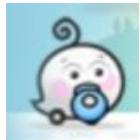


Programmation C++ avec QT (initiation)



Facile



Normal



Difficile



Professionnel



Expert

https://wiki.waze.com/wiki/Your_Rank_and_Points



- 1 - Présentation de la Framework QT5
- 2 - Environnement de développement
- 3 - Les interfaces graphiques : QWidget
- 4 - Les interfaces graphiques : Quick
- 5 - Les interfaces graphiques : Quick & QWidget & C++



- 1.1 – Présentation de QT
- 1.2 - Historique
- 1.3 - Architecture générale
- 1.4 - Documentation



1.1 – Présentation de QT



Qt est un environnement de développement complet comprenant :

- Une boîte à **outils graphique** puissante et multiplateformes (widgets) en C++ à la base. Mais également extension Qt Quick basée sur le langage déclaratif QML
 - Java : **Qt Jambi**
 - Python : **PyQt, PySide**
 - C# : **Qyoto**
- Un ensemble de **bibliothèques** utilitaires et d'extensions : accès aux données, connexions réseaux, gestion du multitâche, XML, etc.
- Un **framework**, car il impose une norme dans la façon de concevoir son programme et de le développer. Qt introduit de nouveaux mots-clés et opérateurs et un formalisme dans les fichiers d'en-têtes.
- Utilisé dans de nombreux produits libres (KDE...) ou propriétaires
- **Multi-plateformes**
 - principaux OS : Windows, Mac OS X, Linux/X11
 - plateformes mobiles : iOS, Android, WinRT

<https://fr.wikipedia.org/wiki/Qt>



Développements et licences

- 1988 : Haavard & Eirik créent une librairie graphique orientée objet
- 1993 : Le noyau est terminé et pour objectif « The world's best C++ GUI framework »
- 2001 : Qt 3.0, développé par TrollTech, 500 000 lignes de codes, Linux, Windows, Mac
- 2008 : Qt 4.5 (racheté par Nokia, 250 employés)
- 2009 : Qt 4.6 : animation, GraphicScene, machine à état
- 2012 : Qt Development Frameworks, filiale de Digia
- fin 2012 : Qt 5.0
- 2014, : Digia crée The Qt Company, une filiale dédiée au développement et à la gestion de Qt.

Licences LGPL (gratuites) et commerciales

<https://fr.wikipedia.org/wiki/Qt>



- Les licences Qt
- Qt Essentials
- Qt Add-Ons
- Qt tools
- Qt Scene graph
- Qt WebEngine
- Qt 3D
- Qt Platform Abstraction
- Qt Quick
- Structure



1.3 - Architecture générale : Les licences Qt

Add-Ons (GPLv3+ or LGPLv3)

Canvas 3D	Charts	Qt Quick 2D renderer	Data Visualization	Purchasing
Active Qt	Graphical Effects	NFC	Location	Qt 3D
X11, Windows, Mac Extras	Print Support	Sensors	Concurrent	WebEngine
Android Extras	Image Formats	Positioning	Serial Port	WebSockets
XML & XML Patterns	SVG	Bluetooth	D-Bus	WebChannel

Essentials (GPLv2+/LGPLv3)

GUI	Widgets	Multimedia Widgets	Quick Dialogs	Quick Controls
Core	Network	Multimedia	Quick Layouts	Quick
		SQL	Test	QML

Desktop & mobile platforms (GPLv2+/LGPLv3)

Windows	Mac	Linux Desktop	Android	iOS	WinRT
---------	-----	---------------	---------	-----	-------

Development Tools (GPLv3)

Qt Creator Cross-platform IDE	CPU usage analyzer
Qt Designer GUI Designer	GPU Profiler
Qt Linguist I18N Toolset	Clang static analyzer
Qt Assistant Documentation Tool	Qt Quick Compiler
Moc, uic, rcc Build Tools	Qt Quick Profiler
Qmake Cross-platform Build Tool	Autotest integration

LGPLv2.1

LGPLv3

GPLv3

GPLv3,
*new modules for OSS*Commercial,
unavailable for OSS

<https://www.ics.com/blog/changes-qt-licensing>

1.3 - Architecture générale : Qt Essentials



Les éléments essentiels de Qt définissent la base de Qt sur toutes les plates-formes

Module	La description
Qt Core	Principales classes non graphiques utilisées par d'autres modules.
Qt GUI	Les classes de base pour les composants de l'interface utilisateur graphique (GUI). Comprend OpenGL.
Qt Multimedia	Classes pour les fonctions audio, vidéo, radio et caméra.
Qt Widgets Multimédia	Cours basés sur Widget pour la mise en œuvre de fonctionnalités multimédia.
Qt Network	Les cours facilitent la programmation du réseau et sont plus portables.
Qt QML	Classes pour les langages QML et JavaScript.
Qt Quick	Un cadre déclaratif pour créer des applications hautement dynamiques avec des interfaces utilisateur personnalisées.
Qt Quick Controls	Réutilisable Qt Commandes d'UI basées sur Quick pour créer des interfaces utilisateur classiques de style bureautique.
Qt Dialogues rapides	Types pour créer et interagir avec des boîtes de dialogue système à partir d'une application Qt Quick.
Qt Quick Layouts	Les mises en page sont des éléments qui permettent d'organiser les éléments basés sur Qt Quick 2 dans l'interface utilisateur.
Qt SQL	Classes pour l'intégration de bases de données à l'aide de SQL.
Qt Test	Classes pour l'unité testant les applications Qt et les bibliothèques.
Qt Widgets	Classes pour étendre l'interface graphique Qt avec les widgets C ++.

1.3 - Architecture générale : Qt Add-Ons



Les modules complémentaires Qt Add-ons contiennent des modules additionnels, monos ou multiplateformes, répondant à des besoins spécifiques.

Module	Plateformes de développement	Plateformes ciblées	La description
Active Qt	Windows		Classes pour les applications qui utilisent ActiveX et COM
Qt 3D	Tout		Fonctionnalités pour les systèmes de simulation en temps réel avec prise en charge du rendu 2D et 3D.
Enginio (Déprécié)	Tout	Tout	Une solution Backend-as-a-Service pour faciliter le développement du backend pour les applications connectées et basées sur les données.
Qt Extras Android	Tout	Android	Fournit des API spécifiques à la plateforme pour Android.
Qt Bluetooth	Tout	Android , iOS , Linux et MacOS	Fournit un accès au matériel Bluetooth.
Qt Canvas 3D	Tout		Active les appels de dessin 3D OpenGL tels que les applications Qt Quick utilisant JavaScript.
Qt Concurrent			Classes pour l'écriture de programmes multi-thread sans utiliser de primitives de filetage de bas niveau.
Q-D-Bus	Tout		Classes pour la communication interprocessus sur le protocole D-Bus.
Qt Gamepad	Tout	Android , iOS , macOS , tvOS (y compris la télécommande tvOS), Linux , Windows	Permet aux applications Qt de prendre en charge l'utilisation du matériel du gamepad.
Qt effets graphiques	Tout		Effets graphiques à utiliser avec Qt Quick 2.
Formats d'image Qt	Tout		Plugins pour les formats d'image supplémentaires: TIFF, MNG, TGA, WBMP.

1.3 - Architecture générale : Qt Add-Ons



Module	Plateformes de développement	Plateformes ciblées	La description
Qt Lieu	Tout	Tout	Affiche la carte, la navigation et place le contenu dans une application QML.
Qt Mac Extras	Tout	MacOS	Fournit des API spécifiques à la plate-forme pour MacOS.
Qt NFC	Tout	Android et Linux	Fournit un accès au matériel de communication Near Field (NFC).
Qt OpenGL (Déconseillé)			Cours de support OpenGL. Déconseillé en faveur des classes QOpenGL* dans le module QOpenGL* GUI .
En-têtes de plate-forme Qt			Fournit des classes qui encapsulent des informations spécifiques à la plateforme, liées à une configuration d'exécution donnée d'un plugin de plateforme.
Qt Positionnement	Tout	Android , iOS , MacOS , Linux , WinRT .	Fournit l'accès aux cours de surveillance de position, satellite et zone.
Qt Support d'impression	Tout		Classes pour faciliter l'impression et plus portables.
Qt Achats	Tout	Android , iOS et MacOS.	Permet l'achat dans l'application de produits dans les applications Qt.
Qt Quick Controls 2	Tout		Fournit des types légers de QML pour créer des interfaces utilisateur performantes pour les périphériques embarqués et mobiles. Ces contrôles permettent d'améliorer l'efficacité en employant une architecture de style simplifiée par rapport à Qt Quick Controls . Ces types fonctionnent conjointement avec Qt Quick and Qt Quick Layouts .
Qt Quick Extras	Tout		Fournit un ensemble spécialisé de contrôles qui peuvent être utilisés pour créer des interfaces dans Qt Quick.
Qt Widgets rapides	Tout		Fournit une classe de widget C ++ pour afficher une interface utilisateur rapide Qt.
Qt Script (Déconseillé)	Tout		Classes pour créer des applications Qt scriptables. Déconseillé en faveur des classes QJS* dans le module Qt QML .
Qt SCXML	Tout	Tout	Fournit des classes et des outils pour créer des machines d'état à partir de fichiers SCXML et les intégrer dans les applications.
Qt Script Tools (Déconseillé)	Tout		Composants supplémentaires pour les applications utilisant Qt Script .

1.3 - Architecture générale : Qt Add-Ons



Module	Plateformes de développement	Plateformes ciblées	La description
Capteurs Qt	Tout	Android , Qt pour iOS , WinRT et Mer.	Fournit un accès au matériel des capteurs et à la reconnaissance des gestes de mouvement.
Qt Serial Bus	Linux	Cibles Linux et Boot to Qt .	Fournit un accès à l'interface de bus série industrielle. Actuellement, le module prend en charge les protocoles CAN bus et Modbus.
Qt Serial Port	Tout	Windows , Linux et MacOS .	Fournit un accès au matériel et aux ports série virtuels.
Qt SVG	Tout		Classes pour afficher le contenu des fichiers SVG. Supporte un sous-ensemble de la norme SVG 1.2 Tiny .
Qt WebChannel	Tout	Tout	Fournit l'accès aux objets QObject ou QML à partir de clients HTML pour une intégration transparente des applications Qt avec les clients HTML / JavaScript.
Qt WebEngine	Tout	Windows , Linux et MacOS .	Classes et fonctions pour intégrer du contenu Web dans des applications utilisant le projet de navigateur Chromium .
Qt WebSockets	Tout	Tout	Fournit une communication WebSocket conforme à RFC 6455 .
Qt WebView	Tout	Plateformes avec un moteur Web natif.	Affiche le contenu Web dans une application QML en utilisant des API natives de la plateforme, sans avoir à inclure une pile de navigateur Web complète.
Qt Windows Extras	Tout	Windows	Fournit des API spécifiques à la plateforme pour Windows.
Qt X11 Extras	Tout	Linux / X11	Fournit des API spécifiques à la plateforme pour X11.
Qt XML			Les implémentations C ++ de SAX et DOM.
Qt XML Patterns			Prise en charge de XPath, XQuery , XSLT et validation de schéma XML.
Qt Wayland Compositor	Linux	Cibles Linux et Boot to Qt .	Fournit un cadre pour développer un compositeur de Wayland.



Module	Plateformes de développement	Plateformes ciblées	La description
Compléments disponibles sous Licences commerciales ou GNU General Public License v3			
Qt Charts	Tout		Composants UI pour afficher des graphiques visuellement agréés, pilotés par des modèles de données statiques ou dynamiques.
Affichage des données Qt	Tout	Tout	UI Components pour créer de magnifiques visualisations de données 3D.
Qt Virtual Keyboard	Tout	Ordinateur de bureau Linux et Windows , et Boot to Qt .	Un cadre pour la mise en œuvre de différentes méthodes d'entrée ainsi qu'un clavier QML virtuel. Prend en charge les mises en page localisées et les thèmes visuels personnalisés.



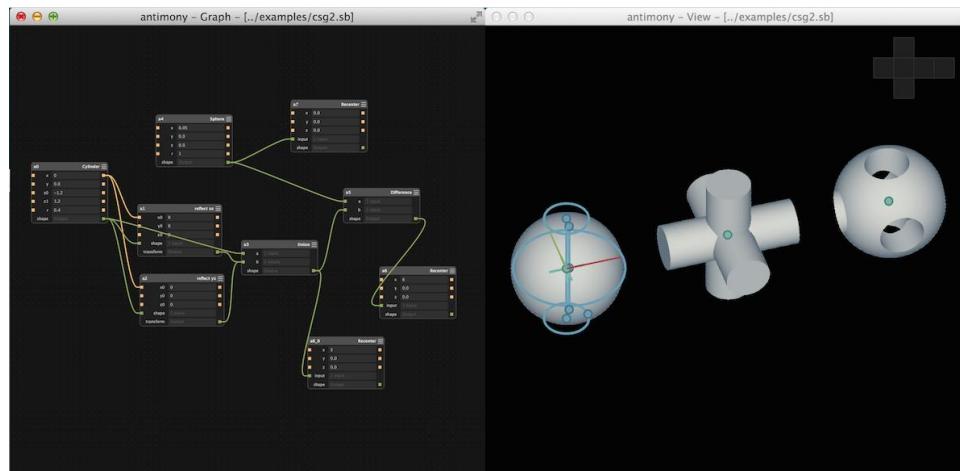
Qt Tools contient les outils officiels de Qt qui s'exécutent sur toutes les platesformes de développement supportées et facilitent le développement et la conception des applications.

Outil	La description
Qt Designer	Classes pour étendre Qt Designer .
Qt Aide	Classes pour intégrer la documentation en ligne aux applications, semblable à Qt Assistant.
Outils QI UI	Classes pour gérer les formulaires créés dans Qt Designer .

1.3 - Architecture générale : Qt Scene graph



Qt Scene graph : Qt Quick 2 utilise un graphique de scène dédié basé et une série d'adaptations dont la valeur par défaut utilise OpenGL ES 2.0 ou OpenGL 2.0 pour son rendu.



<http://doc.qt.io/qt-5/qtquick-visualcanvas-scenegraph.html>



Qt WebEngine fournit des fonctionnalités pour rendre les régions de contenu Web dynamique (anciennement Qt Webkit2)



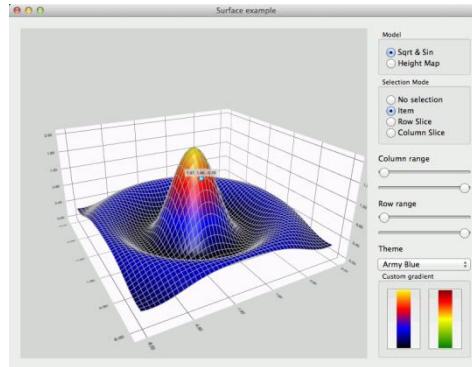
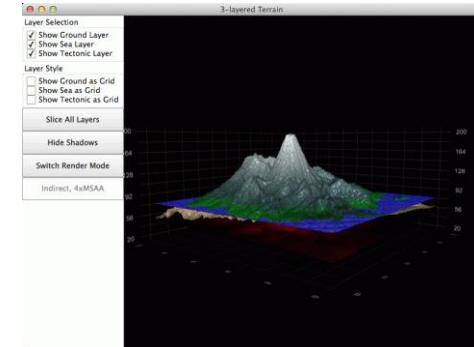
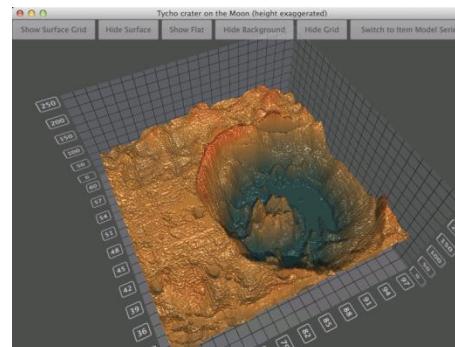
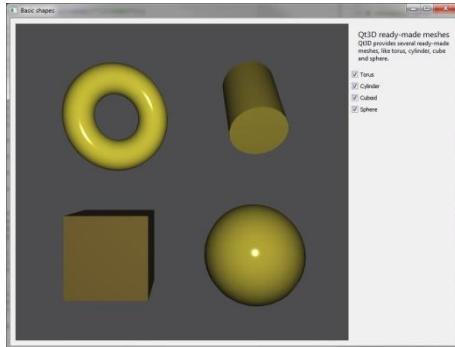
The image contains two screenshots of Qt applications. The left screenshot shows a dialog box with a title bar 'Use default dialogs'. It contains several blue buttons labeled 'Open Authentication Dialog', 'Open Proxy Dialog', 'Open Alert Dialog', 'Open Confirm Dialog', 'Open Prompt Dialog', 'Open Color Dialog', 'Open File Dialog', and 'Open Message Bubble'. Below these buttons is a text field containing the placeholder 'For this field we open message bubble...'. The right screenshot shows a web browser window with the URL 'http://www.qt.io/'. The page displays a medical application interface (Blood Pressure Meter) running on multiple devices (laptop, tablet, smartphone). Below the devices, the text 'One Qt Code Create Powerful Applications & Devices' is visible.

<http://doc.qt.io/qt-5/qtwebengine-index.html>

1.3 - Architecture générale : Qt 3D



Qt 3D fournit des fonctionnalités pour les systèmes de simulation en temps réel avec prise en charge du rendu 2D et 3D dans les applications Qt C ++ et Qt Quick.



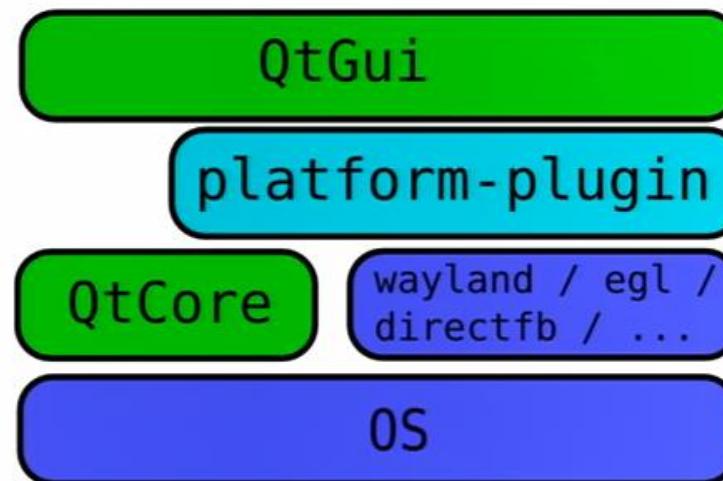
<https://doc.qt.io/qt-5/qt3d-index.html>



Qt Platform Abstraction : (QPA) C'est une couche d'abstraction qui facilite le port de Qt sur différentes plateformes via des plugins en lieu et place de #ifdef dans le code.



Qt Platform Abstraction Architecture



1.3 - Architecture générale : Qt Quick



Qt Quick : est un Framework libre développé et maintenu par *Digia* faisant partie de la bibliothèque Qt.

Il fournit la possibilité de créer des interfaces utilisateur personnalisables et dynamiques avec des effets de transition fluides de manière déclarative.

Ce type d'interface dynamique est de plus en plus utilisée, notamment sur les smartphones.

Qt Quick inclut un langage de script déclaratif appelé **QML** (comparable au XAML créé par Microsoft pour sa bibliothèque WPF).

Qt Quick et QML sont officiellement supportés depuis Qt 4.7 (avec Qt Creator 2.1).



1.3 - Architecture générale : Structure



Qt 5 Structure

Qt Add-Ons

3D, Bluetooth, Contacts,
Concurrent, D-Bus, Graphical
Effects, Image Formats, JS
Backend, Location, OpenGL,
Organizer, Print Support,...

Qt Essentials

Core, GUI, Multimedia,
Network, QML, Quick, SQL,
Test, WebKit

Qt Development Tools



Qt Creator



Qt Linguist



Qt Designer

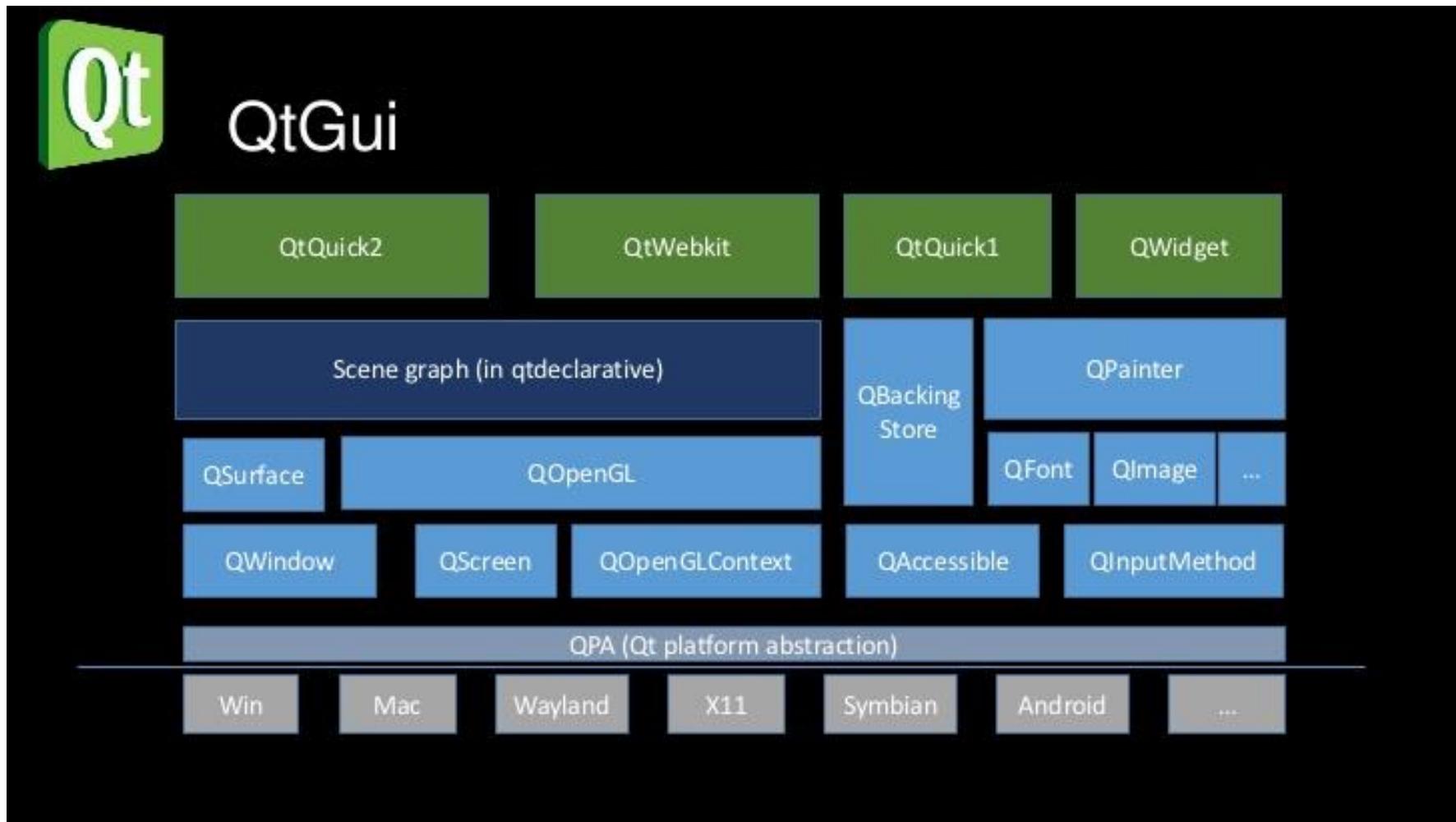


Qt Assistant

qmake

Cross-platform Support on Various Platforms
Linux, OSX, Windows,...

1.3 - Architecture générale : Structure





1.4 - Documentation



<https://doc.qt.io/>

The screenshot shows the main page of the Qt Documentation. At the top, there's a navigation bar with links to Wiki, Documentation, Forum, Bug Reports, and Code Review. On the right side of the header are links for Blog, Contact Us, and a user profile icon. Below the header is a search bar labeled "Google Custom Search" with a magnifying glass icon.

The page is divided into several sections:

- Getting Started**: Includes links to Getting Started Guides, What's New in Qt 5, Qt Licensing, and Examples and Tutorials.
- Advanced Topics**: Includes links to Qt Reference Documentation, Development Topics, Supported Platforms, and Qt Tools.
- Highlighted Features**: Includes links to Qt for Device Creation, Qt for Automation, Qt Automotive Suite, and Qt 3D Studio.
- Qt Versions**: Includes links to Qt 5.10, Qt 5.9, and Qt 5.6.
- Tools Manuals**: Includes links to Qt Creator, Qt Designer, Qt Linguist, Qt Assistant, Emulator, Qt Quick Compiler, Qt Installer Framework, Build Tools: qmake, Build Tools: Qbs, Build Tools: CMake, GammaRay, Qt QML Live, and Qt VS Tools.
- Archives and Snapshots**: Includes links to Documentation Snapshots and View Archives.



1.4 - Documentation



<https://doc.qt.io/qt-5/classes.html>

The screenshot shows the Qt Documentation website for Qt 5.10. The main page title is "Qt Documentation". On the left sidebar, there's a hiring announcement from The Qt Company, links to Reference (All Qt C++ Classes, All QML Types, All Qt Modules, Qt Creator Manual, All Qt Reference Documentation), and Getting Started (Getting Started with Qt, What's New in Qt 5, Examples and Tutorials, Supported Platforms, Qt Licensing). The main content area is titled "All Classes" and contains a list of categories: All Functions, All Namespaces, All Classes by Module, Obsolete Classes, and New Classes and Functions in Qt 5.9. It also mentions "Qt Reference Pages" for more reference pages. Below this, there are two tables of class names starting with '3' and 'A'. The footer includes a search bar, navigation links (Blog, Contact Us, User icon), and a copyright notice.

Qt Documentation

Qt 5.10 > All Classes

The Qt Company is hiring!
Work with the best
#QtPeople in business.
Check out
www.qt.io/careers

Reference

All Qt C++ Classes

All QML Types

All Qt Modules

Qt Creator Manual

All Qt Reference Documentation

Getting Started

Getting Started with Qt

What's New in Qt 5

Examples and Tutorials

Supported Platforms

Qt Licensing

All Classes

This is a list of all Qt 5 classes. The following pages contain different API listings in different categories:

- › All Functions
- › All Namespaces
- › All Classes by Module
- › Obsolete Classes
- › New Classes and Functions in Qt 5.9

For more reference pages including QML types, visit [Qt Reference Pages](#).

A B C D E F G H I J K L M N O P Q R S T U V W X

3	Q3DBars Q3DCamera Q3DInputHandler	Q3DLight Q3DObject Q3DScatter	Q3DScene Q3DSurface Q3DTheme
---	---	-------------------------------------	------------------------------------

A	QAbstract3DAxis QAbstract3DGraph QAbstract3DInputHandler QAbstract3DSeries QAbstractAnimation	QAccessibleInterface QAccessibleObject QAccessiblePlugin QAccessibleStateChangeEvent QAccessibleTableCellInterface	QAudioInputSelectorControl QAudioOutput QAudioOutputSelectorControl QAudioProbe QAudioRecorder
---	---	--	--



1.4 - Documentation



<https://doc.qt.io/qt-5/qtmodules.html>

The screenshot shows the Qt Documentation website for Qt 5.10. The main page title is "All Modules". On the left, there's a sidebar with links to "Contents", "Qt Essentials", "Qt Add-Ons", "Value-Add Modules", "Technology Preview Modules", "Qt Tools", and "Where to Go from Here". Below this is a section titled "The Qt Company Is hiring!" with a link to "www.qt.io/careers". Further down the sidebar are links to "Reference", "All Qt C++ Classes", "All QML Types", "All Qt Modules", "Qt Creator Manual", and "All Qt Reference Documentation". At the bottom of the sidebar are links to "Getting Started", "Getting Started with Qt", "What's New in Qt 5", and "Examples and Tutorials". The main content area starts with a section titled "Qt Essentials" which defines them as the foundation of Qt. It then lists the following modules:

Module	Description
Qt Core	Core non-graphical classes used by other modules.
Qt GUI	Base classes for graphical user interface (GUI) components. Includes OpenGL.
Qt Multimedia	Classes for audio, video, radio and camera functionality.
Qt Multimedia Widgets	Widget-based classes for implementing multimedia functionality.
Qt Network	Classes to make network programming easier and more portable.
Qt QML	Classes for QML and JavaScript languages.
Qt Quick	A declarative framework for building highly dynamic applications with custom user interfaces.
Qt Quick Controls	Reusable Qt Quick based UI controls to create classic desktop-style user interfaces.
Qt Quick Dialogs	Types for creating and interacting with system dialogs from a Qt Quick application.
Qt Quick Layouts	Layouts are items that are used to arrange Qt Quick 2 based items in the user interface.
Qt SQL	Classes for database integration using SQL.
Qt Test	Classes for unit testing Qt applications and libraries.
Qt Widgets	Classes to extend Qt GUI with C++ widgets.

If you use `qmake` to build your projects, the [Qt Core](#) and [Qt GUI](#) modules are included by default. To link only against Qt Core, add the following line to your `.pro` file:



1.4 - Documentation



<http://doc.qt.io/qtcreator/index.html>

Qt Documentation

Qt Creator Manual >

Qt Creator Manual

Qt Creator provides a cross-platform, complete integrated development environment (IDE) for application developers to create applications for multiple desktop, [embedded](#), and mobile device platforms, such as [Android](#) and [iOS](#). It is available for Linux, macOS and Windows operating systems. For more information, see [Supported Platforms](#).

This manual also describes features that are only available if you have the appropriate [Qt license](#). For more information, see [Qt Creator Commercial Features](#).

			
Getting Started <ul style="list-style-type: none">> IDE Overview> User Interface> Configuring Qt Creator> Building and Running an Example> Tutorials	Managing Projects <ul style="list-style-type: none">> Creating Projects> Using Version Control Systems> Configuring Projects> Managing Sessions	Designing User Interfaces <ul style="list-style-type: none">> Developing Qt Quick Applications> Developing Widget Based Applications> Optimizing Applications for Mobile Devices	Coding <ul style="list-style-type: none">> Writing Code> Finding> Refactoring> Configuring the Editor> Modeling> Editing State Charts
			
Building and Running <ul style="list-style-type: none">> Building for Multiple Platforms> Running on Multiple Platforms> Deploying to Devices> Connecting Devices	Testing <ul style="list-style-type: none">> Debugging> Analyzing Code> Running Autotests	Advanced Use <ul style="list-style-type: none">> Supported Platforms> Using Other Build Systems> Using Command Line Options> Keyboard Shortcuts> Using External Tools	Getting Help <ul style="list-style-type: none">> Using the Help Mode> FAQ> Tips and Tricks> Known Issues> Glossary



2.1 - Installation QT

2.2 - Qt Compilation

2.3 - Introduction QT Creator

- Type de projet
- Qt Console
- Qt Widget
- Qt QML

2.4 - Module Qt Core

- .pro
- QObject
- QMetaObject
- Affichage console
- Manipulation de fichier
- Thread
- Qt programmation événementielle
- MVC et Qt
- les containers
- Internationalisation
- Qt ressources system



2.1 - Installation QT : Téléchargement de Qt



<https://www.qt.io/download/>

The screenshot shows the official Qt website's download section. At the top, there's a navigation bar with links for Developers, Blog, English, Contact Us, and BUY QT NOW. Below the navigation is a large banner with the text "Get started with Qt now" and a subtext "Reviewing pricing, choose your plan and start developing with Qt today." A prominent green "Buy Qt" button is located below the banner. The main content area features two main download options: "Desktop & Mobile Applications" and "Embedded Devices". Each option includes a small image, a brief description, and a link. The "Desktop & Mobile Applications" section says "Free commercial 30-day trial and open source downloads available." and "Choose this option if you are building an embedded device or deploying your software". The "Embedded Devices" section says "Commercial offering with free 30-day trial.".

Qt

Developers Blog English Contact Us BUY QT NOW

Products Find an Advisor Resources Customers Company

Get started with Qt now

Reviewing pricing, choose your plan and start developing with Qt today.

Buy Qt

Or choose a download and give Qt a try.

What kind of development project do you have?

Desktop & Mobile Applications

Free commercial 30-day trial and open source downloads available.

Choose this option if you are building an embedded device or deploying your software

Embedded Devices

Commercial offering with free 30-day trial.



- Windows et GNU gcc
- Windows et Visual C++
- GNU/Linux et GNU gcc ou LLVM
- OS X et clang.

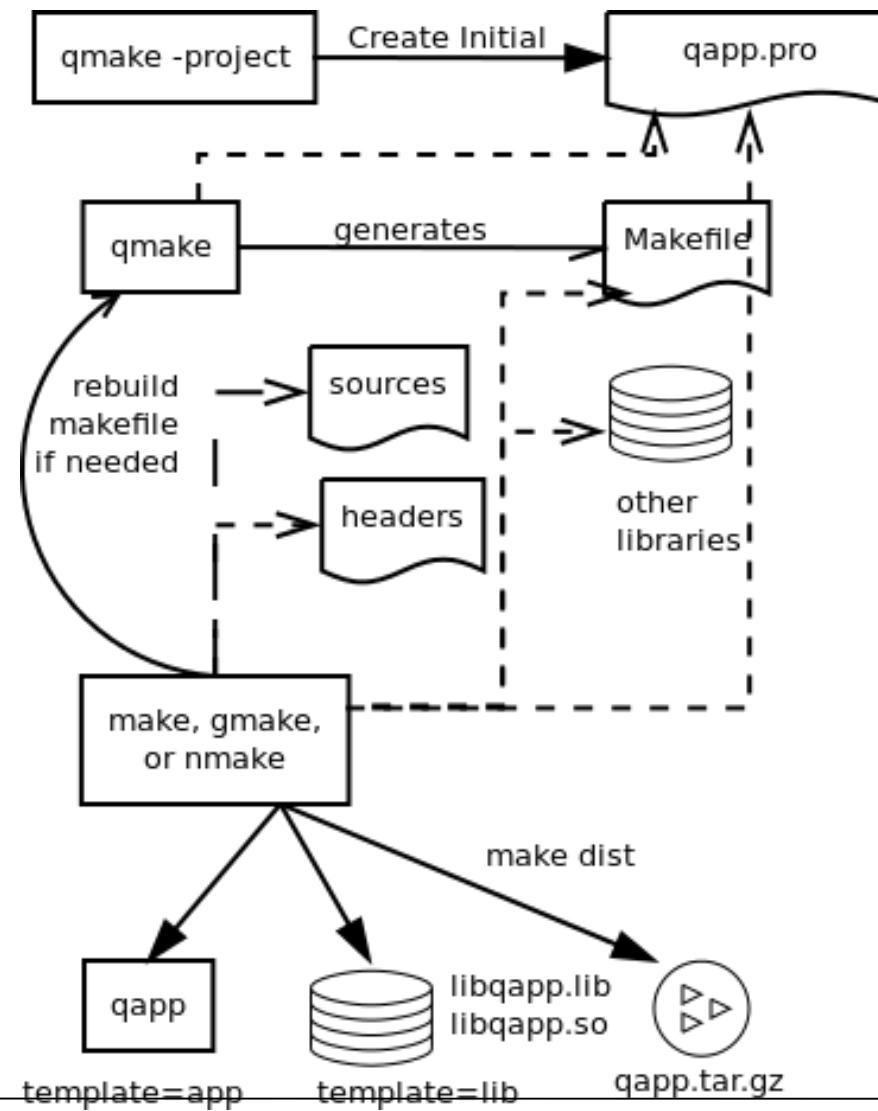


Dans le répertoire de travail

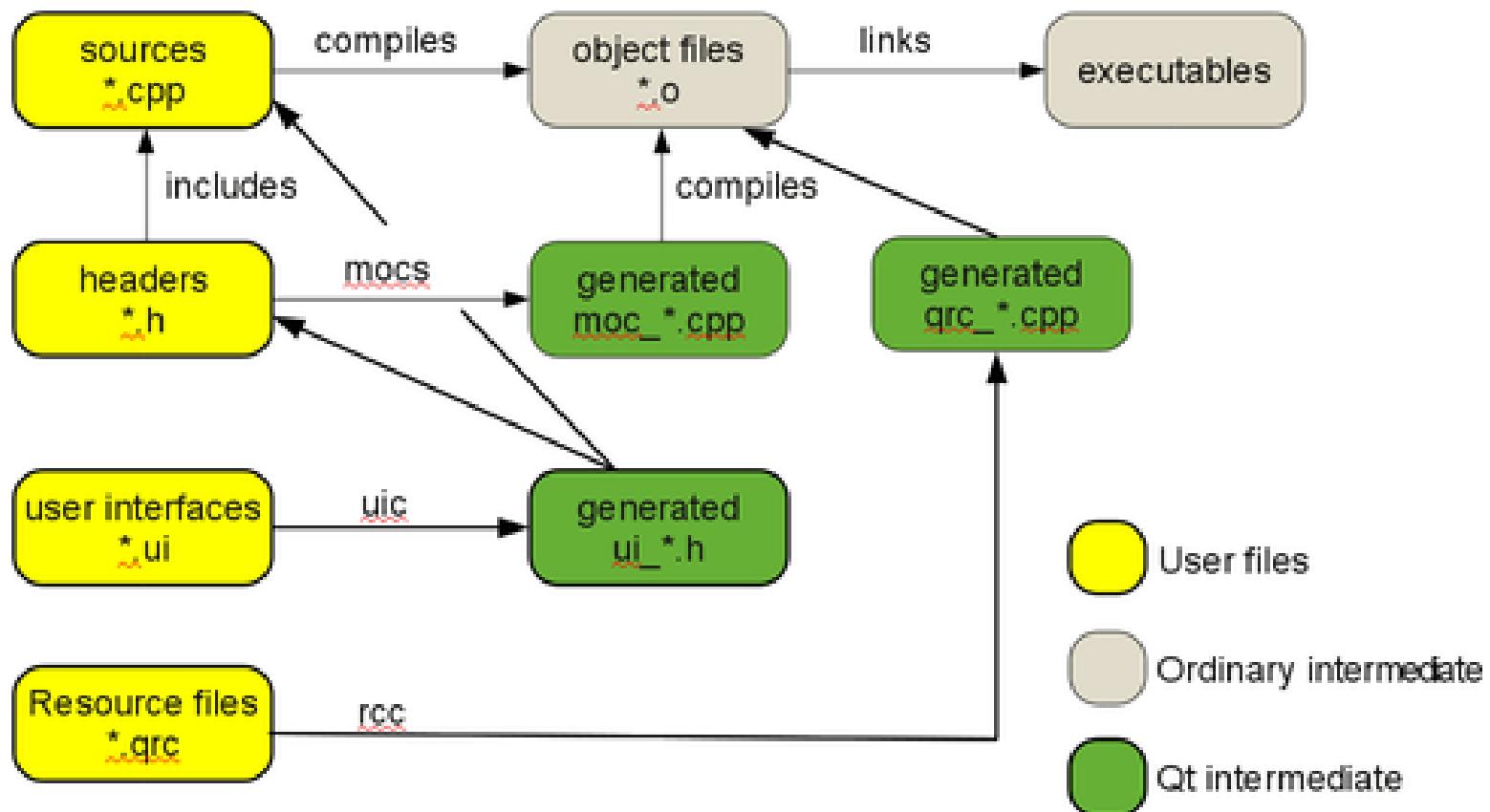
- qmake –project -> génère le **.pro**
- qmake [-makefile] -> génère le **Makefile**
- Make -> **compilation**



2.2 - Qt Compilation : qmake



2.2 - Qt Compilation : fichiers générés





- **moc** : Les fichiers .h et .cpp sont traités par l'application moc(.exe) qui les analyse et crée les méta-classes (moc_*.cpp) si nécessaire.
- **ui** (format XML) : Les fichiers .ui sont traités par l'application uic(.exe) et transformés en fichiers ui_*.h.
- **qrc** (format XML) : Les fichiers .qrc sont traités par l'application rcc(.exe) et transformés en fichiers qrc_*.cpp.

