
```

After the installation, Qt programs will be compiled and run from the command line. In `Chapter 6`, *Connecting Qt with Databases*, we will illustrate how to connect to the database using Qt. Issue the following command to ensure that the relevant libraries are installed for Qt to work with. The database that will'll connect to is MySQL:

```
sudo apt-get install libqt5sql5-mysql
```

Installing Qt on macOS

There are a variety of ways to get Qt installed on a Mac. To begin the process of installing Qt 5 on your Mac, you need to get Xcode installed on your machine. Issue the following commands on the Terminal:

```
xcode-select --install
```

If you get the following output, then you are ready for the next series of steps:

```
xcode-select: error: command line tools are already installed, use
"Software Update" to install updates
```

HomeBrew is a package management software tool that allows you to easily install Unix tools that don't come shipped with the macOS.

If you don't already have it on your machine, you can install it by issuing the following command in a Terminal:

```
 /user/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/
Homebrew/install/master/install)"
```

After that, you should issue yet another set of commands to get Qt installed via the Terminal:

```
curl -O
https://raw.githubusercontent.com/Homebrew/homebrew-core/fdfc724dd532345f5c
6cdf47dc43e99654e6a5fd/Formula/qt5.rb

brew install ./qt5.rb
```

In a few chapters' time, we will be working with the MySql database. To configure Qt 5 with MySql, issue the following command:

```
brew install ./qt5 --with-mysql
```

This command should take a while to complete and, assuming nothing goes wrong, you are ready to write Qt programs.

Installation on Windows

For readers using Windows, installation remains simple, albeit a little less straightforward. We can start by heading over to `http://download.qt.io`.

Select `official_releases/`, then `online_installers/`, and opt to download `qt-unified-windows-x86-online.exe`.

Run the program and opt to create an account. Click through to select the installation folder and don't forget to select the **MinGW 5.3.0 32 bit** option as the compiler when selecting the components that need to be installed.

Most of the commands in this book should run in this IDE.

What is Qt?

Now that we have set up our boxes to start development, let's put together a hello world example. First, however, let's take a brief detour.

Qt is a toolkit for creating **Graphical User Interfaces** (**GUI**), as well as cross-platform applications. GUI applications are programs that employ the use of the mouse to issue commands to the computer for execution. Though Qt can, in some cases, be used without necessarily making use of this, therein lies its utility.

The difficulty in trying to produce the same look, feel, and functionality across multiple operating systems is one big hurdle you have to deal with when writing GUI applications. Qt completely does away with this impediment by providing a means to write code only once and ensuring that it runs on most operating systems without requiring much or any change.

Qt makes use of some modules. These modules group related functionalities together. The following lists some modules and what they do:

- `QtCore`: As the name implies, these modules contains core and important classes for the Qt framework. These include containers, events, and thread management, among others.
- `QtWidgets` and `QtGui`: This module contains classes for calling widgets. Widgets are the components that make up the majority of a graphical interface. These include buttons, textboxes, and labels.
- `QtWebkit`: This module makes it possible to use web pages and apps within a Qt application.
- `QtNetwork`: This module provides classes to connect to and communicate with network resources.

- QtXML: For parsing XML documents, this module contains useful classes.
- QtSQL: This module is feature-rich with classes and drivers that allow for connecting to databases, including My SQL, PostgreSQL, and SQLite.

Hello world in Qt

In this section, we will put together a very simple hello world program. The program will show a simple button within a window. Create a file called hello.cpp in a newly created folder called hello_world. Open the file and insert the code:

```
#include <QApplication>
#include <QLabel>
int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   QLabel label("Hello world !");
   label.show();
   return app.exec();
}
```

This looks like a regular C++ program, with the exception of unfamiliar classes being used.

Like any regular program, the int main() function is the entry point of our application.

An instance of the QApplication class is created, called app, and the arguments passed to the main() function. The app object is required because it sets off the Event loop that continues to run until we close the application. Without the QApplication object, you cannot really create a Qt GUI application.

However, it is possible to use certain aspects of Qt without the need to create an instance of QApplication.

Also, the constructor for QApplication requires that we pass the argc and argv to it.

We instantiate an object of the QLabel class, label. We pass the "Hello World!" string to its constructor. A QLabel represents what we call a widget, which is a term used to describe visual elements on the screen. Labels are used to hold text for display.

By default, created widgets are hidden. To display them, a call to the show() function has to be made.

To start the `Event` loop, the `app.exec()` line is executed. This passes control of the application to Qt.

The `return` keyword will pass an integer back to the operating system, indicating the state of the application when it was closed or exited.

To compile and run our program, navigate to the folder where `hello.cpp` is stored. Type the following command in the Terminal:

