



微算機系統實習

MICROPROCESSOR SYSTEMS LAB.

SPRING, 2021

Instructor : Yen-Lin Chen(陳彥霖), Ph.D.

Professor

Dept. Computer Science and Information Engineering
National Taipei University of Technology





LECTURE 3 – QT視窗程式開發



OUTLINE

- Qt簡介
- 使用Qt Creator開發Qt GUI程式
- Qt Creator 跨平台編譯



QT簡介

QT是什麼？

- Qt目前是Digia公司的產品，是一個跨平台的C++應用程式開發框架。廣泛用於開發GUI程式，這種情況下又被稱為部件工具箱。也可用於開發非GUI程式，比如控制台工具和伺服器。
- Qt使用標準的C++和特殊的代碼生成擴充功能（稱為元物件編譯器（Meta Object Compiler））以及一些巨集。
- 通過語言綁定，其他的程式語言也可以使用Qt。
 - 例如結合Python使用(PyQT)。
- 在Linux平台上C/C++ GUI函式庫常見有GTK+與Qt，GTK+以C語言開發為主，Qt則為C++為主。

QT是什麼？

- Qt是自由且開放原始碼的軟體，在GNU較寬鬆公共許可證（LGPL）條款下發布。所有版本都支援廣泛的編譯器，包括GCC的C++編譯器和Visual Studio。
- Qt也有提供商業授權版本，可以避免LGPL更改函式庫需要公開原始碼的問題
- 課堂上屬於學術用途，所以我們使用的版本為LGPL版本

QT跨平台

- Qt可以跨多種平台
 - Linux
 - MacOS
 - Windows
 - 嵌入式Linux系統 (Embedded Linux)
 - Windows CE
 - Android
 - iOS
 - IBM OS/2
 - Symbian/ Maemo / MeeGo 等Nokia早期系統
(Qt曾經為Nokia的產品，後來賣出)

QT GUI

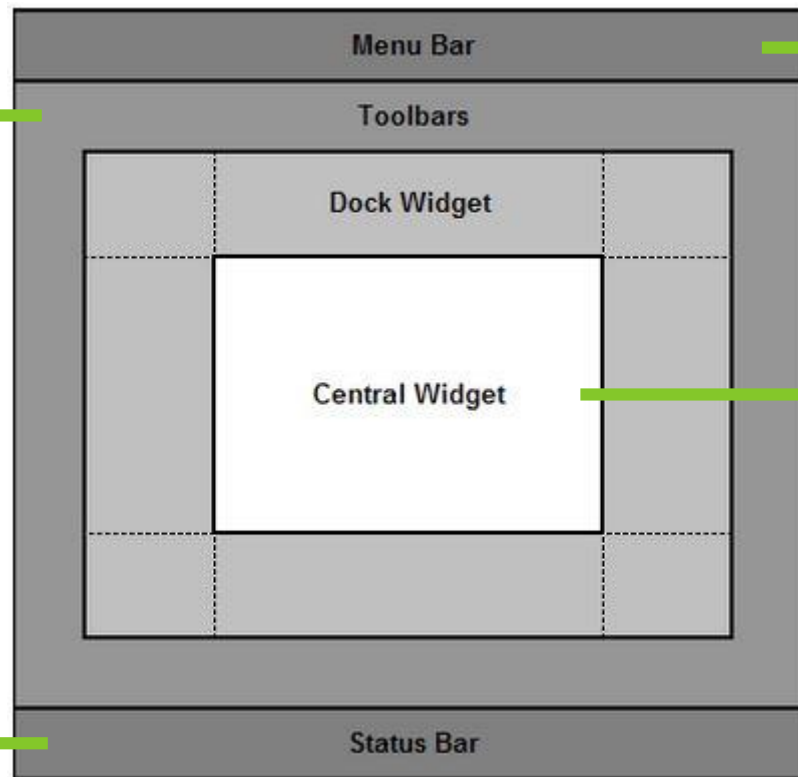
- Qt的圖形使用者介面的基礎是QWidget。
- Qt中所有類型的GUI組件如按鈕、標籤、工具列等都衍生自QWidget，而QWidget本身則為QObject的子類別。
- Widget負責接收滑鼠，鍵盤和來自視窗系統的其他事件，並描繪了自身顯示在螢幕上。
- 每一個GUI組件都是一個widget，widget還可以作為容器，在其內包含其他Widget。

QT GUI 設計介紹與說明

- Qt QMainWindow主視窗類

工具列：
中間矩形區域
最外層為工具
列，Qt 支持
多個工具列，
而且可以並排
顯示或分別放
置四周

狀態欄：
位於底部，根
據使用者操作
可以再狀態欄
顯示信息

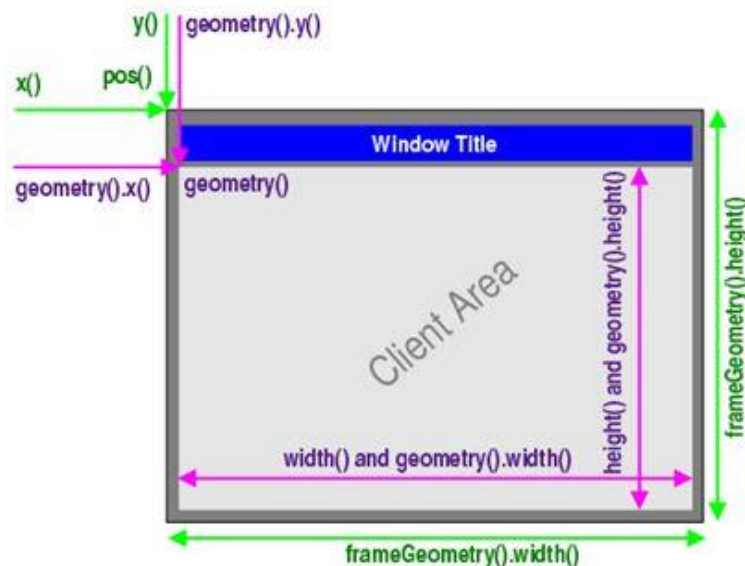


選單欄：
顯示工作選單

工作區：
通常用來顯示
結果並讓使用
者在此操作

QT GUI 設計介紹與說明

- 關於Qt主視窗的視窗大小和位置操作函式，可以分為包含框架和不含框架
 - 含框架：x()、y()、pos()、frameGeometry()、move()。
 - 不含框架：geometry()、width()、height()、rect()、size()。

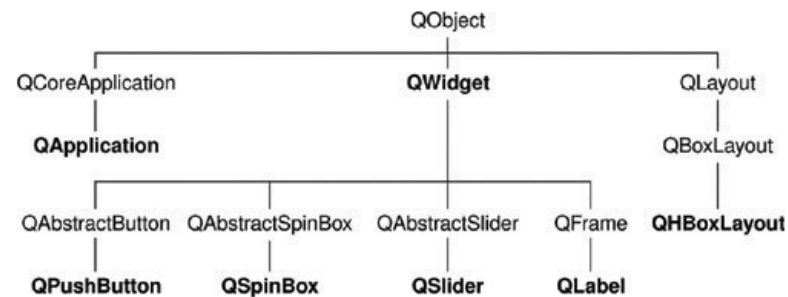


QT常用排版方法

- Qt內建的Layout管理類型有：QHBoxLayout、QVBoxLayout、QGridLayout和QFormLayout。
 - QHBoxLayout：配置widget成橫向一行
 - QVBoxLayout：配置widget成垂直一行
 - QGridLayout：配置widget在平面網格
 - QFormLayout：配置widget用於2欄標籤- field