2.2 - Qt Compilation : Exemple



```
main.cpp ✘ |  
1 #include <QCoreApplication>  
2 #include <iostream>  
3  
4 int main(int , char *[]){  
5 {  
6     std::cout << "Mon premier programme en QT : QtConsole" << std::endl;  
7     return 0;  
8 }  
9 }
```

```
Répertoire de D:\YantraTechnologies\Yantra-Cours\OK_C++&POO\Cours\Exercice\QtFormation\FormationQtMake  
12/03/2018 14:54 <DIR> .  
12/03/2018 14:54 <DIR> ..  
15/09/2016 10:47 173 main.cpp  
1 fichier(s) 173 octets  
2 Rép(s) 225 197 346 816 octets libres
```



qmake –project main.cpp

```
D:\YantraTechnologies\Yantra-Cours\OK_C++&P00\Cours\Exercice\QtFormation\FormationQtMake>qmake -project main.cpp

D:\YantraTechnologies\Yantra-Cours\OK_C++&P00\Cours\Exercice\QtFormation\FormationQtMake>dir
Le volume dans le lecteur D s'appelle Data
Le numéro de série du volume est 28B3-8948

Répertoire de D:\YantraTechnologies\Yantra-Cours\OK_C++&P00\Cours\Exercice\QtFormation\FormationQtMake

12/03/2018  14:57    <DIR>          .
12/03/2018  14:57    <DIR>          ..
12/03/2018  14:57                953 FormationQtMake.pro
15/09/2016  10:47                173 main.cpp
                  2 fichier(s)        1 126 octets
                  2 Rép(s)   225 197 342 720 octets libres
```

2.2 – Qt Compilation : Exemple



```
main.cpp FormationQtMake.pro
1 ######
2 # Automatically generated by qmake (3.1) Mon Mar 12 14:57:23 2018
3 #####
4
5 TEMPLATE = app
6
7 TARGET = FormationQtMake
8 CONFIG += console
9
10 INCLUDEPATH += .
11
12 # The following define makes your compiler warn you if you use any
13 # feature of Qt which has been marked as deprecated (the exact warnings
14 # depend on your compiler). Please consult the documentation of the
15 # deprecated API in order to know how to port your code away from it.
16 DEFINES += QT_DEPRECATED_WARNINGS
17
18 # You can also make your code fail to compile if you use deprecated APIs.
19 # In order to do so, uncomment the following line.
20 # You can also select to disable deprecated APIs only up to a certain version of Qt.
21 #DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0
22
23 # Input
24 SOURCES += main.cpp
```

2.2 – Qt Compilation : Exemple



- qmake
- make

```
D:\YantraTechnologies\Yantra-Cours\OK_C++&P00\Cours\Exercice\QtFormation\FormationQtMake>make  
  
D:\YantraTechnologies\Yantra-Cours\OK_C++&P00\Cours\Exercice\QtFormation\FormationQtMake>  
D:\YantraTechnologies\Yantra-Cours\OK_C++&P00\Cours\Exercice\QtFormation\FormationQtMake>qmake  
  
D:\YantraTechnologies\Yantra-Cours\OK_C++&P00\Cours\Exercice\QtFormation\FormationQtMake>make  
make -f Makefile.Release  
make[1]: Entering directory `d/YantraTechnologies/Yantra-Cours/OK_C++&P00/Cours/Exercice/QtFormation/FormationQtMake'  
g++ -c -fno-keep-inline-dllexport -O2 -std=gnu++11 -Wextra -Wall -W -fexceptions -mthreads -DUNICODE -D_UNICODE -DQT_DEPRECATED_WARNINGS -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_CORE_LIB -DQT_NEEDS_QMAIN -I. -I. -ID:/NewQt/5.10.0/mingw53_32/include -ID:/NewQt/5.10.0/mingw53_32/include/QtGui -ID:/NewQt/5.10.0/mingw53_32/include/QtANGLE -ID:/NewQt/5.10.0/mingw53_32/include/QtCore -Irelease -ID:/NewQt/5.10.0/mingw53_32/mkspecs/win32-g++ -o release/main.o main.cpp  
g++ -Wl,-s -Wl,-subsystem,windows -mthreads -o release/FormationQtMake.exe release/main.o -lmingw32 -LD:/NewQt/5.10.0/mingw53_32/lib D:/NewQt/5.10.0/mingw53_32/lib/libqtmain.a -LC:/utils/my_sql/my_sql/lib -LC:/utils/postgresql/pgsql/lib -Lshell32 D:/NewQt/5.10.0/mingw53_32/lib/libQt5Core.a D:/NewQt/5.10.0/mingw53_32/lib/libQt5Gui.a D:/NewQt/5.10.0/mingw53_32/lib/libQt5Core.a  
make[1]: Leaving directory `d/YantraTechnologies/Yantra-Cours/OK_C++&P00/Cours/Exercice/QtFormation/FormationQtMake'  
  
D:\YantraTechnologies\Yantra-Cours\OK_C++&P00\Cours\Exercice\QtFormation\FormationQtMake>
```

2.2 – Qt Compilation : Exemple



```
D:\YantraTechnologies\Yantra-Cours\OK_C++&POO\Cours\Exercice\QtFormation\FormationQtMake>dir
Le volume dans le lecteur D s'appelle Data
Le numéro de série du volume est 28B3-8948

Répertoire de D:\YantraTechnologies\Yantra-Cours\OK_C++&POO\Cours\Exercice\QtFormation\FormationQtMake

12/03/2018 14:58    <DIR>          .
12/03/2018 14:58    <DIR>          ..
12/03/2018 14:58            1 022 .qmake.stash
12/03/2018 14:58    <DIR>          debug
12/03/2018 15:08            969 FormationQtMake.pro
12/03/2018 15:08            176 main.cpp
12/03/2018 15:08            27 155 Makefile
12/03/2018 15:08            18 781 Makefile.Debug
12/03/2018 15:08            18 833 Makefile.Release
12/03/2018 15:08    <DIR>          release
               6 fichier(s)       66 936 octets
               4 Rép(s)   225 197 244 416 octets libres

D:\YantraTechnologies\Yantra-Cours\OK_C++&POO\Cours\Exercice\QtFormation\FormationQtMake>release\FormationQtMake.exe
Mon premier programme en QT : QtConsole

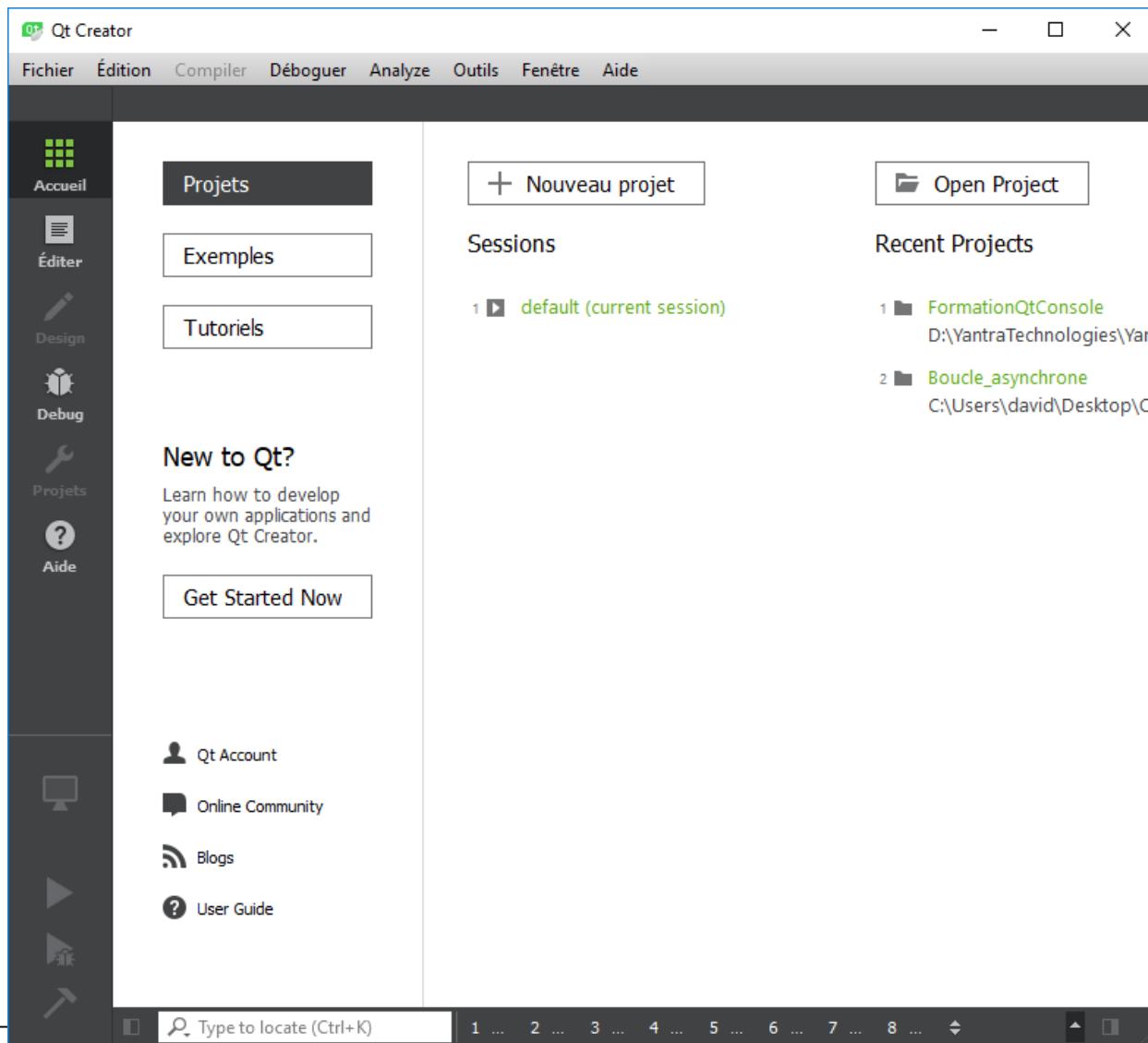
D:\YantraTechnologies\Yantra-Cours\OK_C++&POO\Cours\Exercice\QtFormation\FormationQtMake>
```

<http://doc.qt.io/qt-5/qmake-project-files.html>

<http://doc.qt.io/qt-5/qmake-variable-reference.html>



2.3 - Introduction QT Creator



2.3 - Introduction QT Creator : Type de projet -> application



Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Accueil Exemples Tutoriels Nouveau sur Qt ? Démarrer Qt Account Qt Cloud Services Communauté en ligne Blogs Guide utilisateur Hello world!

Nouveaux projets

Ouvrir le projet

Sessions Projets récents

Nouveau projet

Choisir un modèle :

Modèle Desktop

Application Qt avec widgets

Application Qt Quick

Application Qt4 en console

Interface graphique Qt Quick

Choisir... Cancel

C:\Users\David\Downloads\ENI-Sources\OuvrageQt\Chapitre 1\Shallow_Copy\Shallow_Copy.pro

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control



2.3 - Introduction QT Creator : Type de projet -> Library



Nouveau projet

X

Choisir un modèle :

Tous les modèles ▾

Projets

Application

Library

Autre projet

Projet non Qt

Importer un projet

Fichiers et classes

Bibliothèque C++

Plug-in d'extension Qt Quick 1

Plug-in d'extension Qt Quick 2

Extension Qt Creator

Créer une bibliothèque C++ basée sur qmake.
Cela peut être utilisé pour créer :

- une bibliothèque C++ partagée pour utiliser avec QPluginLoader à l'exécution (plug-ins)
- une bibliothèque C++ partagée ou statique pour utiliser avec un autre projet au moment de la compilation

Plateformes supportées: Desktop

Choose...

Cancel



2.3 - Introduction QT Creator : Type de projet -> Autre projet



Nouveau projet

X

Choisir un modèle :

Tous les modèles ▾

Projets

Application

Library

Autre projet

Projet non Qt

Importer un projet

Fichiers et classes

- Test unitaire Qt
- Widget personnalisé pour Qt4 Designer
- Projet de sous-répertoires
- Empty qmake Project
- Morceau de code

Créer un test unitaire basé sur QTestList pour une fonctionnalité ou une classe. Les tests unitaires vous permettent de vérifier que le code est utilisable et qu'il n'y a pas de régression.

Plateformes supportées: Desktop

Choose...

Cancel

2.3 - Introduction QT Creator : Type de projet -> Projet non Qt



Nouveau projet

Choisir un modèle :

Tous les modèles ▾

Projets

- Application
- Library
- Autre projet
- Projet non Qt
- Importer un projet

Fichiers et classes

Projet C

Projet C++

Projet C (compilation avec CMake)

Projet C complet (compilation avec Qbs)

Projet C++ (compilation avec CMake)

Projet C++ complet (compilation avec C

Créer un projet C utilisant qmake mais pas la bibliothèque Qt.

Plateformes supportées: Desktop

Choose... Cancel



2.3 - Introduction QT Creator : Type de projet -> Importer un projet



Nouveau projet

Choisir un modèle :

Tous les modèles ▾

Projets

- Application
- Library
- Autre projet
- Projet non Qt
- Importer un projet

Fichiers et classes

Bazaar Clone (Or Branch)

Git Repository Clone

Mercurial Clone

Subversion Checkout

Projet Momentics Cascades

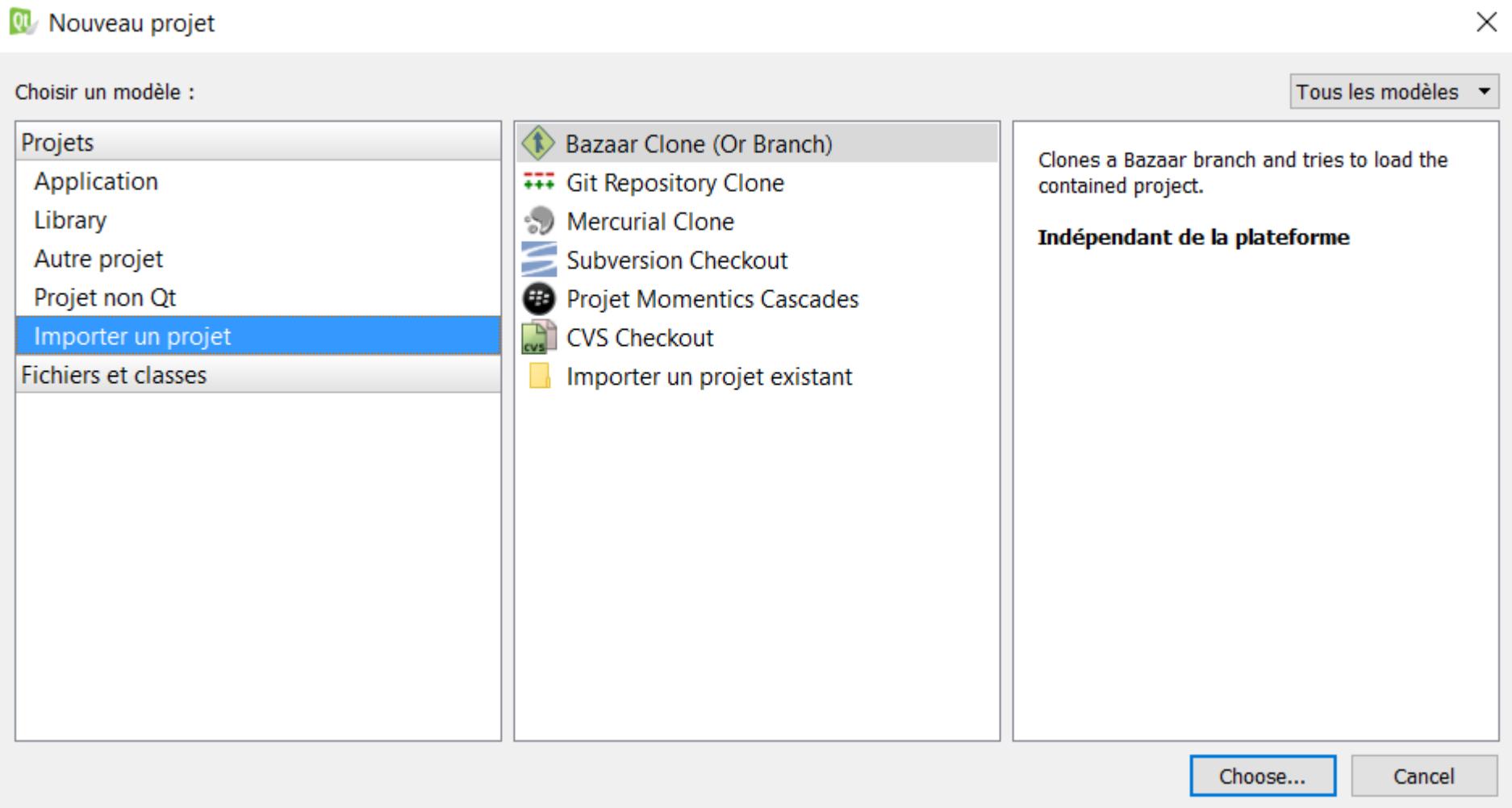
CVS Checkout

Importer un projet existant

Clones a Bazaar branch and tries to load the contained project.

Indépendant de la plateforme

Choose... Cancel





2.3 - Introduction QT Creator : Qt console



Application Qt4 en console

Location

Kits

Résumé

Introduction et emplacement du projet

This wizard generates a Qt Console Application project. The application derives from QCoreApplication and does not provide a GUI.

Nom :

Créer dans :

Utiliser comme emplacement par défaut pour les projets

← **Application Qt4 en console**

Location

Kits

Résumé

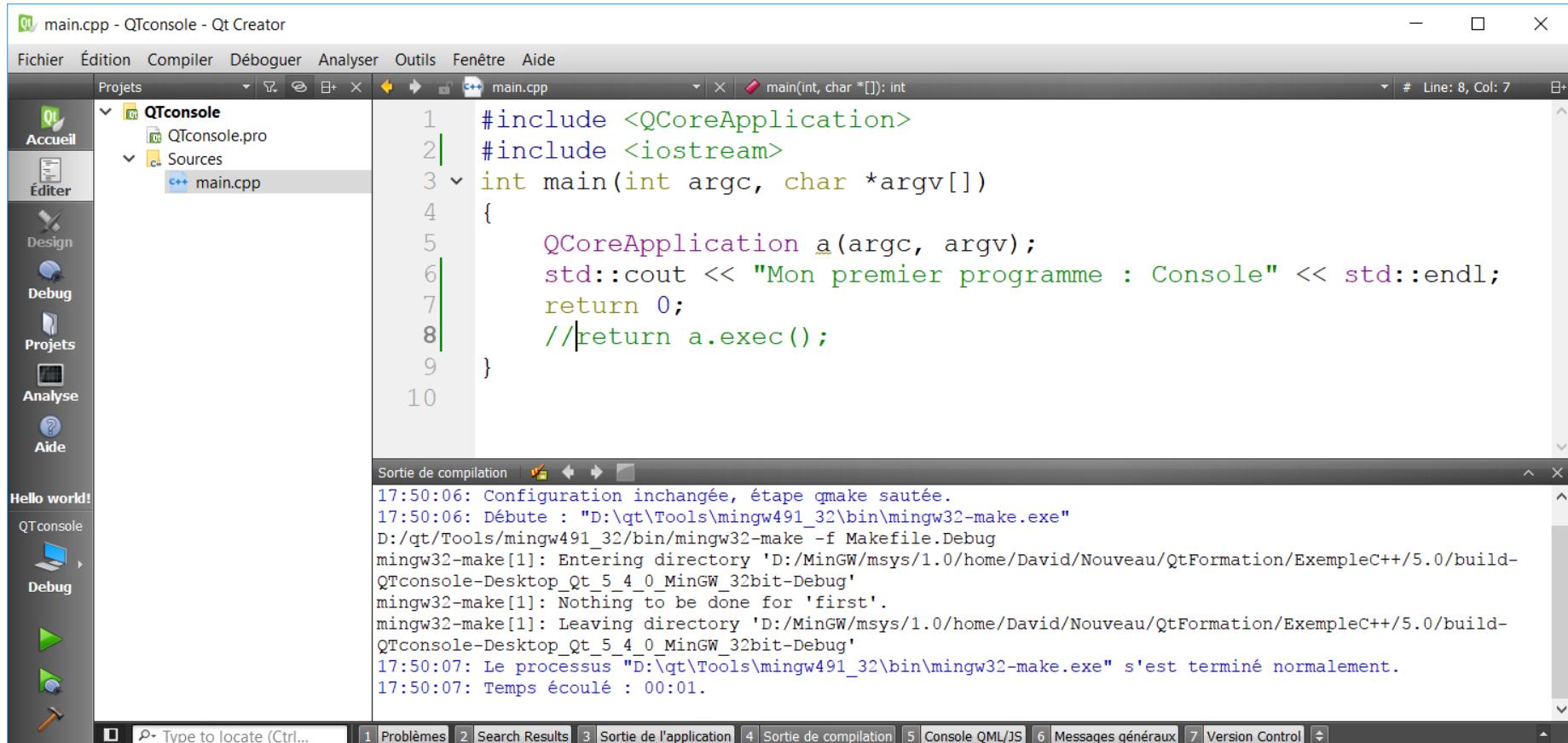
Sélection de kit

Qt Creator peut utiliser les kits suivant pour le projet **QTconsole** :

Select all kits

Desktop Qt 5.4.0 MinGW 32bit

2.3 - Introduction QT Creator : Qt console



main.cpp - QTconsole - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets main.cpp main(int, char *[]): int

QTconsole
QTconsole.pro
Sources main.cpp

```
1 #include <QCoreApplication>
2 #include <iostream>
3 int main(int argc, char *argv[])
4 {
5     QCoreApplication a(argc, argv);
6     std::cout << "Mon premier programme : Console" << std::endl;
7     return 0;
8     //return a.exec();
9 }
10
```

Sortie de compilation

```
17:50:06: Configuration inchangée, étape qmake sautée.
17:50:06: Débuté : "D:\qt\Tools\mingw491_32\bin\mingw32-make.exe"
D:/qt/Tools/mingw491_32/bin/mingw32-make -f Makefile.Debug
mingw32-make[1]: Entering directory 'D:/MinGW/msys/1.0/home/David/Nouveau/QtFormation/ExempleC++/5.0/build-QTconsole-Desktop_Qt_5_4_0_MinGW_32bit-Debug'
mingw32-make[1]: Nothing to be done for 'first'.
mingw32-make[1]: Leaving directory 'D:/MinGW/msys/1.0/home/David/Nouveau/QtFormation/ExempleC++/5.0/build-QTconsole-Desktop_Qt_5_4_0_MinGW_32bit-Debug'
17:50:07: Le processus "D:\qt\Tools\mingw491_32\bin\mingw32-make.exe" s'est terminé normalement.
17:50:07: Temps écoulé : 00:01.
```

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control



2.3 - Introduction QT Creator : Qt console



QTconsole.pro - QTconsole - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets QTconsole QTconsole.pro Sources main.cpp

```
1 #-----#
2 #
3 # Project created by QtCreator 2017-09-19T17:30:40
4 #
5 #-----#
6
7 QT      += core
8 QT      -= gui
9
10 QMAKE_CXXFLAGS += -std=c++11
11
12 TARGET = QTconsole
13
14 CONFIG   += console
15 CONFIG   += c++11
16 CONFIG   -= app_bundle
17
18 TEMPLATE = app
19
```

Sortie de compilation

```
17:50:06: Configuration inchangée, étape qmake sautée.
17:50:06: Débute : "D:\qt\Tools\mingw491_32\bin\mingw32-make.exe"
D:/qt/Tools/mingw491_32/bin/mingw32-make -f Makefile.Debug
mingw32-make[1]: Entering directory 'D:/MinGW/msys/1.0/home/David/Nouveau/QtFormation/ExempleC++/5.0/build-QTconsole-Desktop_Qt_5_4_0_MinGW_32bit-Debug'
mingw32-make[1]: Nothing to be done for 'first'.
mingw32-make[1]: Leaving directory 'D:/MinGW/msys/1.0/home/David/Nouveau/QtFormation/ExempleC++/5.0/build-QTconsole-Desktop_Qt_5_4_0_MinGW_32bit-Debug'
17:50:07: Le processus "D:\qt\Tools\mingw491_32\bin\mingw32-make.exe" s'est terminé normalement.
17:50:07: Temps écoulé : 00:01.
```

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control



2.3 - Introduction QT Creator : Qt console



The screenshot shows the Qt Creator interface with a project named "QTconsole". The left sidebar has icons for Accueil, Éditer, Design, Debug, Projets, Analyse, Aide, and Hello world!. A red arrow points to the "Debug" icon under "Hello world!". The main window shows the project structure with "QTconsole.pro" and "Sources/main.cpp". The code editor displays the contents of main.cpp, which includes a copyright notice and the QT core dependency. A terminal window titled "D:\qt\Tools\QtCreator\bin\qtcreator_process_stub.exe" shows the output of the program: "Mon premier programme : Console" and "Appuyez sur <ENTRÉE> pour fermer cette fenêtre...". The bottom status bar shows build logs: "17:53:22: Le processus \"D:\qt\Tools\mingw491_32\bin\mingw32-make.exe\" s'est terminé normalement." and "17:53:22: Temps écoulé : 00:01.". The bottom navigation bar includes tabs for Problèmes, Search Res..., Sortie de l...', Sortie de c..., Console Q..., Messages ..., and Version Co...".

2.3 - Introduction QT Creator : Qt widget



Application Qt avec widgets

Location Kits Details Résumé

Introduction et emplacement du projet

Cet assistant génère un projet d'application Qt avec widgets. L'application dérive par défaut de QApplication et inclut un widget vide.

Nom :

Créer dans :

Utiliser comme emplacement par défaut pour les projets

2.3 - Introduction QT Creator : Qt widget



← Application Qt avec widgets

Sélection de kit

Qt Creator peut utiliser les kits suivant pour le projet **QtWidget** :

Select all kits

<input checked="" type="checkbox"/>  Desktop Qt 5.4.0 MinGW 32bit	Manage...	Détails ▾
--	-----------	-----------

Suivant > **Annuler**



2.3 - Introduction QT Creator : Qt widget

←  Application Qt avec widgets

X

Information sur la classe

Location

Kits

Details

Résumé

Nom de la classe :

Classe parent :

Fichier d'en-tête :

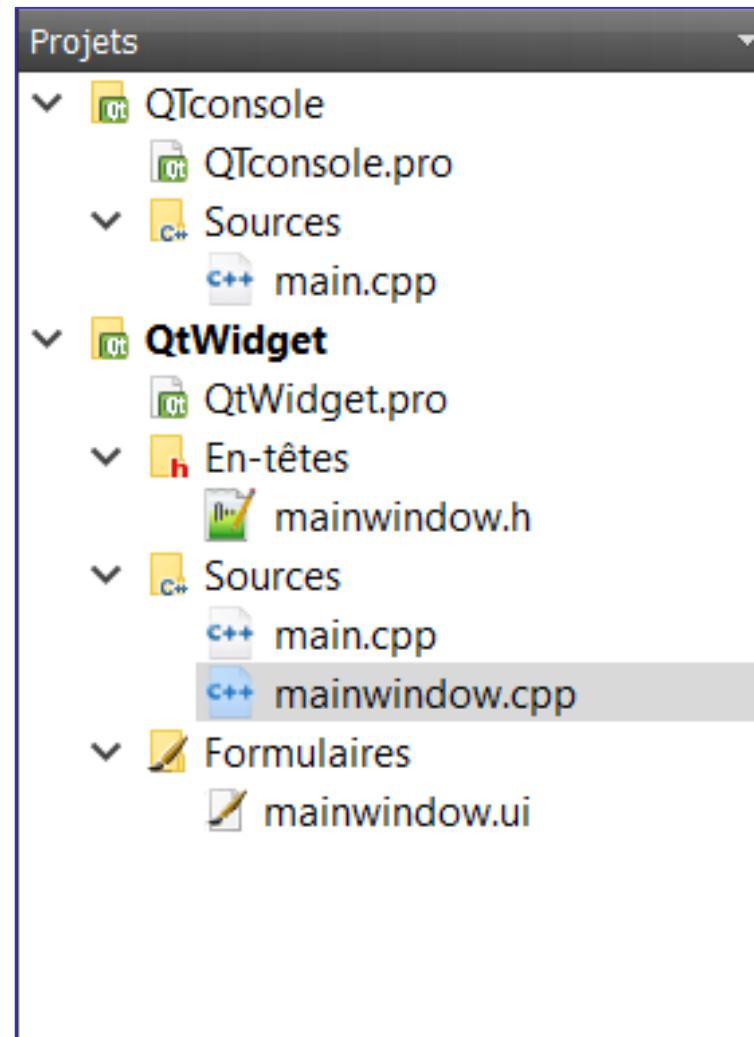
Fichier source :

Générer l'interface graphique :

Fichier d'interface :

Suivant > Annuler

2.3 - Introduction QT Creator : Qt widget



Projets

- QTconsole
 - QTconsole.pro
 - Sources
 - main.cpp
- QtWidget
 - QtWidget.pro
 - En-têtes
 - mainwindow.h
 - Sources
 - main.cpp
 - mainwindow.cpp
 - Formulaires
 - mainwindow.ui



2.3 - Introduction QT Creator : Qt widget



QtWidget.pro - QWidget - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets > QtConsole
QtWidget
QtWidget.pro
En-têtes mainwindow.h
Sources main.cpp mainwindow.cpp
Formulaires mainwindow.ui

```
1 # -----
2 #
3 # Project created by QtCreator 2017-09-19T17:57:57
4 #
5 # -----
6
7 QT      += core gui
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
10
11 TARGET = QtWidget
12 TEMPLATE = app
13
14 QMAKE_CXXFLAGS += -std=c++11
15
16 SOURCES += main.cpp \
               mainwindow.cpp
17
18 HEADERS  += mainwindow.h
19
20 FORMS    += mainwindow.ui
21
22 |
```

Sortie de l'application

Compilation

Type to locate (Ctrl...) 1 Problèmes 2 Search Re... 3 Sortie de l... 4 Sortie de ... 5 Console Q... 6 Messages ... 7 Version C...

2.3 - Introduction QT Creator : Qt widget



```
main.cpp <Selectionner un symbole>
1 #include "mainwindow.h"
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     MainWindow w;
8     w.show();
9
10    return a.exec();
11 }
12
```

2.3 - Introduction QT Creator : Qt widget



```
mainwindow.h <Selectionner un symbole>
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5
6 namespace Ui {
7     class MainWindow;
8 }
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     explicit MainWindow(QWidget *parent = 0);
16     ~MainWindow();
17
18 private:
19     Ui::MainWindow *ui;
20 };
21
22 #endif // MAINWINDOW_H
23
```



2.3 - Introduction QT Creator : Qt widget



The screenshot shows the Qt Creator interface for a QWidget application named "mainwindow.ui".

Left Sidebar:

- Qt Accueil**: Hello world!
- Éditeur**: Layouts (Vertical Layout, Horizontal Layout, Grid Layout, Form Layout), Spacers (Horizontal Spacer, Vertical Spacer), Buttons (Push Button, Tool Button, Radio Button, Check Box, Command Link Button, Button Box), Item Views...del-Based (List View, Tree View, Table View, Column View), Item Widgets-Based (List Widget, Tree Widget, Table Widget).
- Design**: Shows a main window with a central label containing the text "Mon premier programme : QWidget".
- Debug**: Shows the output console.
- Projets**: Shows the project structure.
- Analysé**: Shows code analysis results.
- Aide**: Help documentation.

Central Area:

- Toolbar:** Includes icons for file operations, search, and other common functions.
- Object Inspector (Top Right):** Lists objects and their classes:
 - MainWindow: QMainWindow
 - centralWidget: QWidget
 - label: QLabel
 - menuBar: QMenuBar
 - menuQuit: QMenu
 - mainToolBar: QToolBar
 - statusBar: QStatusBar
- Properties Panel (Bottom Right):** Properties for the "label" object:
 - text**: Mon premier programme : QWidget
 - textFormat**: AutoText
 - pixmap**
 - scaledContents**:
 - alignment**: AlignementCentreH, AlignementCentreV
 - Horizontal**: AlignementCentreH
 - Vertical**: AlignementCentreV
- Action Editor (Bottom Left):** Shows an action named "actionQuit" with the text "quit".

Output Console (Bottom):

```
C:\Users\David\Documents\build-QtWidget/Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtWidget.exe s'est terminé avec le code 0

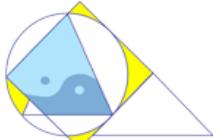
Démarrage de C:\Users\David\Documents\build-QtWidget/Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtWidget.exe...
C:\Users\David\Documents\build-QtWidget/Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtWidget.exe s'est terminé avec le code 0
```



2.3 - Introduction QT Creator : Qt widget



```
mainwindow.cpp MainWindow::~MainWindow()
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9     this->setWindowTitle("Ma premiere fenetre : Widget");
10
11     QFont f( "Arial", 12, QFont::Bold);
12     this->ui->label->setFont( f );
13
14     QAction *actionQuitter = ui->menuQuit->addAction("&Quitter");
15     connect(actionQuitter, SIGNAL(triggered()), qApp, SLOT(quit()));
16 }
17
18 MainWindow::~MainWindow()
19 {
20     delete ui;
21 }
22
```



The screenshot shows the Qt Creator IDE interface. The left sidebar contains icons for Accueil, Éditer, Design, Debug, Projets, Analyse, Aide, and Hello world!. The main window has a menu bar with Fichier, Édition, Compiler, Déboguer, Analyser, Outils, Fenêtre, and Aide. The top toolbar includes buttons for Projects, File, Open, Save, Build, Run, and Help, along with tabs for ui_mainwindow.h and <Aucun symbole>. The status bar at the bottom shows the current file (ui_mainwindow.h), line (Line: 6, Col: 14), and other navigation information.

Projects

Qtconsole

QtWidget

QtWidget.pro

En-têtes

mainwindow.h

Sources

main.cpp

mainwindow.cpp

Formulaires

mainwindow.ui

```
1 //*****  
2 ** Form generated from reading UI file 'mainwindow.ui'  
3 **  
4 ** Created by: Qt User Interface Compiler version 5.4.0  
5 **  
6 ** WARNING! All changes made in this file will be lost when recompiling UI file!  
7 *****  
8  
9 #ifndef UI_MAINWINDOW_H  
10 #define UI_MAINWINDOW_H  
11  
12 #include <QtCore/QVariant>  
13 #include <QtWidgets/QAction>  
14 #include <QtWidgets/QApplication>  
15 #include <QtWidgets/QButtonGroup>  
16 #include <QtWidgets/QHeaderView>  
17 #include <QtWidgets.QLabel>  
18 #include <QtWidgets/QMainWindow>  
19 #include <QtWidgets/QMenu>  
20 #include <QtWidgets/QMenuBar>  
21 #include <QtWidgets/QStatusBar>  
22 #include <QtWidgets/QToolBar>  
23 #include <QtWidgets/QWidget>  
24  
25 QT_BEGIN_NAMESPACE  
26  
27 class Ui_MainWindow  
28 {  
29 public:  
30     QAction *actionQuit;  
31     QWidget *centralWidget;  
32     QLabel *label;  
33     QMenuBar *menuBar;  
34     QMenu *menuQuit;  
35     QToolBar *mainToolBar;  
36     QStatusBar *statusBar;  
37  
38 void setupUi(QMainWindow *MainWindow)  
39 {  
40     if (MainWindow->objectName().isEmpty())  
41         MainWindow->setObjectName(QStringLiteral("MainWindow"));  
42     MainWindow->resize(507, 354);  
43     actionQuit = new QAction(MainWindow);  
44     actionQuit->setObjectName(QStringLiteral("actionQuit"));  
45     centralWidget = new QWidget(MainWindow);  
46     centralWidget->setObjectName(QStringLiteral("centralWidget"));  
47     label = new QLabel(centralWidget);  
48     label->setObjectName(QStringLiteral("label"));  
49 }
```



Nouvelle application Qt Quick

Location

Component Set

Kits

Résumé

Introduction et emplacement du projet

This wizard generates a Qt Quick Application project.

Nom :

Créer dans :

Utiliser comme emplacement par défaut pour les projets



← Nouvelle application Qt Quick

X

Sélectionner l'ensemble des composants Qt Quick

Location

Component Set Ensemble de composants Qt Quick : **Qt Quick Controls 1.3** ▾

Kits

Résumé

Creates a deployable Qt Quick 2 application that contains a .ui.qml file and uses Qt Quick Controls. You can review Qt Quick 2 UI projects in the QML Scene and you need not build them. This project requires that you have installed Qt Quick Controls for your Qt version. Requires Qt 5.4 or newer.

Suivant > Annuler



← Nouvelle application Qt Quick

Location
Component Set
Kits
Résumé

Sélection de kit

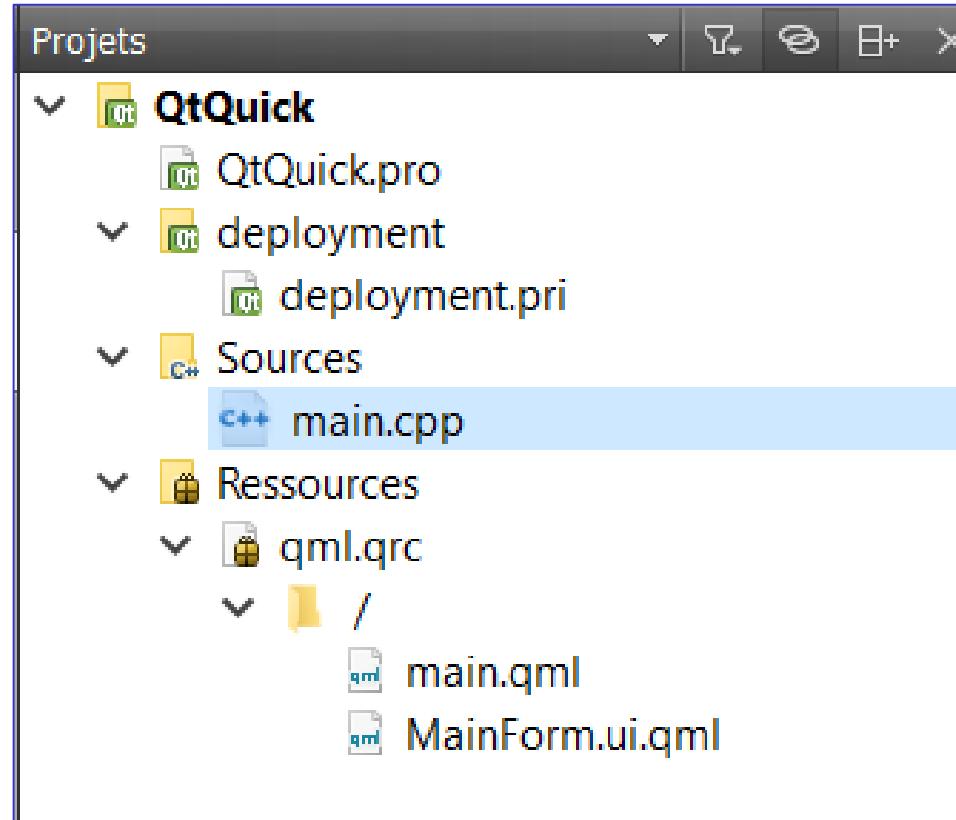
Qt Creator peut utiliser les kits suivant pour le projet **QtQuick** :

Select all kits

<input checked="" type="checkbox"/>  Desktop Qt 5.4.0 MinGW 32bit	Détails ▾
--	-----------

Suivant > **Annuler**

2.3 - Introduction QT Creator : Qt QML





2.3 - Introduction QT Creator : Qt QML



QtQuick.pro - QtQuick - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Line: 15, Col: 1

Projets

QtQuick

- QtQuick.pro
- deployment

 - deployment.pri

- Sources

 - main.cpp

- Ressources

 - qml.qrc
 - /
 - main.qml
 - MainForm.ui.qml

```
1 TEMPLATE = app
2
3 QT += qml quick widgets
4
5 SOURCES += main.cpp
6
7 RESOURCES += qml.qrc
8
9 QMAKE_CXXFLAGS += -std=c++11
10 # Additional import path used to resolve QML modules in Qt Creator's code model
11 QML_IMPORT_PATH =
12
13 # Default rules for deployment.
14 include(deployment.pri)
15 |
```

Accueil
Éditeur
Design
Debug
Projets
Analyse
Aide

Hello world!

QtQuick
Debug

Type to locate (Ctrl...)

1 Problèmes 2 Search R... 3 Sortie d... 4 Sortie de... 5 Console ... 6 Message... 7 Version ...



2.3 - Introduction QT Creator : Qt QML



deployment.pri - QtQuick - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Line: 28, Col: 1

Projets

QtQuick

- QtQuick.pro
- deployment

 - deployment.pri
 - Sources

 - main.cpp

 - Ressources

 - qml.qrc

 - /

 - main.qml
 - MainForm.ui.qml

```
1 android-no-sdk {
2     target.path = /data/user/qt
3     export(target.path)
4     INSTALLS += target
5 } else:android {
6     x86 {
7         target.path = /libs/x86
8     } else: armeabi-v7a {
9         target.path = /libs/armeabi-v7a
10    } else {
11        target.path = /libs/armeabi
12    }
13    export(target.path)
14    INSTALLS += target
15 } else:unix {
16     isEmpty(target.path) {
17         qnx {
18             target.path = /tmp/$${TARGET}/bin
19         } else {
20             target.path = /opt/$${TARGET}/bin
21         }
22         export(target.path)
23     }
24     INSTALLS += target
25 }
26
27 export(INSTALLS)
28 |
```

Sortie de l'application

Type to locate (Ctrl...)

1 Problèmes 2 Search Re... 3 Sortie de ... 4 Sortie de ... 5 Console Q... 6 Messages... 7 Version C...

2.3 - Introduction QT Creator : Qt QML



main.cpp - QtQuick - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets

QtQuick

- QtQuick.pro
- deployment

 - deployment.pri

- Sources

 - main.cpp

- Ressources

 - qml.qrc
 - /
 - main.qml
 - MainForm.ui.qml

```
#include <QApplication>
#include <QQmlApplicationEngine>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

    return app.exec();
}
```

Sortie de l'application

Type to locate (Ctrl...)

1 Problèmes 2 Search Re... 3 Sortie de ... 4 Sortie de ... 5 Console Q... 6 Messages... 7 Version C...



2.3 - Introduction QT Creator : Qt QML



MainForm.ui.qml - QtQuick - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Bibliothèque Types QML Ressources Importations

État de base

Mon premier programme : QtQuick

Sortie de l'application

Propriétés

Type Text identifiant text1

Géométrie Position X 182 Y 161 Taille W 296 H 48

Visibilité Visibilité Est visible Retarder Opacité 1,00

Text Layout Avancé

Texte premier programme : QtQuick
Mode de dé... NoWrap
Alignment
Format AutoText

Couleur du texte Couleur du t... #000000

Couleur du style Couleur du s... #000000

Police Police Arial Taille 18 pixels Style de pol... B I U S Style Normal

Navigateur Item RowLayout button1 text1

Type to locate (Ctrl...)

Problèmes Search Re... Sortie de ... Sortie de ... Console Q... Messages... Version C...



2.3 - Introduction QT Creator : Qt QML



MainForm.ui.qml - QtQuick - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets QtQuick Sources Ressources

QtAccueil Éditer Design Debug Projets Analyse Aide

Hello world!

QtQuick Debug

MainForm.ui.qml

```
import QtQuick 2.4
import QtQuick.Controls 1.3
import QtQuick.Layouts 1.1

Item {
    width: 640
    height: 480

    property alias button1: button1

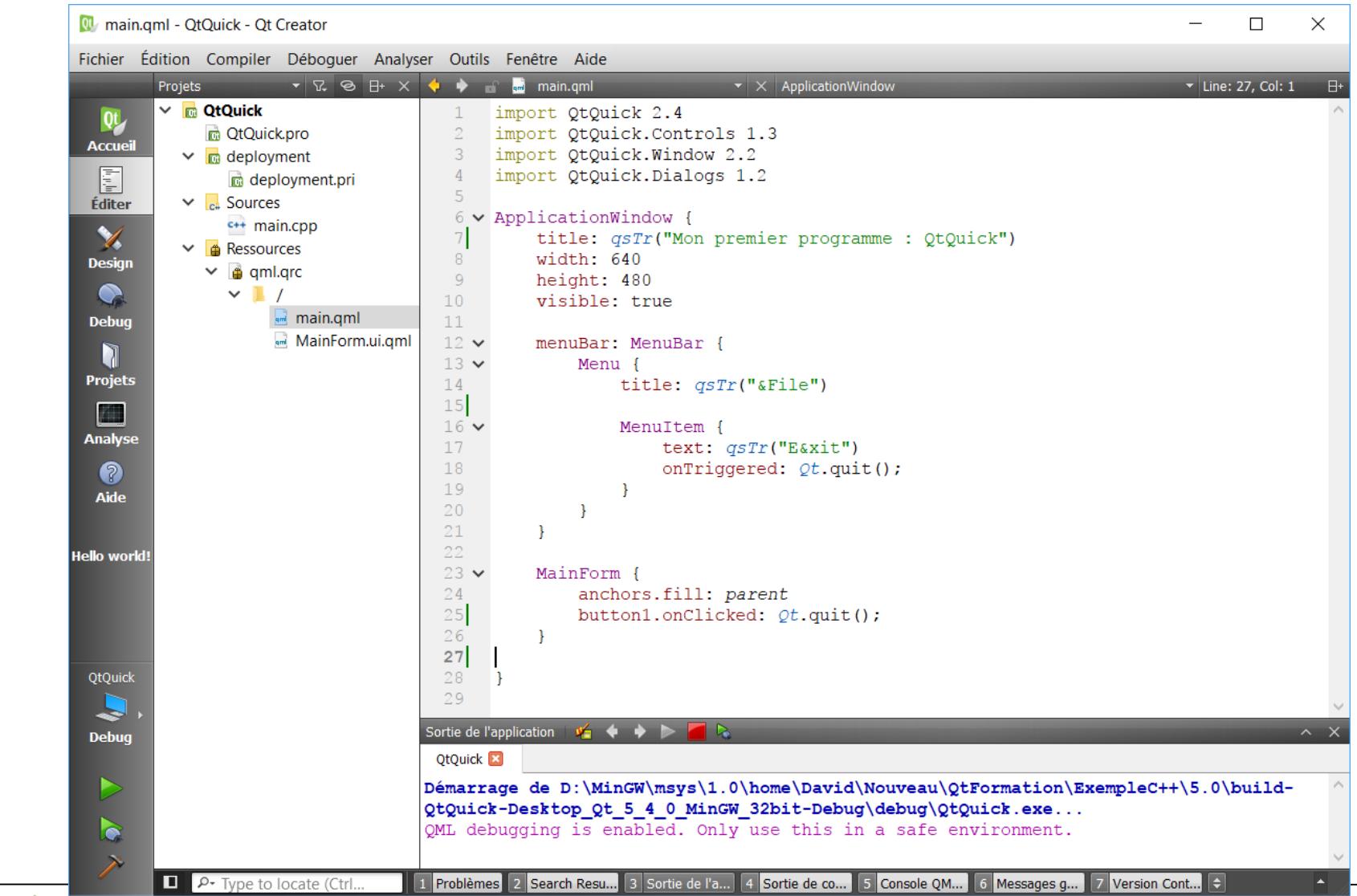
    RowLayout {
        anchors.verticalCenterOffset: 129
        anchors.horizontalCenterOffset: 1
        anchors.centerIn: parent

        Button {
            id: button1
            text: qsTr("Quit")
        }
    }

    Text{
        id: text1
        x: 182
        y: 161
        width: 296
        height: 48
        text: qsTr("Mon premier programme : QtQuick")
        horizontalAlignment: Text.AlignHCenter
        font.pixelSize: 18
    }
}
```

Type to locate (Ctrl...) 1 Problèmes 2 Search Resu... 3 Sortie de l'a... 4 Sortie de co... 5 Console QM... 6 Messages g... 7 Version Cont...

