AMBIENT-IT

+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 24/01/2024

Sign up

Qt 6 training

4 days (28 hours)

Presentation

Qt is THE industrial framework for cross-platform applications on desktops, embedded systems and cell phones. Supported platforms include Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS and many more!

Qt is not just a programming language. It's a framework written in C++. A preprocessor, the MOC (Meta-Object Compiler), is used to extend the C++ language with features such as signals and slots. Prior to compilation, the MOC analyzes source files written in Qt extended C++ and generates standards-compliant C++ sources from them. So your Apps can be compiled by any standards-compliant C++ compiler such as Clang, GCC, ICC, MinGW and MSVC.

The creation of a human-machine interface is a major challenge for any application. Many solutions exist, but many are laborious to implement, not very ergonomic and difficult to maintain.

When it comes to creating applications in C++, the most common and effective solution is to use Qt6. This course will teach you how to master this powerful tool and create user interfaces. As with all our training courses, we'll introduce you to the latest version.

Qt6 lets you create applications compatible with Windows, Mac OS and Linux, using the graphical elements of all these systems. You'll learn how to create pages with buttons connected to your code. You'll also learn how to draw your application using the QtCreator graphical editor, which can be used with Java and Python.

As with all our training courses, this one will introduce you to the latest version of Qt, namely Qt 6.

Objectives

- Discover the power of the Qt6 Framework
- How to use the QtCreator graphic designer
- How to develop a new application
- Learn the basics of QML
- Understanding the Framework's advanced mechanisms

Target audience

- Application developer

Prerequisites

- Basic knowledge of C++ (object and pointer concepts)

Technical requirements

- Have Qt6 installed
- An editor like QT creator
- A recent compiler

Qt 6 training program

Day 1 - Reinforcing C++ / Qt / OOP basics

C++ and Qt today

- C++ evolution from 2011 to today
- Qt evolution from version 6.0 to the present day
- Qt Widgets vs. QML
- A word about Object-

Oriented Programming

licenses

- Reviewing the fundamentals
- Main Design Patterns... and anti-patterns!
- Introduction to SOLID principles

Modern C++ fundamentals

- Value semantics vs. reference semantics

- Robust, automated resource management
- Object-oriented programming

without inheritance Essential Qt

concepts

- MVC (Model-View-Controller)
- C++ extensions thanks to MOC
- Qt's specific object model
- Advanced signal/slot functionalities
- Divergences and pitfalls between traditional C++ and Qt

Day 2 - Discovering QML / QtQuick

Introduction to QML / QtQuick

- Language syntax and main concepts
- Key differences from widgets
- Introduction to the main text and graphics components
- Added TableView support for hiding rows and columns Discover

QML with QtCreator

- Creating reusable components
- Positioning elements
- How property binding works
- Practical work: Creating a QML application with

QtCreator Interacting with the user

- Mouse and multi-touch screen management
- Keyboard management
- Animation and state transitions

Presenting complex data

- Repeater and Delegate
- Using a template
- Using a model proxy

Day 3 - Going deeper into Qt / QML

Advanced notions

- Internal workings of the QML engine
- Using a loader
- Dynamic component creation C++ /

QML interactions

- Linking C++ code to QML presentation layers
- Writing a QML extension in C++
- Managing the life cycle of C++ objects

exposed to QML State machine and Qt

- Principle and benefits
- How QStateMachine works
- Qt SCXML module

QML in a real application

- Translate your interfaces (Internationalization)
- Apply custom themes/styles
- Optimizing and debugging your code
- Best practices and pitfalls to avoid

Day 4 - Architecting and developing a complex project

Being efficient with QMake

- Organizing your project into modules
- Support for multiple platforms and compilation modes
- Integration of an external library (Qwt)
- Unit testing and continuous

integration Multitasking and

asynchronous programming

- Parallelism vs. competition
- Synchronous vs. asynchronous programming with Qt
- A modern approach to parallelism with Qt

Plugin development with Qt

- The Qt plug-in system
- Technical constraints: ABI and binary compatibility
- Out-of-process plugins: Qt's IPC mechanisms

- Plugins vs. scripting (scalability)

Tools and conclusion

- Tools and other useful resources that could not be presented
- Tips and tricks in bulk
- Questions and answers on all topics covered
- Conclusion

New functions

- Qt GUI
  - Add QImage:convertTo new API
  - QPainterPath: supports clear, reserve and capacity methods with the same semantics as QVector::clear() (allocations are preserved)
- Qt Network
  - Windows: Support for Secure Channel for SSL connection
  - OCSP stapling support
- Qt QML
  - Improved support for enumerations declared in C++.
  - JavaScript "null" binding value is now optimized at compile time.
  - QML now generates function tables on 64-bit windows, enabling the stack to be unrolled via JITed functions (QTBUG-50061).
- Qt Quick
  - Added TableView support for hiding rows and columns
- Qt Quick 2 controls
  - Add SplitView
  - Add a cache property to the icon
- Qt Bluetooth
- Qt 3D
  - Added support for importing and exporting OpenGL texture handles
  - Added framegraph nodes for fence objects
  - Addition of a priority-based levy
  - Initial support for glTF 2.0 scene import added
- Composer Qt Wayland
- Qt WebEngine
  - Application-local client certificate store
  - QML support for customer certificates
  - PDF display via internal Chromium extension
  - Web Notifications API
  - Thread- and page-specific url request interceptors.
- Qt WebSockets
- Qt Location
  - Added support for GeoPolygons with holes, reflected in MapPolygons and MapPolygonObjects.
  - Introduction of interoperability with GeoJson with import/export functionality
- Test Qt

- Qt Multimedia
  - Seamless playback in QML VideoOutput using the flushMode property
  - GStreamer support added for Windows/macOS
  - Adding HTTP headers and audio roles for Android
  - Add QT_MULTIMEDIA_PREFERRED_PLUGINS to specify preferred plugins
  - Allows WMF to be built with DirectShow or disabled by configuration option -no-wmf or -no-directshow
  - Introduction of QT_PA_CHANNEL_MAP for QAudioOutput and PulseAudio
  - Video/audio probes in DirectShow
  - QMedia The QMediaResource class is now obsolete
- Wayland
  - New Shell integration for fullscreen-shell-unstable-v1.
- Qt Lottie (TP)
- Qt for WebAssembly
- Qt for automation
- Qt KNX
  - Secure customer API added
- Qt OPC UA
  - API C++ in TP
  - Add QML API (TP)
  - Adding a secure C++ client API (TP)
  - UaCpp and Open62542 are identical in terms of functionality
- Qt CoAP (TP)
- Obsolete modules
  - Qt script
  - Qt 1 quick controls
  - Qt XmlModel XmlListModel

Additional modules (optional) - 2 additional days

- Database (SQLite)
- Serialization with QStream
- Networking with QNetwork
- Responsive design with QML

Companies concerned

This training course is aimed at both individuals and companies, large or small, wishing to train their teams in a new advanced computer technology, or to acquire specific business knowledge or modern methods.

Positioning on entry to training

Positioning at the start of training complies with Qualiopi quality criteria. As soon as registration is finalized, the learner receives a self-assessment questionnaire which enables us to assess his or her estimated level of proficiency in different types of technology, as well as his or her expectations and personal objectives for the training to come, within the limits imposed by the selected format. This questionnaire also enables us to anticipate any connection or security difficulties within the company (intra-company or virtual classroom) which could be problematic for the follow-up and smooth running of the training session.

Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

Organization

The course alternates theoretical input from the trainer, supported by examples, with brainstorming sessions and group work.

Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

Sanction

A certificate will be issued to each trainee who completes the course.