QT GUI 物件繼承

- QWidget繼承QObject, 是建立使用者界面的主要類別。
- 建置時沒有指定父元件的元件稱為視窗, 需指定父元件的元件稱為子元件 (如: 標籤、按鈕等等)
- QMainWindow就是一種視窗, 也稱為頂級部件 (top-level widget)



QT GUI 設計介紹與說明

- 通常在main()函數內只生成頂層視窗
- 在父元件的建構式內，new出我們需要的子元件
- 只要最頂層的視窗（top-level widget）能夠正確釋放記憶體，且父子元件都連接成鏈，每一個組件記憶體就能被正確釋放

QT物件訊號處理（訊號與槽）

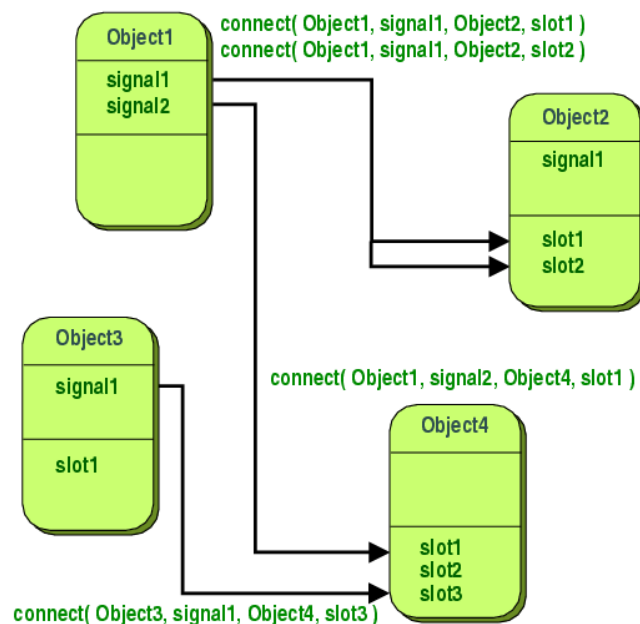
- Qt利用訊號與槽（**signals/slots**）機制取代傳統的callback來進行物件之間的溝通。當操作事件發生的時候，物件會發送出一個訊號（signal）
- 槽（slot）則是一個函式接受特定信號並且執行槽本身設定的動作。
- 訊號與槽之間，則透過QObject的靜態函數connect來連結。
- 訊號在任何執行點上皆可發射，甚至可以在槽裡再發射另一個訊號，訊號與槽的連結不限定為一對一的連結，一個訊號可以連結到多個槽或多個訊號連結到同一個槽，甚至訊號也可連接到訊號。

QT物件訊號處理 (SIGNAL AND SLOT)

- 以往的callback缺乏類型安全，在呼叫處理函式時，無法確定是傳遞正確型態的參數。但訊號和其接受的槽之間傳遞的資料型態必須要相符合，否則編譯器會提出警告。訊號和槽可接受任何數量、任何型態的參數，所以訊號與槽機制是完全類型安全。
- 訊號與槽機制也確保了低耦合性，發送訊號的類別並不知道是哪個槽會接受，也就是說一個訊號可以呼叫所有可用的槽。此機制會確保當在"連接"訊號和槽時，槽會接受訊號的參數並且正確執行。

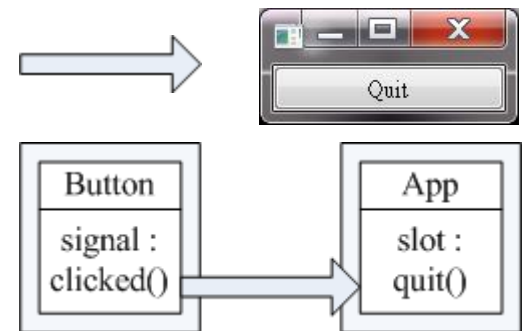
QT物件訊號處理 (SIGNAL AND SLOT)

- Qt 在不同Object之間，傳遞訊息是透過Signal與Slot的機制，當一個Object發出Signal後，將會啟動相對應的Slot進行運作。



QT物件訊號處理 (SIGNAL AND SLOT)

```
#include <QApplication>
#include <QPushButton>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    QPushButton *button = new QPushButton("Quit");
    QObject::connect(button, SIGNAL(clicked(bool)), &a, SLOT(quit()));
    button->show();
    return a.exec();
}
```



● 首先建立兩個物件之間的溝通連結

Button中signal : clicked() // 當按鈕被按下時觸發事件

App中slot : quit() //當被呼叫時離開程式

● 執行後，當Quit button被按下時，將會傳送訊號signal : clicked()，並啟動slot : quit()，當完成後便會關閉視窗

QT更多參考資料

- Qt介紹
 - <https://zh.wikipedia.org/wiki/Qt>
- Qt物件種類與使用
 - <http://doc.qt.io/qt-5/qtwidgets-index.html>
- LGPL版本開發者網頁
 - <https://www1.qt.io/developers/>



使用QT CREATOR 開發QT GUI程式

QT CREATOR介紹

- Qt Creator 是一款跨平台的整合開發環境，特別針對Qt開發者，是Qt SDK組成的一部分，可執行於Windows, Linux/X11及Mac OS X等桌面作業系統，允許開發者為多桌面環境及行動裝置平台建立應用程式。
- Qt Creator 包括一個可視化偵錯工具和整合的 GUI 版面和外型設計師，這個編輯器的功能包括語法高亮度顯示和自動完成。
- Qt Creator 在 Linux 上，使用 GCC 的 C++ 編譯器。在 Windows，預設安裝它可以使用 MinGW 或 MSVC。
- Qt Creator 整合了跨平台自動化建構系統qmake 與 CMake，可以匯入不使用 qmake 或 CMake 的專案，並指定 Qt Creator忽略你的建構系統。