2.3 - Introduction QT Creator : Qt QML

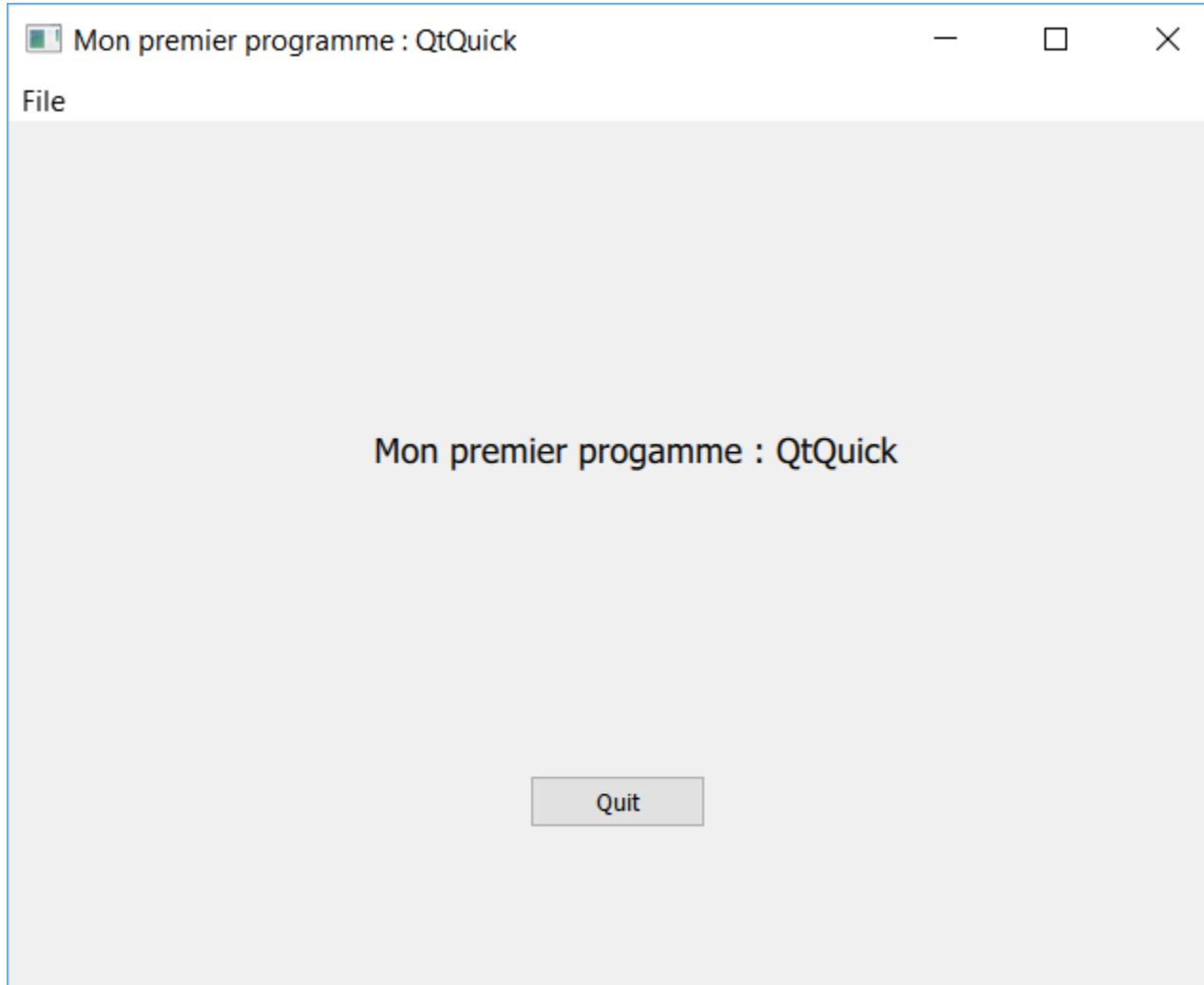


The screenshot shows the Qt Creator interface with a QML project named "main.qml". The left sidebar displays project files like QtQuick.pro, deployment.pri, main.cpp, and qml.qrc. The main editor window shows the QML code for an ApplicationWindow and MainForm. The bottom status bar indicates the application is running in debug mode on a desktop system.

```
1 import QtQuick 2.4
2 import QtQuick.Controls 1.3
3 import QtQuick.Window 2.2
4 import QtQuick.Dialogs 1.2
5
6 ApplicationWindow {
7     title: qsTr("Mon premier programme : QtQuick")
8     width: 640
9     height: 480
10    visible: true
11
12    menuBar: MenuBar {
13        Menu {
14            title: qsTr("&File")
15
16            MenuItem {
17                text: qsTr("E&xit")
18                onTriggered: qt.quit();
19            }
20        }
21    }
22
23    MainForm {
24        anchors.fill: parent
25        button1.clicked: qt.quit();
26    }
27
28
29 }
```



2.3 - Introduction QT Creator : Qt QML





- QObject
- Types de base : QChar, QDate, QString, QStringList, QTime, QTextStream...
- File systems : QDir, QFile...
- Container : QList, QMap, QPair, QSet, QVector...
- Graphique : QLine, Qpoint, QRect, QSize ...
- Thread : QThread, QMutex, QSemaphore ...
- Autres : QTimer, QTimeLine ...

QtCore contient plus de 200 classes

<http://doc.qt.io/qt-5/qtc...>



Ajouter dans le fichier .pro : QT += core

Remarque : tous les modules QT on besoin de QtCore



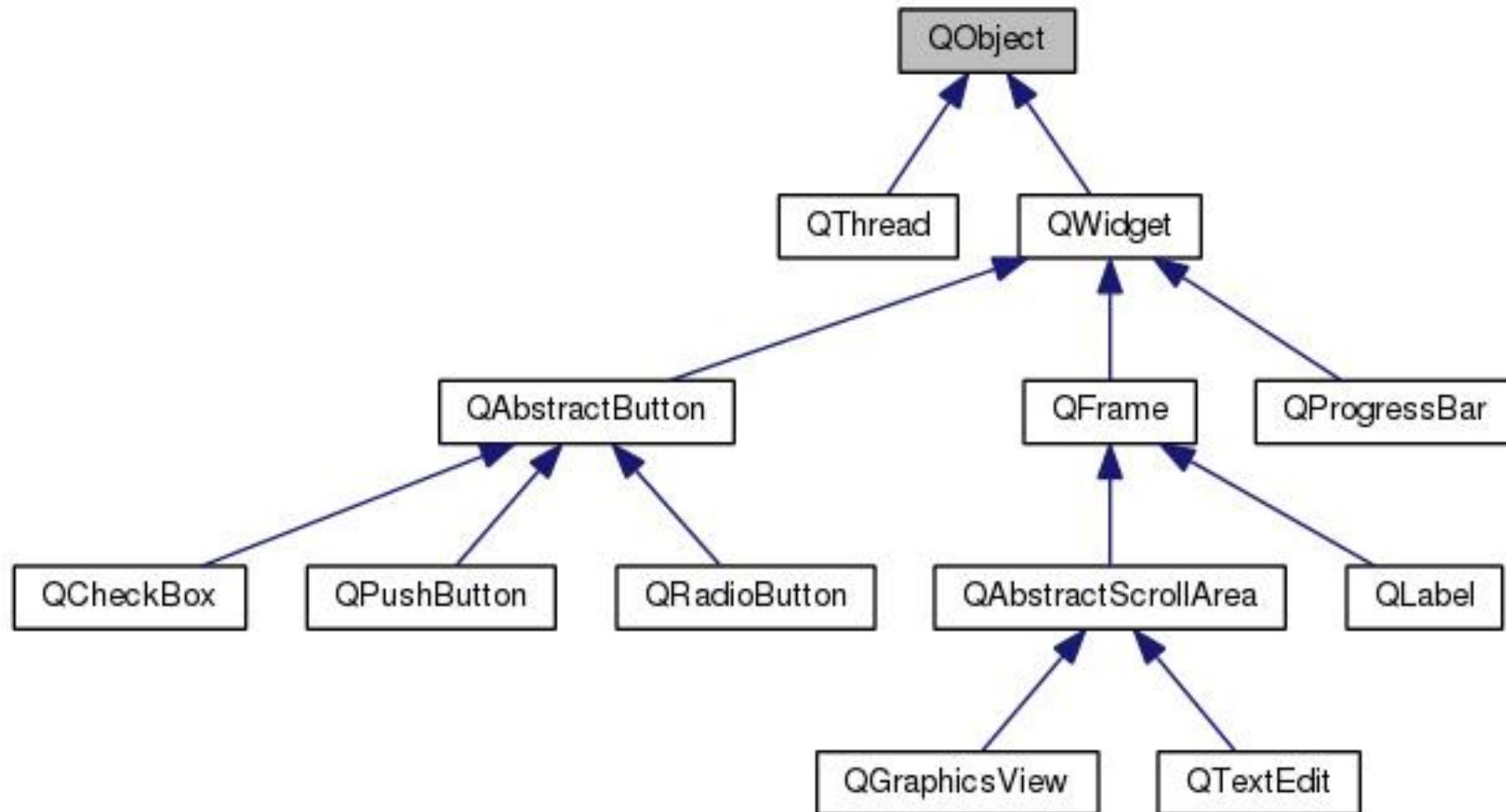
La classe **QObject** est la classe de base de tous les objets Qt. La caractéristique centrale de ce modèle est un mécanisme de communication homogène entre objets, appelé signaux et slots.

Vous pouvez :

- connecter un signal à un slot avec la fonction **connect()** et détruire cette connexion avec la fonction **disconnect()**.
- Vous pouvez les bloquer temporairement avec la fonction **blockSignals()**.

Les fonctions protégées **connectNotify()** et **disconnectNotify()** rendent possibles le suivi des connexions.

2.4 - Module Qt Core : QObject



https://wiki.qt.io/Qt_for_Beginners



Le système de méta-objet est basé sur trois éléments :

- La classe **QObject** fournit une classe de base pour les objets pouvant tirer parti du système de méta-objets.
- La macro **Q_OBJECT** à l'intérieur de la section privée de la déclaration de classe et qui est utilisée pour activer les fonctionnalités de méta-objet, telles que les propriétés dynamiques, les signaux et les emplacements.
- Le compilateur de méta-objets (moc) fournit à chaque sous-classe QObject le code nécessaire pour implémenter les fonctionnalités de méta-objet.

Header:	<code>#include <QMetaObject></code>
qmake:	<code>QT += core</code>



La classe QObject permet la gestion des timers, afin de permettre des actions différées.

lancer un timer :

```
public bool  
startTimer(int délaiEnMillisecondes)
```

arriver à échéance du timer appelle de la fonction:

```
protected void  
timerEvent(QTimerEvent*)
```

```
class UneClasse : public QObject {  
  
public:  
    UneClasse() {}  
    ~UneClasse() {}  
    UneClasse(const UneClasse&) = delete;  
    UneClasse& operator=(const UneClasse& val) = delete;  
  
    void démarrer() {  
        qDebug() << "Démarrage du timer (3 secondes)";  
        if(!startTimer(3*1000)) {  
            qWarning() << "Le timer n'a pas démarré";  
        }  
    }  
  
protected:  
    void timerEvent(QTimerEvent* event) {  
        qDebug() << "Le timer a été appelé sur l'objet";  
        //Destruction du timer  
        killTimer(event->timerId());  
    }  
};
```

2.4 - Module Qt Core : QMetaObject



La classe **QMetaObject** contient des méta-information sur les objets Qt.

Chaque classe qui hérite de **QObject** a un méta-objet associé, une instance de la classe **QMetaObject**.

Les méthodes que vous êtes le plus susceptibles de trouver utiles sont les suivantes:

- **className ()** renvoie le nom d'une classe.
- **superClass ()** retourne le méta-objet de la superclasse.
- **method ()** et **methodCount ()** fournissent des informations sur les méta-méthodes d'une classe (signaux, slots et autres fonctions membres).
- **enumerator ()** et **enumeratorCount ()** fournissent des informations sur les énumérateurs d'une classe.
- **propertyCount ()** et **property ()** fournissent des informations sur les propriétés d'une classe.

Les méthodes d'index **indexOfMethod ()**, **indexOfEnumerator ()** et **indexOfProperty ()** mappent les noms des fonctions membres, des énumérateurs ou des propriétés aux index du méta-objet.

Entête:	#include <QMetaObject>
qmake:	QT += noyau

<http://doc.qt.io/qt-5/qmetaobject.html>



- **qDebug ()** est utilisé pour écrire une sortie de débogage personnalisée.
- **qInfo ()** est utilisé pour les messages d'information.
- **qWarning ()** est utilisé pour signaler les avertissements et les erreurs récupérables dans votre application.
- **qCritical ()** est utilisé pour écrire les messages d'erreur critiques et signaler les erreurs du système.
- **qFatal ()** est utilisé pour écrire des messages d'erreur fatale peu avant

```
qDebug () << "Widget" << widget << "à la position" << widget -> pos ();
```

<https://doc.qt.io/qt-5.10/debug.html>

2.4 - Module Qt Core : Manipulation de fichier



```
// référence : https://qt.developpez.com/faq/?page=modules-qtcore-fichiers-lectureecriture
// Création d'un objet QFile
QString fileName = "Qt.txt";
QFile file(fileName);
// On ouvre notre fichier en lecture seule et on vérifie l'ouverture
if (!file.open(QIODevice::WriteOnly | QIODevice::Text))
    return 1;

// Création d'un objet QTextStream à partir de notre objet QFile
QTextStream flux(&file);
// On choisit le codec correspondant au jeu de caractères que l'on souhaite ; ici, UTF-8
flux.setCodec("UTF-8");
// Écriture des différentes lignes dans le fichier
flux << "Bonjour," << endl <<
    "Nous sommes le " << QDate::currentDate().toString("ddd d MMMM yyyy") << endl;
flux << QString("àéé").toUtf8() << endl;
file.close();
```

2.4 - Module Qt Core : Manipulation de fichier



```
// reference : https://qt.developpez.com/faq/?page=modules-qtcore-fichiers-lectureecriture
QString fileName = "Qt.txt";
QFile file(fileName);
file.open(QIODevice::ReadOnly | QIODevice::Text);
QTextStream flux(&file);
flux.setCodec("UTF-8");

QString ligne;

QTextCodec::setCodecForLocale( QTextCodec::codecForName("IBM-850") );

while(! flux.atEnd())
{
    ligne = flux.readLine();
    qDebug() << ligne;
    //traitement de la ligne
}

// QString tout = flux.readAll();
file.close();
```



La classe QVariant agit comme une union des types de données les plus courants Qt.

```
#include <QDebug>

int main(int argc, char *argv[])
{
    QVariant v(123);                      // The variant now contains an int
    int x = v.toInt();                    // x = 123
    qDebug() << v << " int " << x ;
    v = QVariant("hello");                // The variant now contains a QByteArray
    v = QVariant(QObject::tr("hello"));    // The variant now contains a QString
    int y = v.toInt();                  // y = 0 since v cannot be converted to an int
    QString s = v.toString();            // s = tr("hello") (see QObject::tr())
    qDebug() << v << " int " << y << " QString " << s << "\n";
    return 0;
}
```

```
QVariant(int, 123)  int  123
QVariant(QString, "hello")  int  0  QString  "hello"
```

http://www.bogotobogo.com/Qt/Qt5_QVariant_meta_object_system_MetaData.php



Types

typedef	<code>QFunctionPointer</code>		
typedef	<code>QtMessageHandler</code>		
enum	<code>QtMsgType</code> { <code>QtDebugMsg</code> , <code>QtInfoMsg</code> , <code>QtWarningMsg</code> , <code>QtCriticalMsg</code> , <code>QtFatalMsg</code> , <code>QtSystemMsg</code> }		
typedef	<code>qint8</code>		
typedef	<code>qint16</code>		
typedef	<code>qint32</code>		
typedef	<code>qint64</code>		
typedef	<code>qintptr</code>		
typedef	<code>qlonglong</code>		
typedef	<code>qptrdiff</code>	8 bits	<code>quint8</code>
typedef	<code>qreal</code>	16 bits	<code>quint16</code>
typedef	<code>quint8</code>	32 bits	<code>quint32</code>
typedef	<code>quint16</code>		<code>quint64</code>
typedef	<code>quint32</code>		<code>qlonglong</code>
typedef	<code>quint64</code>	64 bits	
typedef	<code>quintptr</code>		
typedef	<code>qulonglong</code>		
typedef	<code>uchar</code>		
typedef	<code>uint</code>	Pointeur	<code>quintptr</code>
typedef	<code>ulong</code>		
typedef	<code>ushort</code>		

2.4 - Module Qt Core : Macros QT



QT_POINTER_SIZE	Correspond à la taille d'un pointeur en octets (4 ou 8 octets). Équivalent à sizeof(void*).
QT_REQUIRE_VERSION <code>(int argc, char** argv, const char * version)</code>	Permet de s'assurer que l'application s'exécute avec une version suffisamment récente de Qt
QT_VERSION	Correspond à la représentation « compressée » de Qt. Par exemple, la version 5.0.1 sera transcrive 0x050001, la 4.8.3 sera transcrive 0x040803.
QT_VERSION_STR	Équivalent à un appel de la fonction qVersion().
Q_ASSERT(bool)	Permet en phase de débogage, de faire échouer le programme si la condition n'est pas réalisée.
Q_BYTE_ORDER	Peut être utilisée pour déterminer l'ordre des octets que votre système utilise pour le stockage des données en mémoire, c'est-à-dire si votre système est petit boutiste ou grand boutiste. Il est défini par Qt à la valeur d'une des macros Q_LITTLE_ENDIAN ou Q_BIG_ENDIAN.
Q_BIG_ENDIAN	Cette macro représente une valeur que vous pouvez comparer avec la macro Q_BYTE_ORDER pour déterminer le boutisme de votre système. Dans un système grand boutiste, l'octet significatif est stocké à l'adresse la plus basse. Les autres octets suivront dans l'ordre décroissant d'importance.



2.4 - Module Qt Core : Macros QT



Q_LITTLE_ENDIAN	Cette macro représente une valeur que vous pouvez comparer avec la macro Q_BYTE_ORDER pour déterminer le boutisme de votre système. Dans un système petit boutiste, l'octet de poids faible est stocké à l'adresse de poids faible. Les autres octets suivent dans l'ordre de poids croissant.
Q_DECL_IMPORT, Q_DECL_EXPORT	Constantes pour les déclarations de fonctions exportables dans les DLL de Windows.
Q_OS_(ANDROID, CYGWIN, DARWIN, IOS, LINUX, MAX, MACOS, UNIX, WIN, WIN32, WIN64, WINRT)	Constante correspondant au système d'exploitation sur lequel tourne l'application. Si vous voulez qu'une partie de votre programme ne soit exécutée que sous Linux utilisez #ifdef(Q_OS_LINUX).
Q_PROCESSOR_(X86, ALPHA, ARM, ARM_V5, ARM_V6, ARM_V7, IA64, MIPS, MIPS_32, MIPS_64, MIPS_I à MIPS_V, X86_32, X86_64)	Constante correspondant à l'architecture sur laquelle tourne l'application. Si vous voulez qu'une partie de votre programme ne soit exécutée que sur une architecture compatible Intel 64 bits, utilisez #ifdef(Q_PROCESSOR_X86_64).
forever Q_FOREVER	Cette macro est fournie par commodité pour l'écriture de boucles infinies.

<http://qt.apidoc.info/5.2.0/qtcore/qtglobal.html>



2.4 - Module Qt Core : Macros QT



```
QT_DISABLE_DEPRECATED_BEFORE
QT_POINTER_SIZE
QT_REQUIRE_VERSION(int argc, char ** argv, const char * version)
QT_TRANSLATE_NOOP3( context, sourceText, comment)
QT_TRANSLATE_NOOP( context, sourceText)
QT_TRID_NOOP( id)
QT_TR_NOOP( sourceText)
QT_VERSION
QT_VERSION_CHECK
QT_VERSION_STR
void Q_ASSERT(bool test)
void Q_ASSERT_X(bool test, const char * where, const char * what)
void Q_ASSUME(bool expr)
Q_BIG_ENDIAN
Q_BYTE_ORDER
Q_CC_BOR
Q_CC_CDS
Q_CC_COKEAU
Q_CC_DEC
Q_CC_EDG
Q_CC_GHS
Q_CC_GNU
Q_CC_HIGHC
Q_CC_HPACC
Q_CC_INTEL
Q_CC_KAI
Q_CC_MIPS
Q_CC_MSVC
Q_CC_OC
Q_CC_PGI
Q_CC_SUN
Q_CC_SYM
Q_CC_USLC
Q_CC_WAT
void Q_CHECK_PTR(void * pointer)
Q_DECLARE_TYPEINFO( Type, Flags)
Q_DECL_CONSTEXPR
Q_DECL_EXPORT
Q_DECL_FINAL
Q_DECL_IMPORT
Q_DECL_NOEXCEPT
Q_DECL_NOEXCEPT_EXPR
```

<http://qt.apidoc.info/5.2.0/qtcore/qtglobal.html>



- Définition : Thread
- Threads et multitâches
- Avantages et inconvénients
- Crédit et exécution des threads
- Annulation et fin des threads
- Définition : Mutex
- Fonctions : Mutex



Un thread ou fil (d'exécution) ou tâche ou processus léger est similaire à un processus, car tous deux représentent l'exécution d'un ensemble d'instructions du langage machine d'un processeur.

Du point de vue de l'utilisateur, ces exécutions semblent se dérouler en parallèle.

Toutefois, là où chaque processus possède sa propre mémoire virtuelle, les threads d'un même processus se partagent sa mémoire virtuelle.

Par contre, tous les threads possèdent leur propre pile d'appel.

[http://fr.wikipedia.org/wiki/Thread_\(informatique\)](http://fr.wikipedia.org/wiki/Thread_(informatique))



Les threads se distinguent du **multi-processus** plus classique par le fait que deux processus sont typiquement **indépendants** et peuvent interagir uniquement à travers une API fournie par le système telle que IPC.

D'un autre côté les **threads** partagent une information sur l'état du processus, des zones de mémoires ainsi que d'autres ressources. Puisqu'il n'y a pas de changement de mémoire virtuelle, la commutation de contexte entre 2 threads est **moins coûteuse** que la commutation de contexte entre 2 processus.

On peut y voir un avantage de la programmation utilisant des threads multiples.

2.4 - Module Qt Core : Thread -> Avantages et inconvénients



Dans certains cas, les programmes utilisant des threads sont plus rapides que des programmes architecturés plus classiquement, en particulier sur les machines multi-processeurs.

Le principal surcoût dû à l'utilisation de processus multiples provient de la communication entre processus séparés.

Le partage de certaines ressources entre **threads** permet une communication plus efficace entre les différents threads d'un processus. Là où deux **processus** séparés doivent utiliser un **mécanisme** fourni par le **système** pour communiquer car **les threads partagent une partie de l'état du processus**.



La programmation utilisant des threads est plus difficile que la programmation multi-processus car l'accès à certaines ressources partagées doit être **restreint par le programme lui-même**, pour éviter que l'état d'un processus ne devienne temporairement incohérent, tandis qu'un autre thread va avoir besoin de consulter cette portion de l'état du processus.

Il est donc obligatoire de mettre en place des mécanismes de synchronisation (à l'aide de sémaphores par exemple), tout en conservant à l'esprit que l'utilisation de la synchronisation peut aboutir à des situations d'inter-blocage ce qui augmente la complexité du programme.

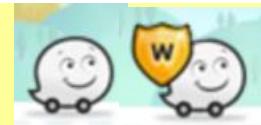


- QThread::create
- QThread:: start
- QThread:: wait

```
QTextCodec::setCodecForLocale( QTextCodec::codecForName("IBM-850") );
QCoreApplication a(argc, argv);
{
    QThread* thread = QThread::create([&](){ qDebug() << " traitement à faire ...."; thread->exit(); } );
    thread->start();
    thread->wait();
    delete thread;
    qDebug() << " Fin \n\n";
}
```

traitement à faire
Fin

2.4 - Module Qt Core : Thread



```
class MonTraitement : public QThread
{
    Q_OBJECT
public:
    MonTraitement(QObject *parent=nullptr):QThread(parent)
    {}

    void run()
    {
        qDebug() << " traitement à faire ....";
        for ( int i = 0; i < 2; ++i)
            qDebug() << int(this->currentThreadId())<< " " << i;

    }

    ~MonTraitement() { qDebug() << "Fin"; }
};
```

```
{
    MonTraitement thread ;
    thread.start();
    thread.wait();
}
```

```
traitement à faire ....
17316  0
17316  1
Fin
```



2.4 - Module Qt Core : Thread



```
class ExempleArretThread : public QThread
{
    Q_OBJECT
public:
    ExempleArretThread(QObject *parent = nullptr);
    void run();
    ~ExempleArretThread();
};
```

void QThread::usleep(unsigned long usecs)
-> microseconde

void QThread::msleep(unsigned long msecs)
-> milliseconde

void QThread::sleep(unsigned long secs)
-> seconde

```
{
    ExempleArretThread thread ;
    thread.start();
    thread.sleep(6);
    thread.terminate();
}
```

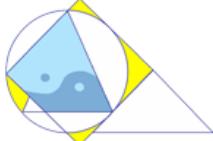
```
ExempleArretThread::ExempleArretThread(QObject *parent)
:QThread(parent)
{}

void ExempleArretThread::run()
{
    qDebug() << " traitement à faire ....";
    for ( int i = 1; i <=5; ++i)
    {
        qDebug() << i << " secondes";
        QThread::sleep(1);
    }
    qDebug() << "\n";
}

ExempleArretThread::~ExempleArretThread()
{
    qDebug() << "Fin";
}
```

```
traitement à faire ....
1 secondes
2 secondes
3 secondes
4 secondes
5 secondes

Fin
```



2.4 - Module Qt Core : Thread



Yantra Technologies

```
class ExempleArretThread_2 : public QThread
{
    Q_OBJECT
public:
    ExempleArretThread_2(MonTraitement* mt) :QThread(), a_traitement(mt) {}

    void run()
    {
        qDebug() << " ExempleArretThread_2::run " << int ( this->currentThreadId() ) ;
        for ( int i = 1; i <=5; ++i)
            a_traitement->run();
    }

    ~ExempleArretThread_2() { qDebug() << "Fin"; }

private:
    MonTraitement* a_traitement;
};

{

    MonTraitement mt ;
    ExempleArretThread_2 thread1(&mt) ;
    ExempleArretThread_2 thread2(&mt) ;
    ExempleArretThread_2 thread3(&mt) ;

    thread1.start();
    thread2.start();
    thread3.start();

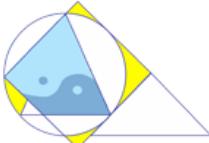
    thread1.wait();
    thread2.wait();
    thread3.wait();

}
```

```
ExempleArretThread_2::run  17332
traitement à faire ....
ExempleArretThread_2::run  1284
traitement à faire ....
17332  0
17332  1
traitement à faire ....
1284  0
ExempleArretThread_2::run  9616
1284  1
traitement à faire ....
traitement à faire ....
9616  0
1284  0
9616  1
1284  1
traitement à faire ....
traitement à faire ....
9616  0
1284  0
9616  1
1284  1
traitement à faire ....
traitement à faire ....
9616  0
1284  0
9616  1
1284  1
traitement à faire ....
traitement à faire ....
9616  0
1284  0
9616  1
1284  1
traitement à faire ....
9616  0
9616  1
Fin
Fin
Fin
Fin
```



Yantra Technologies



2.4 - Module Qt Core : Thread -> QMutex



```
class ExempleArretThread_3 : public QThread
{
    Q_OBJECT
    static QMutex mutex;
public:
    ExempleArretThread_3(MonTraitement* mt) :QThread(), a_traitement(mt) {}

    void run()
    {
        mutex.lock();
        qDebug() << " ExempleArretThread_3::run " << int ( this->currentThreadId() ) ;
        for ( int i = 1; i <=5; ++i)
            a_traitement->run();
        mutex.unlock();
    }

    ~ExempleArretThread_3() { qDebug() << "Fin"; }

private:
    MonTraitement* a_traitement;
};
```

```
{
    MonTraitement mt ;
    ExempleArretThread_3 thread1(&mt) ;
    ExempleArretThread_3 thread2(&mt) ;
    ExempleArretThread_3 thread3(&mt) ;

    thread1.start();
    thread2.start();
    thread3.start();

    thread1.wait();
    thread2.wait();
    thread3.wait();
```

```
ExempleArretThread_3::run 12072
traitement à faire ....
12072  0
12072  1
ExempleArretThread_3::run 11312
traitement à faire ....
11312  0
11312  1
ExempleArretThread_3::run 3860
traitement à faire ....
3860  0
3860  1
Fin
Fin
Fin
Fin
Fin
```



2.4 - Module Qt Core : Thread -> QSemaphore



```

class ExempleArretThread_4 : public QThread
{
    Q_OBJECT
    static QSemaphore mutex;
public:
    ExempleArretThread_4(MonTraitement* mt) :QThread(), a_traitement(mt) {}

    void run()
    {
        mutex.acquire(1);
        qDebug() << " ExempleArretThread_4::run " << int ( this->currentThreadId() ) ;
        for ( int i = 1; i <=5; ++i)
            a_traitement->run();
        mutex.release(1);
    }

    ~ExempleArretThread_4() { qDebug() << "Fin"; }

private:
    MonTraitement* a_traitement;
};

```

```

// QSemaphore ExempleArretThread_4::mutex(1);

MonTraitement mt ;
ExempleArretThread_4 thread1(&mt) ;
ExempleArretThread_4 thread2(&mt) ;
ExempleArretThread_4 thread3(&mt) ;

thread1.start();
thread2.start();
thread3.start();

thread1.wait();
thread2.wait();
thread3.wait();

```

```

ExempleArretThread_4::run 16700
traitement à faire ....
16700 0
16700 1
ExempleArretThread_4::run 13060
traitement à faire ....
13060 0
13060 1
ExempleArretThread_4::run 19820
traitement à faire ....
19820 0
19820 1
Fin
Fin
Fin
Fin

```



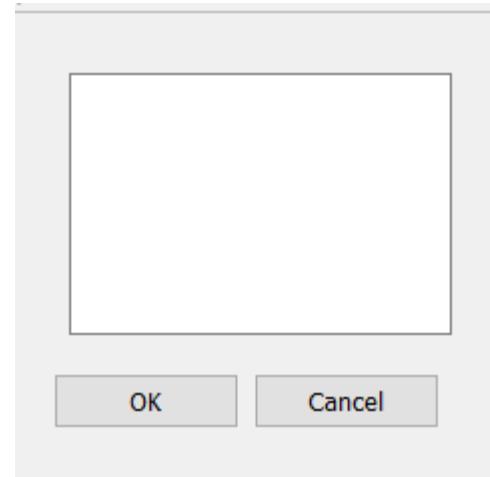
La programmation événementielle est architecturée autour d'une boucle principale fournie et divisée en deux sections : la première section détecte les événements, la seconde les gère.

Pour chaque événement à gérer, il faut lui associer une action à réaliser (fonction ou méthode) : c'est le gestionnaire événement (handler).

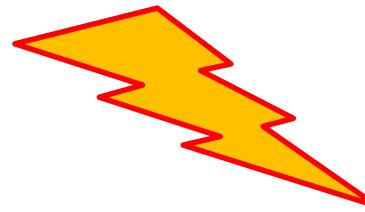
Ensuite, à chaque fois que l'événement sera détecté par la boucle d'événements, le gestionnaire d'événements sera alors exécuté.



Widgets (objet Qt)



Signal (clicked())



Slot (méthode)

```
void MainWindow::on_buttonBox_clicked(QAbstractButton *button)
{
    std::cout << "Slot : BoutonBox " <<
        button->text().toStdString() <<
    std::endl;
}
```



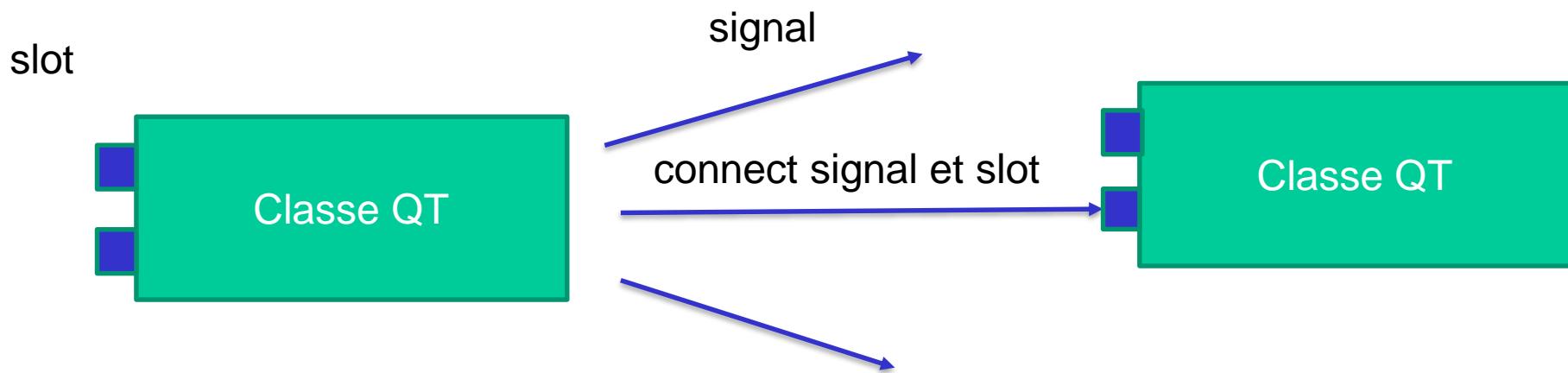
Les signaux et slots permettent d'interconnecter des objets Qt entre eux

- un **signal** est émis lorsqu'un événement particulier se produit.
- un **slot** est une **fonction** qui va être appelée en réponse à un **signal** particulier.
- L'association d'un **signal** à un **slot** est réalisée par une **connexion** avec l'appel **connect()**.



Tout objet Qt (dérivant de QObject) peut définir des signaux, et des slots pour recevoir des signaux en provenance d'autres objets Qt.

Dans QT une classe est un composant qui peut envoyer et recevoir des données sans le savoir.



2.4 - Module Qt Core : Qt Programmation événementielle

→ signal et slot : exemple



```
#include <QObject>
#include "personne.h"
#include <memory>
#include <iostream>

namespace PControleur {

    class Communication : public QObject
    {
        Q_OBJECT
public:
    explicit Communication(QObject *parent = 0);
    ~Communication();

    Q_PROPERTY(QString name READ getName WRITE setName NOTIFY nameChanged)
    Q_INVOKABLE QString toStringPersonne() const;
    static void DeclareQML();

signals:
    void change_Personne();
    void create_Personne();
    void nameChanged(QString );
public slots:
    void add_Personne(const QString& nom, const QString& prenom, const QString& adresse);
    void mod_Personne(const QString& nom, const QString& prenom, const QString& adresse);

    const QString &getName() const { return a_nameCurrent; }
    void setName(const QString &name) { a_nameCurrent=name; emit nameChanged("OK : nom changer"); }

    void voir(QString name) const { std::cout << name.toStdString() << std::endl; }
private:
    std::shared_ptr<PPersonne::Personne> current;
    QString a_nameCurrent;
};

}

#endif // COMMUNICATION_H
```



Définir un signal

- Un signal est défini dans le fichier d'en-tête déclarant la classe :

```
signals:  
    void change_Personne();  
    void create_Personne();  
    void nameChanged(QString );
```

2.4 – Module Qt Core : Q_PROPERTY



La macro **Q_PROPERTY** permet de déclarer des propriétés dans les classes qui héritent de **QObject**

- **READ** permet de définir le nom de la fonction qui retournera la valeur du membre.
- **WRITE** permet de définir le nom de la fonction qui écrira une nouvelle valeur dans le membre.
- **MEMBER** permet de définir le nom du membre privé correspondant. Si vous utilisez ce mot-clé, il est inutile d'utiliser les mots-clés READ et WRITE.
- **RESET** permet de définir le nom d'une fonction qui réinitialisera la valeur du membre à sa valeur naturelle, `QString("")` pour une chaîne de caractères, ou 0 pour un entier.
- **NOTIFY** permet de définir le nom d'un signal qui sera émis dès que la valeur du membre aura été modifiée.
- **CONSTANT** permet de définir le membre comme constant, le mot-clé WRITE est donc incompatible avec celui-ci.
- **FINAL** permet d'indiquer que les accesseurs ne peuvent pas être surchargés.

```
Q_PROPERTY(  
    type nom  
    READ getter [WRITE setter]  
    | MEMBER membre [READ getter | WRITE setter]  
    RESET reset  
    NOTIFY signal  
    CONSTANT bool  
    FINAL bool  
)
```



2.4 – Module Qt Core : Exemple Q_PROPERTY



```
#ifndef MYCLASS_H
#define MYCLASS_H

#include <QObject>

class MyClass : public QObject
{
    Q_OBJECT
    Q_PROPERTY(Priority priority MEMBER a_priority READ priority WRITE setPriority)
    Q_PROPERTY(QString string READ toString)

    Q_ENUMS(Priority)

public:
    MyClass(QObject* parent = nullptr)
        :QObject(parent),
         a_priority(Priority::High) {}
    ~MyClass() {}

    enum Priority : int { High=0, Low=1, VeryHigh=2, VeryLow=3 };

    void setPriority(Priority priority) { this->a_priority= priority; }
    Priority priority() const { return this->a_priority; }

    QString toString() const {
        QString res;
        res = QString::number(a_priority);
        return res;
    }

private:
    Priority a_priority;
};

#endif // MYCLASS_H
```



2.4 – Module Qt Core : Exemple Q_PROPERTY



```
#include "myclass.h"

#include <QCoreApplication>
#include <QDebug>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MyClass *myinstance = new MyClass();
    QObject *object = myinstance;
    qDebug() << "MyClass : " << myinstance->toString();
    qDebug() << "QObject : " << object->property("string");
    qDebug() << "QObject : " << object->property("string").toString();

    myinstance->setPriority(MyClass::VeryHigh);
    object->setProperty("priority", "VeryHigh");

    qDebug() << "MyClass : " << myinstance->toString();
    qDebug() << "QObject : " << object->property("string");
    qDebug() << "QObject : " << object->property("string").toString();

    return a.exec();
}
```

```
MyClass : "0"
QObject : QVariant(QString, "0")
QObject : "0"
MyClass : "2"
QObject : QVariant(QString, "2")
QObject : "2"
```



Définir un slot

- Un slot est similaire à une méthode de la classe,
- Il doit être déclaré dans l'en-tête, dans une section slots,
- Il doit être implémenté dans la définition de la classe.

```
Q_PROPERTY(QString name READ getName WRITE setName NOTIFY nameChanged)  
Q_INVOKABLE QString toStringPersonne() const ;
```

```
public slots:  
void add_Personne(const QString& nom, const QString& prenom, const QString& adresse);  
void mod_Personne(const QString& nom, const QString& prenom, const QString& adresse);  
  
const QString &getName() const { return a_nameCurrent; }  
void setName(const QString &name) { a_nameCurrent=name; emit nameChanged("OK : nom changer"); }  
  
void voir(QString name) const { std::cout << name.toStdString() << std::endl; }
```

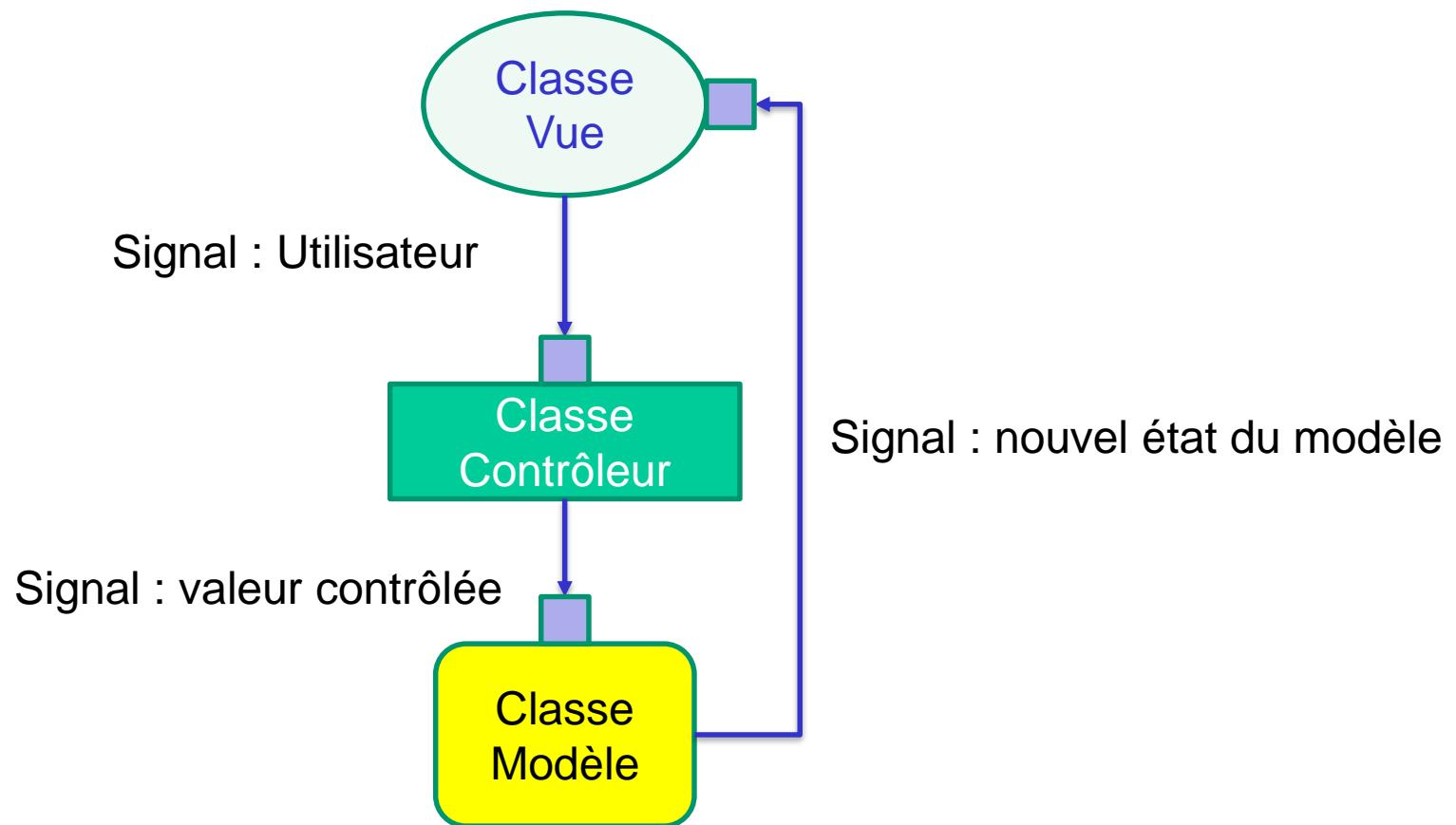


La méthode `QObject::connect` connecte un signal avec un slot.