```
% qmake -project
```

This will create the `hello_world.pro` file. The name `hello_world` is the name of the folder where `hello.cpp` is located. The generated file should change, depending on the location you stored the `hello.cpp` file.

Open the `hello_world.pro` file with any text editor of your choice. The following lines deserve some explanation:

```
TEMPLATE = app
```

The value, `app`, here means that the final output of the project will be an application. Alternatively, it could be a library or sub-directory:

```
TARGET = hello_world
```

The name, `hello_world`, here is the name of the application or (library) that will be executed:

```
SOURCES += hello.cpp
```

Since `hello.cpp` is the only source file in our project, it is added to the `SOURCES` variable.

We need to generate a `Makefile` that will detail the steps needed to compile our hello world program. The benefit of this autogenerated `Makefile` is that it takes away the need for us to know the various nuances involved in compiling the program on the different operating systems.

While in the same project directory, issue the following command:

```
% qmake
```

This generates a `Makefile` in the directory.

Now, issue the following command to compile the program:

```
% make
```

The following error will be produced (along with further information) as the output from running the `make` command:

```
#include <QApplication>
         ^~~~~~~~~~~~
```

Earlier on, we mentioned that various components and classes are packaged into modules. The `QApplication` is being utilized in our application, but the correct module has not been included. During compilation, this omission results in an error.

To fix this issue, open the `hello_world.pro` file and insert the following lines after the line:

```
INCLUDEPATH += .
QT += widgets
```

This will add the `QtWidget` module, along with the `QtCore` modules, to the compiled program. With the correct module added, run the `make` command again on the command line:

```
% make
```

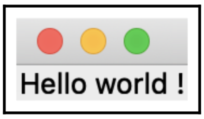
A `hello_world` file will be generated in the same folder. Run this file from the command line as follows:

```
% ./hello_world
```

On a macOS, the full path to the executable will be specified with the following path from the command line:

```
./hello_world.app/Contents/MacOS/hello_world
```

This should produce the following output:



Hello world !

Well, there is our first GUI program. It displays the **Hello world !** in a label. To close the application, click on the **Close** button of the window.

Let's add a dash of **Qt Style Sheet** (**QSS**) to give our label a little effect!

Modify the `hello.cpp` file as follows:

```
#include <QApplication>
#include <QLabel>
int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   QLabel label("Hello world !");
   label.setStyleSheet("QLabel:hover { color: rgb(60, 179, 113)}");
   label.show();
   return app.exec();
}
```

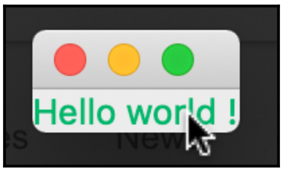
The only change here is `label.setStyleSheet("QLabel:hover { color: rgb(60, 179, 113)}");`.

A QSS rule is passed as an argument to the `setStyleSheet` method on the `label` object. The rule sets every label within our application to show the color green when the cursor hovers over it.

Run the following commands to recompile the application and run it:

```
% make
% ./hello_world
```

The program should appear as in the following screenshot. The label turns green when the mouse is placed over it:



Summary

This chapter laid the foundation for getting to know Qt and what it can be used for. Installing of Qt on macOS and Linux was outlined. A small hello world application was written and compiled, all from the command line, without any need for an IDE. This meant that we were also introduced to the various steps that lead to the final program.

Finally, the hello world application was modified to employ QSS in a bid to show what other things can be done to a widget.

In `Chapter 2`, *Creating Widgets and Layouts*, we will explore more widgets in Qt and how to organize and group them.