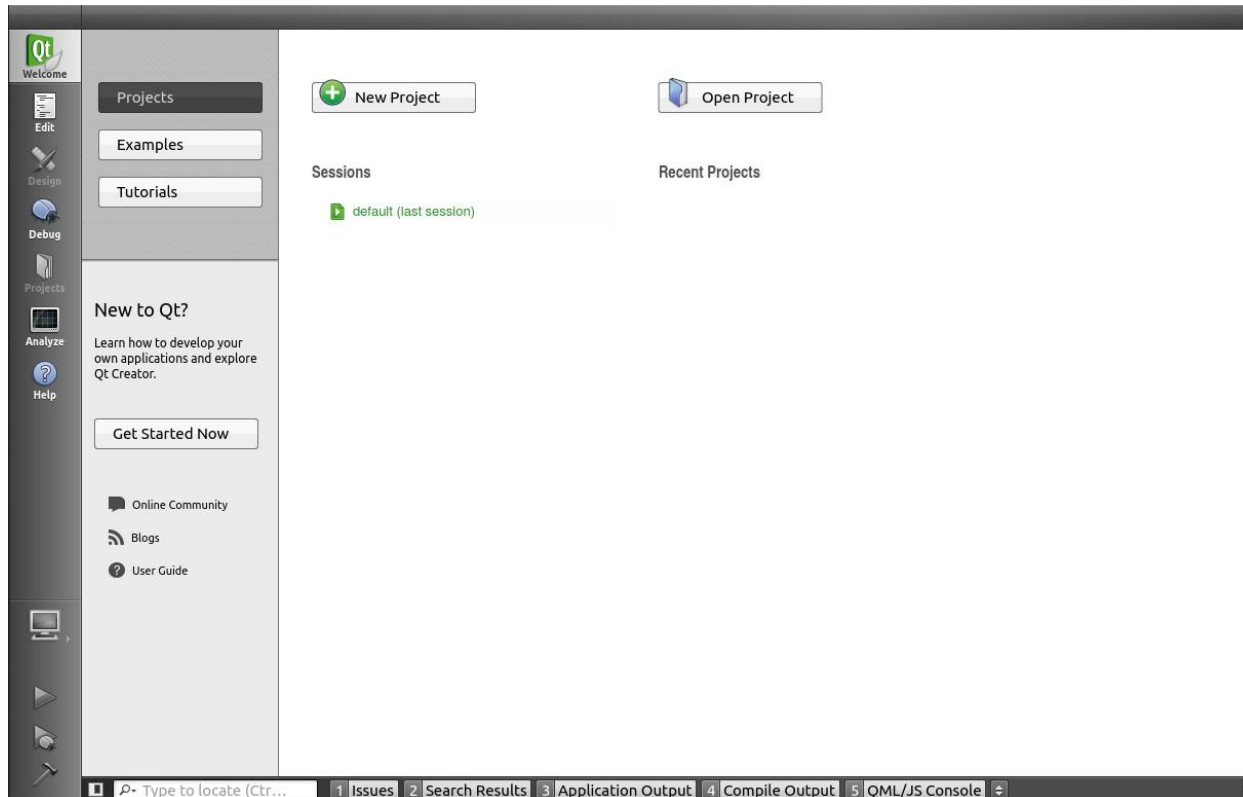
QT CREATOR

- 請先於Ubuntu系統上安裝Qt Creator，執行下列指令進行安裝
 - `sudo apt-get install qtcreator`
 - 也可於軟體中心搜尋並安裝
- 安裝好後可於應用程式放置區找到Qt Creator