Le **signal** et le **slot** doivent avoir la même signature.

```
QObject::connect (this, SIGNAL(nameChanged(QString)) ,  
                  this, SLOT(voir(QString))) ;
```

2.4 – Module Qt Core : MVC en Qt



2.4 - Module Qt Core : les containers



QList<T>	Liste accessible par index qui privilégie la vitesse de lecture. Les données peuvent être ajoutées au début, à la fin ou à l'intérieur.
QLinkedList<T>	Listes chaînées offrant de meilleures performances dans l'insertion d'éléments au centre d'une longue liste.
QVector<T>	Similaire à QList en stockant les données de manière contiguë.
QStack<T>	Structure du type QVector LIFO (Last In First Out).
QQueue<T>	Structure du type QList FIFO (First In First Out).
QSet<T>	Stocke les données de manière hachée pour les retrouver très rapidement.
QMap <T>	Stocke les paires (clé, valeur) et fournit une recherche rapide de la valeur associée à une clé.

2.4 - Module Qt Core : les containers



```
QList<int> List;  
  
for(int i = 0; i < 6; i++) List.append(i);  
List.removeOne(4);  
foreach(int n, List) qDebug() << n;
```

```
QVector<QString> vector(0);  
vector.append("one");  
vector.append("two");  
vector.append("three");  
for ( auto val:vector)  
    qDebug() << val;
```

2.4 - Module Qt Core : les containers



```
QMap<int, QString> Map;  
  
Map.insert(1,"A");  
Map.insert(2,"B");  
Map[3] = "C";  
  
foreach(int i, Map.keys()) qDebug() << Map[i];  
  
QMapIterator<int, QString> iter(Map);  
  
while(iter.hasNext())  
{  
    iter.next();  
    qDebug() << iter.key() << " : " << iter.value();  
}
```

2.4 - Module Qt Core : Internationalisation



Qt intègre son propre système de traduction. La prise en charge de plusieurs langues est extrêmement simple dans des applications Qt et ajoute peu de surcharge de travail pour le programmeur.

Selon le manuel de Qt Linguist, l'internationalisation est assurée pour trois types de personnes :

- les développeurs,
- le chef de projet,
- les traducteurs.

<http://doc.qt.io/qt-5/qtlinguist-index.html>



Les développeurs entrent des chaînes de caractères dans leur propre langue. Ils doivent permettre la traduction de ces chaînes grâce à la méthode **tr()**.

```
QString chaine = "Une chaine ne pouvant pas être traduite";
QString chainetraduisible = QObject::tr("Une chaine pouvant être traduite") ;
QString chainetraduisiblecommentaire = QObject::tr("Une chaine pouvant être traduite","commentaire") ;

qDebug() << chaine;
qDebug() << chainetraduisible;
qDebug() << chainetraduisiblecommentaire;
```

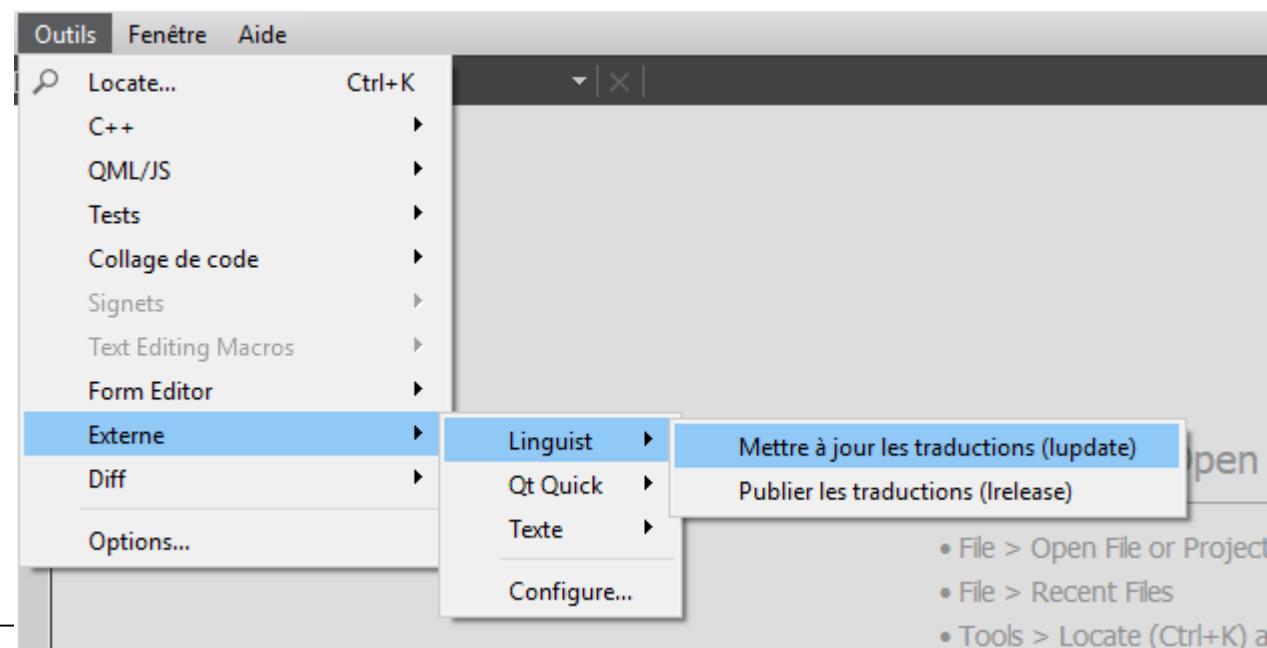
2.4 - Module Qt Core : Internationalisation



Le chef de projet déclare les fichiers de traduction (un pour chaque langue) dans le fichier de projet (.pro). L'exportation des textes se fait dans Qt Creator avec l'utilitaire lupdate qui génère les fichiers de traduction en XML avec l'extention « .ts »

```
TRANSLATIONS += \
    fr_FR.ts \
    en_EN.ts
```

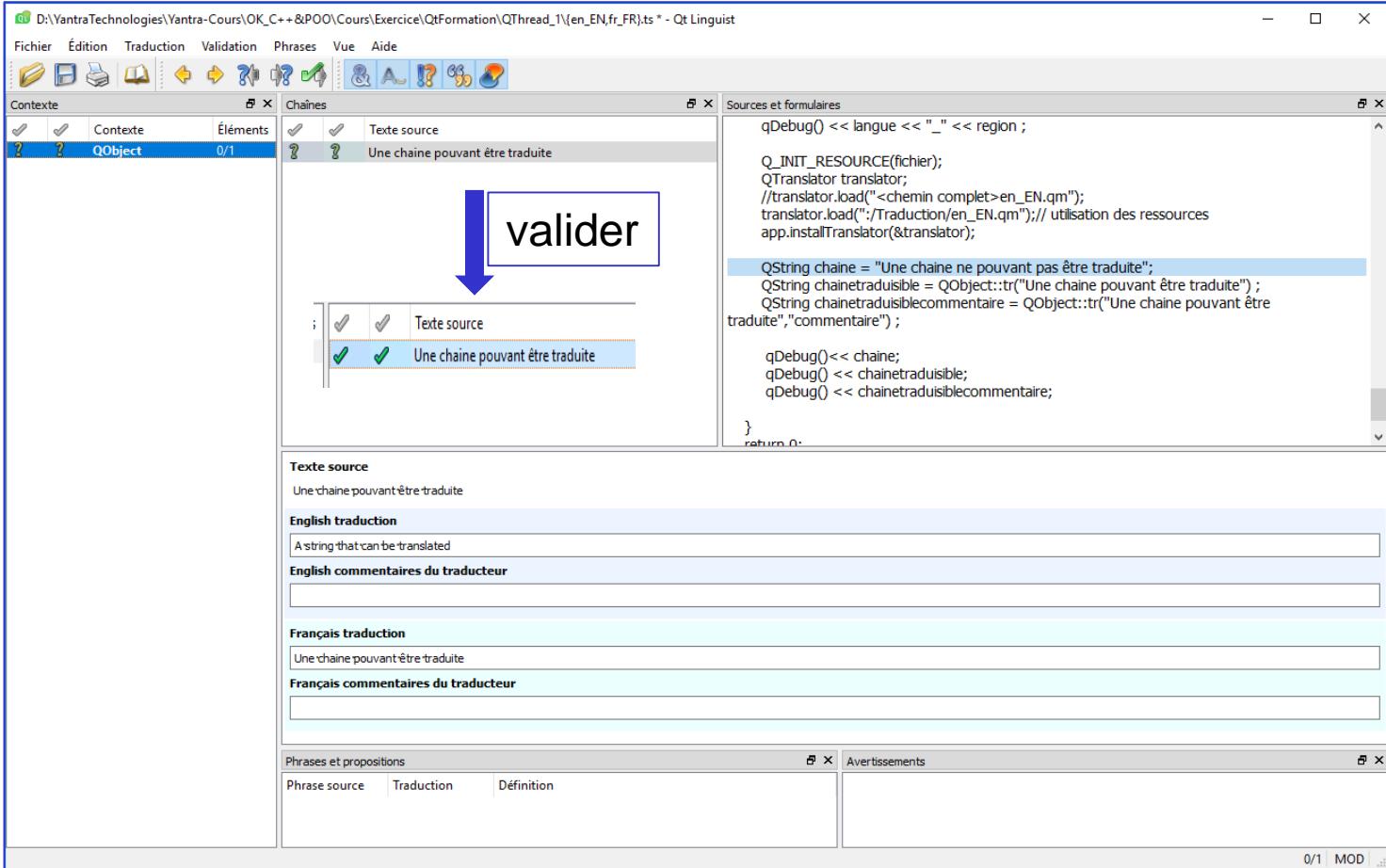
A rajouter dans le .pro



2.4 - Module Qt Core : Internationalisation



Les traducteurs utilisent Qt Linguist pour renseigner les fichiers de traduction.
 « linguist » qui se trouvent dans les binaire QT.

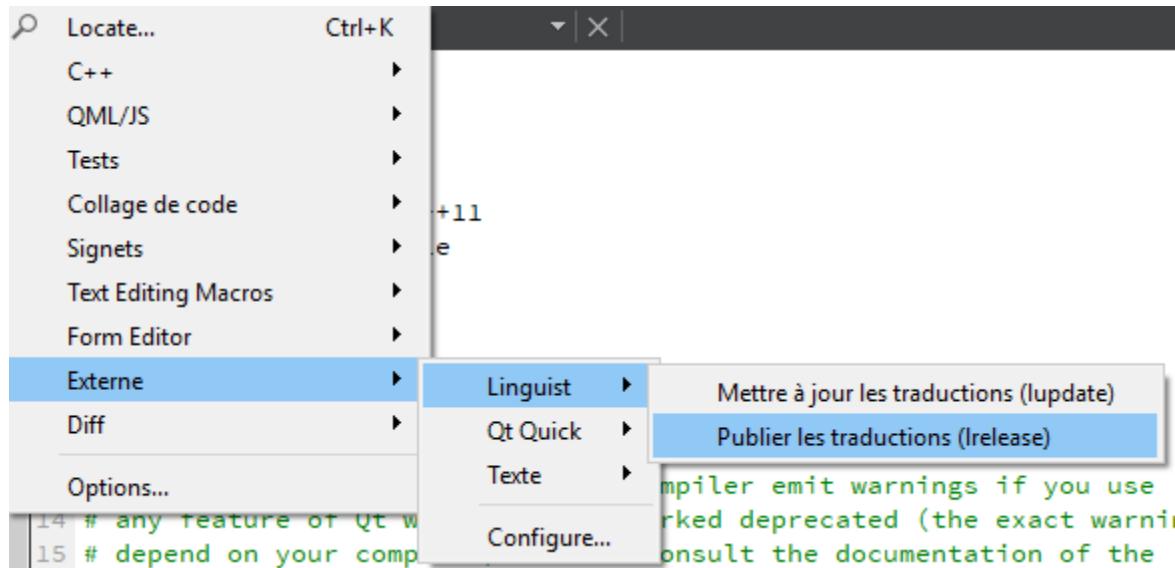


The screenshot shows the Qt Linguist application window. The menu bar includes Fichier, Édition, Traduction, Validation, Phrases, Vue, and Aide. The toolbar contains various icons for file operations and search. The main interface has several panes:

- Contexte**: Shows a list of contexts, with "QObject" selected.
- Chaines**: Displays a list of strings to be translated. One string, "Une chaîne pouvant être traduite", is highlighted with a blue arrow pointing to a button labeled "valider".
- Sources et formulaires**: Shows the source code containing Qt translation macros and comments.
- Texte source**: Shows the original text "Une chaîne pouvant être traduite".
- English traduction**: Contains the English translation "A string that can be translated".
- English commentaires du traducteur**: An empty text area for translator comments.
- Français traduction**: Contains the French translation "Une chaîne pouvant être traduite".
- Français commentaires du traducteur**: An empty text area for translator comments.
- Phrases et propositions**: A pane for managing phrase lists and definitions.
- Avertissements**: A pane for displaying warnings.

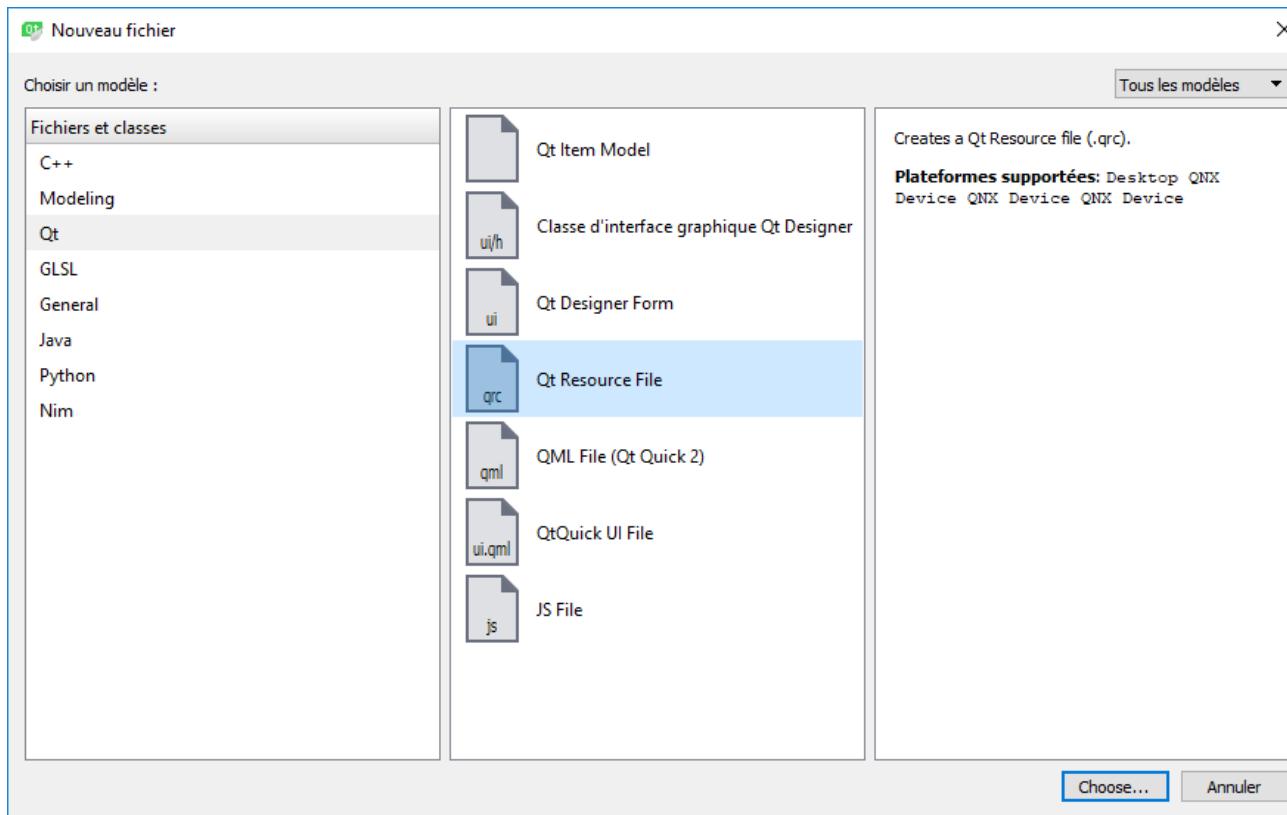


Le chef de projet peut compiler les fichiers .ts à l'aide de l'utilitaire lrelease qui génère des fichiers binaires portant l'extension .qm



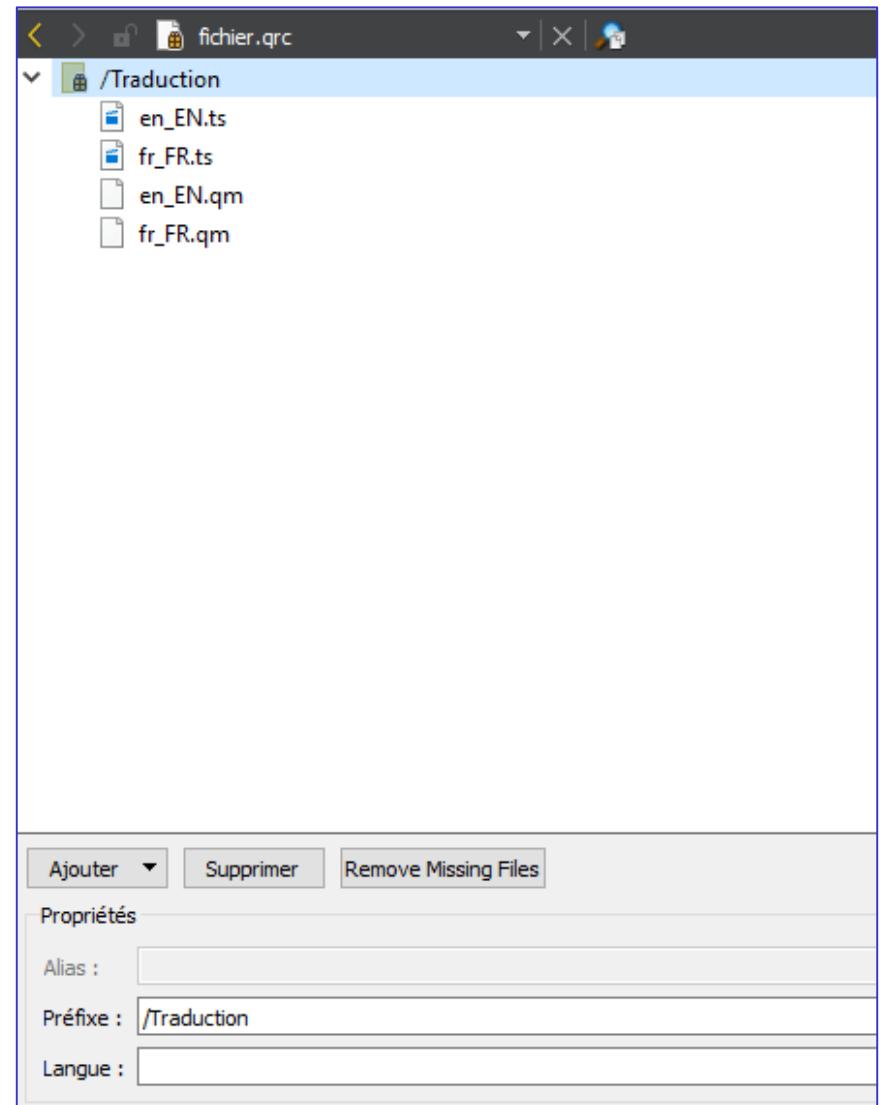


Les ressources permettent d'encapsuler les données de vos images, documents, icônes, etc. dans les données de votre programme et de les rendre disponibles à toutes les classes.





2.4 - Module Qt Core : Qt Ressources system



2.4 - Module Qt Core : Internationalisation



```
QCoreApplication app(argc, argv);
QString langue = QLocale::system().name().split("_")[0] ;
QString region = QLocale::system().name().split("_")[1] ;
qDebug() << langue << "_" << region ;

Q_INIT_RESOURCE(fichier);
QTranslator translator;
//translator.load("<chemin complet>en_EN.qm");
translator.load(":/Traduction/en_EN.qm");// utilisation des ressources
app.installTranslator(&translator);

QString chaine = "Une chaine ne pouvant pas être traduite";
QString chainetraduisible = QObject::tr("Une chaine pouvant être traduite") ;
QString chainetraduisiblecommentaire = QObject::tr("Une chaine pouvant être traduite","commentaire") ;

qDebug()<< chaine;
qDebug() << chainetraduisible;
qDebug() << chainetraduisiblecommentaire;
```

"fr" _ "FR"
"Une chaine ne pouvant pas être traduite"
"A string that can be translated"
"A string that can be translated"



La classe QSettings permet de facilement manipuler des fichiers d'option (ini)



2.4 - Module Qt Core : QSetting



```
void writeOption() {
    // Création du fichier en précisant que l'on travaille avec un fichier de format INI.
    QSettings settings("Developpez.ini", QSettings::IniFormat);

    // Création du groupe [EquipeQt]
    settings.beginGroup("EquipeQt");

    // Création des différentes clés et valeurs correspondantes
    settings.setValue("membre1", "dourouc05");
    settings.setValue("membre2", "Amnell");
    settings.setValue("membre3", "IrmatDen");

    // On ferme le groupe [EquipeQt]
    settings.endGroup();

    // Création du groupe [ModerateursQt]
    settings.beginGroup("ModerateursQt");

    // Création des différentes clés et valeurs correspondantes
    settings.setValue("moderateur1", "yan");
    settings.setValue("moderateur2", "superjaja");

    // On ferme le groupe [ModerateursQt]
    settings.endGroup();
}
```

[EquipeQt]
membre1=dourouc05
membre2=Amnell
membre3=IrmatDen

[ModerateursQt]
moderateur1=yan
moderateur2=superjaja

2.4 - Module Qt Core : QSetting



```

void loadOption() {
    // Ouverture du fichier INI
    QSettings settings("Developpez.ini", QSettings::IniFormat);

    //base de registre Windows => QSettings settings("HKEY_CURRENT_USER\\Developpez",QSettings::NativeFormat);

    // On récupère la valeur de membre1 du groupe [EquipeQt]
    QString membre1 = settings.value("EquipeQt/membre1","Developpez").toString();
    //membre1 == "dourouc05"

    // On souhaite récupérer la clé membre4 du groupe [EquipeQt].
    // Celui-ci n'existant pas, c'est la valeur par défaut qui est retournée.
    QString defaultValue = settings.value("EquipeQt/membre4","Developpez")
                           .toString();
    // defaultValue == "Developpez"

    // On récupère la valeur du moderateur2 dans le groupe [ModerateursQt]
    QString moderateur2 = settings.value("ModerateursQt/moderateur2","developpez")
                           .toString();
    // moderateur2 == "superjaja"

    foreach(const QString& str,settings.allKeys())
    {
        //str == Moderateur1, Moderateur2 puis ResponsableQt
        qDebug() << str;
    }
}

```

"EquipeQt/membre1"
 "EquipeQt/membre2"
 "EquipeQt/membre3"
 "ModerateursQt/moderateur1"
 "ModerateursQt/moderateur2"



2.4 - Module Qt Core : QJsonObject



QJsonObject permet de manipuler des fichiers JSON en Qt

```
void writeJson() {
    QJsonObject json_obj;
    json_obj["name"] = "AVengers";
    json_obj["str"] = 34;
    json_obj["enemy"] = "Thanos";

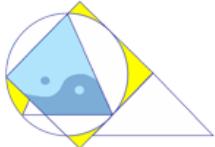
    QJsonDocument json_doc(json_obj);
    QString json_string = json_doc.toJson();

    QFile save_file("test.json");
    if(!save_file.open(QIODevice::WriteOnly)){
        throw QString("failed to open save file");
    }
    save_file.write(json_string.toLocal8Bit());
    save_file.close();
}
```

```
{
    "enemy": "Thanos",
    "name": "AVengers",
    "str": 34
}
```

<http://erickveil.github.io/2016/04/06/How-To-Manipulate-JSON-With-C++-and-Qt.html>





2.4 - Module Qt Core : QJsonObject



```
void readJson() {
    QFile file_obj("fichier_qt.json");
    if(!file_obj.open(QIODevice::ReadOnly)){
        throw QString("failed to open read file");
    }

    QTextStream file_text(&file_obj);
    QString json_string;
    json_string = file_text.readAll();
    file_obj.close();

    QByteArray json_bytes = json_string.toLocal8Bit();
    auto json_doc=QJsonDocument::fromJson(json_bytes);

    if(json_doc.isNull()) throw QString("Failed to create JSON doc.");
    if(!json_doc.isObject()) throw QString("JSON is not an object.");

    QJsonObject json_obj=json_doc.object();

    if(json_obj.isEmpty()) throw QString("JSON object is empty.");

    QVariantMap json_map = json_obj.toVariantMap();
    qDebug()<< json_map["name"].toString();
    qDebug()<< json_map["str"].toInt();
    qDebug()<< json_map["enemy"].toString();
}
```

"Avengers"
34
"Thanos"

<http://erickveil.github.io/2016/04/06/How-To-Manipulate-JSON-With-C++-and-Qt.html>



3.1 - Exemple :

- main
- QMainWindow
- signal & slot

3.2 - Application SDI (Single-Document Interface)

3.3 - Application MDI (Multiple Documents Interface)

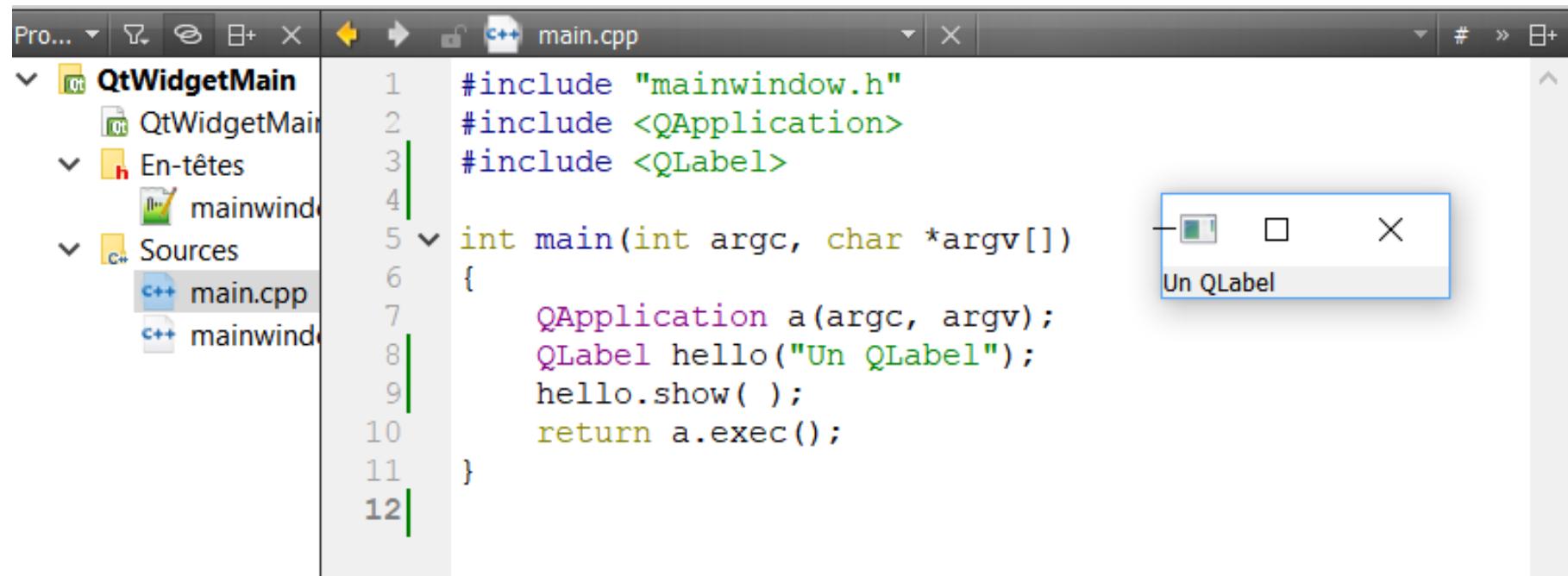
3.4 - Qt Designer

- Layouts
- Spacers
- Buttons
- Item Views
- Item Widgets
- Containers
- Input Widgets
- Display Widgets
- Qt Char
- QPainter
- 2D OpenGL
- QWidget interaction avec le C++

>	Layouts
>	Spacers
>	Buttons
>	Item Views (Model-Based)
>	Item Widgets (Item-Based)
>	Containers
>	Input Widgets
>	Display Widgets

3.1 - les interfaces graphiques : QtGui/ QWidget

-> Exemple : main



The screenshot shows a Qt-based IDE interface. On the left, the project tree displays a project named "QtWidgetMain" with files like "mainwindow.h", "mainwindow.cpp", and "main.cpp". The "Sources" folder contains "main.cpp". The main editor window shows the following C++ code:

```
#include "mainwindow.h"
#include <QApplication>
#include <QLabel>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QLabel hello("Un QLabel");
    hello.show();
    return a.exec();
}
```

To the right of the code editor, there is a small window titled "Un QLabel" which displays a simple window with a close button and the text "Un QLabel".



3.1 - les interfaces graphiques : QtGui/ QWidget

-> Exemple : main



main.cpp - QWidgetMain - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Pro... X main.cpp # Line: 15, Col: 1

Accueil
Éditer
Design
Debug
Projets
Analyse
Aide

QtWidgetMain
QtWidgetMain.h
En-têtes
Sources
main.cpp
mainwindow.h

```
#include "mainwindow.h"
#include <QApplication>
#include <QLabel>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    QLabel hello("Un QLabel", &w);
    w.show();

    return a.exec();
}
```

Sortie de l'application

QtWidgetMain

```
Démarrage de D:\MinGW\msys\1.0\home\David\Nouveau\QtFormation\ExempleC++
\5.1\build-QtWidgetMain-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug
\QtWidgetMain.exe...
```

Type to locate (Ctrl...) 1 Problèmes 2 Search ... 3 Sortie d... 4 Sortie d... 5 Console... 6 Messag... 7 Version ...

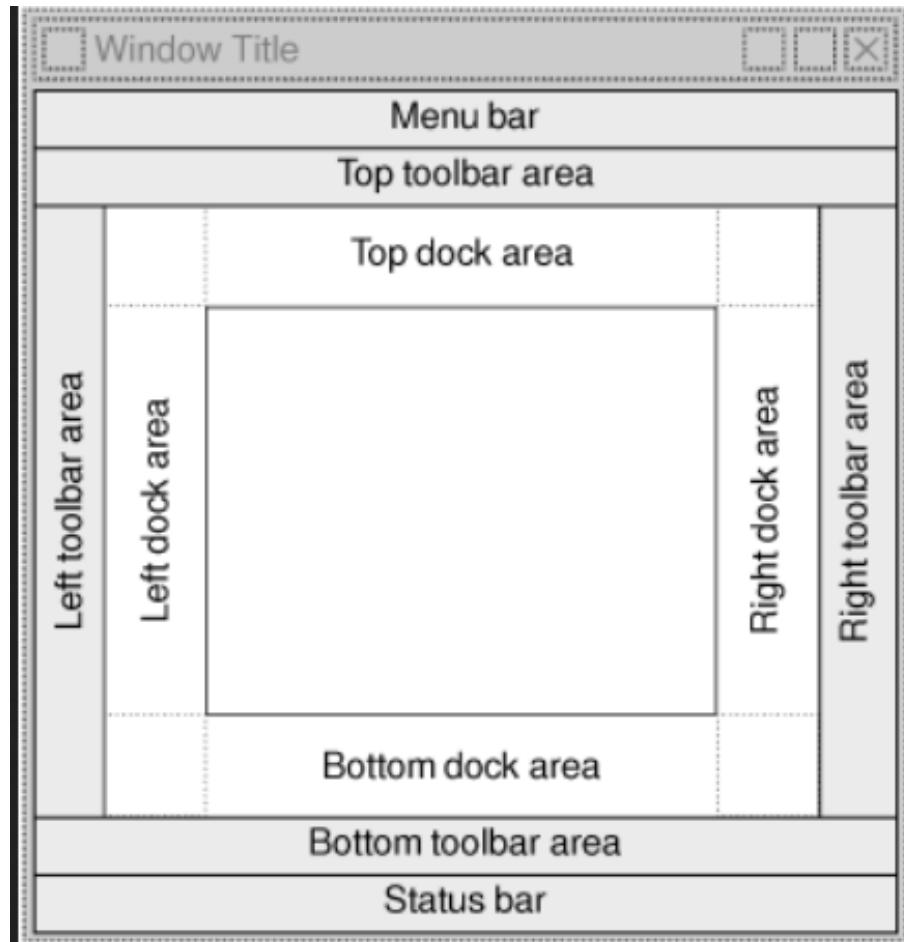
3.1 - les interfaces graphiques : QtGui/ QWidget -> Exemple : QMainWindow



```
mainwindow.h MainWindow # Line: 6, Col: 38
1 ifndef MAINWINDOW_H
2 define MAINWINDOW_H
3
4 include <QMainWindow>
5
6 class MainWindow : public QMainWindow
7 {
8     Q_OBJECT
9
10 public:
11     MainWindow(QWidget *parent = 0);
12     ~MainWindow();
13 }
14
15 endif // MAINWINDOW_H
```

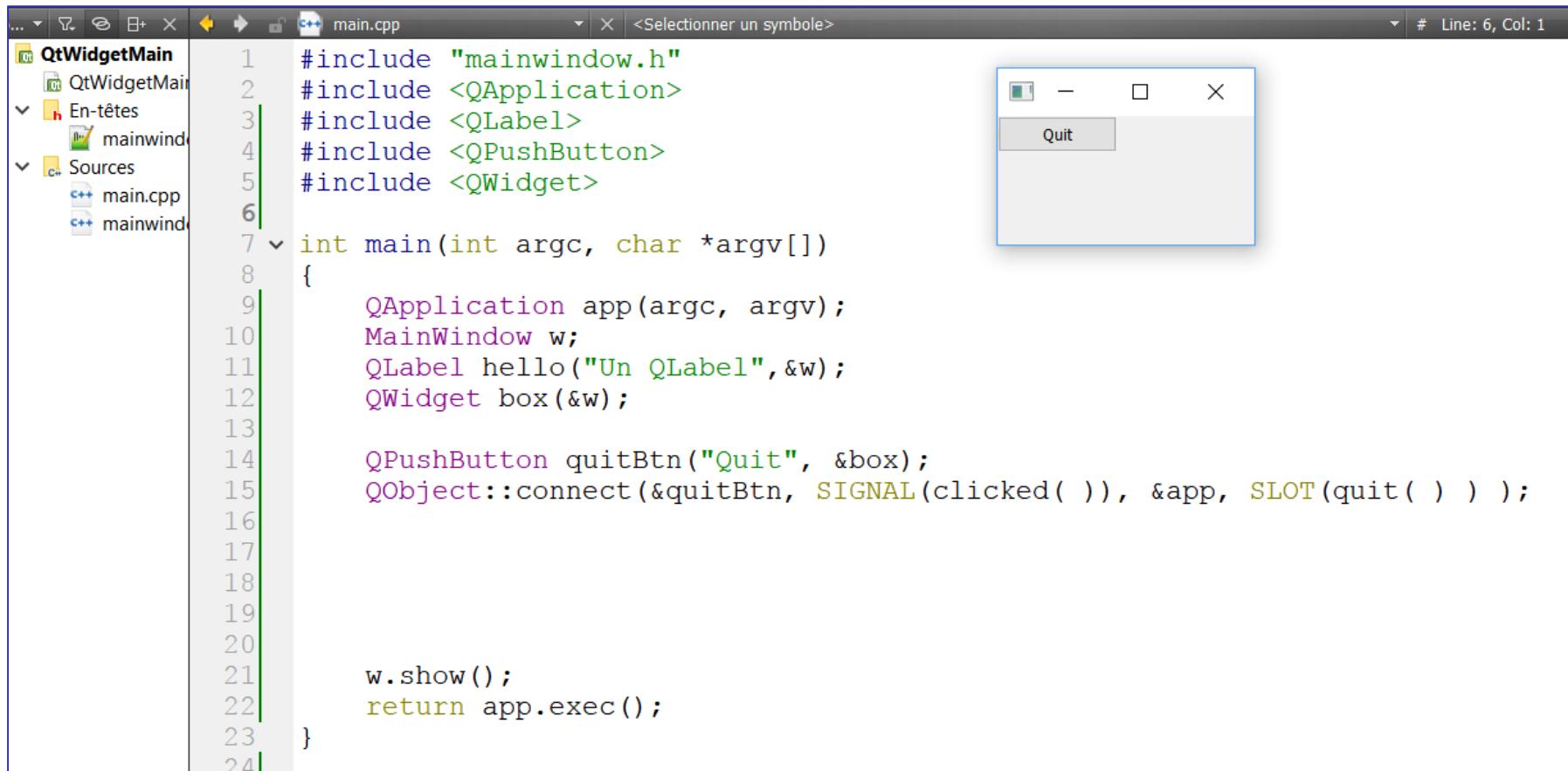


La classe QMainWindow fournit une fenêtre principale pour l'application



3.1 - les interfaces graphiques : QtGui/ QWidget*

-> Exemple : signal & slot



The screenshot shows a Qt-based IDE interface. On the left, the project tree displays a single project named "QtWidgetMain" with files "mainwindow.h", "mainwindow.cpp", and "main.cpp". The main editor area contains the following C++ code:

```
#include "mainwindow.h"
#include <QApplication>
#include <QLabel>
#include <QPushButton>
#include <QWidget>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    MainWindow w;
    QLabel hello("Un QLabel", &w);
    QWidget box(&w);

    QPushButton quitBtn("Quit", &box);
    QObject::connect(&quitBtn, SIGNAL(clicked()), &app, SLOT(quit()));

    w.show();
    return app.exec();
}
```

To the right of the code editor, a small window preview shows a simple application window with a single button labeled "Quit".



3.1 - les interfaces graphiques : QtGui/ QWidget

-> Exemple : signal & slot



The screenshot shows a Qt-based IDE interface. On the left, the project structure is displayed under 'QtWidgetMain':

- QtWidgetMain (selected)
- QtWidgetMain.h
- En-têtes
- mainwindow.h
- Sources
- main.cpp
- mainwindow.h

The main editor window contains the following C++ code:

```
#include "mainwindow.h"
#include <QApplication>
#include <QLabel>
#include <QPushButton>
#include <QWidget>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    MainWindow w;
    w.setWindowTitle("Ma fenetre");
    w.resize(300,120);
    QLabel hello("Un QLabel", &w);
    hello.move(150,50);
    QWidget box(&w);
    box.move(200,90);

    QPushButton quitBtn("Quit", &box);
    QObject::connect(&quitBtn, SIGNAL(clicked()), &app, SLOT(quit()));
    w.show();
    return app.exec();
}
```

To the right, a preview window titled "Ma fenetre" is shown. It contains the text "Un QLabel" and a "Quit" button.

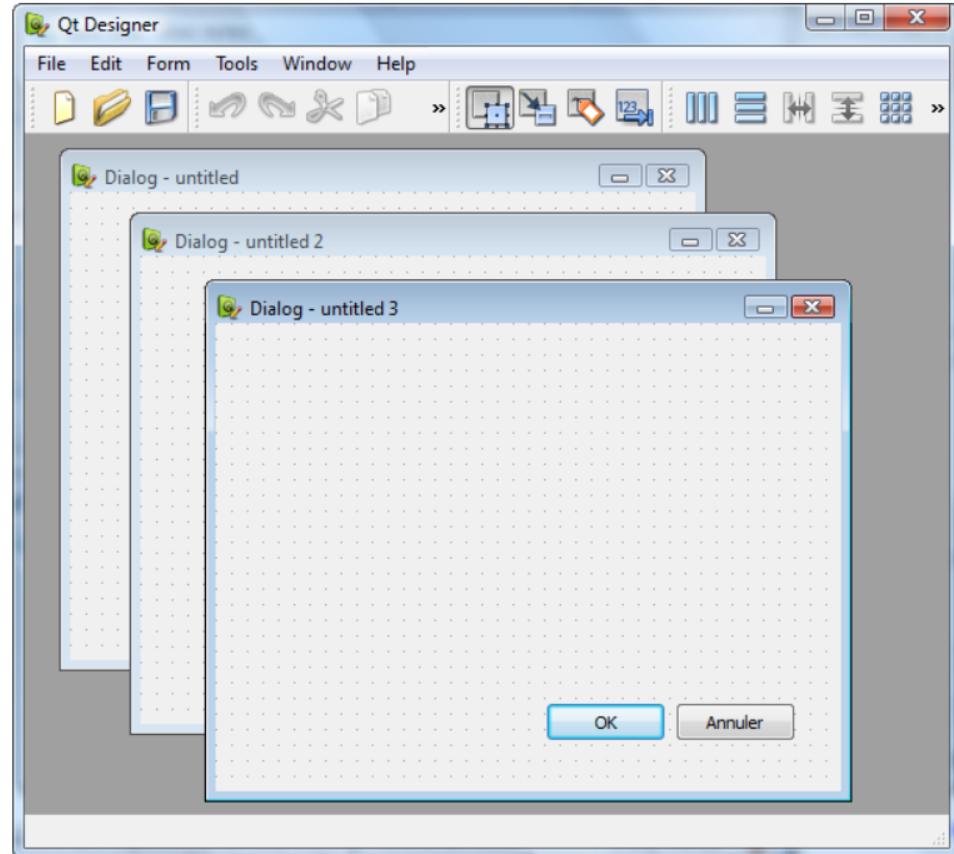


- **Les SDI (Single Document Interface)** : elles ne peuvent afficher qu'un document à la fois.
C'est le cas du Bloc-Notes par exemple.