QT CREATOR

- 打開Qt Creator



QT CREATOR

- 按下 New Project 建立新專案

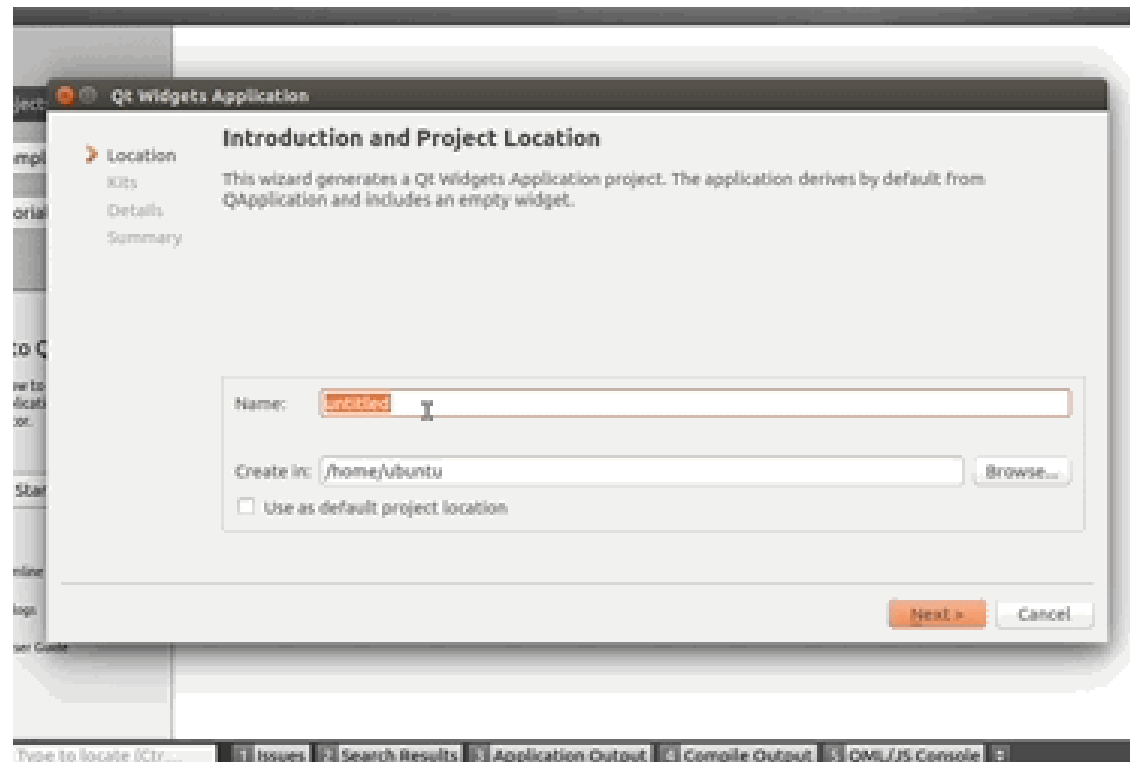


- 選擇 Qt Widgets Application



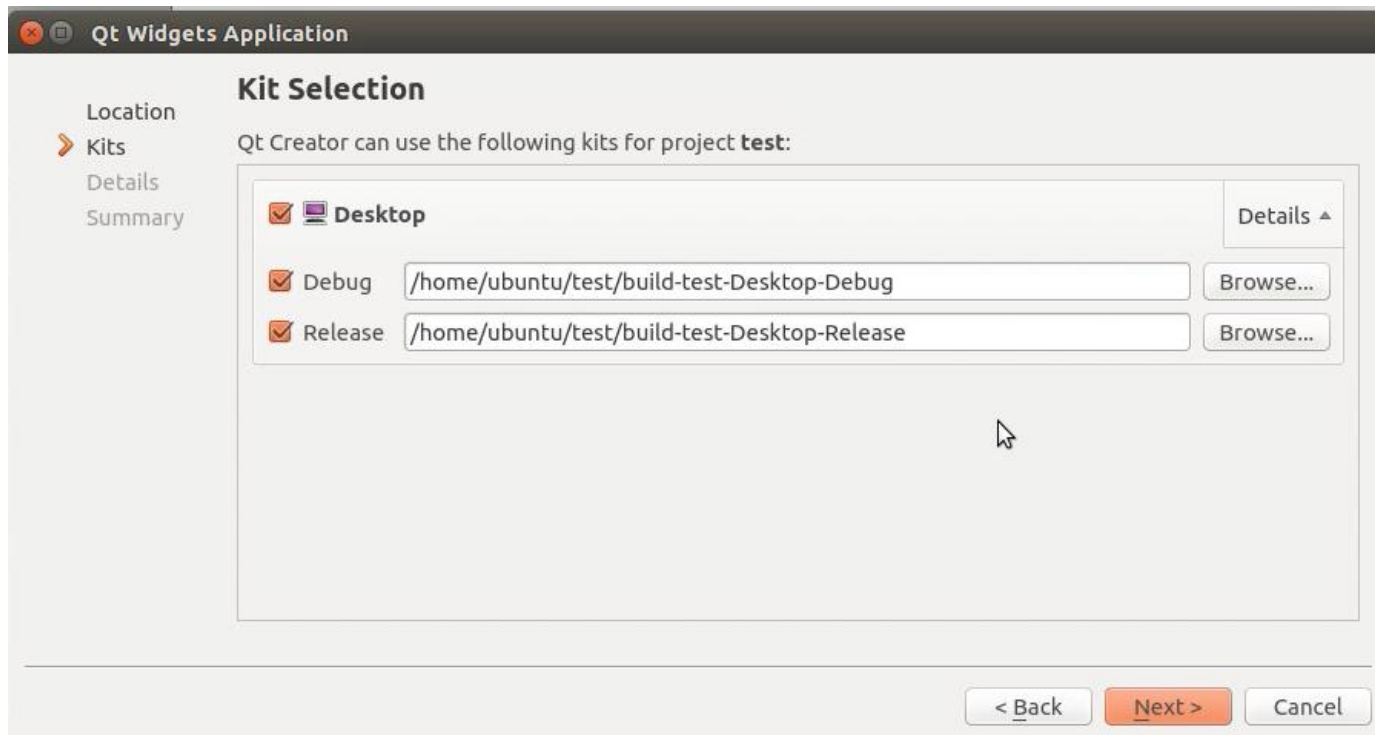
QT CREATOR

- 選擇專案名稱與要存放的資料夾路徑



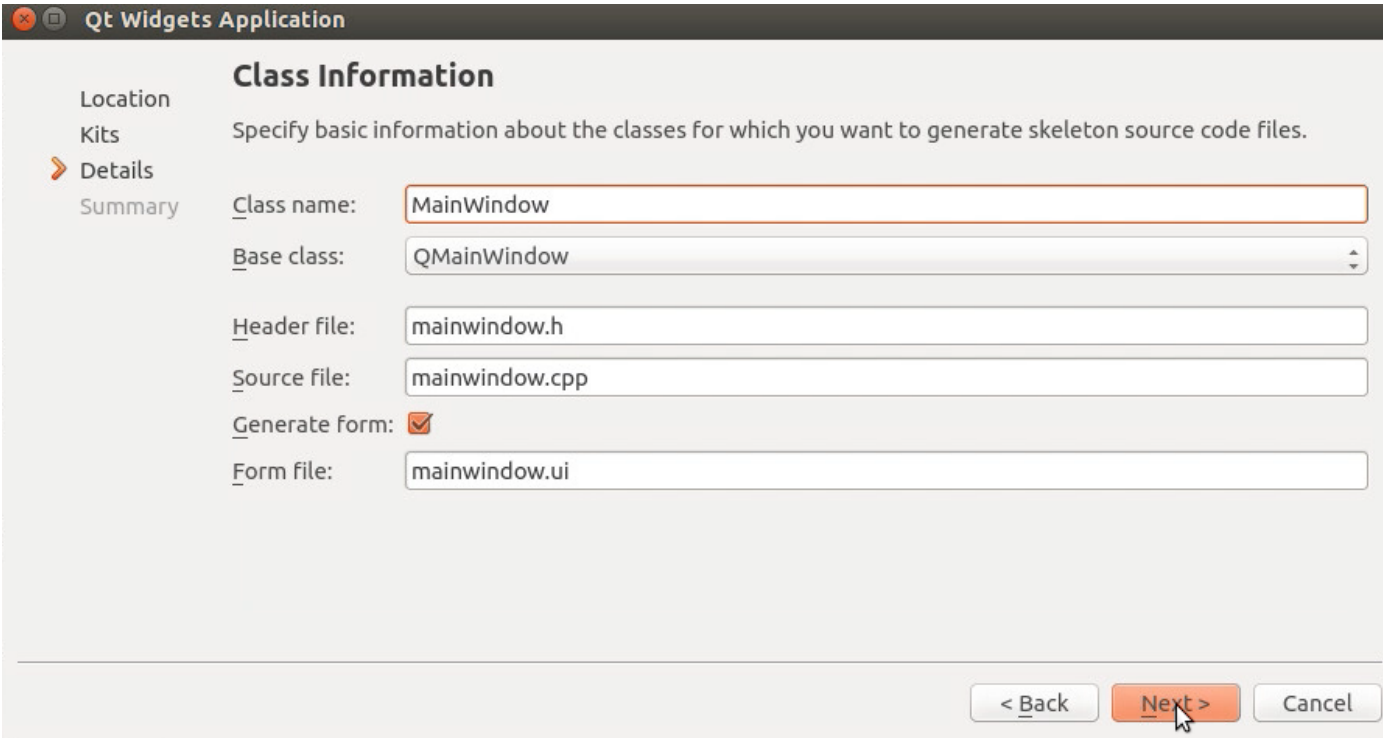
QT CREATOR

- 對於編譯的細節設定 (此次範例要編譯出Ubuntu虛擬機上執行的程式，所以不用更改預設設定)



QT CREATOR

- 設定主視窗class (不用更改預設設定)



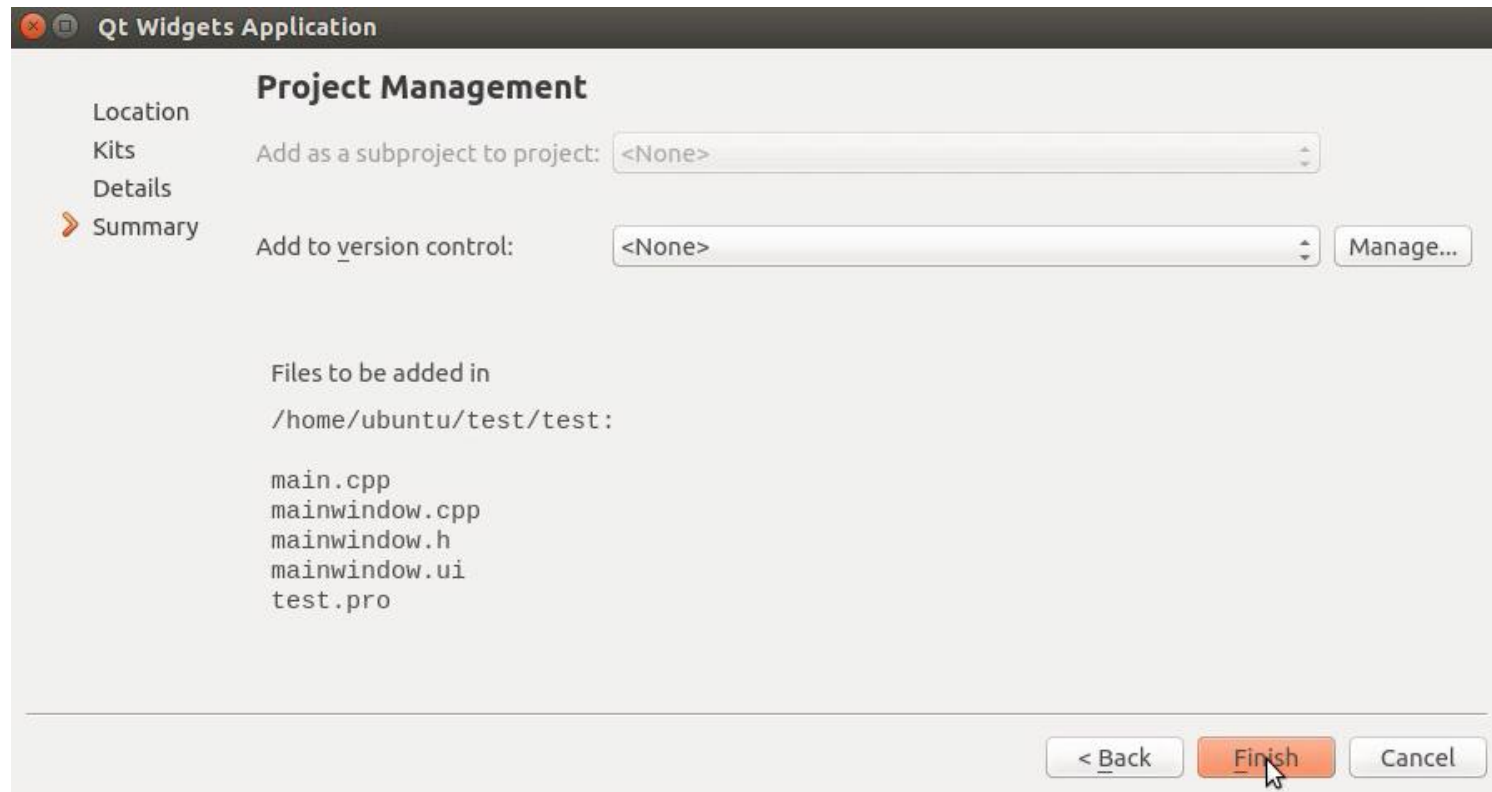
The image shows the 'Qt Widgets Application' dialog box in Qt Creator, specifically the 'Class Information' tab. The dialog is used to specify basic information for generating skeleton source code files. The fields are as follows:

Field	Value
Class name:	MainWindow
Base class:	QMainWindow
Header file:	mainwindow.h
Source file:	mainwindow.cpp
Generate form:	<input checked="" type="checkbox"/>
Form file:	mainwindow.ui

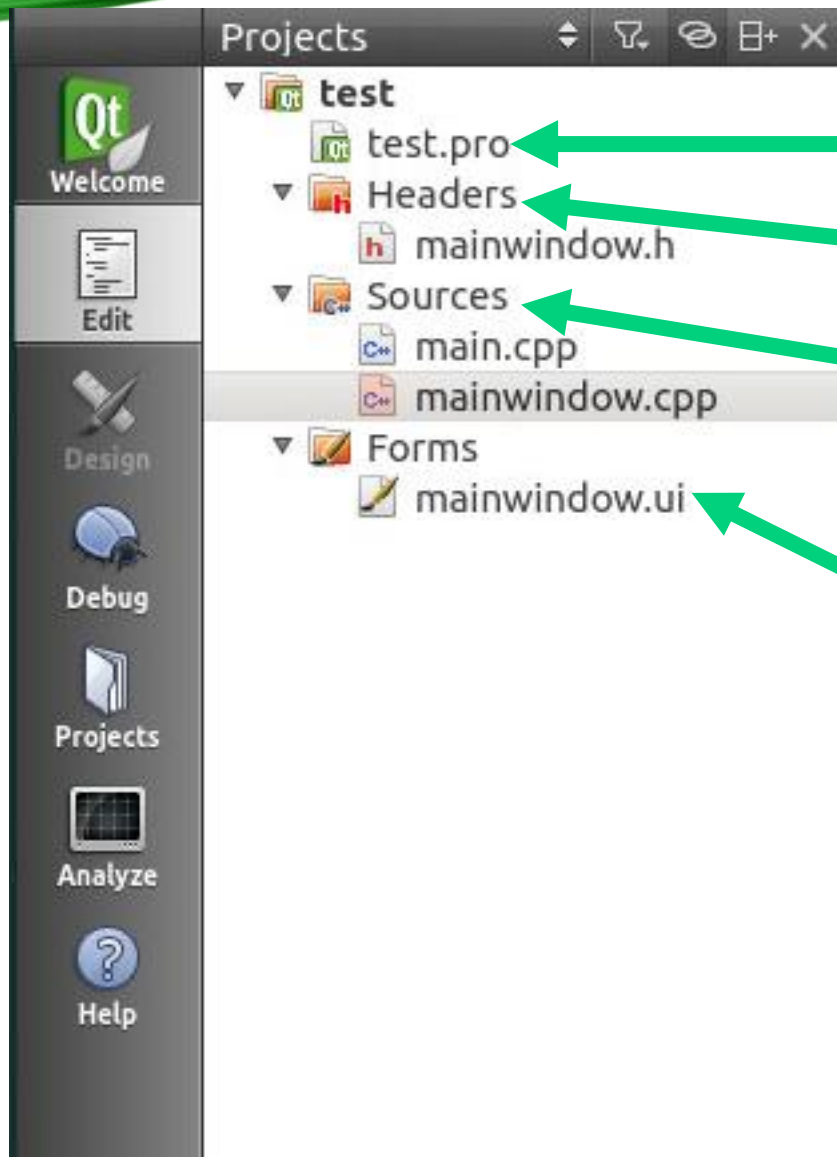
At the bottom right, there are three buttons: '< Back', 'Next >', and 'Cancel'. A mouse cursor is pointing at the 'Next >' button.