3.3 - les interfaces graphiques : QtGui/ QWidget -> Application MDI (Multiple Documents Interface)



- **Les MDI (Multiple Document Interface) :** elles peuvent afficher plusieurs documents à la fois. Elles affichent des sous-fenêtres dans la zone centrale. C'est le cas par exemple de Qt Designer.





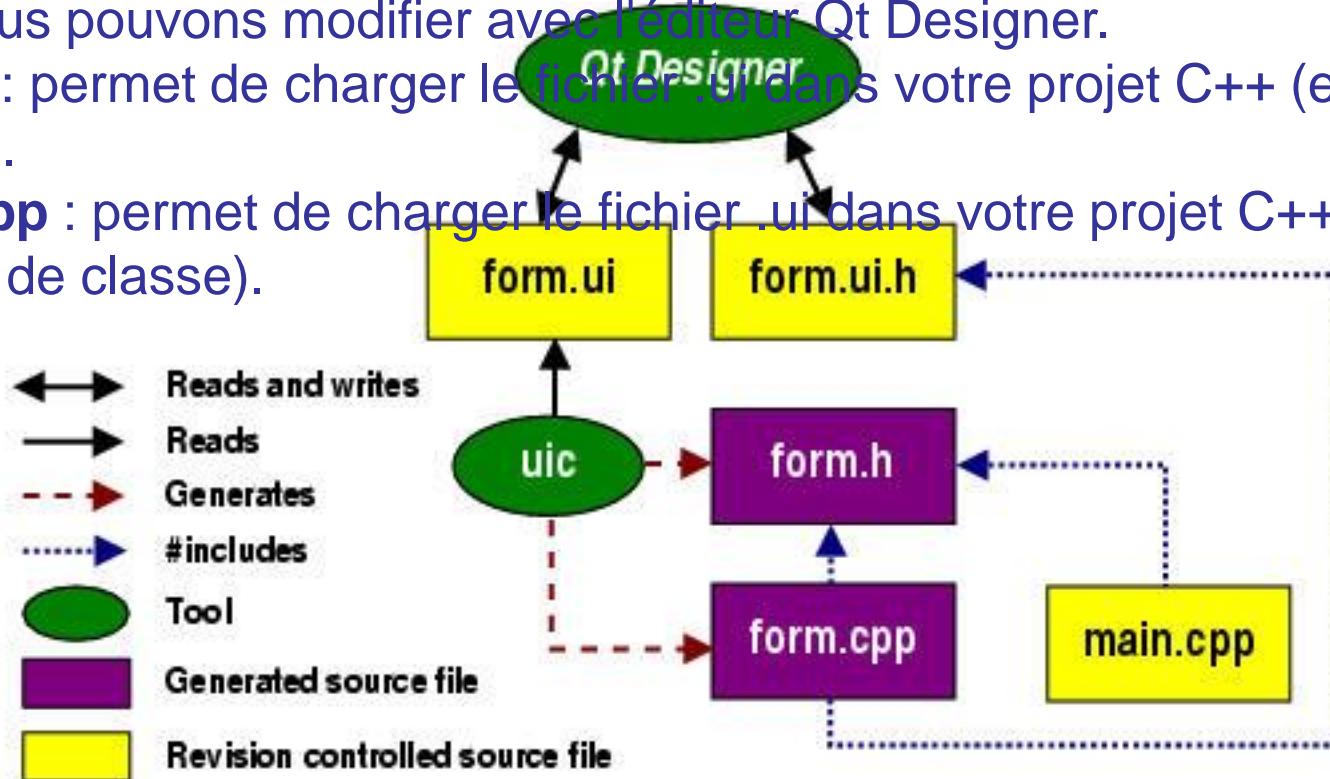
Qt Designer permet de **dessiner vos fenêtres visuellement**, et de modifier les propriétés des widgets, d'utiliser des layouts, et d'effectuer la connexion entre signaux et slots.

Qt Designer existe sous forme de programme indépendant, mais il est aussi intégré au sein de Qt Creator dans la section Design.

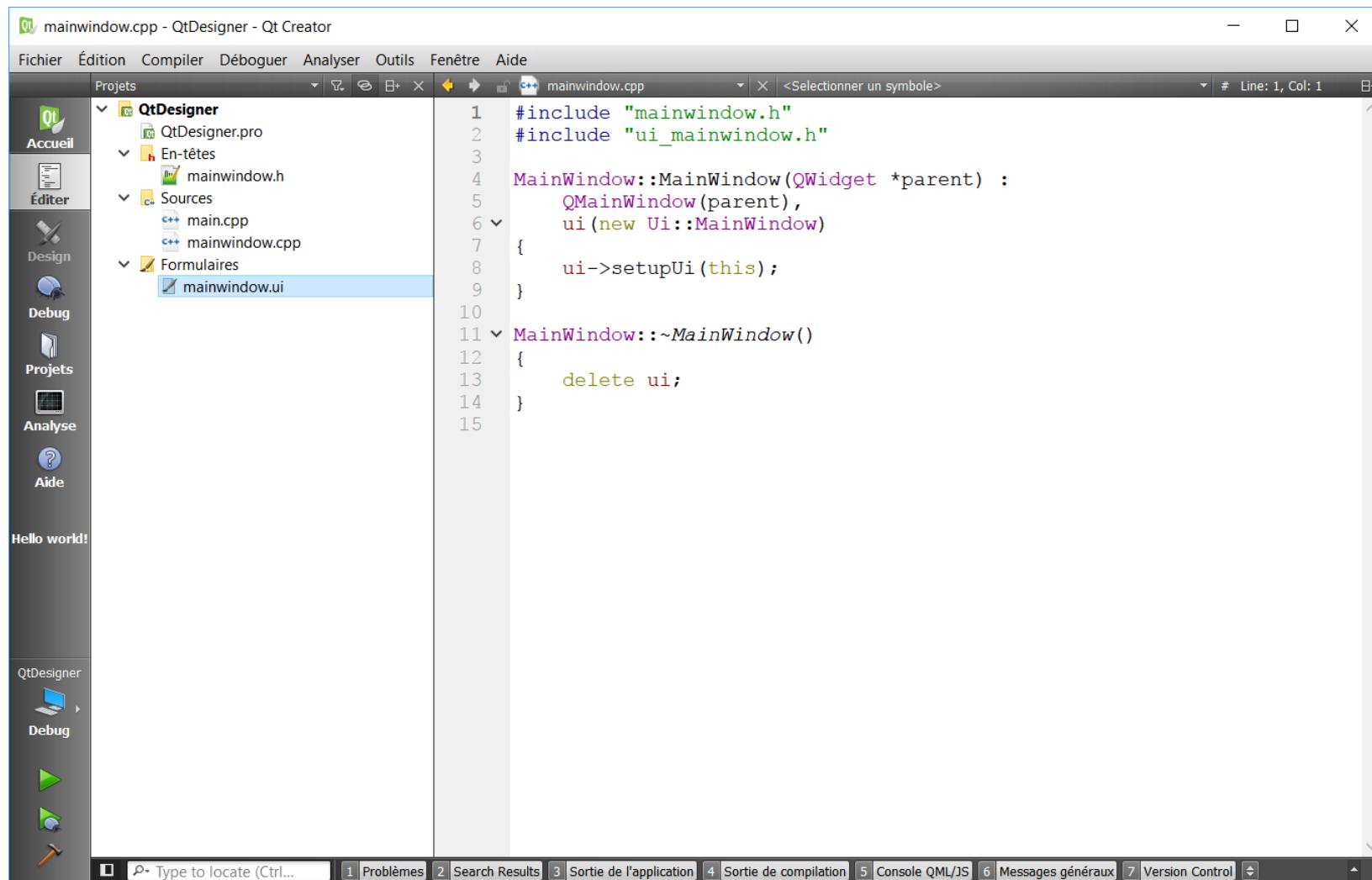


QDesigner génère 3 fichiers Exemple :

- **form.ui** : fichier qui contiendra l'interface graphique (de type XML), fichier que nous pouvons modifier avec l'éditeur Qt Designer.
- **form.h** : permet de charger le fichier .ui dans votre projet C++ (en-tête de classe).
- **form.cpp** : permet de charger le fichier .ui dans votre projet C++ (code source de classe).



3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer



The screenshot shows the Qt Creator IDE interface. The left sidebar has icons for Accueil, Éditer, Design, Debug, Projets, Analyse, and Aide. The central area shows a Qt Designer project named "QtDesigner". The Projects view lists "QtDesigner.pro", "mainwindow.h", "main.cpp", and "mainwindow.ui". The main editor window displays the following C++ code:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

The status bar at the bottom includes tabs for Type to locate (Ctrl...), Problèmes, Search Results, Sortie de l'application, Sortie de compilation, Console QML/JS, Messages généraux, and Version Control.



3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer



mainwindow.ui - QtDesigner - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

mainwindow.ui

Filter

- > Layouts
- > Spacers
- > Buttons
- > Item Views (Model-Based)
- > Item Widgets (Item-Based)
- > Containers
- > Input Widgets
- > Display Widgets

Taper ici

Objet Classe

MainWindow	QMainWindow
centralWidget	QWidget
menuBar	QMenuBar
mainToolBar	QToolBar
statusBar	QStatusBar

Filter

MainWindow : QMainWindow

- ▼ QObject
- ▼ QWidget
- ▼ QMainWindow

Nom Utilisé Texte Raccourci Vérifiable In

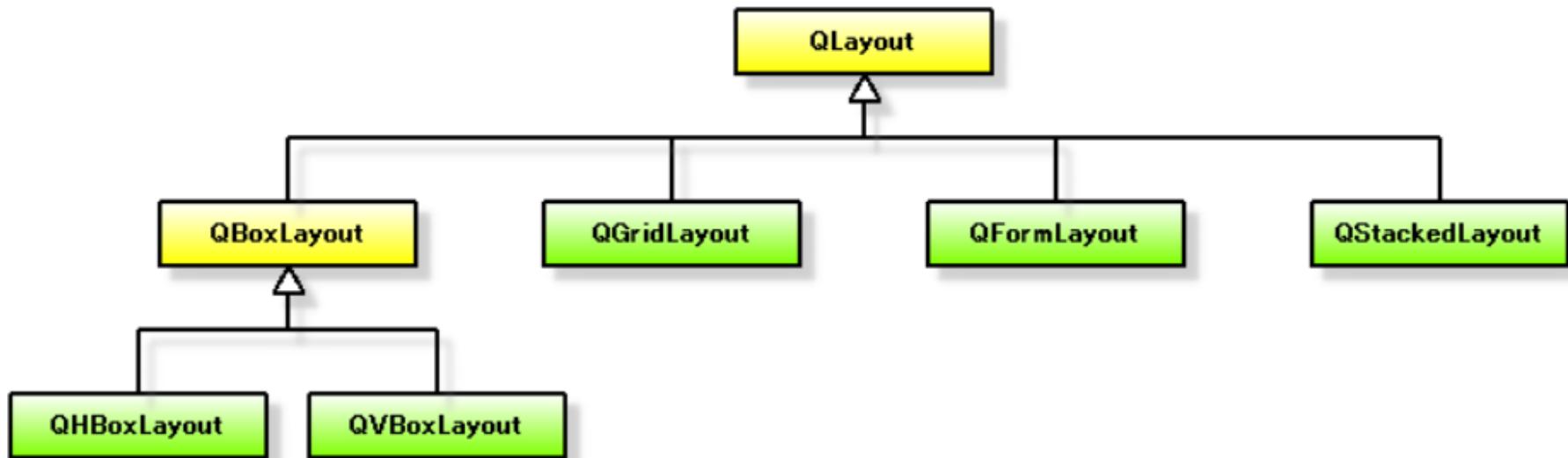
Action Editor Signals & Slots Editor

Type to locate (Ctrl...)

1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control



Qt inclut un ensemble de classes de type **QLayout** qui sont utilisées pour décrire la façon dont les *widgets* sont disposés dans l'interface utilisateur d'une application.

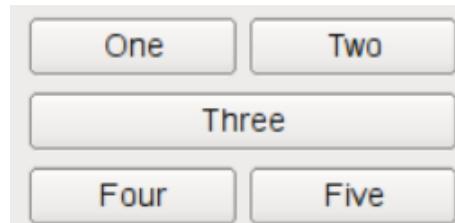




- **QLayout** : La classe de base pour la gestion de la disposition.
 - **QFormLayout** : Gère des formulaires, associant des widgets de saisie avec leur étiquette.



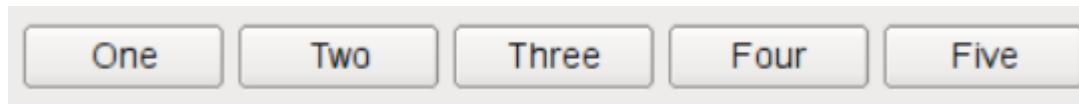
- **QGridLayout** : Dispose les widgets dans une grille.



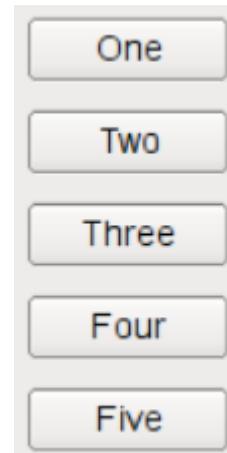
- **QStackedLayout** : Une pile de widgets où un seul widget est visible à la fois.

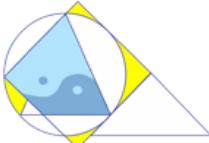


- **QLayout -> QVBoxLayout** : Aligne les widgets enfants horizontalement ou verticalement.
 - **QHBoxLayout** : Aligne les widgets horizontalement.



- **QVBoxLayout** : Aligne les widgets verticalement.





3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Layouts



mainwindow.ui - QtDesigner - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

mainwindow.ui*

Filter

Layouts

- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

Spacers

Buttons

- PushButton
- Tool Button
- RadioButton
- CheckBox
- Command Link Button
- Button Box

Item Views (Model-Based)

Item Widget...Item-Based

Containers

Input Widgets

Display Widgets

- Label
- Text Browser
- Graphics View
- Calendar
- LCD Number
- Progress Bar
- Horizontal Line
- Vertical Line
- OpenGL Widget
- QQuickWidget
- QWebView

Taper ici

Vertical Layout

Horizontal Layout

Grid Layout

Form Layout

Objet Classe

- push...n_17 QPushButton
- push...on_6 QPushButton
- push...on_7 QPushButton
- push...on_8 QPushButton
- push...on_9 QPushButton
- hori...yout QBoxLayout
- push...on_3 QPushButton
- push...on_4 QPushButton
- push...on_5 QPushButton
- label QLabel
- label_2 QLabel
- label_3 QLabel
- label_4 QLabel

Filter

label_4 : QLabel

QObject

QWidget

QFrame

QLabel

text Form Layout

textFormat AutoText

pixmap

scaledContents

alignment AlignementGauche, Alignements

Horizontal AlignmentGauche

Vertical AlignmentCentreV

wordWrap

margin 0

Nom Utilisé Texte Raccourci Vérifiable Info-bulle

Action Editor Signals & Slots Editor

Type to locate (Ctrl...)

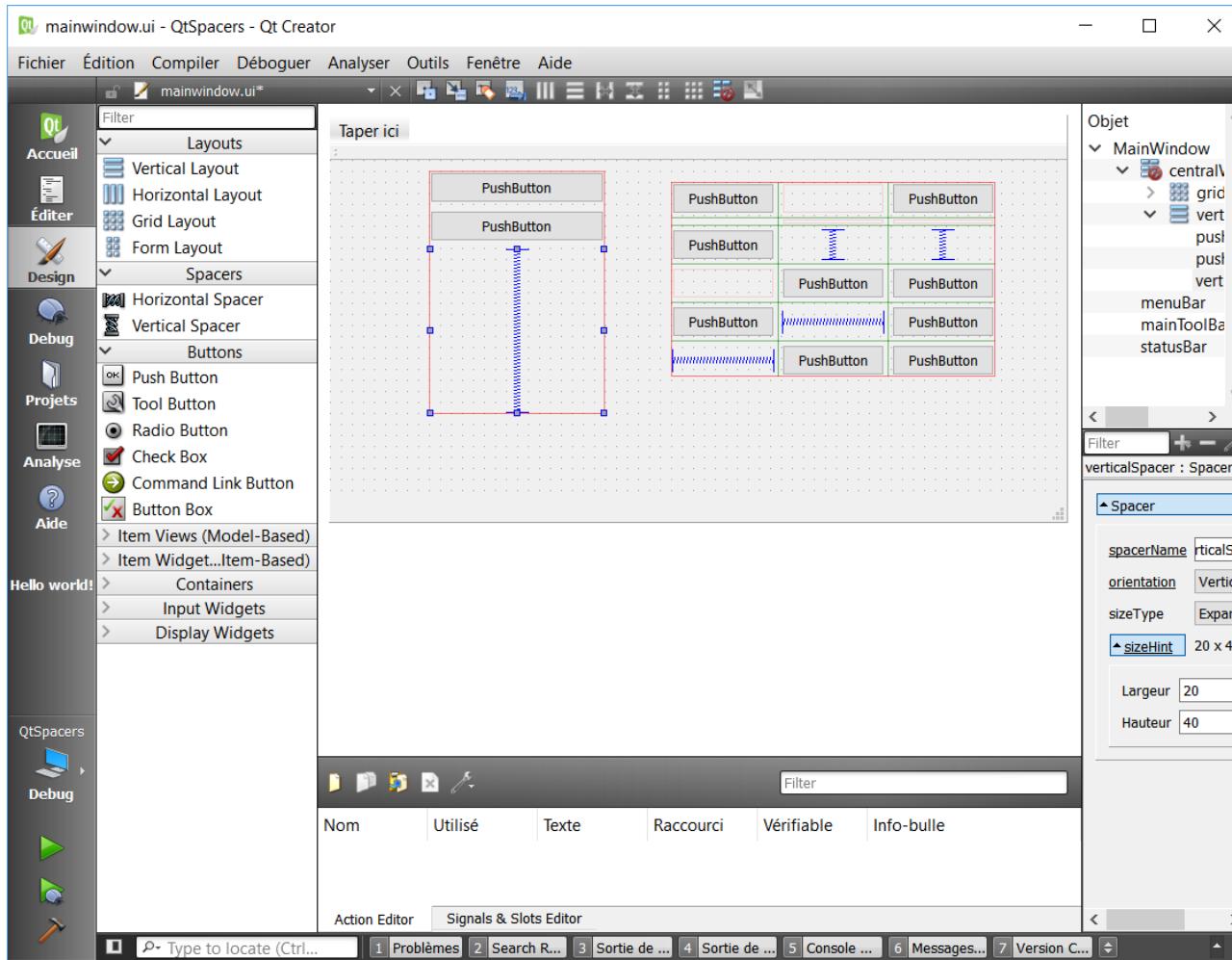
1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control

3.4 - les interfaces graphiques : QtGui/ QWidget

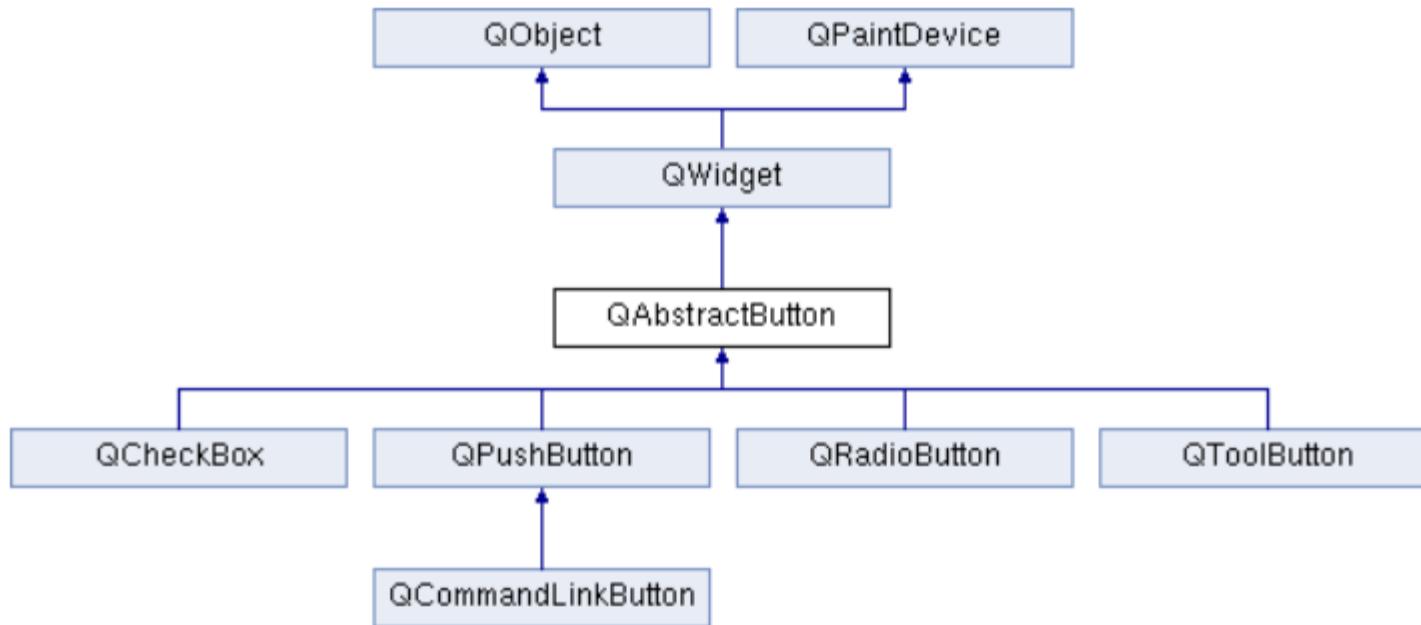
-> Qt Designer : Spacers



Un Spacer est d'un widget **QSpacerItem**, invisible qui sert à créer de l'espace sur la fenêtre.



3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Buttons



3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Buttons



mainwindow.ui - QtButtons - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Accueil Éditer Design Debug Projets Analyse Aide

QtButtons Debug

mainwindow.ui*

Layouts

- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

Spacers

- Horizontal Spacer
- Vertical Spacer

Buttons

- PushButton
- Tool Button
- RadioButton
- CheckBox
- Command Link Button
- Button Box

Item Views (Model-Based)

- List View
- Tree View
- Table View
- Column View

Item Widgets (Item-Based)

- List Widget
- Tree Widget
- Table Widget

Containers

- GroupBox
- Scroll Area
- Tool Box
- Tab Widget
- Stacked Widget
- Frame

mainwindow

centralWidget

buttonBox

checkbox

comma...utton

pushButton

radioButton

toolButton

menuBar

mainToolBar

statusBar

QMainWindo...

QW...

QD...

QC...

QC...

QP...

QR...

QT...

QMen...

QToolB...

QStatu...

Filter

MainWindow : QMainWindow

QObject

QWidget

QMainWindow

iconSize 30

toolButtonStyle

animated

documentMode

tabShape

dockNestingEnabled

dockOptions Ani...

unifiedTitleAndToolBarOnMac

Taper ici

PushButton

RadioButton

CheckBox

...

OK Cancel

→ CommandLinkButton

Émetteur Signal Receveur Slot

Action Editor Signals & Slots Editor

Type to locate (Ctrl...)

Problèmes Search ... Sortie d... Sortie d... Console ... Messag... Version ...

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Buttons - slot & signal



mainwindow.ui - QtButtons - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Accueil Éditeur Design Debug Projets Analyse Aide Hello world! QtButtons Debug Containers

Layouts Vertical Layout Horizontal Layout Grid Layout Form Layout Spacers Horizontal Spacer Vertical Spacer Buttons Push Button Tool Button Radio Button CheckBox Command Link Button Button Box Item Views (Model-Based) List View Tree View Table View Column View Item Widgets (Item-Based) List Widget Tree Widget Table Widget Group Box Scroll Area Tool Box Tab Widget Stacked Widget Frame

Taper ici

Modifier le texte...
Modifier objectName...
Transformer en
Modifier toolTip...
Modifier whatsThis...
Modifier la feuille de style...
Contrainte de taille
Promouvoir en...
Aller au slot...
Placer en arrière plan
Amener au premier plan
Couper Ctrl+X
Copier Ctrl+C
Coller Ctrl+V
Tout sélectionner Ctrl+A
Supprimer
Mettre en page

MainWindow centralWidget buttonBox checkBox comma...utton pushButton radioButton toolButton menuBar mainToolBar statusBar

pushButton : QPushButton

QObject QWidget QAbstractButton

text QPushButton icon iconSize 20 x 20 shortcut Press shortcut checkable checked autoRepeat autoExclusive

Émetteur Signal Receveur Slot Action Editor Signals & Slots Editor

Type to locate (Ctrl...)

1 Problèmes 2 Search ... 3 Sortie d... 4 Sortie d... 5 Console ... 6 Messag... 7 Version ...



3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Buttons - slot & signal



mainwindow.ui - QtButtons - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

mainwindow.ui*

Filter

Layouts

- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

Spacers

- Horizontal Spacer
- Vertical Spacer

Buttons

- PushButton
- Tool Button
- Radio Button
- Check Box
- Command Link Button
- Button Box

Item Views (Model-Based)

- List View
- Tree View
- Table View
- Column View

Item Widgets (Item-Based)

- List Widget
- Tree Widget
- Table Widget

Containers

- GroupBox
- Scroll Area
- Tool Box
- Tab Widget
- Stacked Widget

Frame

Taper ici

PushButton

RadioButton

CheckBox

...

OK Cancel

Aller au slot

Sélectionner signal

clicked()	QAbstractButton
clicked(bool)	QAbstractButton
pressed()	QAbstractButton
released()	QAbstractButton
toggled(bool)	QAbstractButton
destroyed()	QObject
destroyed(QObject*)	QObject
objectNameChanged(QString)	QObject
customContextMenuRequested(QPoint)	QWidget
windowIconChanged(QIcon)	QWidget
windowIconTextChanged(QString)	QWidget
windowTitleChanged(QString)	QWidget

OK Cancel

Émetteur Signal Receveur Slot

Action Editor Signals & Slots Editor

Type to locate (Ctrl...)

Problèmes Search ... Sortie d... Sortie d... Console ... Messag... Version ...

shortcuts

Press shortcut

checkbox checked autoRepeat autoExclusive

143

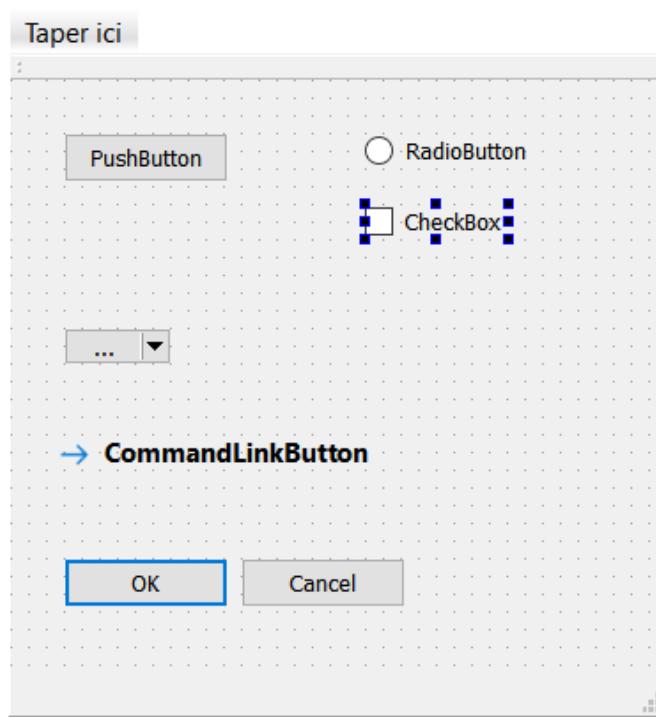
3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Buttons - slot & signal



```
mainwindow.cpp MainWindow::on_pushButton_clicked(): void
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QDebug>
4
5 MainWindow::MainWindow(QWidget *parent) :
6     QMainWindow(parent),
7     ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10 }
11
12 ~MainWindow()
13 {
14     delete ui;
15 }
16
17 void MainWindow::on_pushButton_clicked()
18 {
19     qDebug() << QString(__FILE__) + " : " + __FUNCTION__ ;
20 }
21
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Buttons - slot & signal

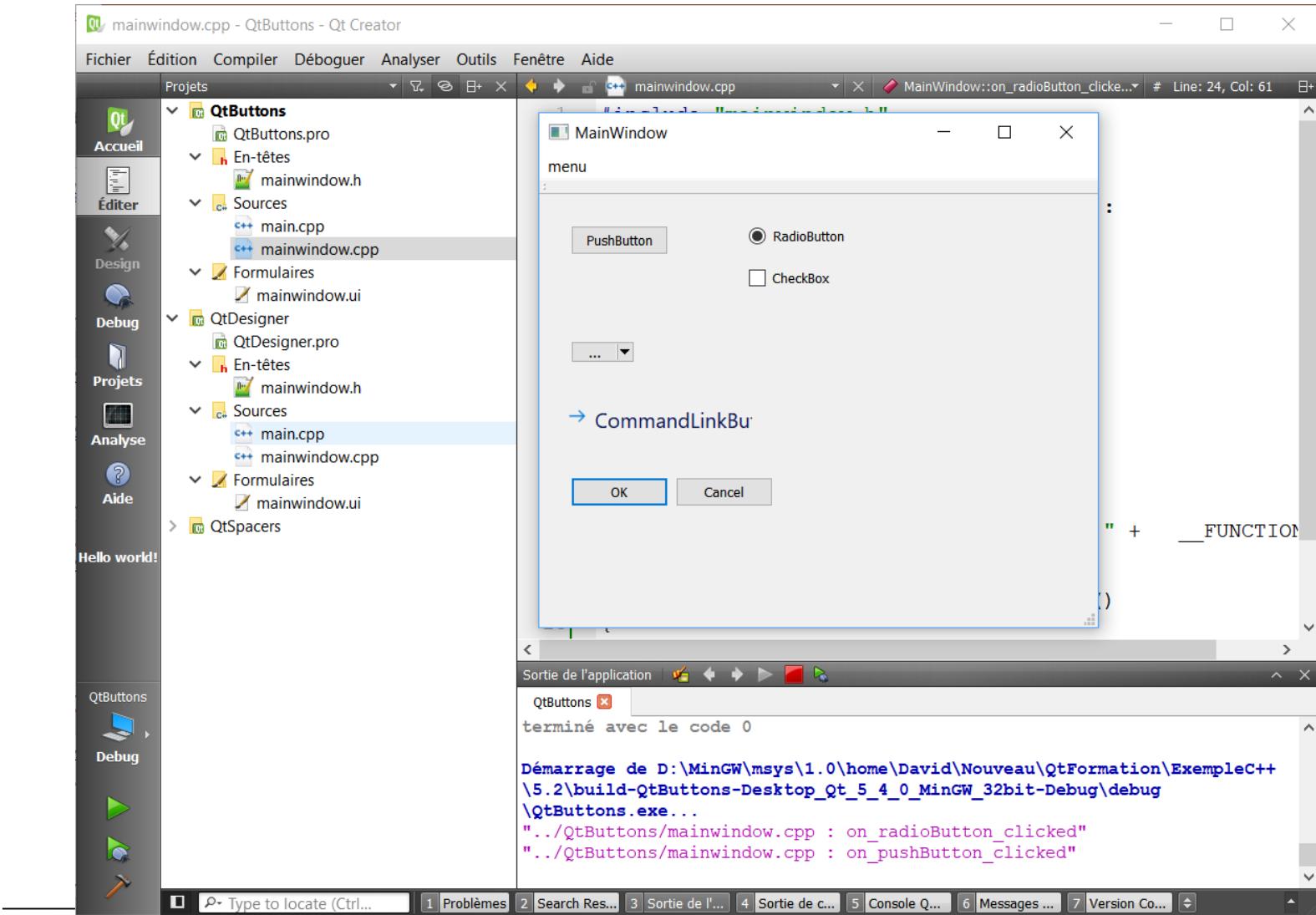


3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Buttons - slot & signal



```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QDebug>
4
5 MainWindow::MainWindow(QWidget *parent) :
6     QMainWindow(parent),
7     ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10}
11
12 ~MainWindow()
13{
14     delete ui;
15}
16
17 void MainWindow::on_pushButton_clicked()
18{
19     qDebug() << QString(__FILE__) + " : " + __FUNCTION__ ;
20}
21
22 void MainWindow::on_RADIOButton_clicked()
23{
24     qDebug() << QString(__FILE__) + " : " + __FUNCTION__ ;
25}
26
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Buttons - slot & signal

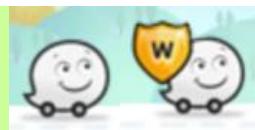


The screenshot shows the Qt Creator IDE interface. On the left, the Projects panel displays two projects: "QtButtons" and "QtDesigner". The "QtButtons" project is currently selected, showing files like mainwindow.h, main.cpp, and mainwindow.cpp. The "QtDesigner" project also has similar files. The central workspace shows a dialog window titled "MainWindow" with a "menu" bar. Inside the dialog, there is a "PushButton" button, a "RadioButton" button (which is checked), and a "CheckBox" button. Below these are "OK" and "Cancel" buttons. The status bar at the bottom shows the path "D:\MinGW\msys\1.0\home\David\Nouveau\QtFormation\ExempleC++\5.2\build-QtButtons-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtButtons.exe..." and the command "cd ..\QtButtons\mainwindow.cpp". The terminal window shows the application has started and terminated successfully with code 0.

```
Démarrage de D:\MinGW\msys\1.0\home\David\Nouveau\QtFormation\ExempleC++\5.2\build-QtButtons-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtButtons.exe...
"../QtButtons/mainwindow.cpp : on_pushButton_clicked"
"../QtButtons/mainwindow.cpp : on_checkBox_clicked"
```

3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Buttons - slot & signal



mainwindow.cpp - QtButtons - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets mainwindow.cpp <Selectionner un symbole> # Line: 4, Col: 1

Qt Accueil

- QtButtons
 - QtButtons.pro
 - En-têtes
 - mainwindow.h
 - Sources
 - main.cpp
 - mainwindow.cpp
 - Formulaires
 - mainwindow.ui
- QtDesigner
 - QtDesigner.pro
 - En-têtes
 - mainwindow.h
 - Sources
 - main.cpp
 - mainwindow.cpp
 - Formulaires
 - mainwindow.ui
- QtSpacers

6 Q MainWindow(parent),
 7 ui(new Ui::MainWindow)
 8 {
 9 ui->setupUi(this);
 10 }
 11 MainWindow::~MainWindow()
 12 {
 13 delete ui;
 14 }
 15 void MainWindow::on_pushButton_clicked()
 16 {
 17 qDebug() << QString(__FILE__) + ":" + __FUNCTION__;
 18 }
 19 void MainWindow::on_radioButton_clicked()
 20 {
 21 qDebug() << QString(__FILE__) + ":" + __FUNCTION__;
 22 }
 23 void MainWindow::on_buttonBox_clicked(QAbstractButton *button)
 24 {
 25 qDebug() << QString(__FILE__) + ":" + __FUNCTION__;
 26 qDebug() << button->text();
 27 }
 28 void MainWindow::on_buttonBox_clicked(QAbstractButton *button)
 29 {
 30 qDebug() << QString(__FILE__) + ":" + __FUNCTION__;
 31 qDebug() << button->text();
 32 }

MainWindow

menu

PushButton RadioButtonItem
 CheckBox

... ▾

→ CommandLinkBu

OK Cancel

Sortie de l'application

QtButtons

```
../QtButtons/mainwindow.cpp : on_pushButton_clicked"
"OK"
../QtButtons/mainwindow.cpp : on_pushButton_clicked"
"Cancel"
```

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'a... 4 Sortie de co... 5 Console QML... 6 Messages g... 7 Version Cont...



3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Buttons - slot & signal



mainwindow.h - QtButtons - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets mainwindow.h <Selectionner un symbole> # Line: 5, Col: 27

Accueil

Éditeur

Design

Debug

Projets

Analyse

Aide

Hello world!

QtButtons

QtButtons

mainwindow.h

Sources

main.cpp

mainwindow.cpp

Formulaires

mainwindow.ui

QtDesigner

QtDesigner

mainwindow.h

Sources

main.cpp

mainwindow.cpp

Formulaires

mainwindow.ui

QtSpacers

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QAbstractButton>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();
    void on_radioButton_clicked();
    void on_buttonBox_clicked(QAbstractButton *button);

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

Sortie de l'application

QtButtons

.../QtButtons/mainwindow.cpp : on_buttonBox_clicked
"OK"
.../QtButtons/mainwindow.cpp : on_buttonBox_clicked
"Cancel"

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'a... 4 Sortie de co... 5 Console QML... 6 Messages g... 7 Version Cont...

3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Buttons - slot & signal



mainwindow.ui - QtButtons - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Accueil Éditeur Design Debug Projets Analyse Aide Hello worlds QtButtons Debug

mainwindow.ui

Layouts

- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

Spacers

- Horizontal Spacer
- Vertical Spacer

Buttons

- Push Button
- Tool Button
- RadioButton
- CheckBox
- Command Link Button
- Button Box

Item Views (Model-Based)

- List View
- Tree View
- Table View
- Column View

Item Widgets (Item-Based)

- List Widget
- Tree Widget
- Table Widget

Containers

menu | Taper ici

action1

Taper ici

Ajouter séparateur

RadioButton

CheckBox

...

CommandLinkButton

OK Cancel

Objet Classe

menuMenu QAction mainToolBar QAction

actionAction1 : QAction

QObject

checkable

checked

enabled

icon

text action1

iconText action1

toolTip action1

statusTip

whatsThis

font A [MS Shell Dlg 2, 8]

shortcut Ctrl+D

shortcutContext WindowShortcut

autoRepeat

visible

menuRole TextHeuristicRole

Nom Utilisé Texte Raccourci Vérifiable Info-bulle

actionAction1	<input checked="" type="checkbox"/>	action1	Ctrl+D	<input type="checkbox"/>	action1
---------------	-------------------------------------	---------	--------	--------------------------	---------

Action Editor Signals & Slots Editor

Sortie de l'application

```
QtButtons x
"../QtButtons/mainwindow.cpp : print"
"../QtButtons/mainwindow.cpp : print"
D:\MinGW\msys\1.0\home\David\Nouveau\QtFormation\ExempleC++\5.2\build-QtButtons-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtButtons.exe s'est terminé avec le code 0
```

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages généraux Version Control

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Buttons - slot & signal & connect



```

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <QDebug>
4
5 MainWindow::MainWindow(QWidget *parent) :
6     QMainWindow(parent),
7     ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10
11     this->connect(this->ui->actionAction1, SIGNAL(triggered()), this, SLOT(print()));
12 }
13
14
15 ~MainWindow()
16 {
17     delete ui;
18 }
19
20 void MainWindow::print()
21 {
22     qDebug() << QString(__FILE__) + " : " + __FUNCTION__ ;
23 }
24
25 void MainWindow::on_pushButton_clicked()

```

```

mainwindow.h
namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void print();
    void on_pushButton_clicked();
    void on_RADIOButton_clicked();
    void on_buttonBox_clicked(QAbstractButton *button);

private:
    Ui::MainWindow *ui;
};

print(): void

```

3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Item Views



ItemWidgets

List View

	Name	Size	Type
1	debug		File
2	release		File
3	Makefile	17 KB	File
4	Makefile.Debug	29 KB	Debug File
5	Makefile.Release	29 KB	Release File
	ui_mainwindow.h		h File

Tree View

Name	Size	Type
debug		File Folder
release		File Folder
Makefile		File
Makefile.Debug		Debug File
Makefile.Release		Release File
ui_mainwindow.h		h File

Table View

	Name	Size	Type
1	debug		File
2	release		File
3	Makefile	17 KB	File
4	Makefil...	29 KB	De...
5	Makefil...	29 KB	Rele...