QT CREATOR

- 完成設定 Finish



QT CREATOR



專案配置檔

標頭檔放置區

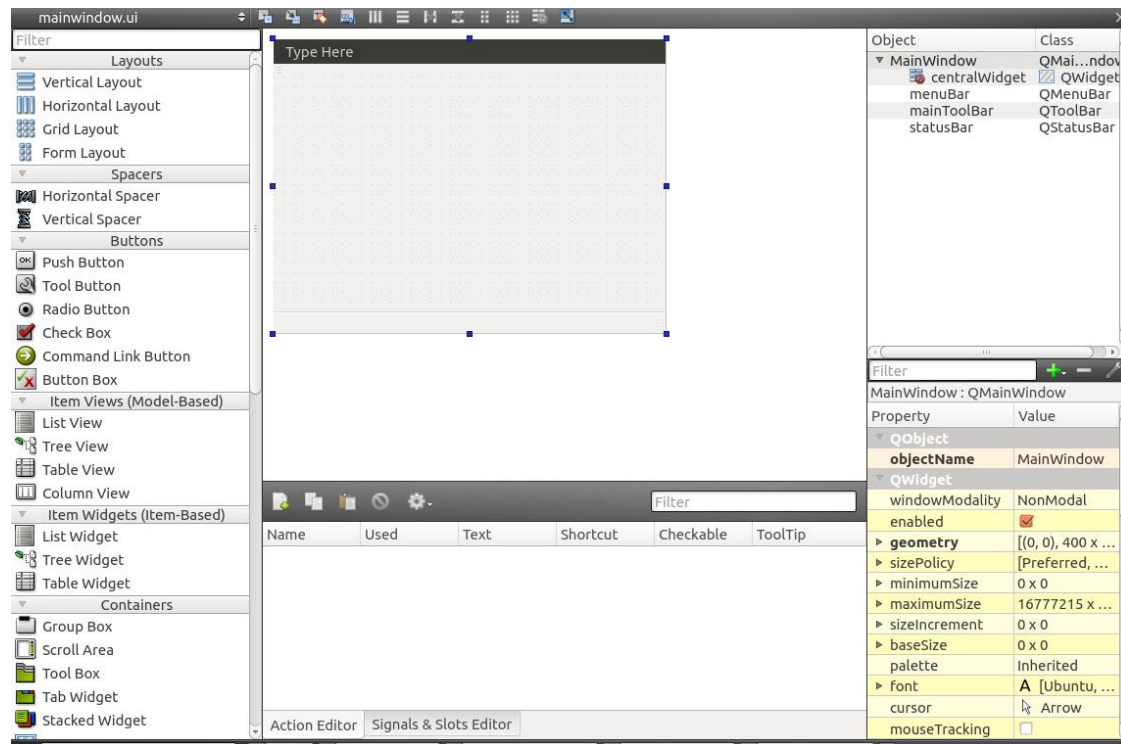
程式檔放置區

(mainwindow.cpp為
mainwindow.ui的程式碼實作)