Column View

Name
debug
release
Makefile
Makefile.Debug
Makefile.Release
ui_mainwindow.h



3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Item Views



mainwindow.ui - QtItem - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Accueil Éditer Design Debug Projets Analyse Aide Hello world! QtItem Debug

mainwindow.ui

Filter

DOCK Widgets

QAxWidget

Input Widgets

- ComboBox
- Font ComboBox
- LineEdit
- TextEdit
- PlainTextEdit
- SpinBox
- DoubleSpinBox
- TimeEdit
- DateEdit
- Date/TimeEdit
- Dial
- HorizontalScrollBar
- VerticalScrollBar
- HorizontalSlider
- VerticalSlider
- KeySequenceEdit

Display Widgets

- Label
- TextBrowser
- GraphicsView
- Calendar
- LCDNumber
- ProgressBar
- HorizontalLine
- VerticalLine
- OpenGLWidget
- QQuickWidget
- QWebView

Taper ici

Central Widget

gridLayout

label

label_2

label_3

label_4

listView

tableView

treeView

menuBar

mainToolBar

statusBar

Objet Classe

MainWindow

- centralWidget
- gridLayout
- label
- label_2
- label_3
- label_4
- listView
- tableView
- treeView
- menuBar
- mainToolBar
- statusBar

QMainWindow

QWidget

QGridLayout

QColumnView

QLabel

QLabel

QLabel

QLabel

QListView

QTableView

QTreeView

QMenuBar

QToolBar

QStatusBar

Filter listView : QListWidget

QWidget

QFrame

QAbstractScrollArea

verticalScrollBarPolicy ScrollBarAsNeeded

horizontalScrollBarPolicy ScrollBarAsNeeded

sizeAdjustPolicy AdjustIgnored

QAbstractItemView

autoScroll

autoScrollMargin 16

editTriggers DoubleClicked|EditKeyPressed

tabKeyNavigation

showDropIndicator

dragEnabled

Action Editor Signals & Slots Editor

Émetteur Signal Receveur Slot

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages généraux Version Control

<https://qt.developpez.com/doc/4.7/model-view-programming/>



3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Item Views



mainwindow.cpp - QtItem - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets Accueil Éditer Design Debug Projets Analyse Aide Hello world! QtItem

QtItem

En-têtes Sources Formulaires

mainwindow.h main.cpp mainwindow.cpp mainwindow.ui

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QFileSystemModel>

namespace Ui {
class ItemWidgets;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::ItemWidgets *ui;
private:
    QFileSystemModel model;
};

#endif // MAINWINDOW_H
```

mainwindow.h

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::ItemWidgets),
    model()

{
    ui->setupUi(this);

    model.setRootPath(QDir::currentPath());
    ui->listView->setModel(&model);
    ui->listView->setRootIndex(model.index(QDir::currentPath()));

    ui->treeView->setModel(&model);
    ui->treeView->setRootIndex(model.index(QDir::currentPath()));

    ui->tableView->setModel(&model);
    ui->tableView->setRootIndex(model.index(QDir::currentPath()));

    ui->columnView->setModel(&model);
    ui->columnView->setRootIndex(model.index(QDir::currentPath()));
}
```

mainwindow.cpp

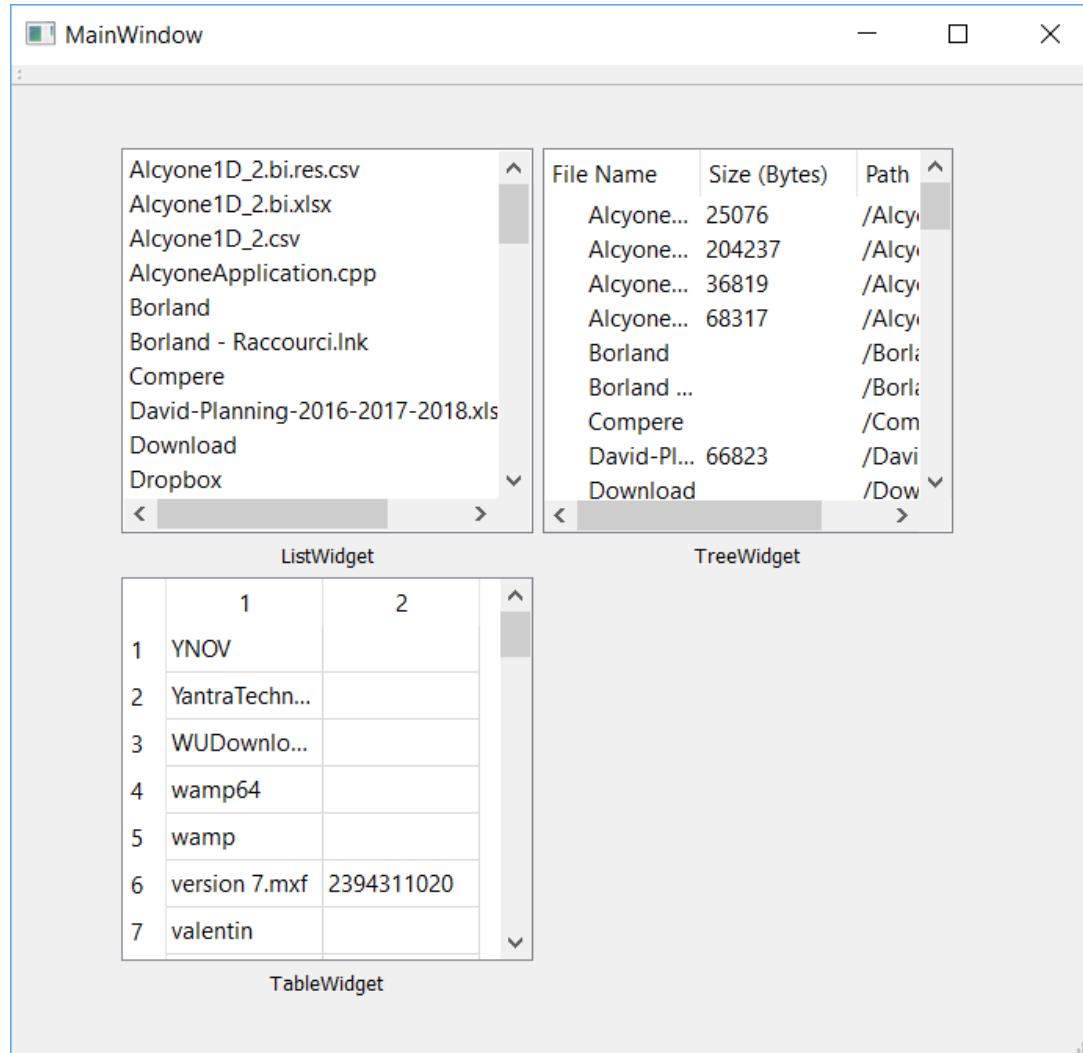
```
MainWindow::~MainWindow()
{
    delete ui;
```

Problèmes

Type to locate (Ctrl...)

1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Item Widgets



The screenshot shows a Qt Designer interface for a "MainWindow". The window contains three item widgets:

- ListWidget**: A vertical list of file names.
- TreeWidget**: A tree view showing file names, sizes, and paths.
- TableWidget**: A grid table with two columns and 7 rows.

ListWidget Data:

- Alcyone1D_2.bi.res.csv
- Alcyone1D_2.bi.xlsx
- Alcyone1D_2.csv
- AlcyoneApplication.cpp
- Borland
- Borland - Raccourci.lnk
- Compere
- David-Planning-2016-2017-2018.xls
- Download
- Dropbox

TreeWidget Data:

File Name	Size (Bytes)	Path
Alcyone...	25076	/Alcy...
Alcyone...	204237	/Alcy...
Alcyone...	36819	/Alcy...
Alcyone...	68317	/Alcy...
Borland		/Borla...
Borland ...		/Borla...
Compere		/Com...
David-Pl...	66823	/Davi...
Download		/Dow...

TableWidget Data:

	1	2
1	YNOV	
2	YantraTechn...	
3	WUDownlo...	
4	wamp64	
5	wamp	
6	version 7.mxf	2394311020
7	valentin	



3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Item Widgets



mainwindow.ui - QtItemWidget - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Accueil Éditeur Design Debug Projets Analyse Aide

mainwindow.ui

Filter

Layouts

- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

Spacers

- Horizontal Spacer
- Vertical Spacer

Buttons

- Push Button
- Tool Button
- Radio Button
- Check Box
- Command Link Button
- Button Box

Item Views (Model-Based)

- List View
- Tree View
- Table View
- Column View

Item Widgets (Item-Based)

- List Widget
- Tree Widget
- Table Widget

Containers

Input Widgets

Display Widgets

Nom Utilisé Texte Raccourci Vérifiable Info-bulle

Action Editor Signals & Slots Editor

Problèmes

- cannot open output file debug/QtItemWidget.exe: Permission denied
- error: Id returned 1 exit status

collect2.exe

QtItemWidget

Qt Item Widgets

Objet Classe

- MainWindow
- centralWidget
- gridLayout
- label
- label_2
- label_3
- listWidget
- tableWidget
- treeWidget
- menuBar
- mainToolBar
- statusBar

Filter

tableWidget : QTableWidget

verticalScrollMode : ScrollPerItem

horizontalScrollMode : ScrollPerItem

QTableView

showGrid : checked

gridStyle : SolidLine

sortingEnabled : unchecked

wordWrap : checked

cornerButtonEnabled : checked

QTableWidget

rowCount : 0

columnCount : 2

Header

Type to locate (Ctrl...)

1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Item Widgets



```

mainwindow.cpp*  MainWindow::MainWindow(QWidget *)
include "mainwindow.h"
include "ui_mainwindow.h"

v MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow) {
    ui->setupUi(this);

    QTreeWidgetItem* headerItem = new QTreeWidgetItem();
    headerItem->setText(0,QString("File Name"));
    headerItem->setText(1,QString("Size (Bytes)"));
    headerItem->setText(2,QString("Path"));
    ui->treeWidget->setHeaderItem(headerItem);

    QDir* rootDir = new QDir("/");
    QFileInfoList filesList = rootDir->entryInfoList();

    foreach(QFileInfo fileInfo, filesList)
    {
        QTreeWidgetItem *itemTree = new QTreeWidgetItem();
        QListWidgetItem *itemList = new QListWidgetItem();
        QTableWidgetItem *itemTable0 = new QTableWidgetItem();
        QTableWidgetItem *itemTable1 = new QTableWidgetItem();
        itemTree->setText(0,fileInfo.fileName());
        itemList->setText(fileInfo.fileName());
        itemTable0->setText(fileInfo.fileName());

        if(fileInfo.isFile())
        {
            itemTree->setText(1,QString::number(fileInfo.size()));
            itemTable1->setText(QString::number(fileInfo.size()));
        }
        itemTree->setText(2,fileInfo.filePath());

        ui->treeWidget->addTopLevelItem(itemTree);
        ui->listWidget->addItem(itemList);

        ui->tableWidget->insertRow(0);
        ui->tableWidget->setItem(0,0,itemTable0);
        ui->tableWidget->setItem(0,1,itemTable1);
    }
}

v MainWindow::~MainWindow()
{
    delete ui;
}

```



QGroupBox : Cadre de groupe avec un titre

QScrollArea : Zone de défilement sur un autre widget

QToolBox : Colonne de widgets en onglets

QTabWidget : Conteneur de widgets en onglets

QStackedWidget : Pile de widgets où un seul est visible à la fois

QFrame : La classe de base des widgets qui peuvent avoir un cadre

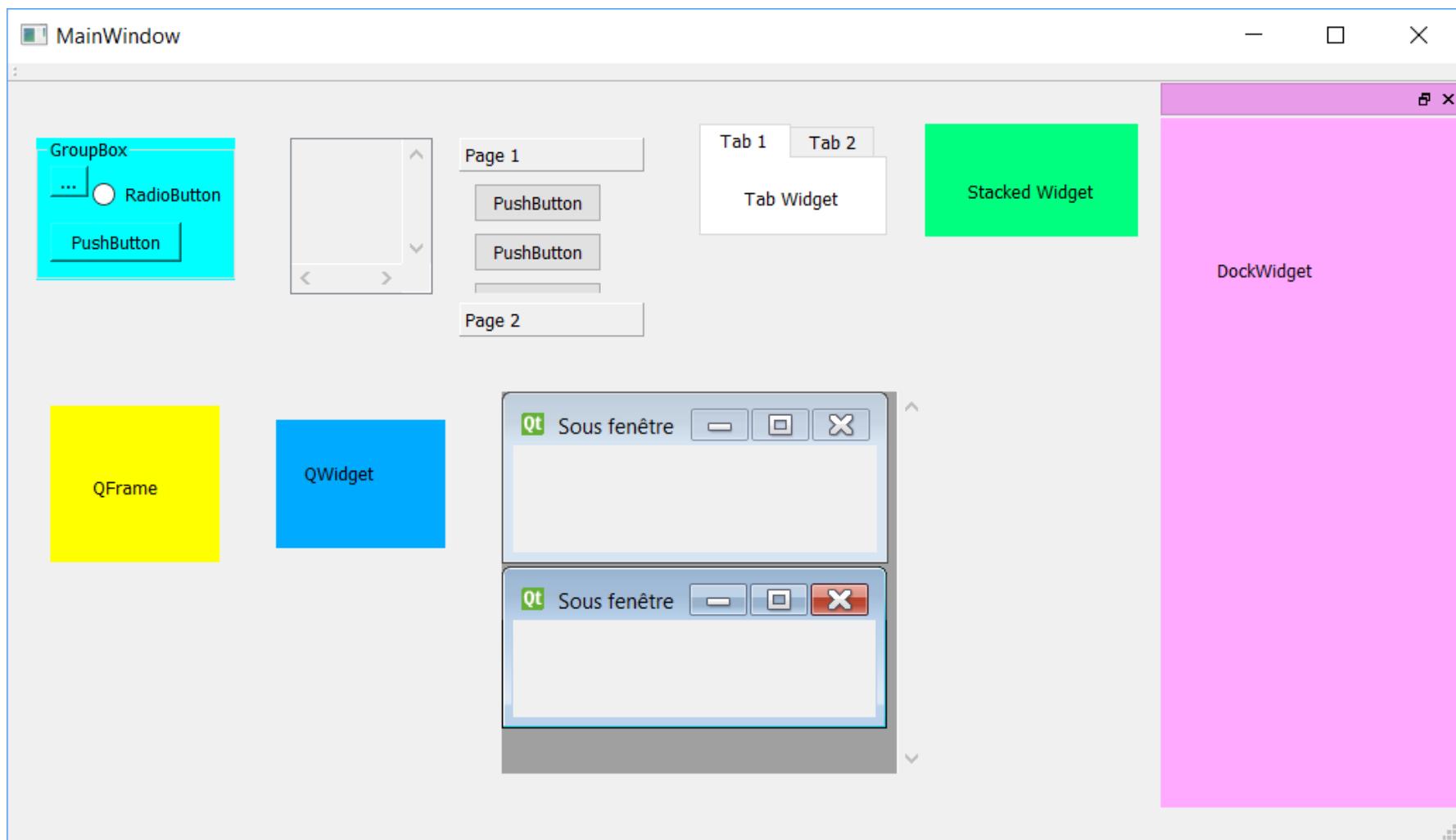
QWidget : Cette classe fournit la capacité de base d'affichage à l'écran et de gestion des événements. Tous les éléments graphiques que Qt fournit sont hérités de QWidget ou sont utilisés avec une classe fille de QWidget.

QMdiArea : Cette classe fournit une zone dans laquelle des fenêtres multidocuments (MDI) peuvent être affichées.

QDockWidget : Cette classe fournit un widget qui peut être parqué (*docked*) à l'intérieur d'une QMainWindow ou flotter en tant que fenêtre de premier niveau sur le bureau.

QAxWidget : Cette classe est un QWidget qui enveloppe un contrôle ActiveX

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Containers





3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Containers



mainwindow.ui - QtContainers - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

mainwindow.ui

Filter

Accueil

Éditeur

Design

Debug

Projets

Analyse

Aide

Hello world!

QtContainers

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages généraux Version Control

Taper ici

Containers

- Layouts
- Spacers
- Buttons
- Item Views (Model-Based)
- Item Widgets (Item-Based)
- Containers
 - GroupBox
 - Scroll Area
 - Tool Box
 - Tab Widget
 - Stacked Widget
 - Frame
 - Widget
 - MdiArea
 - Dock Widget
 - QAxWidget
- Input Widgets
- Display Widgets

Objet

MainWindow

- centralWdg
- frame
- label_2
- groupBo
- pushButt
- radioButt
- toolButt

radioButton : QRadioButton

minimumSize

Largeur 0
Hauteur 0

maximumSize

sizeIncrement

baseSize

palette

font

cursor

mouseTracking

focusPolicy

contextMenuPolicy

acceptDrops

toolTip

toolTipDuration

Action Editor Signals & Slots Editor

Nom Utilisé Texte Raccourci Vérifiable Info-bulle Filter

The screenshot shows the Qt Designer interface with a central workspace displaying several container widgets. On the left, there's a sidebar with categories like 'Containers' (selected), 'Layouts', 'Spacers', etc. On the right, the 'Objet' (Object) tree shows the hierarchy of the main window, including 'MainWindow' with its children like 'centralWdg', 'frame', 'label_2', and 'groupBo'. Below the tree, properties for a selected 'radioButton' are listed, including 'minimumSize', 'maximumSize', 'sizeIncrement', 'baseSize', 'palette', 'font', 'cursor', 'mouseTracking', 'focusPolicy', 'contextMenuPolicy', 'acceptDrops', 'toolTip', and 'toolTipDuration'. At the bottom, tabs for 'Action Editor' and 'Signals & Slots Editor' are visible.



QComboBox : Bouton combiné avec liste popup

QFontComboBox : **ComboBox** permettant à l'utilisateur de choisir une police d'écriture

QLineEdit : Éditeur de texte mono-ligne

QTextEdit : Widget qui permet d'éditer et d'afficher des textes simples et riches

QPlainTextEdit : Widget qui est utilisé pour éditer et afficher du texte brut

QSpinBox : Champ de saisie numérique

QDoubleSpinBox : Champ de saisie acceptant des doubles

QTimeEdit : Widget pour éditer le temps basé sur le widget **QDateTimeEdit**

QDateEdit : Widget pour éditer des dates basé sur le widget **QDateTimeEdit**

QDateTimeEdit : Widget pour éditer des dates et heures

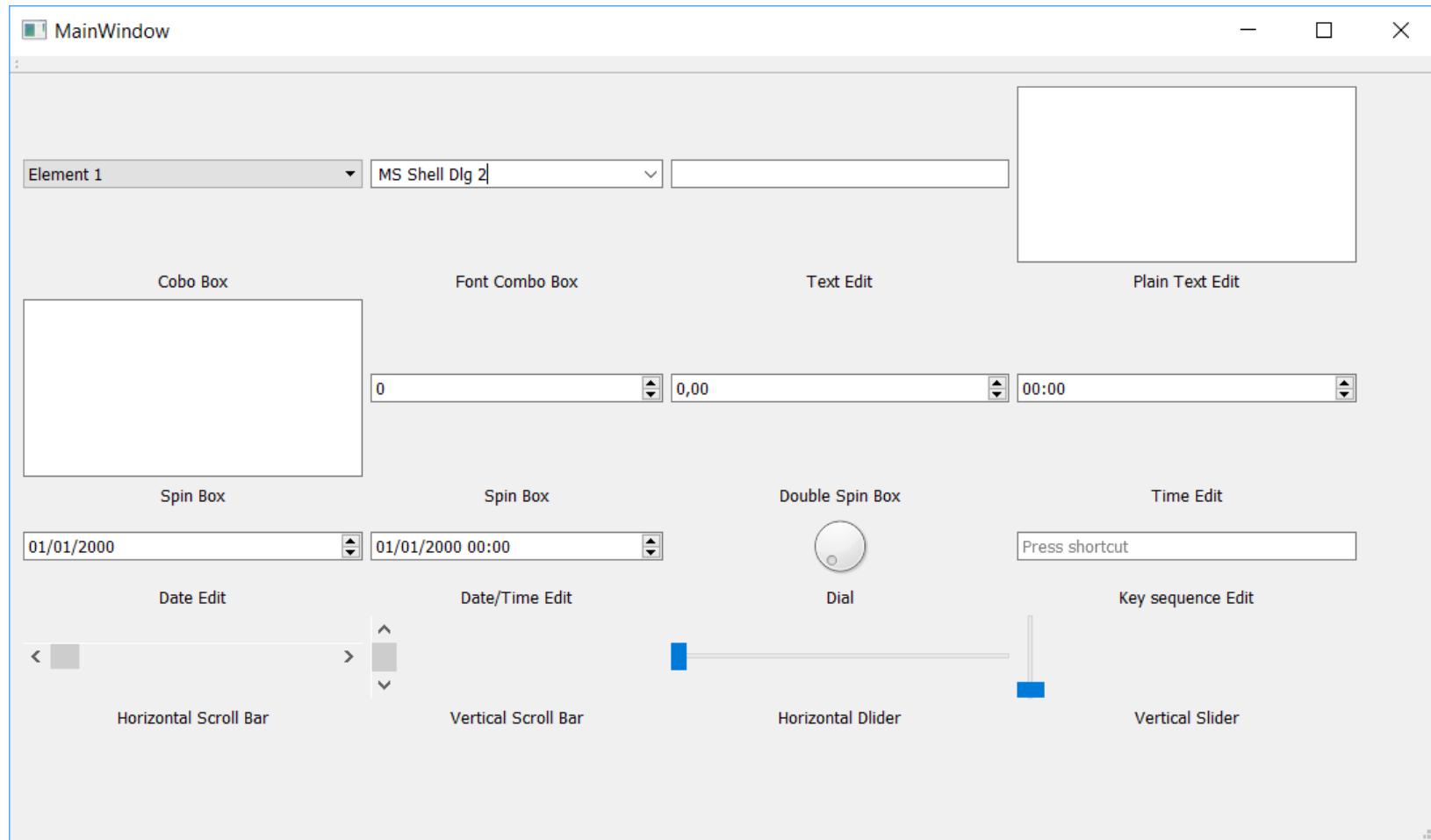
QDial : Cadran à aiguille rond (comme un compteur de vitesse ou un potentiomètre)

QScrollBar : Barre de défilement horizontale ou verticale

QSlider : Slider vertical ou horizontal

QKeySequenceEdit : Widget qui permet d'entrer un QKeySequence (encapsule une séquence de touches utilisée par les raccourcis) .

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Input Widgets





3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Input Widgets



mainwindow.ui - QtInput - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Qt Accueil Éditeur Design Projets Analyse Aide Hello world! QTinput Debug

mainwindow.ui - QtInput - Qt Creator

Taper ici

Element 1 MS Shell Dlg 2

Cobo Box Font Combo Box Text Edit Plain Text Edit

Spin Box Spin Box Double Spin Box Time Edit

Date Edit Date/Time Edit Dial Key sequence Edit

Horizontal Scroll Bar Vertical Scroll Bar Horizontal Slider Vertical Slider

Horizontal Scroll Bar Vertical Scroll Bar Horizontal Slider Vertical Slider

Émetteur Signal Receveur Slot

Action Editor Signals & Slots Editor

Objet Classe

comboBox	QComboBox
dateEdit	QDateEdit
date...Edit	QDateTimeEdit
dial	QDial
doubleBox	QDoubleSpinBox
font...Box	QFontComboBox
horiz...lBar	QScrollBar
horiz...lider	QSlider
keyS...Edit	QKeySequenceEdit
label	QLabel
label_11	QLabel
label_12	QLabel

Filter comboBox : QComboBox

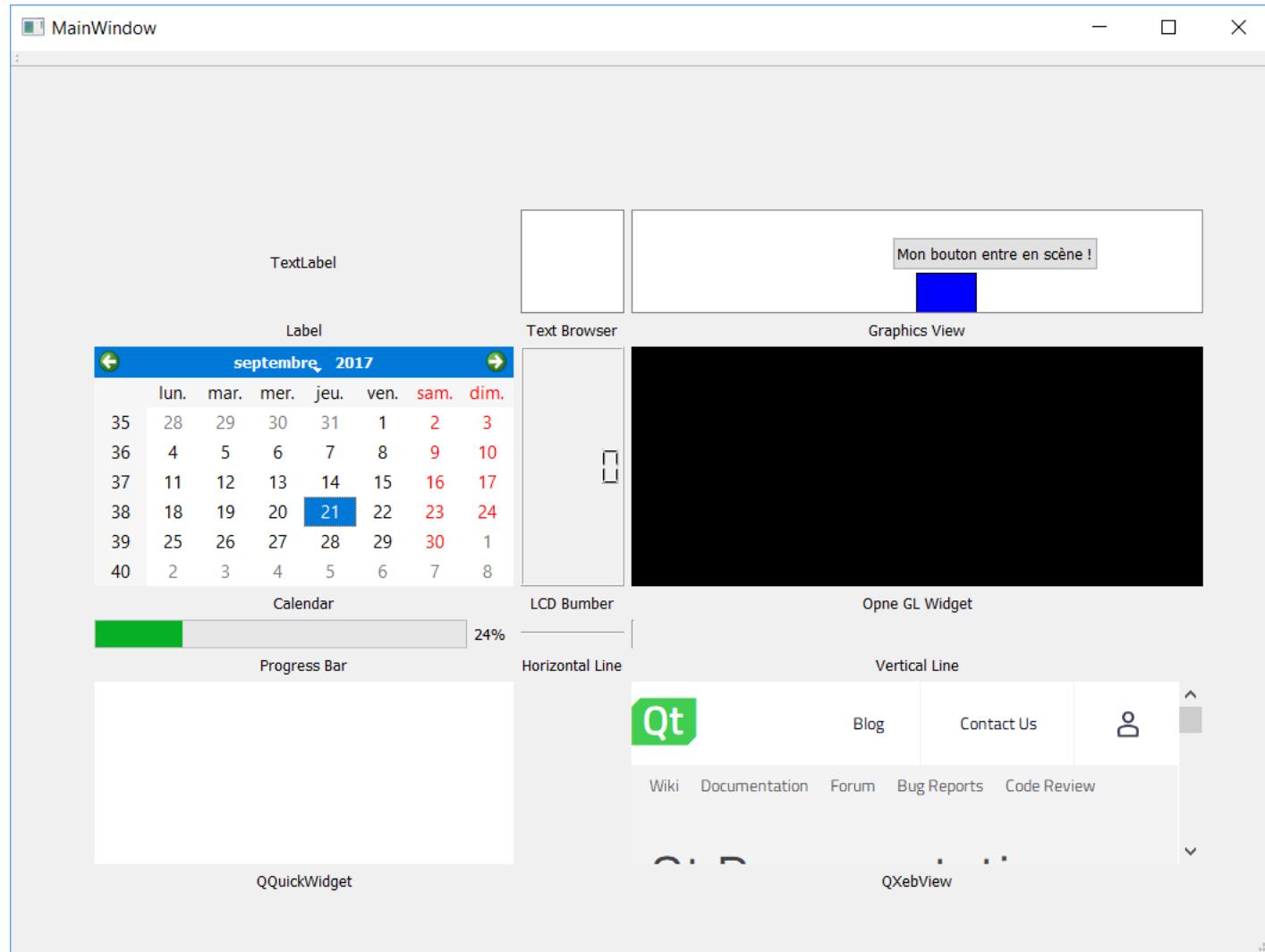
accessibleName
accessibleDescription
layoutDirection
autoFillBackground
styleSheet
locale French, France
inputMethodHints

QComboBox

editable
currentText Element 1
currentIndex 0
maxVisibleItems 10
maxCount 2147483647
insertPolicy InsertAtBottom
sizeAdjustPolicy AdjustToContentsOnFirstShow
minimumContentsLength 0
iconSize 20 x 20
duplicatesEnabled
frame
modelColumn 0

1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Display Widgets





QLabel : widget qui fournit un affichage de texte ou d'image

QTextBrowser : widget qui fournit un navigateur de texte enrichi avec une navigation hypertexte.

QGraphicsView : widget pour l'affichage du contenu d'une **QGraphicsScene** (surface pour gérer un grand nombre d'éléments graphiques 2D)

QLCDNumber : affiche un nombre avec des chiffres de type LCD

QCalendarWidget : widget de calendrier mensuel permettant à l'utilisateur de sélectionner une date

QProgressBar : fournit une barre de progression horizontale ou verticale.

Line : Frame définissant une ligne horizontale ou verticale fourni par QtDesigner (separ = new QFrame; separ->setFrameStyle(QFrame::VLine | QFrame::Raised);)

QOpenGLWidget : fournit des fonctionnalités pour afficher des graphiques OpenGL intégrés dans une application Qt.

QQuickWidget : widget pour afficher une interface utilisateur de type Qt Quick

QWebView : widget qui permet d'afficher et de modifier des documents Web

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Display Widgets



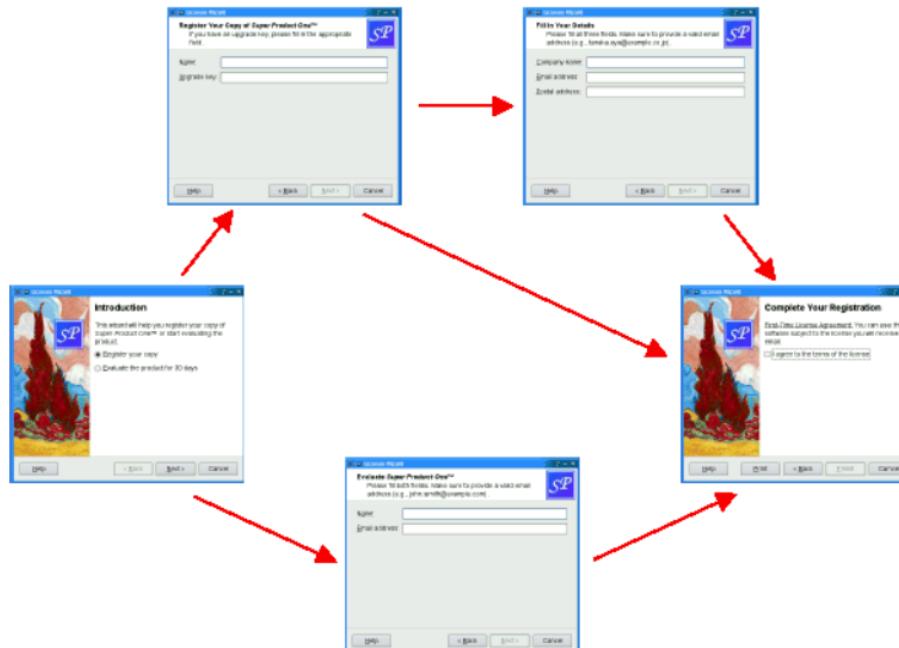
QDialog : classe de base des fenêtres de dialogue

- **QFileDialog** : fournit une boîte de dialogue qui permet aux utilisateurs de sélectionner des fichiers ou des répertoires
- **QProgressDialog** : fournit un aperçu sur la progression d'une opération lente
- **QColorDialog** : fournit un widget de dialogue pour spécifier les couleurs
- **QFontDialog** : fournit un widget de dialogue pour sélectionner une police de caractère
- **QErrorMessage** : fournit une boîte de dialogue d'affichage des messages d'erreur
- **QInputDialog** : fournit une boîte de dialogue de commodité simple pour obtenir une seule valeur de l'utilisateur
- **QMessageBox** : fournit une boîte de dialogue modale pour informer l'utilisateur ou pour demander à l'utilisateur une question et recevoir une réponse
- **QPageSetupDialog** : fournit une boîte de dialogue de configuration pour les options liées à la page sur une imprimante
- **QAbstractPrintDialog** : fournit une implémentation de base pour les boîtes de dialogue d'impression utilisées pour configurer les imprimantes
 - **QPrintDialog** : fournit une boîte de dialogue pour spécifier la configuration de l'imprimante.
- **QPrintPreviewDialog** : fournit une boîte de dialogue pour la prévisualisation et la configuration des mises en page pour la sortie de l'imprimante.



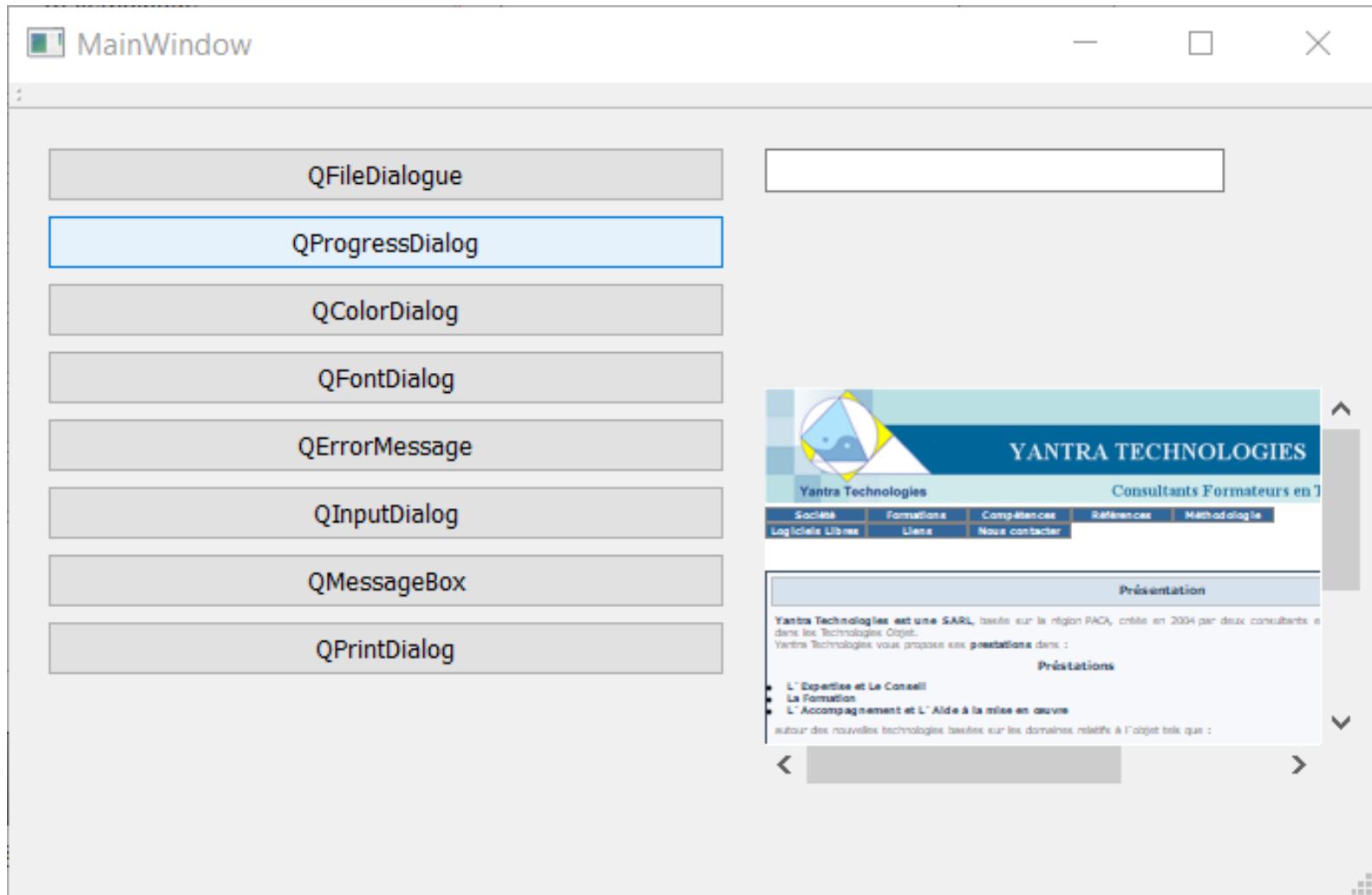
Qdialog : classe de base des fenêtres de dialogue

- **QWizard** : fournit un cadre pour les assistants



<http://doc.qt.io/qt-4.8/all-examples.html>

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Display Widgets



The screenshot shows the Qt Designer interface with a window titled "MainWindow". Inside the window, there is a vertical list of dialog boxes, each labeled with a class name. The "QProgressDialog" box is highlighted with a blue border. The other boxes are: QFileDialog, QColorDialog, QFontDialog, QErrorMessage, QInputDialog, QMessageBox, and QPrintDialog.



The screenshot shows a web page for "YANTRA TECHNOLOGIES". The header features the company logo and navigation links for "Société", "Formations", "Compétences", "Références", "Méthodologie", "Logiciels Utiles", "Liens", and "Nous contacter". The main content area has a title "Présentation" and a paragraph about the company's expertise in consulting, training, and accompaniment. It also lists "Présentations" and "autour des nouvelles technologies basées sur les domaines relatifs à l'objet tel que :



3.4 - les interfaces graphiques : QtGui/ QWidget

-> Qt Designer : Display Widgets



mainwindow.ui - QDialogue - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

mainwindow.ui

Filter

Widget MdiArea Dock Widget QAxWidget

Input Widgets

- ComboBox
- Font ComboBox
- LineEdit
- Text Edit
- PlainTextEdit
- SpinBox
- DoubleSpinBox
- TimeEdit
- DateEdit
- Date/TimeEdit
- Dial
- HorizontalScrollBar
- VerticalScrollBar
- HorizontalSlider
- VerticalSlider
- KeySequenceEdit

Display Widgets

- Label
- TextBrowser
- GraphicsView
- Calendar
- LCDNumber
- ProgressBar
- HorizontalLine
- VerticalLine
- OpenGLWidget
- QQQuickWidget
- QWebView

Taper ici

mainwindow.ui

centralWidget

gridLayout

pushButton

pushButton_2

pushButton_3

pushButton_4

pushButton_5

pushButton_6

pushButton_7

pushButton_8

verticalSpacer

lineEdit

webView

menuBar

mainToolBar

statusBar

Objet

MainWindow

centralWidget

gridLayout

pushButton

pushButton_2

pushButton_3

pushButton_4

pushButton_5

pushButton_6

pushButton_7

pushButton_8

verticalSpacer

lineEdit

webView

menuBar

mainToolBar

statusBar

Class

C

C

C

C

C

C

C

C

C

S

C

C

QM

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

QM

C

QMe

QToo

QStat

Filter

webView : QWebView

cursor Flèche

mouseTracking checked

focusPolicy WheelFocus

contextMenuPolicy NoContext

Action Editor Signals & Slots Editor

Nom Utilisé Texte Raccourci Vérifiable Info-bulle

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages générales Version Control

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Display Widgets



```
void MainWindow::on_pushButton_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Open file"),
                                                    "../..",
                                                    tr("Image Files (*.png *.jpg *.bmp);;Text files (*.txt);; Any files (*)"));
    this->ui->lineEdit->setText(fileName);
}

void MainWindow::on_pushButton_2_clicked()
{
    int numTasks = 100000;
    QProgressDialog progress("Task in progress...", "Cancel", 0, numTasks, this);
    progress.setWindowModality(Qt::WindowModal);

    for (int i = 0; i < numTasks; i++) {
        progress.setValue(i);

        if (progress.wasCanceled())
            break;
    }
    progress.setValue(numTasks);
}
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Display Widgets



```
void MainWindow::on_pushButton_4_clicked()
{
    bool ok;
    QFont c = QFontDialog::getFont(&ok, QFont("Helvetica [Cronyx]", 10), this);
    this->ui->lineEdit->setText(c.toString());
}

void MainWindow::on_pushButton_5_clicked()
{
    QErrorMessage dialog;
    dialog.showMessage("Message d'erreur");
    dialog.exec();
}

void MainWindow::on_pushButton_6_clicked()
{
    bool ok;
    QString text = QInputDialog::getText(this, tr("QInputDialog::getText()"),
                                         tr("User name:"), QLineEdit::Normal,
                                         QDir::home().dirName(), &ok);
    this->ui->lineEdit->setText(text);
}
```



3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Display Widgets



```
void MainWindow::on_pushButton_7_clicked()
{
    QMessageBox msgBox;
    msgBox.setText("The document has been modified.");
    msgBox.setInformativeText("Do you want to save your changes?");
    msgBox.setStandardButtons(QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel);
    msgBox.setDefaultButton(QMessageBox::Save);
    int ret = msgBox.exec();
    QString text;
    switch (ret) {
        case QMessageBox::Save:
            text="Save";
            break;
        case QMessageBox::Discard:
            text="Discard";
            break;
        case QMessageBox::Cancel:
            text="Cancel";
            break;
        default:
            text="...";
            break;
    }
    this->ui->lineEdit->setText(QString::number(ret)+ " " + text);

    QMessageBox msgBox2;
    msgBox2.setText("The document has been modified.");
    msgBox2.exec();
}
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Designer : Display Widgets



```
void MainWindow::on_pushButton_8_clicked()
{
    QPrinter* printer = new QPrinter(QPrinter::HighResolution);
    QPrintPreviewDialog* preview = new QPrintPreviewDialog(printer, this);
    preview->setWindowFlags( Qt::Window );
    connect(preview, SIGNAL(paintRequested(QPrinter *)), SLOT(printPreview(QPrinter *)));
    preview->exec();
}

void MainWindow::printPreview(QPrinter *printer)
{
    ui->webView->print(printer);
}
```



Yantra Technologies

3.4 - les interfaces graphiques : Exemple Copier-Coller



Text Editor

test a copier

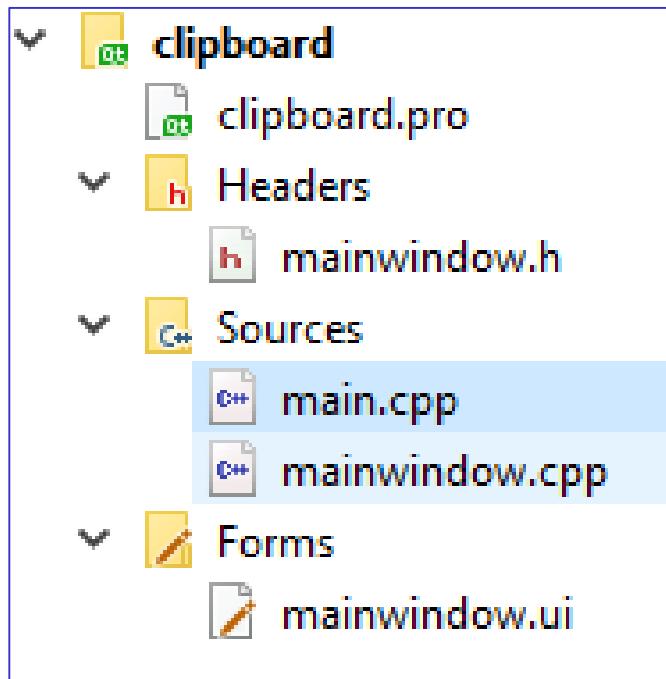
Clipboard Contents

Formats => text/html : application/vnd.oasis.opendocument.text : text/plain

```
text/html => <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">
<html><head><meta name="qrichtext" content="1" /><meta http-equiv="Content-Type" content="text/html; charset=utf-8" /><style type="text/css">
p, li { white-space: pre-wrap; }
</style></head><body>
<p style="margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;"><!--StartFragment-->test a copier<!--EndFragment--></p></body></html>
application/vnd.oasis.opendocument.text => PK□□□
text/plain => test a copier
```



3.4 - les interfaces graphiques : Exemple Copier-Coller



```
mainwindow.h
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QClipboard>

namespace Ui {
    class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget* parent = 0);
    ~MainWindow();

public slots:
    void clipboardChanged(QClipboard::Mode);

private:
    Ui::MainWindow* ui;
};

#endif // MAINWINDOW_H
```



3.4 - les interfaces graphiques : Exemple Copier-Coller



```
mainwindow.cpp      ▾ X | #  MainWindow::clipboardChanged(QClipboard::Mode): void
#include <QtGui>
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget* parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    connect (qApp->clipboard(), SIGNAL(changed(QClipboard::Mode)),
             this, SLOT(clipboardChanged(QClipboard::Mode)));
}

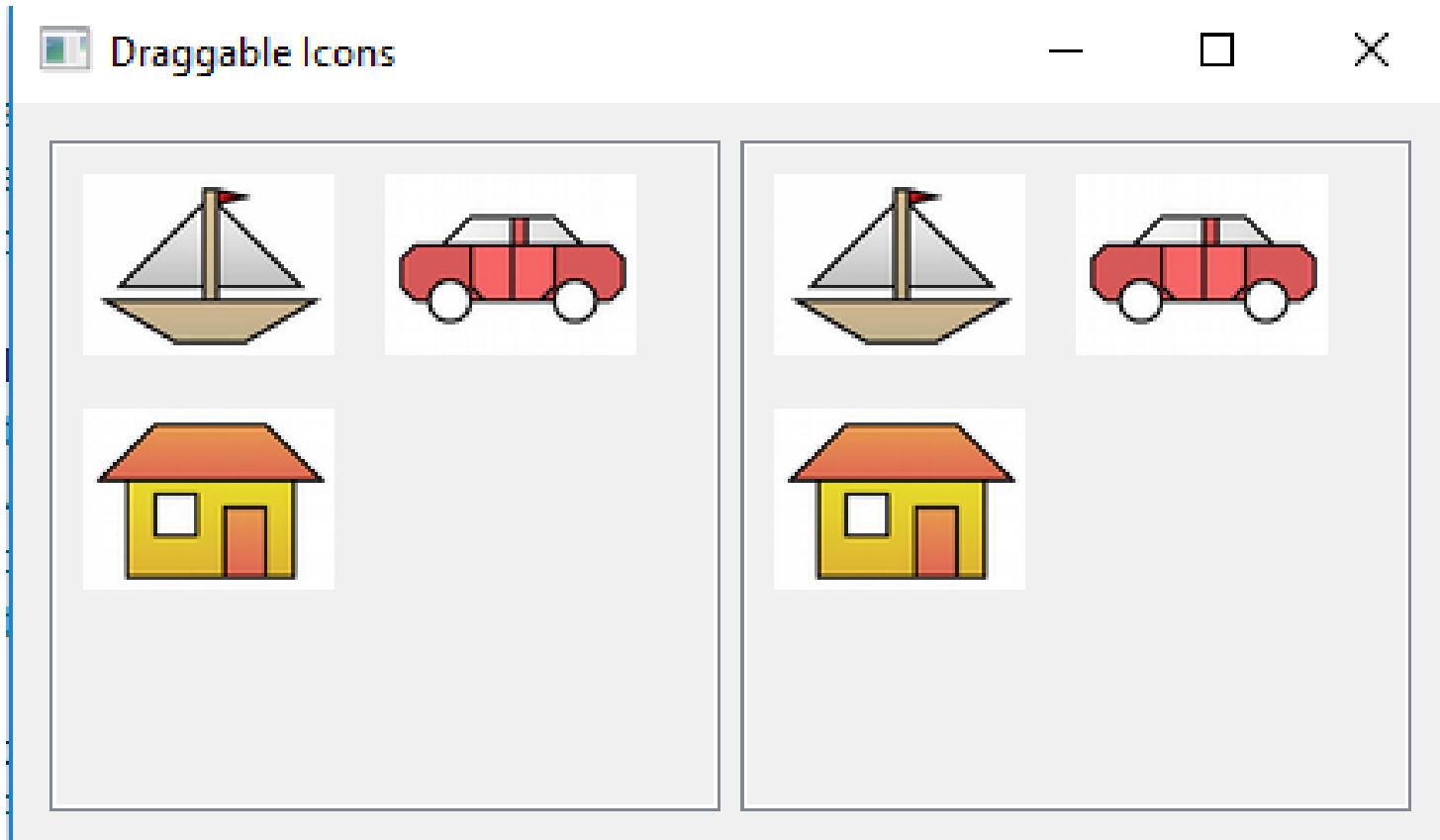
void MainWindow::clipboardChanged(QClipboard::Mode) {
    QStringList stringlist;
    QClipboard *clipboard = qApp->clipboard();
    constQMimeData *minedata = clipboard->mimeType();

    stringlist << "Formats => " + minedata->formats().join(" : ")<<"\n";
    foreach (QString format, minedata->formats()) {
        QByteArray ba = minedata->data(format);
        stringlist << "\t" + format + " => " + QString(ba) ;
    }
    ui->clipboardContentsEdit->setText(stringlist.join("\n"));
}

MainWindow::~MainWindow()
{
    delete ui;
}
```



3.4 - les interfaces graphiques : Exemple Drag and Drop



<http://doc.qt.io/qt-5/qtwidgets-draganddrop-draggableicons-example.html>



3.4 - les interfaces graphiques : Exemple Drag and Drop



The screenshot shows the project structure of a Qt application named "draggableicons". The project includes a .pro file, Headers, Sources, and Resources. The Sources folder contains dragwidget.h and dragwidget.cpp, with dragwidget.cpp currently selected. The Resources folder contains a qrc file with images for boat, car, and house.

```
#include <QApplication>
#include <QHBoxLayout>

#include "dragwidget.h"

int main(int argc, char *argv[])
{
    Q_INIT_RESOURCE(draggableicons);

    QApplication app(argc, argv);

    QWidget mainWidget;
    QHBoxLayout *horizontalLayout = new QHBoxLayout(&mainWidget);
    horizontalLayout->addWidget(new DragWidget());
    horizontalLayout->addWidget(new DragWidget());

    mainWidget.setWindowTitle(QObject::tr("Draggable Icons"));
    mainWidget.show();

    return app.exec();
}
```



3.4 - les interfaces graphiques : Exemple Drag and Drop



```
#ifndef DRAGWIDGET_H
#define DRAGWIDGET_H

#include <QFrame>

QT_BEGIN_NAMESPACE
class QDragEnterEvent;
class QDropEvent;
QT_END_NAMESPACE

//! [0]
class DragWidget : public QFrame
{
public:
    explicit DragWidget(QWidget *parent = nullptr);

protected:
    void dragEnterEvent(QDragEnterEvent *event) override;
    void dragMoveEvent(QDragMoveEvent *event) override;
    void dropEvent(QDropEvent *event) override;
    void mousePressEvent(QMouseEvent *event) override;
};

//! [0]

#endif // DRAGWIDGET_H
```



3.4 - les interfaces graphiques : Exemple Drag and Drop



```
DragWidget::DragWidget(QWidget *parent)
    : QFrame(parent)
{
    setMinimumSize(200, 200);
    setFrameStyle(QFrame::Sunken | QFrame::StyledPanel);
    setAcceptDrops(true);

    QLabel *boatIcon = new QLabel(this);
    boatIcon->setPixmap(QPixmap(":/images/boat.png"));
    boatIcon->move(10, 10);
    boatIcon->show();
    boatIcon->setAttribute(Qt::WA_DeleteOnClose);

    QLabel *carIcon = new QLabel(this);
    carIcon->setPixmap(QPixmap(":/images/car.png"));
    carIcon->move(100, 10);
    carIcon->show();
    carIcon->setAttribute(Qt::WA_DeleteOnClose);

    QLabel *houseIcon = new QLabel(this);
    houseIcon->setPixmap(QPixmap(":/images/house.png"));
    houseIcon->move(10, 80);
    houseIcon->show();
    houseIcon->setAttribute(Qt::WA_DeleteOnClose);
}
```



3.4 - les interfaces graphiques : Exemple Drag and Drop



Nous enverrons des données de **pixmap** pour l'icône et des informations sur le clic de l'utilisateur dans le widget d'icône,
nous construisons un **QByteArray** et faisons un package des détails dans un **QDataStream**,

```
void DragWidget::mousePressEvent(QMouseEvent *event)
{
    QLabel *child = static_cast<QLabel*>(childAt(event->pos()));
    if (!child)
        return;

    QPixmap pixmap = *child->pixmap();

    QByteArray itemData;
    QDataStream dataStream(&itemData, QIODevice::WriteOnly);
    dataStream << pixmap << QPoint(event->pos() - child->pos());
    // [1]

    //! [2]
    QMimeData *mimeData = new QMimeData;
    mimeData->setData("application/x-dnditemdata", itemData);
    //! [2]

    //! [3]
    QDrag *drag = new QDrag(this);
    drag->setMimeData(mimeData);
    drag->setPixmap(pixmap);
    drag->setHotSpot(event->pos() - child->pos());
    //! [3]

    QPixmap tempPixmap = pixmap;
    QPainter painter;
    painter.begin(&tempPixmap);
    painter.fillRect(pixmap.rect(), QColor(127, 127, 127, 127));
    painter.end();

    child->setPixmap(tempPixmap);

    if (drag->exec(Qt::CopyAction | Qt::MoveAction, Qt::CopyAction) == Qt::MoveAction) {
        child->close();
    } else {
        child->show();
        child->setPixmap(pixmap);
    }
}
```

Pour activer le glisser de l'icône, nous devons agir sur un événement **QMouseEvent**

Pour l'interopérabilité, les opérations de glisser-déposer décrivent les données qu'elles contiennent à l'aide des types MIME



3.4 - les interfaces graphiques : Exemple Drag and Drop



```

void DragWidget::mousePressEvent(QMouseEvent *event)
{
    QLabel *child = static_cast<QLabel*>(childAt(event->pos()));
    if (!child)
        return;

    QPixmap pixmap = *child->pixmap();

    QByteArray itemData;
    QDataStream dataStream(&itemData, QIODevice::WriteOnly);
    dataStream << pixmap << QPoint(event->pos() - child->pos())
//! [1]

//! [2]
    QMimeData *mimeData = new QMimeData;
    mimeData->setData("application/x-dnditemdata", itemData);
//! [2]

//! [3]
    QDrag *drag = new QDrag(this);
    drag->setMimeData(mimeData);
    drag->setPixmap(pixmap);
    drag->setHotSpot(event->pos() - child->pos());
//! [3]

    QPixmap tempPixmap = pixmap;
    QPainter painter;
    painter.begin(&tempPixmap);
    painter.fillRect(pixmap.rect(), QColor(127, 127, 127, 127));
    painter.end();

    child->setPixmap(tempPixmap);

    if (drag->exec(Qt::CopyAction | Qt::MoveAction, Qt::CopyAction) == Qt::MoveAction) {
        child->close();
    } else {
        child->show();
        child->setPixmap(pixmap);
    }
}

```

Nous définissons un pixmap qui sera affiché à côté du curseur pendant l'opération

Nous définissons la position du **hotspot** qui le place curseur sous le pixmap,

L'opération glisser-déposer elle-même est gérée par un objet **QDrag**

Nous passons les données sur l'image : **mineData** à l'objet **drag**





3.4 - les interfaces graphiques : Exemple Drag and Drop



```
void DragWidget::dragEnterEvent(QDragEnterEvent *event)
{
    if (event->mimeData()->hasFormat("application/x-dnditemdata")) {
        if (event->source() == this) {
            event->setDropAction(Qt::MoveAction);
            event->accept();
        } else {
            event->acceptProposedAction();
        }
    } else {
        event->ignore();
    }
}

void DragWidget::dragMoveEvent(QDragMoveEvent *event)
{
    if (event->mimeData()->hasFormat("application/x-dnditemdata")) {
        if (event->source() == this) {
            event->setDropAction(Qt::MoveAction);
            event->accept();
        } else {
            event->acceptProposedAction();
        }
    } else {
        event->ignore();
    }
}
```



3.4 - les interfaces graphiques : Exemple Drag and Drop



```
void DragWidget::dropEvent(QDropEvent *event)
{
    if (event->mimeData()->hasFormat("application/x-dnditemdata")) {
        QByteArray itemData = event->mimeData()->data("application/x-dnditemdata");
        QDataStream dataStream(&itemData, QIODevice::ReadOnly);

        QPixmap pixmap;
        QPoint offset;
        dataStream >> pixmap >> offset;

        QLabel *newIcon = new QLabel(this);
        newIcon->setPixmap(pixmap);
        newIcon->move(event->pos() - offset);
        newIcon->show();
        newIcon->setAttribute(Qt::WA_DeleteOnClose);

        if (event->source() == this) {
            event->setDropAction(Qt::MoveAction);
            event->accept();
        } else {
            event->acceptProposedAction();
        }
    } else {
        event->ignore();
    }
}
```



QtChar module permet de créer des diagrammes à barres, circulaires, radars et autres.

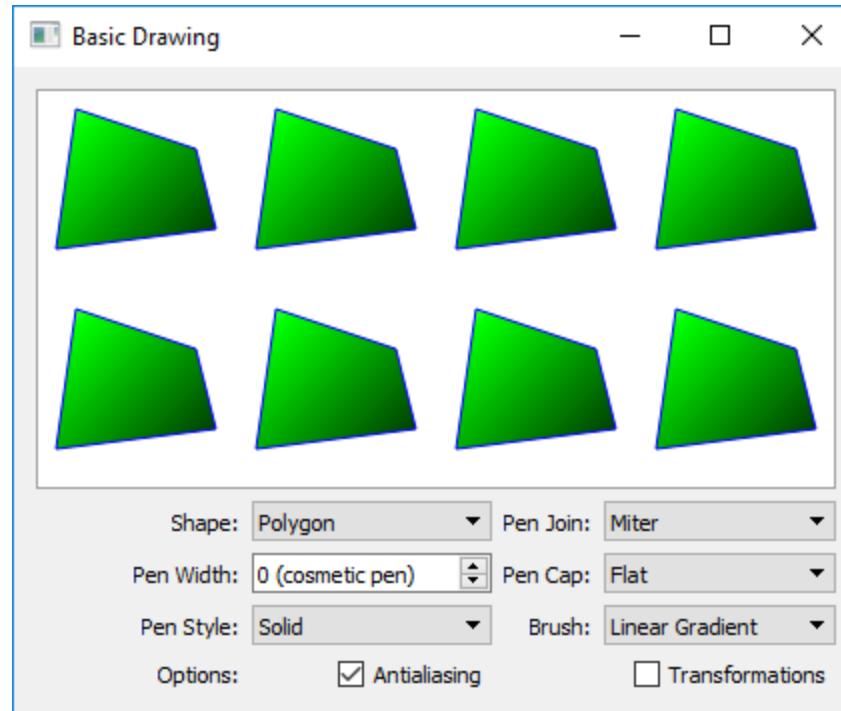
Module optionnel disponible avec le kit depuis la version 5.6 sous licence libre.

QT += charts

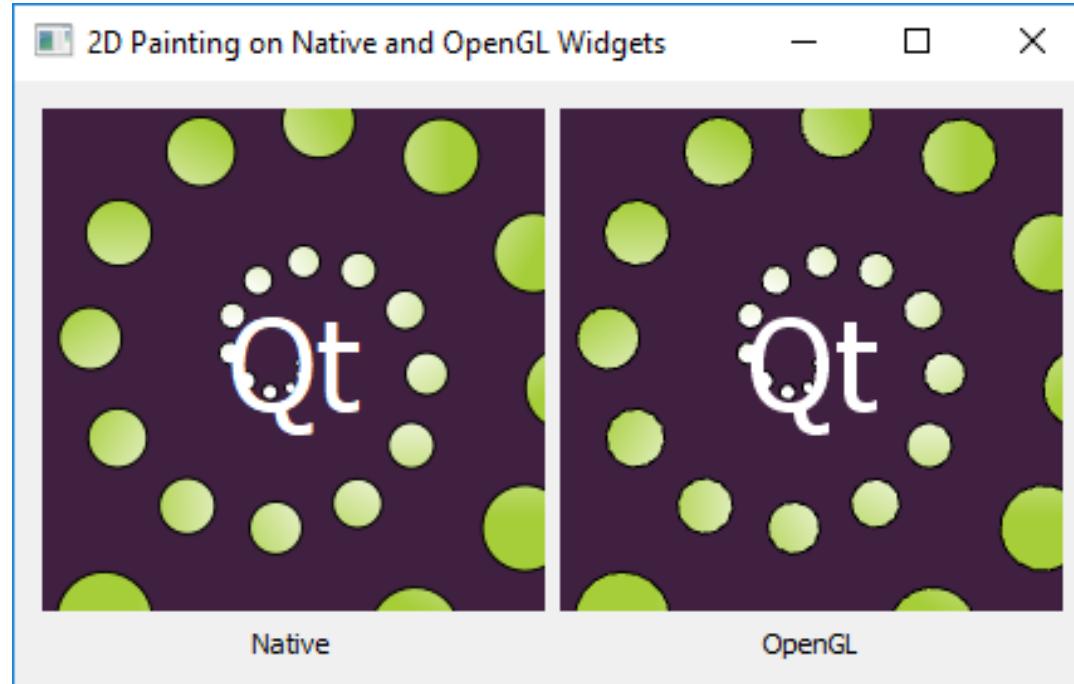
3.4 - les interfaces graphiques : QtGui/ QWidget -> Qt Charts



3.4 - les interfaces graphiques : QtGui/ QWidget -> QPainter



3.4 - les interfaces graphiques : QtGui/ QWidget -> 2D OpenGL



3.4 - les interfaces graphiques : QtGui/ QWidget -> 2D OpenGL



http://guillaume.belz.free.fr/doku.php?id=qt_opengl_-_overpainting

C++ , Qt, OpenGL, CUDA

S'enregistrer Connexion

Derniers changements Gestionnaire de médias Index

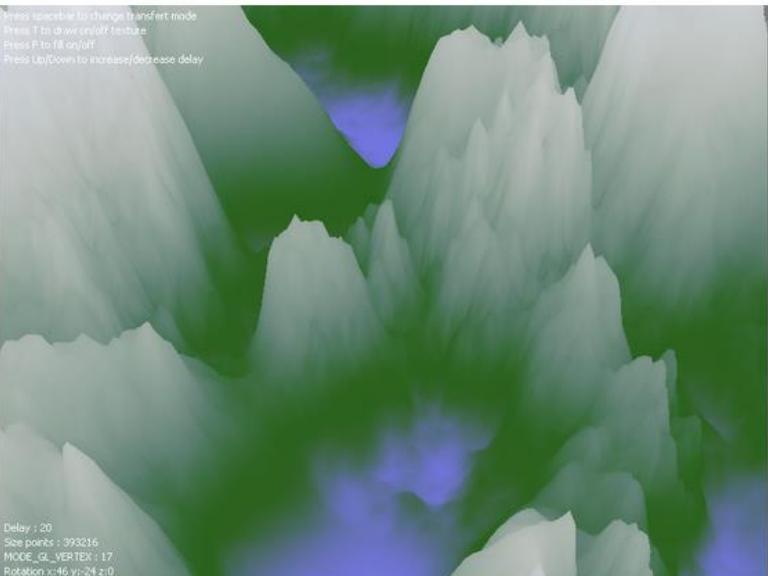
Vous êtes ici: C++, Qt, OpenGL, CUDA » Qt OpenGL - Générer un terrain

Qt OpenGL - Générer un terrain

- » Introduction à OpenGL et Qt 5.4
- » Support d'OpenGL dans Qt
- » Qt OpenGL - Générer un terrain
- » Qt OpenGL - Envoyer des données au processeur graphique
- » Qt OpenGL - Utilisation du pipeline programmable
- » Qt OpenGL - Ajouter des lumières et des textures
- » Qt OpenGL - Réaliser un rendu offscreen
- » Qt OpenGL - Overpainting : dessiner en 2D avec QPainter sur une scène 3D
- » Qt OpenGL - Gestion des extensions avec QGLContext::getProcAddress()
- » Qt OpenGL - Annexes

Cette partie présente la structure de base utilisée pour réaliser notre application de génération de terrain. Après avoir rappelé les mécanismes de base pour réaliser du rendu 3D dans une application Qt, nous détaillerons la problématique du chargement de notre terrain. Enfin, un système simple de mesure des performances, utilisé tout au long du tutoriel, sera mis en place.

Press spacebar to change transfert mode
Press T to draw on/off texture
Press F to fil on/off
Press Up/Down to increase/decrease delay



Delay : 20
Size points : 393216
MODE_GL_VERTEX : 17
Rotation x:46 y:-24 z:0

3.4 - les interfaces graphiques : QtGui/ QWidget -> interaction avec le C++



MainWindow

File

Entrer vos coordonnées

Nom

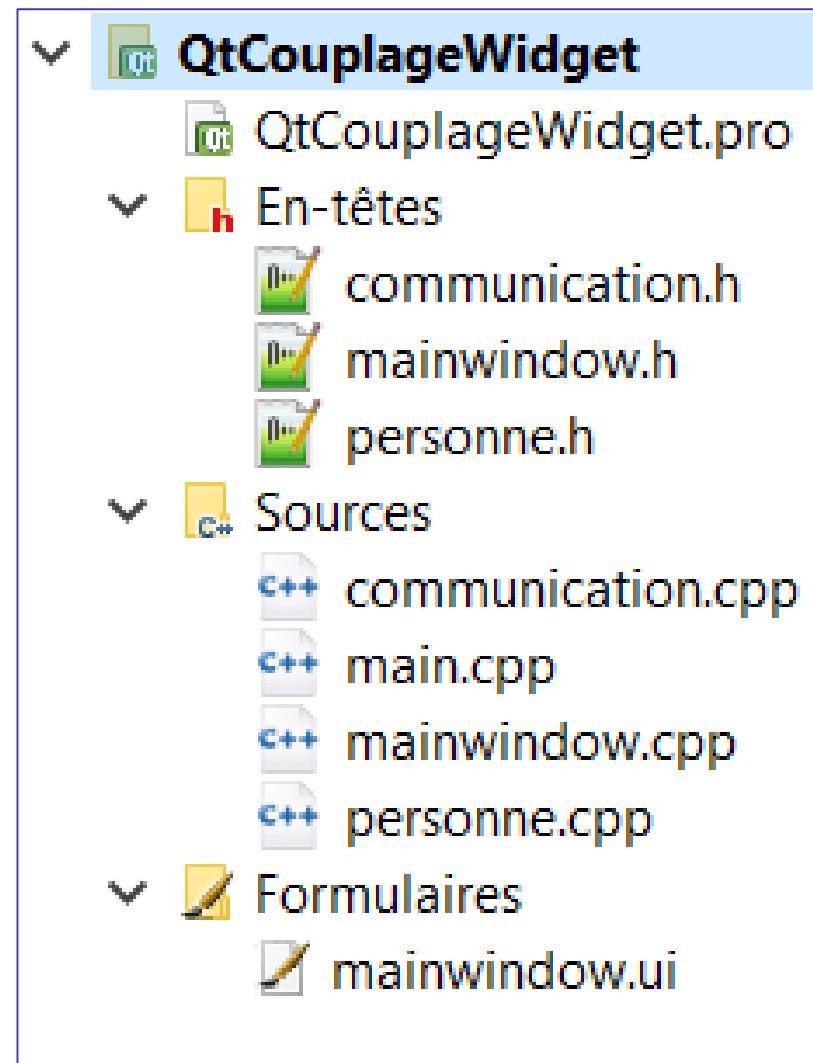
Prénom

Adresse

Message validation

Vous allez valider vos coordonnées

3.4 - les interfaces graphiques : QtGui/ QWidget -> interaction avec le C++



3.4 - les interfaces graphiques : QtGui/ QWidget -> interaction avec le C++



```
QtCouplageWidget.pro*  
#  
# Project created by QtCreator 2017-10-04T16:42:56  
#  
#-----|  
QT      += core gui  
  
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets  
  
TARGET = QtCouplageWidget  
TEMPLATE = app  
  
SOURCES += main.cpp \  
          mainwindow.cpp \  
          personne.cpp \  
          communication.cpp  
  
HEADERS  += mainwindow.h \  
            personne.h \  
            communication.h  
  
FORMS     += mainwindow.ui  
  
QMAKE_CXXFLAGS += -Wno-unused-but-set-variable  
QMAKE_CXXFLAGS += -Wno-unused-variable  
QMAKE_CXXFLAGS += -Wno-unused-parameter  
QMAKE_CXXFLAGS += -D_DEBUG  
QMAKE_CXXFLAGS += -std=c++11  
QMAKE_CXXFLAGS += -std=c++14  
QMAKE_CXXFLAGS += -Wno-cpp  
#QMAKE_CXXFLAGS += -Weffc++  
QMAKE_CXXFLAGS += -Wall  
QMAKE_CXXFLAGS += -Wextra  
QMAKE_CXXFLAGS += -fbounds-check  
QMAKE_CXXFLAGS += -malign-double  
QMAKE_CXXFLAGS += -mfpmath=sse  
QMAKE_CXXFLAGS += -msse2  
QMAKE_CXXFLAGS += -DLinux  
QMAKE_CXXFLAGS += -DNQML
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



main.cpp - QtCouplage - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

mainwindow.ui

Filter Taper ici

File Entrer vos coordonnées

Nom :

Prénom :

Adresse :

Valider

Effacer

Aide

Button

Objet Classe

pushButton_2	QPushButton
pushButton_3	QPushButton
pushButton_4	QPushButton
lineEdit	QTextEdit
lineEdit_2	QTextEdit
menuBar	QMenuBar

Filter textEdit_2 : QTextEdit

QObject

QWidget

enabled

geometry [(140, 380), 241 x 131]

X 140

Y 380

Largeur 241

Hauteur 131

sizePolicy [Expanding, Expanding, 0, 0]

Police horizontale Expanding

Police verticale Expanding

Étirement horizontal 0

Étirement vertical 0

minimumSize 0 x 0

Largeur 0

Hauteur 0

maximumSize 16777215 x 16777215

sizeIncrement 0 x 0

baseSize 0 x 0

palette Modifier la palette

font A [MS Shell Dlg 2, 8]

+ -

Émetteur Signal Receveur Slot

actionExit triggered() MainWindow close()

Action Editor Signals & Slots Editor

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages généraux Version Control

3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



```
mainwindow.h          X  getCaption() const: unsigned
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <communication.h>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

    unsigned getCpt() const { return this->cpt; }

private slots:
    void on_pushButton_clicked();
    void on_pushButton_2_clicked();
    void on_pushButton_3_clicked();
    void on_pushButton_4_clicked();
    void messageOpen();

private:
    Ui::MainWindow *ui;
    PControleur::Communication id_Communication;
    unsigned cpt;
};

#endif // MAINWINDOW_H
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



```
c++ mainwindow.cpp*      X  MainWindow::messageOpen(): void
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <iostream>
#include <qmessagebox.h>
#include <QDebug>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow), id_Communication(parent), cpt(0)
{
    ui->setupUi(this);
    ui->pushButton_4->setVisible(false);
    ui->textEdit_2->setVisible(false);

    connect(ui->actionOpen, SIGNAL(triggered()), this, SLOT(messageOpen()));
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::messageOpen() { [...] }

void MainWindow::on_pushButton_clicked() [...] 
void MainWindow::on_pushButton_2_clicked() [...] 
void MainWindow::on_pushButton_3_clicked() [...] 
void MainWindow::on_pushButton_4_clicked() [...] 
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



```
void MainWindow::messageOpen () {  
  
    auto message = new QMessageBox(this);  
    message->setWindowTitle("Action Message");  
    message->setText("Open action triggered");  
    message->show();  
}
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



```
void MainWindow::on_pushButton_clicked()
{
    QMessageBox message(this);
    message.setWindowTitle("Message validation");
    message.setStandardButtons(QMessageBox::Yes | QMessageBox::No | QMessageBox::Abort);
    if ( ui->pushButton->text() != "Valider" ) {
        message.setText("Vous allez modifier vos coordonnées");
        switch (message.exec()) { [...] }

    } else if ( ui->textEdit->toPlainText().replace(' ', ' ') != "" ||
                ui->lineEdit->text().replace(' ', ' ') != "" ||
                ui->lineEdit_2->text().replace(' ', ' ') != "" )
    {
        message.setText("Vous allez valider vos coordonnées");
        switch (message.exec()) { [...] }
    }
    else {
        QMessageBox message(this);
        message.setWindowTitle("Message");
        message.setText("Veuillez remplir tous les champs.");
        message.setStandardButtons(QMessageBox::Ok);
        message.exec();
    }
}
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



```
if ( ui->pushButton->text() != "Valider" ) {
    message.setText("Vous allez modifier vos coordonnées");
    switch (message.exec()) {
        case QMessageBox::Yes:
            ui->pushButton->setText("Valider");
            ui->pushButton_4->setEnabled(true);
            ui->pushButton_4->setVisible(false);
            ui->lineEdit->setReadOnly(false);
            ui->lineEdit_2->setReadOnly(false);
            ui->textEdit->setReadOnly(false);
            ui->pushButton_2->setVisible(true);

            this->id_Communication.add_Personne(ui->lineEdit->text(),
                                                    ui->lineEdit_2->text(),
                                                    ui->textEdit->toPlainText());

            break;
        case QMessageBox::No:
            std::cout << "Modifications non validees" << std::endl;
            break;
        case QMessageBox::Abort:
            std::cout << "Reste sur la page" << std::endl;
            break;
        default:
            break;
    }
}
```

3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



```
    } else if ( ui->textEdit->toPlainText().replace(' ', ' ') != "" ||  
                ui->lineEdit->text().replace(' ', ' ') != "" ||  
                ui->lineEdit_2->text().replace(' ', ' ') != "" )  
{  
    message.setText("Vous allez valider vos coordonnées");  
    switch (message.exec()) {  
    case QMessageBox::Yes:  
        ui->pushButton->setText("Modifier");  
        ui->pushButton_4->setEnabled(true);  
        ui->pushButton_4->setVisible(true);  
        ui->lineEdit->setReadOnly(true);  
        ui->lineEdit_2->setReadOnly(true);  
        ui->textEdit->setReadOnly(true);  
        ui->pushButton_2->setVisible(false);  
  
        cpt++;  
        id_Communication.setName(ui->lineEdit->text() + QString::number( cpt ) );  
        id_Communication.add_Personne(ui->lineEdit->text(),  
                                       ui->lineEdit_2->text(),  
                                       ui->textEdit->toPlainText());  
        ui->textEdit_2->setText( id_Communication.toStringPersonne() );  
  
        break;  
    case QMessageBox::No:  
        std::cout << "Modifications non validees" << std::endl;  
        break;  
    case QMessageBox::Abort:  
        std::cout << "Reste sur la page" << std::endl;  
        break;  
    default:  
        break;  
    }  
}
```

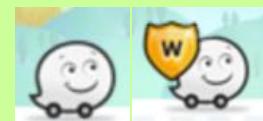
3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



```
void MainWindow::on_pushButton_2_clicked()
{
    QMessageBox message(this);
    message.setWindowTitle("Message validation");
    message.setText("voulez vous effacer tous les champs ?");
    message.setStandardButtons(QMessageBox::Yes | QMessageBox::No);

    switch (message.exec()) {
    case QMessageBox::Yes:
        ui->lineEdit->setText("");
        ui->lineEdit_2->setText("");
        ui->textEdit->setText("");
        break;
    case QMessageBox::No:
        std::cout << "Modifications non validees" << std::endl;
        break;
    }
}
```

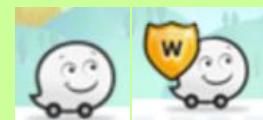
3.4 - les interfaces graphiques : QtGui/ QWidget -> QWidget interaction avec le C++



```
void MainWindow::on_pushButton_3_clicked()
{
    QMessageBox message(this);
    message.setWindowTitle("Message");
    message.setText("Aide : Il faut remplir tous les champs et valider");
    message.setStandardButtons(QMessageBox::Ok);
    message.exec();
}
```

3.4 - les interfaces graphiques : QtGui/ QWidget

-> QWidget interaction avec le C++



```
void MainWindow::on_pushButton_4_clicked()
{
    if ( ui->groupBox->size().height() == 600) {
        ui->groupBox->setGeometry(10,10,780,420);
        ui->centralWidget->setGeometry(0,0,830,455);
        this->setGeometry(this->geometry().x(),this->geometry().y(),830,520);
        ui->centralWidget->show();
        ui->textEdit_2->setVisible(false);
        ui->textEdit_2->setEnabled(false);
    } else {

        ui->groupBox->setGeometry(10,10,780,600);
        ui->centralWidget->setGeometry(0,0,827,620);
        this->setGeometry(this->geometry().x(),this->geometry().y(),827,700);
        ui->textEdit_2->setVisible(true);
        ui->textEdit_2->setEnabled(false);
    }
}
```



Le module QtTest est intégré à votre projet grâce à l'ajout du mot-clé testlib dans la déclaration QT de votre fichier.

.pro. : QT += testlib

<http://doc.qt.io/qt-5/qtest.html>

- **QCOMPARE**(*actual, expected*)
- **QVERIFY2**(*condition, message*)
- **QVERIFY**(*condition*)

- **QTEST_MAIN**(*TestClass*)



4 – QTTest : assertion



QCOMPARE(valeur, reference)	Compare une valeur avec une valeur de référence.
QFAIL(message)	Arrete l'exécution du scénario avec un message d'erreur.
QSKIP(message)	Arrete l'exécution du test sans arrêter l'exécution du scénario.
QTRY_COMPARE_WITH_TIMEOUT (valeur, reference, timeout)	Compare 2 valeurs de manière répétée jusqu'à ce qu'elles soient égales ou que le timeout soit atteint. Les événements de la run-loop sont exécutés pendant ce temps, permettant aux autres traitements de modifier la valeur testée éventuellement. Remarque : Il s'agit là d'un test en condition de fonctionnement de l'application et non d'un test statique.
QTRY_COMPARE (valeur, reference)	identique à QTRY_COMPARE_WITH_TIMEOUT avec un timeout prédéfini à 5 secondes.
QVERIFY(condition)	vérifie que la valeur est vraie.
QVERIFY2(condition, message)	vérifie que la condition est vraie et affiche un message dans le cas contraire.
QWARN(message)	affiche un message d'avertissement.
QTRY_VERIFY_WITH_TIMEOUT (condition, timeout)	exécute le test jusqu'à ce que la condition soit vraie ou que le timeout soit atteint. Les événements de la run-loop sont traités pendant ce temps.
QTRY_VERIFY(condition)	identique à la précédente avec un timeout prédéfini à 5 secondes.

4 - QTest



```
// reference : http://doc.qt.io/qt-5/qttestlib-tutorial1-example.html
#include <QtTest/QtTest>

class TestQString: public QObject
{
    Q_OBJECT
private slots:
    void toUpper();
};

void TestQString::toUpper()
{
    QString str = "Hello";
    QVERIFY(str.toUpper() == "HELLO");
    QVERIFY2(str.toUpper() == "HELLO", "toUpper - Erreur 001");
    QCOMPARE(str.toUpper(), QString("HELLO"));
}

QTEST_MAIN(TestQString)
#include "TestQString.moc"
```

```
***** Start testing of TestQString *****
Config: Using QTest library 5.10.0, Qt 5.10.0 (i386-little_endian-ilp32 shared (dynamic) debug build; by GCC 5.3.0)
PASS  : TestQString::initTestCase()
PASS  : TestQString::toUpper()
PASS  : TestQString::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped, 0 blacklisted, 1ms
***** Finished testing of TestQString *****
```



QFETCH (*type, name*) : récupère dans le jeu de données une nouvelle ligne de données du **type** dans la colonne spécifiée **name**.

void QTest::addColumn (const char * *name*, T * *dummy* = 0) : permet de créer une nouvelle colonne avec le type T et de lui donner un nom

QTestData & QTest::newRow (const char * *dataTag*) : permet d'ajouter une nouvelle ligne au jeu de test en lui donnant un nom



4 - QTTest



```
class TestQString: public QObject
{
    Q_OBJECT
private slots:
    void toUpper();
    void toLower_data ();
    void toLower();
};
```

```
void TestQString :: toLower_data ()
{
    QTest ::addColumn < QString > ( "chaine" );
    QTest ::addColumn < QString > ( "resultat" );

    QTest :: newRow ( "all lower" ) << "BONJOUR" << "bonjour" ;
    QTest :: newRow ( "mixed" )      << "Bonjour" << "bonjour" ;
    QTest :: newRow ( "all upper" ) << "bonjour" << "bonjour" ;
}

void TestQString :: toLower ()
{
    QFETCH ( QString , chaine);
    QFETCH ( QString , resultat);

    QCMPARE (chaine.toLower() , resultat);
}
```

index	name	string	result
0	all lower	"hello"	HELLO
1	mixed	"Hello"	HELLO
2	all upper	"HELLO"	HELLO



```
***** Start testing of TestQString *****
Config: Using QTest library 5.10.0, Qt 5.10.0 (i386-little_endian-ilp32 shared (dynamic) debug build; by GCC 5.3.0)
PASS : TestQString::initTestCase()
PASS : TestQString::toUpper()
PASS : TestQString::toLower(all lower)
PASS : TestQString::toLower(mixed)
PASS : TestQString::toLower(all upper)
PASS : TestQString::cleanupTestCase()
Totals: 6 passed, 0 failed, 0 skipped, 0 blacklisted, 1ms
***** Finished testing of TestQString *****
```



4 - QTest



```
#include <QString>
#include <QtTest>
#include <QtWidgets>

class TestGuiTest : public QObject
{
    Q_OBJECT

public:
    TestGuiTest();

private Q_SLOTS:
    void testCas1();
};

TestGuiTest::TestGuiTest()
{}

void TestGuiTest::testCas1()
{
    QLineEdit lineEdit;
    QTest::keyClicks(&lineEdit, "hello world");
    QCOMPARE(lineEdit.text(), QString("hello world"));
}

QTEST_MAIN(TestGuiTest)

#include "tst_testguitest.moc"
```

```
***** Start testing of TestGuiTest *****
Config: Using QtTest library 5.10.0, Qt 5.10.0 (i386-little_endian-ilp32 shared (dynamic) debug build; by GCC 5.3.0)
PASS  : TestGuiTest::initTestCase()
PASS  : TestGuiTest::testCas1()
PASS  : TestGuiTest::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped, 0 blacklisted, 407ms
***** Finished testing of TestGuiTest *****
```

<http://doc.qt.io/qt-5/qttutorial3-example.html>



4 - QTest



```
class TestGuiTest : public QObject
{
    Q_OBJECT
public:
    TestGuiTest();
private Q_SLOTS:
    void testCas1();

    void testGui_data();
    void testGui();
```

```
void TestGuiTest::testGui_data()
{
    QTest::addColumn<QTestEventList>("events");
    QTest::addColumn<QString>("expected");

    QTestEventList list1;
    list1.addKeyClick('a');
    QTest::newRow("char") << list1 << "a";

    QTestEventList list2;
    list2.addKeyClick('a');
    list2.addKeyClick(Qt::Key_Backspace);
    QTest::newRow("there and back again") << list2 << "";
}

void TestGuiTest::testGui()
{
    QFETCH(QTestEventList, events);
    QFETCH(QString, expected);
    QLineEdit lineEdit;

    events.simulate(&lineEdit);
    QCOMPARE(lineEdit.text(), expected);
}
```

```
***** Start testing of TestGuiTest *****
Config: Using QtTest library 5.10.0, Qt 5.10.0 (i386-little_endian-ilp32 shared (dynamic) debug build; by GCC 5.3.0)
PASS : TestGuiTest::initTestCase()
PASS : TestGuiTest::testCas1()
PASS : TestGuiTest::testGui(char)
PASS : TestGuiTest::testGui(there and back again)
PASS : TestGuiTest::cleanupTestCase()
Totals: 5 passed, 0 failed, 0 skipped, 0 blacklisted, 328ms
***** Finished testing of TestGuiTest *****
```



4 - QTTest



```
class TestQString: public QObject
{
    Q_OBJECT
private slots:
    void toUpper();

    void toLower_data ();
    void toLower();

    void benchmark_simple();

    void benchmark_multiple_data();
    void benchmark_multiple();
};
```

```
void TestQString::benchmark_multiple_data()
{
    QTest::addColumn<bool>("useLocaleCompare");
    QTest::newRow("locale aware compare") << true;
    QTest::newRow("standard compare") << false;
}

void TestQString::benchmark_multiple()
{
    QFETCH(bool, useLocaleCompare);
    QString str1 = QLatin1String("This is a test string");
    QString str2 = QLatin1String("This is a test string");

    int result;
    if (useLocaleCompare) {
        QBENCHMARK {
            result = str1.localeAwareCompare(str2);
        }
    } else {
        QBENCHMARK {
            result = (str1 == str2);
        }
    }
    Q_UNUSED(result);
}
```

```
PASS : TestQString::benchmark_simple()
RESULT : TestQString::benchmark_simple():
    0.00014 msecs per iteration (total: 76, iterations: 524288)
PASS : TestQString::benchmark_multiple(locale aware compare)
RESULT : TestQString::benchmark_multiple(): "locale aware compare":
    0.00014 msecs per iteration (total: 77, iterations: 524288)
PASS : TestQString::benchmark_multiple(standard compare)
RESULT : TestQString::benchmark_multiple(): "standard compare":
    0.000036 msecs per iteration (total: 77, iterations: 2097152)
```



La classe QSignalSpy permet l'introspection de l'émission du signal.