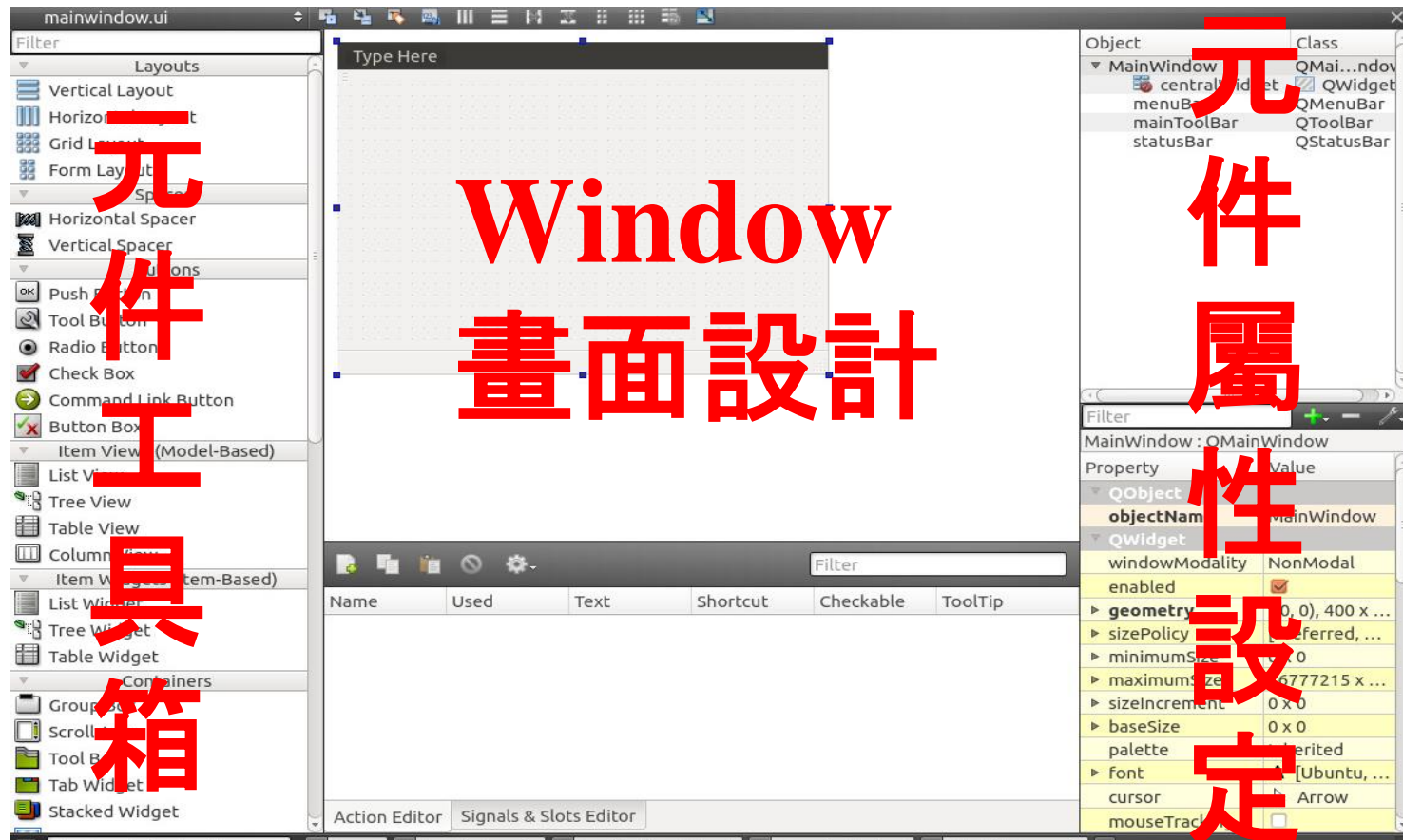
UI設計配置文件（點兩下可以開
啟UI配置）

QT CREATOR

- 點兩下 `mainwindow.ui`



QT CREATOR



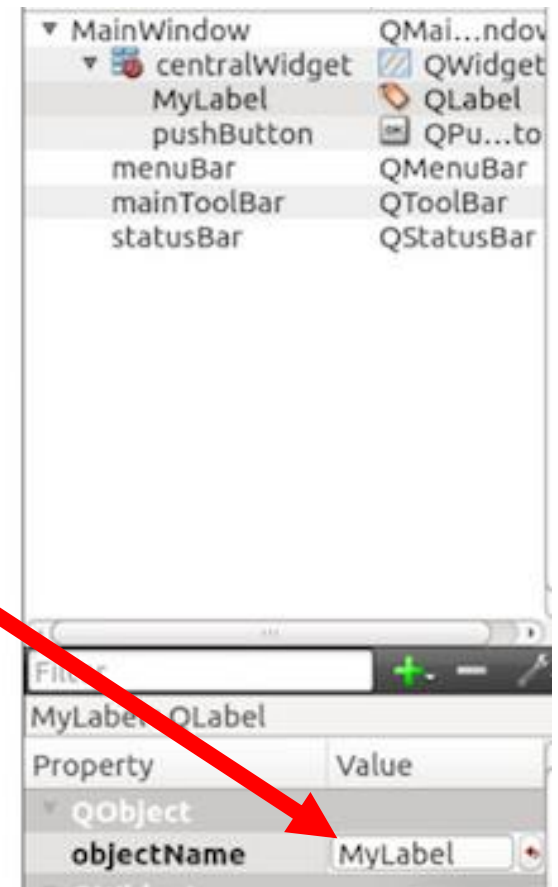
QT CREATOR

- 拉兩個物件分別是Push Button與Label於Window畫面上



QT CREATOR

- 將Label命名成MyLabel



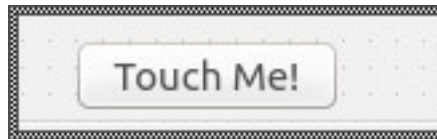
QT CREATOR

- 將Push Button命名成MyButton1



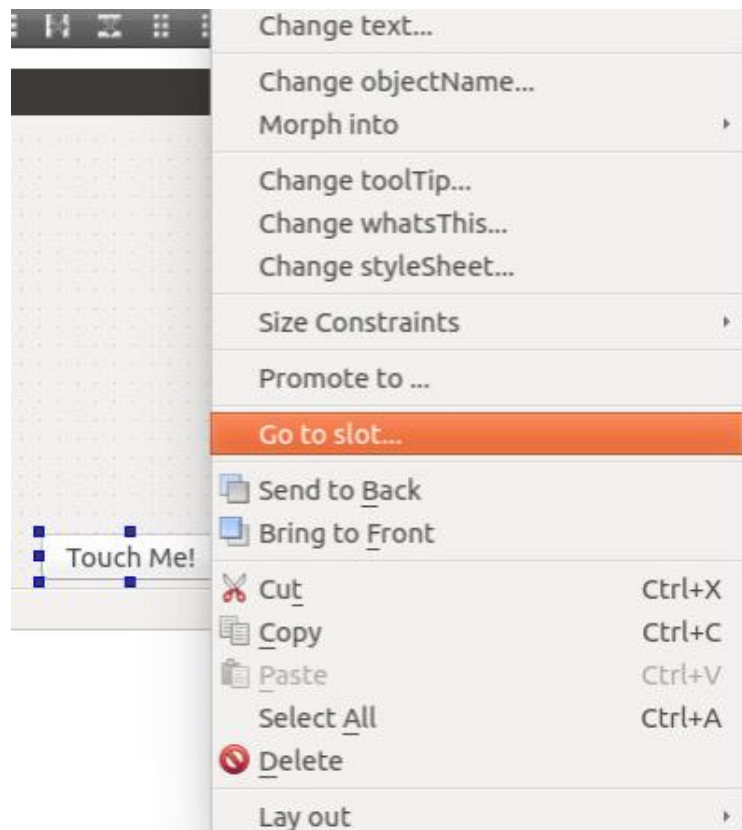
QT CREATOR

- 在Push Button上點兩下標題改成”Touch Me!”



QT CREATOR

- 點擊Push Button右鍵選擇 Go to slot... 來建立對應的訊號槽



QT CREATOR

- 選擇`clicked()`建立點擊此按鈕後會觸發的訊號槽



QT CREATOR

- IDE會幫我們再mainwindow.h和mainwindow.cpp中建立對應的程式碼如下圖

```
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 slots:
19     void on_MyButton1_clicked();
20
21 private:
22     Ui::MainWindow *ui;
23 };
24
25 #endif // MAINWINDOW_H
```

mainwindow.h



```
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  }
10
11 MainWindow::~MainWindow()
12 {
13     delete ui;
14 }
15
16 void MainWindow::on_MyButton1_clicked()
17 {
18
19 }
```

mainwindow.cpp



QT CREATOR

- 我們可以於mainwindow.cpp中剛剛建立的訊號觸發的function中撰寫按下按鈕後要執行的程式動作

mainwindow.cpp

```
15  
16 ▼ void MainWindow::on_MyButton1_clicked()  
17 {  
18     在這裡寫程式  
19 }
```

QT CREATOR

- 範例是撰寫一個按下按鈕時會更改Label上的文字，按一下顯示“Hello Qt Creator!!” 按第二下顯示 “This is an apple”，第三下回到第一下時候的狀態（循環） * 程式碼於下頁說明

```
void MainWindow::on_MyButton1_clicked()
{
    QString getLebelString = ui->MyLabel->text();

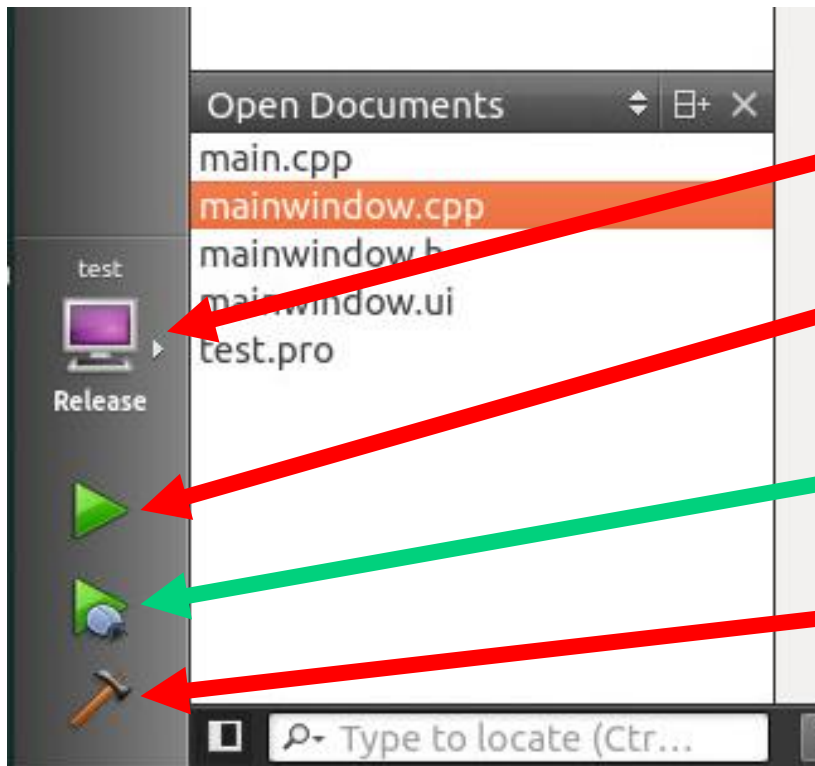
    if (getLebelString == "Hello Qt Creator!!")
    {
        ui->MyLabel->setText("This is an apple");
    }
    else
    {
        ui->MyLabel->setText("Hello Qt Creator!!");
    }
}
```