```
QCheckBox * boîte = . . . ;
Espion QSignalSpy (boîte , SIGNAL (clic (booléen)));

// fait quelque chose qui déclenche le signal
boîte -> animateClick ();

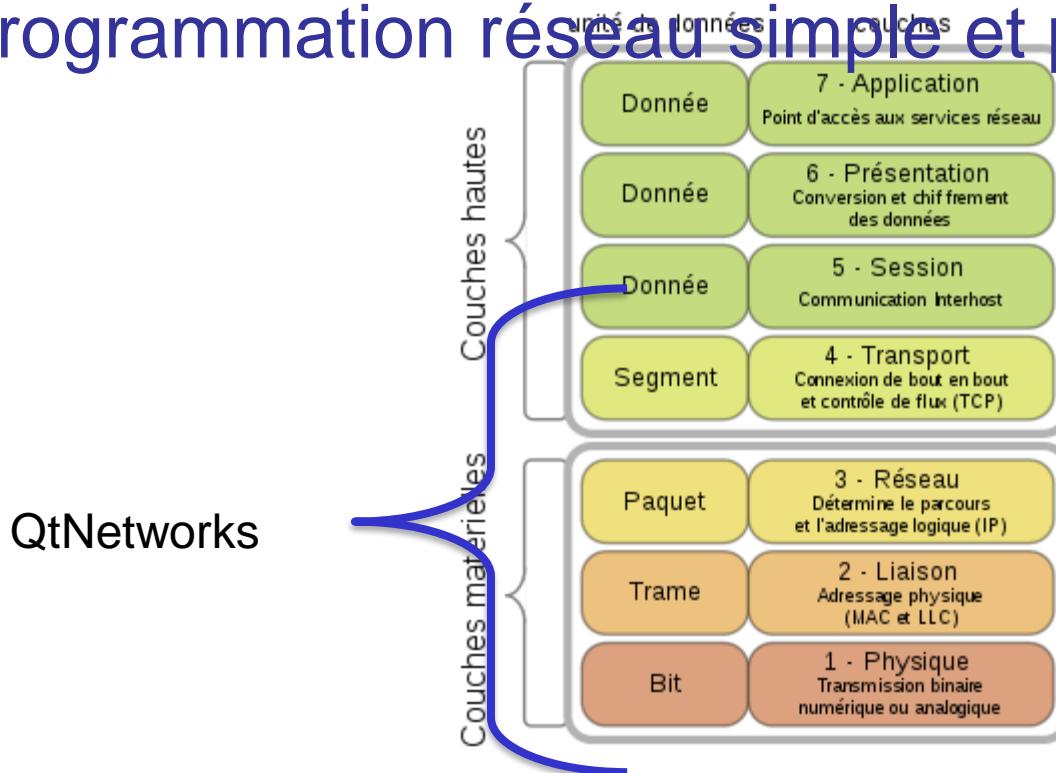
QCOMPARE (espion, compte () , 1 ); // assurez-vous que le signal a été émis exactement une fois
QList < QVariant > arguments = espion . takeFirst (); // prend le premier signal

QVERIFY (arguments à ( 0 ) . ToBool () == true ); // vérifie le premier argument
```

<http://doc.qt.io/qt-5/qsignalspy.html>



Le module QtNetwork fournit des classes pour rendre la programmation réseau simple et portable





- Socket
- Socket -> Client TCP
- Socket -> Serveur TCP



Rappel : Pour que 2 programmes communiquent entre eux via le réseau, il faut :

- Connaître l'adresse IP identifiant l'autre ordinateur (127.0.0.1 local)
- Utiliser un port libre et ouvert (nombre compris entre 1 et 65 536)
- Utiliser le même protocole de transmission des données (FTP, TCP, UDP)



QTcpSocket et QUdpSocket héritent de la classe abstraite QAbstractSocket qui définit la quasi-totalité des fonctions dont elles disposent.

QAbstractSocket hérite elle de QIODevice, tout comme QFile.

La gestion d'une socket s'effectue donc de la même manière que celle d'un fichier et les flux de données se gèrent aussi de la même manière. Tout ce qui est possible avec un fichier l'est avec une socket.

Ajouter dans le .pro
QT += network



Creation client :

➤ QTcpSocket socket = new QTcpSocket(this)

Ouverture d'une connexion

➤ socket->connectToHost("90.83.78.130", 80) ;

➤ Signal :

➤ connected()

➤ stateChanged(AbstractSocket::SocketState)

Envoi d'une requête

➤ socket->write(QString("Données envoyées en texte")) ;



Envoi d'une requête

➤ String

- `socket->write(QString("Données envoyées en texte")) ;`

➤ Paquet

- `QByteArray paquet;`
- `QDataStream out(&paquet, QIODevice::WriteOnly);`
- `QString messageAEnvoyer = msg;`
- `out << (quint16) 0 << messageAEnvoyer;`
- `out.device()->seek(0);`
- `out << (quint16) (paquet.size() - sizeof(quint16));`
- `socket->write(paquet);`



Réception :

- signal readyRead
- QByteArray reponse = socket->readAll()
 - String
 - QString::fromStdString(reponse.toStdString())
 - Paquet
 - QDataStream in(socket);
 - in >> tailleMessage;
 - if (socket->bytesAvailable() < tailleMessage) return;
 - QString messageRecu;
 - in >> messageRecu;



Fermeture et nettoyage

- socket->abort() ; // désactive les connexions précédentes



Creation serveur :

- QTcpServer* serveur = new QTcpServer(this) ;
 - Signal serveur: newConnection
- serveur->listen(QHostAddress::Any, 11111) ; // rend vrai si la connection est réussie

Attente connection

- QTcpSocket *client = serveur->nextPendingConnection();
 - Signaux clients :
 - readyRead
 - disconnected



- Présentation
- Créer un projet via l'éditeur de code
- Créer un projet via l'éditeur graphique
- QTQuick interaction avec le C++

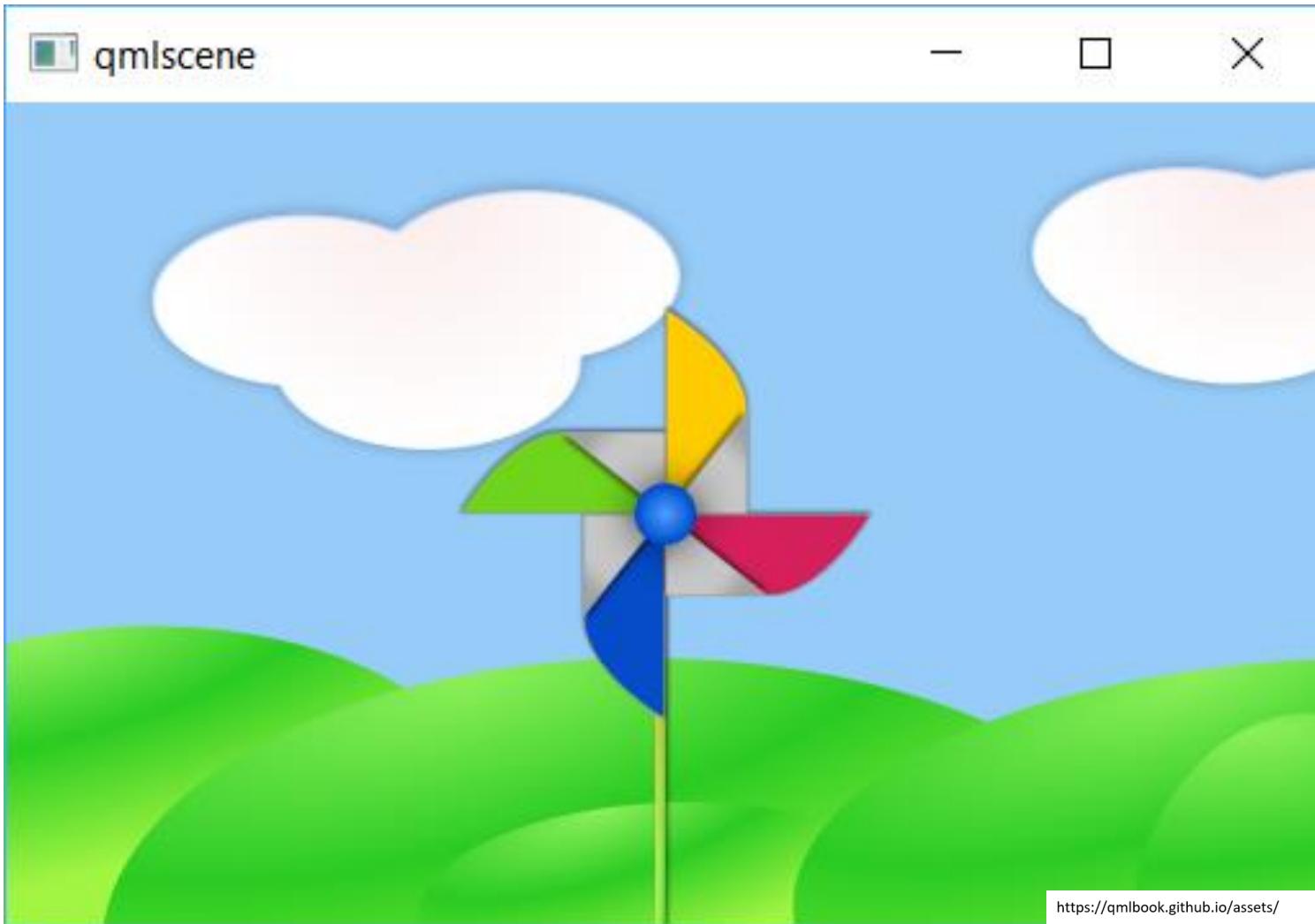


QT Quick est le terme générique pour la technologie d'interface utilisateur utilisée dans Qt5.

QT Quick est une collection de plusieurs technologies :

- QML - langage de balisage pour les interfaces utilisateur
- JavaScript - le langage de script dynamique QT
- C++

6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur de code



<https://qmlbook.github.io/assets/>

© Copyright 2012-2014 Jürgen Bocklage-Ryannel and Johan Thelin. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
Last updated on Mar 21, 2016.

6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur de code



background.png



blur.png



pinwheel.png



pole.png

<https://qmlbook.github.io/assets/>

© Copyright 2012-2014 Jürgen Bocklage-Ryannel and Johan Thelin. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
Last updated on Mar 21, 2016.



Qt Creator va créer plusieurs fichiers pour vous.

- Le fichier **<XXX>.qmlproject** est le fichier de projet où est stocké la configuration de projets pertinents. Ce fichier est géré par Qt Creator afin ne pas les modifier.
- **<XXX>.qml**, est le code de notre application.



6 - les interfaces graphiques : QTQuick

-> Créer un projet via l'éditeur de code



showcase.qmlproject - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets showcase showcase.qmlproject images background.qml

```
/* File generated by Qt Creator, version 2.4.1 */
import QmlProject 1.1

Project {
    mainFile: "background.qml"

    /* Include .qml, .js, and image files from current directory and sub-
     * QmlFiles {
        directory: "."
    }
    JavaScriptFiles {
        directory: "."
    }
    ImageFiles {
        directory: "."
    }
    /* List of plugin directories passed to QML runtime */
    // importPaths: [ "../exampleplugin" ]
}
```

Hello world!

showcase

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'appli... Sortie de compil... Console QML/JS Messages

<https://qmlbook.github.io/assets/>

© Copyright 2012-2014 Jürgen Bocklage-Ryannel and Johan Thelin. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. Last updated on Mar 21, 2016.



6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur de code



background.qml - showcase - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets showcase showcase.qmlproject images background.qml

Qt Accueil Éditer Design Debug Projets Analyse Aide

Hello world!

showcase

```
1 > /* ... */
27
28 // background.qml
29
30 import QtQuick 2.0
31
32 Item {
33     id: window
34     width: background.width
35     height: background.height
36
37     // M1>>
38     Image {
39         id: background
40         source: "images/background.png"
41     }
42     Image {
43         id: pole
44         anchors.horizontalCenter: parent.horizontalCenter
45         anchors.bottom: parent.bottom
46         source: "images/pole.png"
47     }
48     // <<M1
49
50 }
```

qmlscene

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'appli... 4 Sortie de compil... 5 Console QML/JS 6 Messages générés...

<https://qmlbook.github.io/assets/>

© Copyright 2012-2014 Jürgen Bocklage-Ryannel and Johan Thelin. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
Last updated on Mar 21, 2016.

6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur de code



blur.qml - showcase - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

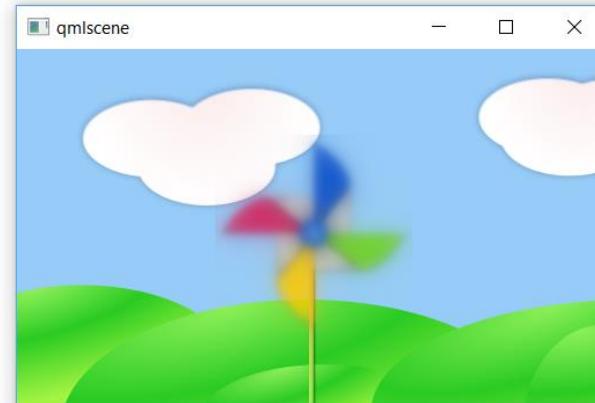
Projets Showcase showcase.qmlproject images blur.qml

```

1 > /* ... */
27
28
29
30 import QtQuick 2.0
31 import QtGraphicalEffects 1.0
32
33 Rectangle {
34     id: window
35     width: background.width
36     height: background.height
37
38     // M1>>
39     Image {
40         id: background
41         source: "images/background.png"
42     }
43     Image {
44         id: pole
45         anchors.horizontalCenter: parent.horizontalCenter
46         anchors.bottom: parent.bottom
47         source: "images/pole.png"
48     }
49
50     // <<M1
51     Image {
52         id: pinwheel
53         anchors.centerIn: parent
54         source: "images/pinwheel.png"
55         layer.effect: FastBlur {
56             id: blur
57             radius: 32
58         }
59         layer.enabled: true
60     }
61 }
62

```

qmlscene



Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages généraux View

<https://qmlbook.github.io/assets/>

© Copyright 2012-2014 Jürgen Bocklage-Ryannell and Johan Thelin. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
Last updated on Mar 21, 2016.

6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur de code



animate.qml - showcase - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

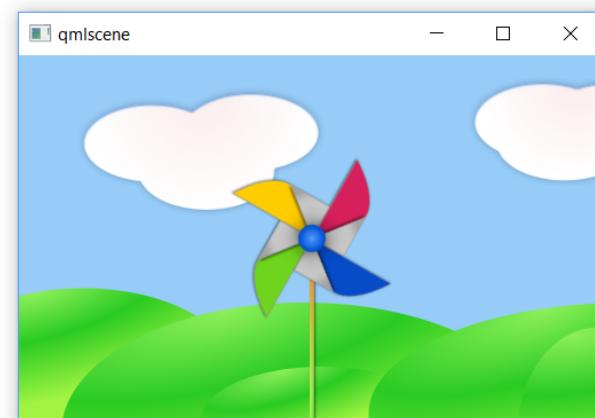
Projets showcase showcase.qmlproject images animate.qml

```

30 import QtQuick 2.0
31
32 Item {
33     id: root
34     width: background.width
35     height: background.height
36
37     property int rotationStep: 45
38
39     Image {
40         id: background
41         source: "images/background.png"
42     }
43
44     Image {
45         id: pole
46         x: (root.width - width)/2+2
47         y: root.height - height
48         source: "images/pole.png"
49     }
50
51     Image {
52         id: pinwheel
53         anchors.centerIn: parent
54         source: "images/pinwheel.png"
55         // visible: false
56         Behavior on rotation {
57             NumberAnimation { duration: 125 }
58         }
59
60         Image {
61             id: blur
62             opacity: 0
63             anchors.centerIn: parent
64             source: "images/blur.png"
65             // visible: false
66             Behavior on rotation {
67                 NumberAnimation { duration: 125 }
68             }
69             Behavior on opacity {
70                 NumberAnimation { duration: 125 }
71             }
72         }
73     }
}

```

qmiscene



Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages générés

<https://qmlbook.github.io/assets/>

© Copyright 2012-2014 Jürgen Bocklage-Ryannel and Johan Thelin. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. Last updated on Mar 21, 2016.



6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur de code



animate.qml - showcase - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

animate.qml Keys.onLeftPressed Line: 79, Col: 25

```
59 }
60
61     Image {
62         id: blur
63         opacity: 0
64         anchors.centerIn: parent
65         source: "images/blur.png"
66         // visible: false
67         Behavior on rotation {
68             NumberAnimation { duration: 125 }
69         }
70         Behavior on opacity {
71             NumberAnimation { duration: 125 }
72         }
73     }
74
75
76
77 // M1>>
78     focus: true
79     Keys.onLeftPressed: {
80         blur.opacity = 1
81         pinwheel.rotation -= root.rotationStep
82         blur.rotation -= root.rotationStep
83     }
84     Keys.onRightPressed: {
85         blur.opacity = 0.5
86         pinwheel.rotation += root.rotationStep
87         blur.rotation += root.rotationStep
88     }
89     Keys.onReleased: {
90         blur.opacity = 0
91     }
92     // <<M1
93 }
```

animate.qmllscene



6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur de code



showcase.qml - showcase - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Pro... Showcase showcase.qml Behavior Line: 55, Col: 10

Accueil Éditeur Design Debug Projets Analyse Aide

Hello world!

```
/* ... */
// showcase.qml

import QtQuick 2.0
import QtGraphicalEffects 1.0

Image {
    id: root
    source: "images/background.png"

    property int blurRadius: 0

    Image {
        id: pole
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.bottom: parent.bottom
        source: "images/pole.png"
    }

    Image {
        id: wheel
        anchors.centerIn: parent
        source: "images/pinwheel.png"
        Behavior on rotation {
            NumberAnimation {
                duration: 250
            }
        }
        layer.effect: FastBlur {
            id: blur
            radius: root.blurRadius
            Behavior on radius {
                NumberAnimation {
                    duration: 250
                }
            }
            layer.enabled: true
        }
    }
}
```

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'ap... Sortie de com... Console QML/JS Messages gén... Version Control

<https://qmlbook.github.io/assets/>



6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur de code



showcase.qml - showcase - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Pro... T D Showcase showcase.qm showcase.qm images showcase.qm

Line: 55, Col: 10

```
39 Image {
40     id: pole
41     anchors.horizontalCenter: parent.horizontalCenter
42     anchors.bottom: parent.bottom
43     source: "images/pole.png"
44 }
45
46
47 Image {
48     id: wheel
49     anchors.centerIn: parent
50     source: "images/pinwheel.png"
51     Behavior on rotation {
52         NumberAnimation {
53             duration: 250
54         }
55     }
56     layer.effect: FastBlur {
57         id: blur
58         radius: root.blurRadius
59         Behavior on radius {
60             NumberAnimation {
61                 duration: 250
62             }
63         }
64     }
65     layer.enabled: true
66 }
67
68 MouseArea {
69     anchors.fill: parent
70     onPressed: {
71         wheel.rotation += 90
72         root.blurRadius = 16
73     }
74     onReleased: {
75         root.blurRadius = 0
76     }
77 }
78 }
79 }
```

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'ap... 4 Sortie de com... 5 Console QML/JS 6 Messages gér Version Control

<https://qmlbook.github.io/assets/>



6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur graphique



Hello World

File

Grid Layout

Row Layout

column Layout

Group Box

Radio Button

Progress Bar

Slider (Horizontal)

toolButton

Check Box

Check Box

Check Box

Text Input

Text Edit

Text Field

TextArea

Yantra Technologies

This screenshot shows a QTQuick-based graphical user interface (GUI) editor window titled "Hello World". The interface includes a menu bar with "File", a toolbar with icons for Grid Layout, Row Layout, column Layout, Group Box, Radio Button, Progress Bar, Slider (Horizontal), and toolButton, and a central workspace displaying various UI components. Components include three check boxes labeled "Check Box", three buttons labeled "Button 1", "Button 2", and "Button 3", a text input field, a text edit field, a text field, and a text area. A red square button is also present. The "Group Box" section contains a radio button, a progress bar, a slider, and a tool button. The "Yantra Technologies" logo is visible in the bottom right corner of the workspace.



6 - les interfaces graphiques : QTQuick

-> Créer un projet via l'éditeur graphique



MainForm.ui.qml - QtQuickCpp - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Bibliothèque Types QML Ressources Importations

État de base

Accueil Éditer Design Debug Projets Analyse Aide Hello world! QtQuickCpp

Qt Quick - Basic

- Border Image Flickable Focus Scope Image
- Item MouseArea Rectangle Text
- Text Edit TextInput

Qt Quick - Controls

- OK Button CheckBox ComboBox Group Box
- Label Progress Bar Radio Button Slider (Horizontal)
- Slider (Vertical) SpinBox TextArea TextField
- OK Tool Button

Qt Quick - Layouts

- Column Row Layout Column Layout

item1

RowLayout

- button1
- button2
- button3

Sortie de l'application

QtQuickCpp x ++\5_3\build-QtQuickCpp-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtQuickCpp.exe...
QML debugging is enabled. Only use this in a safe environment.
qrc:/MainForm.ui.qml:241:1: QML Image: Cannot open: qrc:/LogoYantraPM.jpg

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages généraux Version Control

Propriétés Type RowLayout identifiant Identifiant Géométrie Position X 125 Y 60 Taille W 291 H 28 Visibilité Opacité 1,00 Avancé Origin Échelle 1,00 Rotation 0,00 Z 0 Enabled Smooth Anti-aliasing Accept mouse and keyboard events Smooth sampling active Anti-aliasing active

Yantra Technologies





6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur graphique



MainForm.ui.qml - QtQuickCpp - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets

QtAccueil

QtQuickCpp

- QtQuickCpp.pro
- deployment
- Sources
 - main.cpp
- Ressources
 - qml.qrc
 - /main.qml
 - MainForm.ui.qml
 - /images
 - LogoYantraPM.jpg
- Autres fichiers
 - LogoYantraPM.jpg

168 height: 106
169 orientation: *Qt.Vertical*
170 }
171
172 TextField {
173 id: textField1
174 x: 362
175 y: 18
176 width: 177
177 height: 20
178 placeholderText: *qsTr("Text Field")*
179 }
180
181 TextArea {
182 id: textArea1
183 x: 362
184 y: 49
185 width: 177
186 height: 103
187 text: "TextArea"
188 }
189
190 ToolButton {
191 id: toolButton1
192 x: -8
193 y: 147
194 width: 76
195 height: 36
196 text: "toolButton"
197 checked: true
198 checkable: false
199 }
200
Sortie de l'application

QtQuickCpp

Démarrage de D:\MinGW\msys\1.0\home\David\Nouveau\QtFormation\ExempleC++\5.3\build-QtQuickCpp-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtQuickCpp.exe...
QML debugging is enabled. Only use this in a safe environment.

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'ap... 4 Sortie de co... 5 Console QML... 6 Messages gé... 7 Version Control



6 - les interfaces graphiques : QTQuick

-> Créer un projet via l'éditeur graphique



main.cpp - QtQuickCpp - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets QtQuickCpp QtQuickCpp.pro deployment deployment.pri Sources main.cpp Ressources qml.qrc / main.qml MainForm.ui.qml /images LogoYantraPM.jpg Autres fichiers LogoYantraPM.jpg

```
#include <QApplication>
#include <QQmlApplicationEngine>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));

    return app.exec();
}
```

Hello world!

QtQuickCpp Debug

Type to locate (Ctrl...) 1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control



6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur graphique



main.qml - QtQuickCpp - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Projets

QtQuickCpp

- deployment deployment.pri
- Sources main.cpp
- Ressources qml.qrc
 - / main.qml
 - MainForm.ui.qml
 - /images LogoYantraPM.jpg
- Autres fichiers LogoYantraPM.jpg

main.qml

```
1 import QtQuick 2.4
2 import QtQuick.Controls 1.3
3 import QtQuick.Window 2.2
4 import QtQuick.Dialogs 1.2
5
6 ApplicationWindow {
7     title: qsTr("Hello World")
8     width: 640
9     height: 480
10    visible: true
11
12    menuBar: MenuBar {
13        Menu {
14            title: qsTr("&File")
15            MenuItem {
16                text: qsTr("&Open")
17                onTriggered: messageDialog.show(qsTr("Open action triggered"));
18            }
19            MenuItem {
20                text: qsTr("E&xit")
21                onTriggered: Qt.quit();
22            }
23        }
24    }
25
26    MainForm {
27        anchors.fill: parent
28        button1.onClicked: messageDialog.show(qsTr("Button 1 pressed"))
29        button2.onClicked: messageDialog.show(qsTr("Button 2 pressed"))
30        button3.onClicked: messageDialog.show(qsTr("Button 3 pressed"))
31    }
32
33    MessageDialog {
34        id: messageDialog
35        title: qsTr("May I have your attention, please?")
36
37        function show(caption) {
38            messageDialog.text = caption;
39            messageDialog.open();
40        }
41    }
42}
43
```

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages généraux Version Control

6 - les interfaces graphiques : QTQuick -> Créer un projet via l'éditeur graphique



qml.qrc - QtQuickCpp - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Line: 10, Col: 1

Projets

Accueil

Éditer

Design

Debug

Projets

Analyse

Aide

Hello world!

QtQuickCpp

QtQuickCpp.pro

deployment

Sources

main.cpp

Ressources

qml.qrc

/

main.qml

MainForm.ui.qml

/images

LogoYantraPM.jpg

Autres fichiers

LogoYantraPM.jpg

```
<RCC>
    <qresource prefix="/">
        <file>main.qml</file>
        <file>MainForm.ui.qml</file>
    </qresource>
    <qresource prefix="/images/">
        <file>LogoYantraPM.jpg</file>
    </qresource>
</RCC>
```

Sortie de l'application

Démarrage de D:\MinGW\msys\1.0\home\David\Nouveau\QtFormation\ExempleC++\5.3\build-QtQuickCpp-Desktop_Qt_5_4_0_MinGW_32bit-Debug\debug\QtQuickCpp.exe...
QML debugging is enabled. Only use this in a safe environment.

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'ap... Sortie de co... Console QML... Messages gé... Version Control



6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



Formulaire Qt en QML

File

Entrer vos coordonnées

Nom : Palermo

Prénom : David

Adresse : Pertuis

Valider Effacer Aide

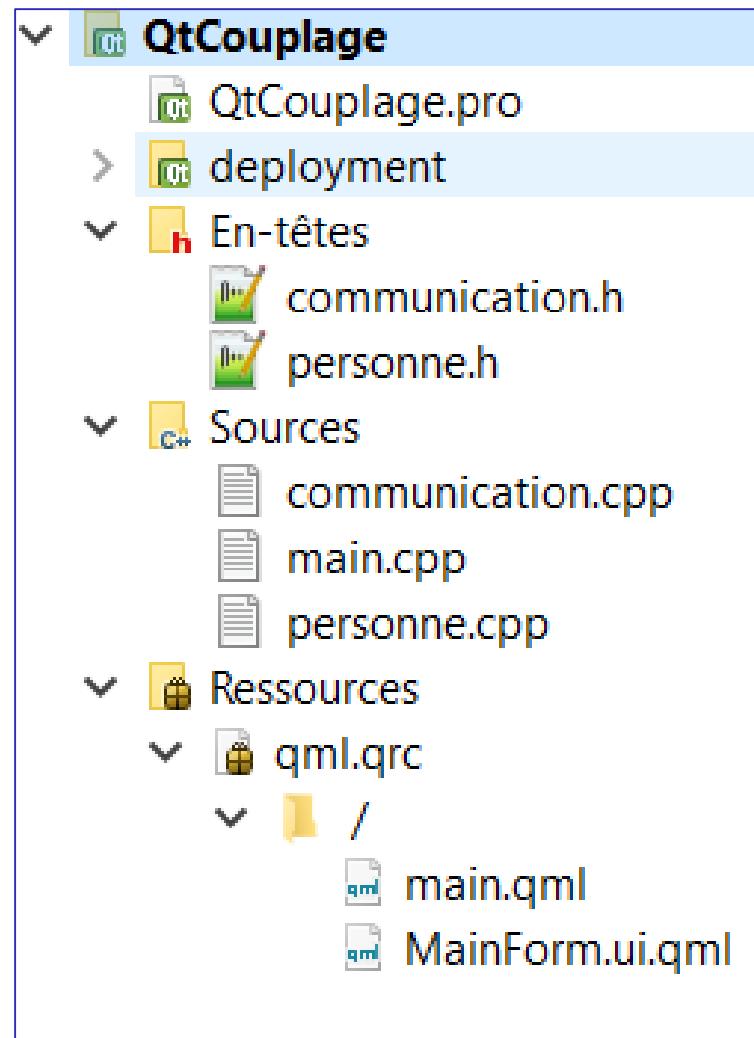
Panneau de validation

? X

Vous allez valider vos coordonnées

Oui Non Abandonner

6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



```
Qt QtCouplage.pro Line: 32,
TEMPLATE = app

QT += qml quick widgets

SOURCES += main.cpp \
    personne.cpp \
    communication.cpp

RESOURCES += qml.qrc

# Additional import path used to resolve QML modules in Qt Creator's code model
QML_IMPORT_PATH =

# Default rules for deployment.
include(deployment.pri)

HEADERS += \
    personne.h \
    communication.h

QMAKE_CXXFLAGS += -Wno-unused-but-set-variable
QMAKE_CXXFLAGS += -Wno-unused-variable
QMAKE_CXXFLAGS += -Wno-unused-parameter
QMAKE_CXXFLAGS += -D_DEBUG
QMAKE_CXXFLAGS += -std=c++11
QMAKE_CXXFLAGS += -std=c++14
QMAKE_CXXFLAGS += -Wno-cpp
#QMAKE_CXXFLAGS += -Weffc++
QMAKE_CXXFLAGS += -Wall
QMAKE_CXXFLAGS += -Wextra
QMAKE_CXXFLAGS += -fbounds-check
QMAKE_CXXFLAGS += -malign-double
QMAKE_CXXFLAGS += -mfpmath=sse
QMAKE_CXXFLAGS += -msse2
QMAKE_CXXFLAGS += -DLinux
```

6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



```
personne.h*          Personne
#ifndef PERSONNE_H
#define PERSONNE_H

#include <string>
#include <vector>

namespace PPersonne {

enum TypeValue { NOM, PRENOM, ADRESSE };

class Personne
{

public:

    Personne(const std::string& nom,
              const std::string& prenom,
              const std::string& adresse);
    ~Personne();

    const std::string& get(TypeValue type) const ;
    void set(TypeValue type, const std::string& valeur) ;

    std::string toString() const ;
private:
    std::string a_nom;
    std::string a_prenom;
    std::string a_adresse;
};

#endif // PERSONNE_H
```



6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



```
personne.cpp*  Personne::Personne(const std::string &, const std::string &, const std::string &)
#include "personne.h"

namespace PPersonne {
Personne::Personne(const std::string& nom,
                  const std::string& prenom,
                  const std::string& adresse):
    a_nom(nom),
    a_prenom(prenom),
    a_adresse(adresse) {}

Personne::~Personne() {}

const std::string& Personne::get(TypeValue type) const {
    switch ( type ) {
        case TypeValue::NOM: return this->a_nom;
        case TypeValue::PRENOM: return this->a_prenom;
        case TypeValue::ADRESSE: return this->a_adresse;
    }
    throw;
}

void Personne::set(TypeValue type, const std::string& valeur) {
    switch ( type ) {
        case TypeValue::NOM: this->a_nom = valeur;
        case TypeValue::PRENOM: this->a_prenom= valeur;
        case TypeValue::ADRESSE: this->a_adresse= valeur;
    }
}

std::string Personne::toString() const {
    std::string res ;
    res+="";
    res = "<nom>" + this->a_nom + "</nom> \n";
    res = res + "<prenom>" + this->a_prenom + "</prenom> \n";
    res = res + "<adresse>" + this->a_adresse + "</adresse> \n";
    res+="</personne>";
    return res;
}
```

6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



La macro
Q_INVOKABLE
permet d'appeler la
méthode dans QML

```
#ifndef COMMUNICATION_H
#define COMMUNICATION_H

#include <QObject>
#include "personne.h"
#include <memory>
#include <iostream>

namespace PControleur {

class Communication : public QObject
{
    Q_OBJECT
public:
    explicit Communication(QObject *parent = 0);
    ~Communication();

    Q_PROPERTY(QString name READ getName WRITE setName NOTIFY nameChanged)
    Q_INVOKABLE QString toStringPersonne() const;
    static void DeclareQML();

signals:
    void change_Personne();
    void create_Personne();
    void nameChanged(QString );
public slots:
    void add_Personne(const QString& nom, const QString& prenom, const QString& adresse);
    void mod_Personne(const QString& nom, const QString& prenom, const QString& adresse);

    const QString &getName() const { return a_nameCurrent; }
    void setName(const QString &name) {
        a_nameCurrent=name;
        emit nameChanged("OK : nom changé");
    }

    void voir(QString name) const { std::cout << name.toStdString() << std::endl; }
private:
    std::shared_ptr< PPersonne::Personne> current;
    QString a_nameCurrent;
};

#endif // COMMUNICATION_H
```



6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



MainForm.ui.qml - QtCouplage - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Bibliothèque Types QML Ressources Importations

Qt Accueil Éditer Design Debug Projets Analyse Aide

>Hello world!

QtQuick - Basic

- Border Image
- Flickable
- Focus Scope
- Image
- Item
- MouseArea
- Rectangle
- Text
- Text Edit

Qt Quick - Controls

- OK
- Button
- Check Box
- Combo Box
- Group Box
- Label
- Progress Bar
- RadioButton
- Slider (Horizontal)
- Slider (Vertical)
- SpinBox
- Text Area
- Text Field

Navigateur

- item1
 - formulaire
 - textnom
 - label1
 - textadresse
 - txtprenom
 - button1
 - button2
 - button3
 - label2

Type to locate (Ctrl...)

1 Problèmes 2 Search Results 3 Sortie de l'application 4 Sortie de compilation 5 Console QML/JS 6 Messages généraux 7 Version Control

Aucun ou plusieurs éléments sélectionnés.





6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



MainForm.ui.qml - QtCouplage - Qt Creator

Fichier Édition Compiler Déboguer Analyser Outils Fenêtre Aide

Bibliothèque Types QML Ressources Importations

Qt Quick - Basic

- Border Image
- Flickable
- Focus Scope
- Image
- Item
- MouseArea
- Rectangle
- Text
- Text Edit
- Text Input

Qt Quick - Controls

- OK
- Check Box
- Combo Box
- Button
- Group Box
- Label
- Progress Bar

Navigateur

- item1
- formulaire
 - textnom
 - label1
 - textadresse
 - textprenom
 - button1
 - button2
 - button3
 - label2
 - label3
 - textArea2
 - button4

Vos coordonnées

Nom : Nom

Prénom : Prénom

Adresse :

Modifier

Aide

Button

Propriétés

Aucun ou plusieurs éléments sélectionnés.

Type to locate (Ctrl...)

Problèmes Search Results Sortie de l'application Sortie de compilation Console QML/JS Messages généraux Version Control

6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



```
main.qml* ApplicationWindow
1 import QtQuick 2.4
2 import QtQuick.Controls 1.3
3 import QtQuick.Window 2.2
4 import QtQuick.Dialogs 1.2
5 import MonControleur 1.0
6
7 ApplicationWindow {
8     id : monapplication
9     title: qsTr("Formulaire Qt en QML")
10    width: 640
11    height: 480
12    visible: true
13    property string action : ""
14
15 Communication {
16     id: id_Communication
17 }
18
19 menuBar: MenuBar { ... }
32
33 MainForm { id: maforme... }
81
82 MessageDialog { id: id_MessageDialogEffacer... }
102
103 MessageDialog { id: id_MessageDialogValidation... }
118
119 MessageDialog { id: id_MessageDialog... }
128
129 }
```

6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



```
menuBar: MenuBar {
    Menu {
        title: qsTr("&File")
        MenuItem {
            text: qsTr("&Open")
            onTriggered: messageDialog.show(qsTr("Open action triggered"));
        }
        MenuItem {
            text: qsTr("E&xit")
            onTriggered: Qt.quit();
        }
    }
}
```

6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



```

MainForm {
    id:maforme
    property int cpt : 0
    anchors.fill: parent
    button1.onClicked:{
        var msg;
        if ( button1.text != "Valider")
        {}
        msg = "Vous allez passer sur le panneaux de modification de vos coordonnées"
        id_MessageDialogValidation.show(qsTr(msg));
        action="";
        id_Communication.add_Personne(textnom.text, textprenom.text, textadresse.text);
    }
    else if ( textnom.text.replace(' ','') != "" &&
              textprenom.text.replace(' ','') !="" &&
              textadresse.text.replace(' ','') != "")
    {
        msg = "Vous allez valider vos coordonnées"
        id_MessageDialogValidation.show(qsTr(msg));
        action="Afficher";
        id_Communication.setName(textnom.text + cpt);
        id_Communication.add_Personne(textnom.text, textprenom.text, textadresse.text);
        textArea2.text = id_Communication.toStringPersonne();
        cpt++;
    }
    } else id_MessageDialog.show(qsTr("Veuillez remplir tous les champs."))

    button2.onClicked: id_MessageDialogEffacer.show(qsTr("voulez vous effacer tous les champs ?"), "");
    button3.onClicked: id_MessageDialog.show(qsTr("Aide : Il faut remplir tous les champs et valider"));
    button4.onClicked: {
        if ( monapplication.height != 480)
        {monapplication.height = 480;}
        else
        {monapplication.height = 700;}
    }
}

```

6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



```

MessageDialog {
    id: id_MessageDialogEffacer
    title: qsTr("Panneau d'effacement")
    icon: StandardIcon.Question
    standardButtons: StandardButton.Yes | StandardButton.No
    Component.onCompleted: visible = false
    onYes:{
        maforme.textnom.text=""
        maforme.textprenom.text=""
        maforme.textadresse.text=""
        console.log("MessageDialogEffacer")
    }
    onNo: console.log("Modifications non validées")
    function show(caption) {id_MessageDialogEffacer.text = caption;id_MessageDialogEffacer.open();}
}

MessageDialog {
    id: id_MessageDialogValidation
    title: qsTr("Panneau de validation")
    icon: StandardIcon.Question
    standardButtons: StandardButton.Yes | StandardButton.No | StandardButton.Abort
    Component.onCompleted: visible = false
    onYes:{ maforme.state=action;monapplication.height = 480;}
    onNo: console.log("Modifications non validées")
    onRejected: console.log("Reste sur la page")
    function show(caption) {id_MessageDialogValidation.text = caption;id_MessageDialogValidation.open();}
}

MessageDialog {
    id: id_MessageDialog
    title: qsTr("Panneau erreur")
    function show(caption) {id_MessageDialog.text = caption;id_MessageDialog.open();}
}

```

6 - les interfaces graphiques : QTQuick -> QTQuick interaction avec le C++



```
#include <QApplication>
#include <QQmlApplicationEngine>
#include <QQmlComponent>
#include <QQmlProperty>
#include <QDebug>
#include "communication.h"
#include <QSignalMapper>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    PControleur::Communication::DeclareQML();

    /*
     * QQmlApplicationEngine engine;
     * engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
     */

    QQmlApplicationEngine engine;
    QQmlComponent component(&engine, QUrl(QStringLiteral("qrc:/main.qml")));
    QObject *object = component.create();
    QObject *objet1 = object->findChild<QObject*>("id_Communication");
    QObject *childObject = object->findChild<QObject*>("maforme");

    //qDebug() << childObject->property("text");
    qDebug() << "Property value:" << QQmlProperty::read(object, "title");
    qDebug() << "Property value:" << QQmlProperty::read(object, "action");
    qDebug() << "Property value:" << QQmlProperty::read(childObject, "cpt").toInt();
    return app.exec();
}
```



- Problématique
- Comparaison QML & QWidget

7 - les interfaces graphiques : QTQuick & QWidget & C++ -> Problématique



Formlaire Qt en QML

File

Entrer vos coordonnées

Nom

Prénom

Adresse

Valider

Modifier

Aide

Valider **Effacer**

Formlaire Qt en QML

File

Vos coordonnées

Nom

Prénom

Adresse

Modifier

Button

Aide

7 - les interfaces graphiques : QTQuick & QWidget & C++ -> Problématique



Formulaire Qt en QML

File

Vos coordonnées

Nom	<input type="text" value="Palermo"/>
Prénom	<input type="text" value="David"/>
Adresse	<input type="text" value="Nice"/>

Button

Button

Button

Modifier **Aide**

Formulaire Qt en QML

File

Vos coordonnées

Nom	<input type="text" value="Palermo"/>
Prénom	<input type="text" value="David"/>
Adresse	<input type="text" value="Nice"/>

Modifier **Aide**

Button

```
<nom>Palermo</nom>
<prenom>David</prenom>
<adresse>Nice</adresse>
</personne>
```

7 - les interfaces graphiques : QTQuick & QWidget & C++ -> Problématique



Formulaire Qt en QML

File

Entrer vos coordonnées

Nom: Palermo
Prénom: David
Adresse: 123

Valider Effacer Aide

Formulaire Qt en QML

File

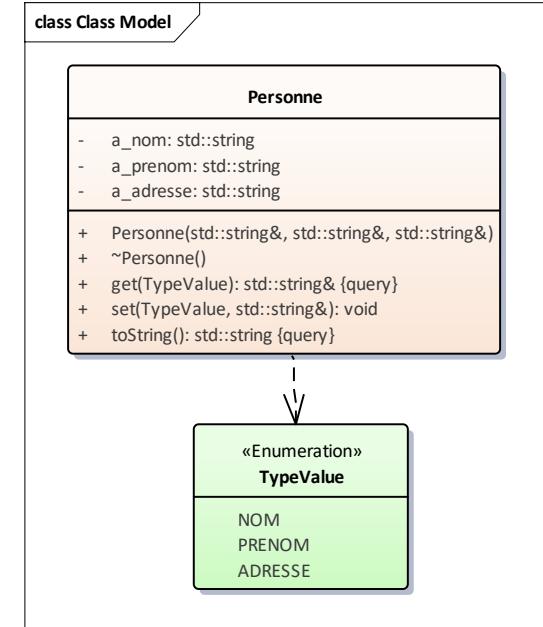
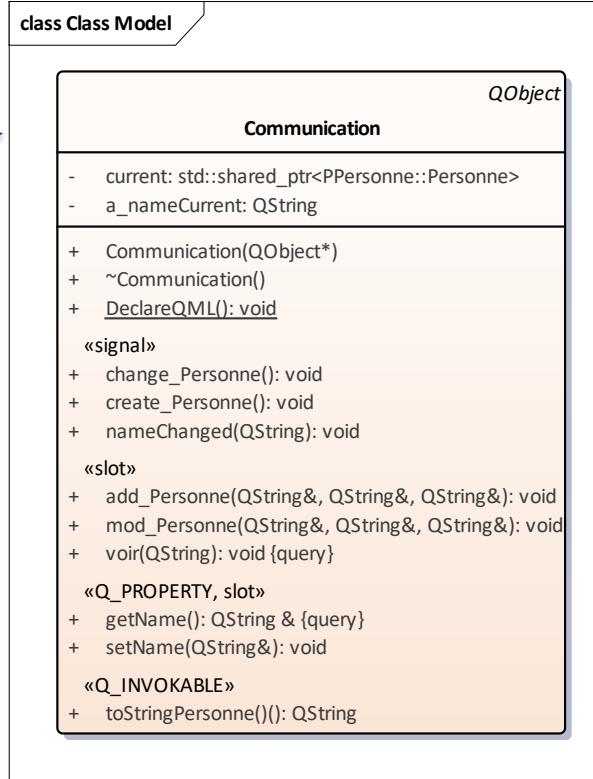
Vos coordonnées

Nom: Palermo
Prénom: David
Adresse: 123

Modifier Aide

Button

```
<nom>Palermo</nom>
<prénom>David</prénom>
<adresse>123</adresse>
</personne>
```



Qt Quick

- main.qml
- MainForm.ui.qml

Qt Widget

- mainwindow.h
- mainwindow.cpp
- mainwindow.ui



QWidget :

- QWidget : Les classes QGraphics ne sont plus vraiment développées
- + QWidget : Les classes QGraphics sont stables et performantes, et sont d'ailleurs utilisées dans un très grand nombre d'applications.

QML & QTQuick :

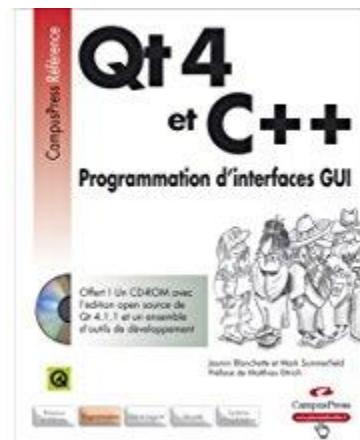
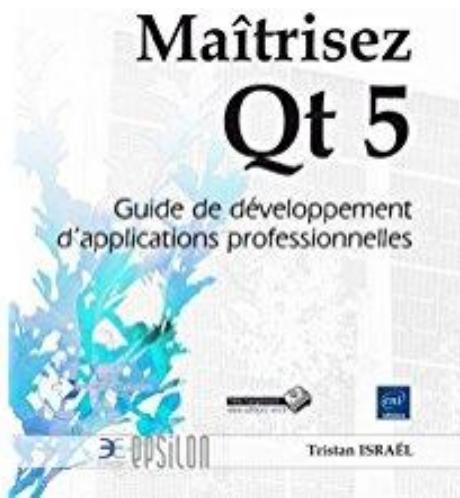
- + QML & QTQuick : Qt indique clairement que Qt Quick est l'avenir pour la création d'interfaces graphiques : les nouvelles fonctionnalités de Qt sont d'ailleurs généralement utilisables aussi bien en C++ qu'en QML (comme Qt 3D).
- + QML & QTQuick : apprentissage Qt Quick
- + QML & QTQuick : mélange C++ / Javascript lourd

<https://qt.developpez.com/actu/113480/La-fin-de-QGraphicsView-faut-il-lui-privilégier-Qt-Quick-dans-de-nouveaux-developpements-Ce-cadriel-ne-devrait-plus-evoluer-dans-le-futur/>



- Plug-in
- QSql
- Multimedia
- WebkitWidgets
- Qt mobile
- Qt temps réel et embarqué

9 - Bibliographie



9 - Bibliographie



- <https://doc.qt.io> : Qt documentation
- <https://wiki.qt.io/Main>
- <http://wiki.qtfr.org/>
- http://www.bogotobogo.com/Qt/Qt5_Creating_QtQuick2_QML_Application_Animation_A.php