QT CREATOR

```
1 void MainWindow::on_MyButton1_clicked()
2 {
3     //QString為Qt的字串型態，可以轉換為C++ String
4     //此段程式碼式向ui上我們定義Label(名稱為MyLabel)取出上面的文字並存在變數中
5     QString getLebelString = ui->MyLabel->text();
6
7     //如果文字是Hello Qt Creator!!
8     if (getLebelString == "Hello Qt Creator!!")
9     {
10         //將Label的文字改成This is an apple
11         ui->MyLabel->setText("This is an apple");
12     }
13     else //若不是
14     {
15         //將Label的文字改成Hello Qt Creator!!
16         ui->MyLabel->setText("Hello Qt Creator!!");
17     }
18 }
```


QT CREATOR

- 編譯（於左下角），點選**編譯 & 執行程式**按鈕



改為Release

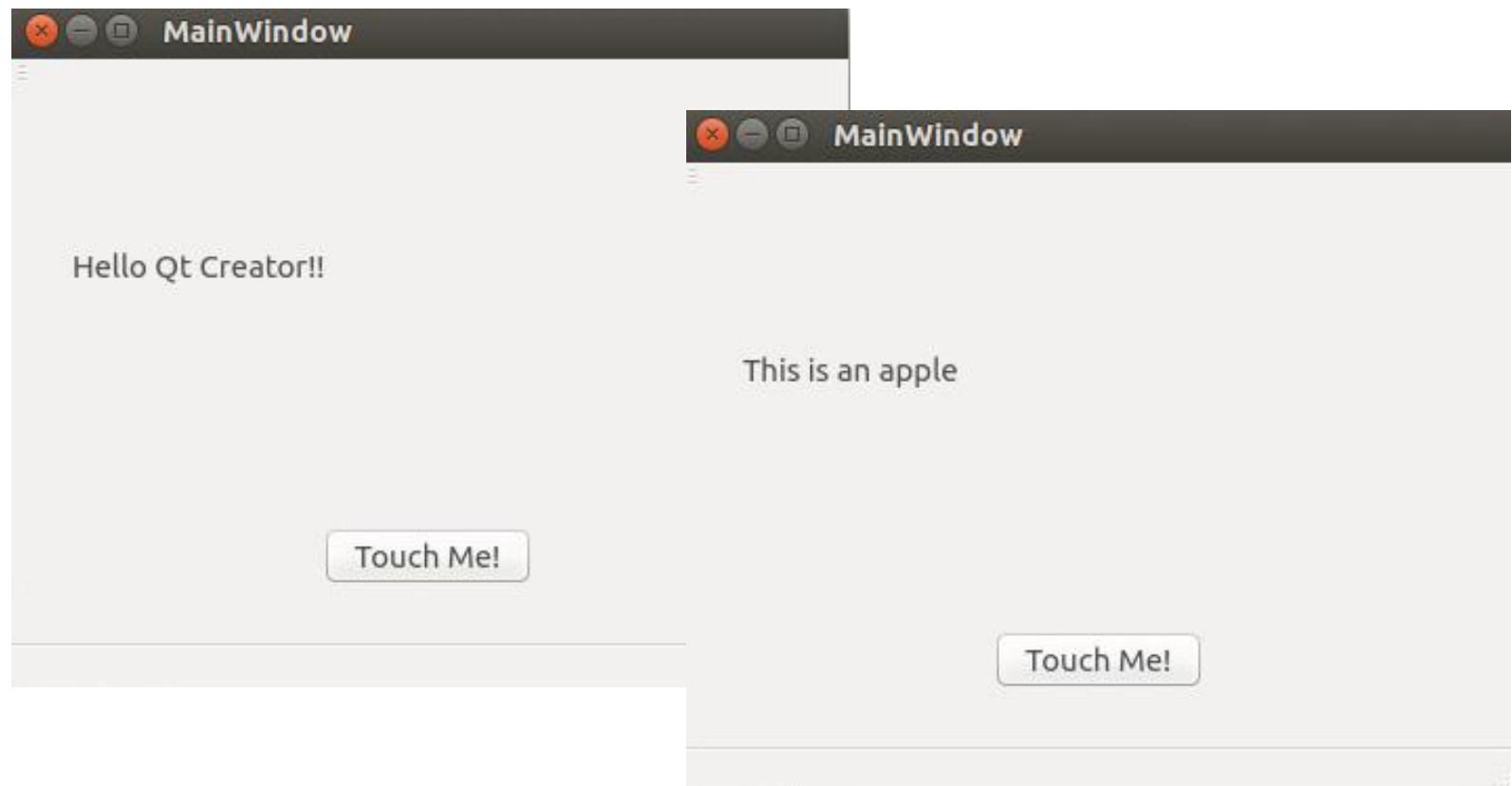
編譯 & 執行程式（會自動編譯
& 執行）

Debug執行用（一般不使用此按
鈕）

重新編譯（不自動執行）

QT CREATOR

- 執行結果





QT CREATOR 跨平台開發

QT CREATOR 跨平台開發

- 步驟1:將Qt Creator專案整個目錄傳送到TX2上。
- 此範例的目錄為x86_untitled

```
nvidia@ubuntu:~$ ls
build-x86_untitled-Desktop-Debug  Music          Templates
Desktop                          my_cron_tasks.ini Videos
Documents                        Pictures
Downloads                        Public
examples.desktop                 selectcomp.txt
nvidia@ubuntu:~$ scp -r x86_untitled nvidia@192.168.137.21:~
x86_untitled.pro                  100% 365    0.4KB/s   00:00
x86_untitled.pro.user            100% 18KB   18.1KB/s  00:00
mainwindow.ui                   100% 630    0.6KB/s   00:00
main.cpp                        100% 172    0.2KB/s   00:00
mainwindow.cpp                  100% 219    0.2KB/s   00:00
mainwindow.h                    100% 291    0.3KB/s   00:00
```

專案目錄

TX2的IP

QT CREATOR 跨平台開發

- 步驟2:遠端連線到TX2上, 使用ls指令, 確認檔案是否傳輸成功, 切換目錄到專案目錄下(此範例為x86_untyped)。

```
nvidia@ubuntu:~$ ssh -Y nvidia@192.168.137.21
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.38-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

34 packages can be updated.
16 updates are security updates.

*** System restart required ***
Last login: Thu Mar 26 08:56:08 2020 from 192.168.137.1
nvidia@tegra-ubuntu:~$ ls
build-lab_3-JetsonTX2-Debug  Pictures
build-untyped-ARM_tx2-Release  Public
Desktop                     qt5_7
Documents                   tegra_multimedia_api
Downloads                   tegrastats
examples.desktop            Templates
jetson_clocks.sh            tx2_cross_test
lab_3                       Videos
Music                       VisionWorks-SFM-0.90-Samples
NVIDIA_CUDA-9.0_Samples     weston.ini
opencv-2.4.9                x86_untyped
nvidia@tegra-ubuntu:~$ cd x86_untyped/
nvidia@tegra-ubuntu:~/x86_untyped$
```

QT CREATOR 跨平台編譯

- 步驟3:輸入qmake 指令, qmake會根據專案檔 (.pro) 裡面的資訊自動生成適合平台的 Makefile。

```
nvidia@tegra-ubuntu:~/x86Untitled$ ls
main.cpp      mainwindow.h  x86Untitled.pro
mainwindow.cpp mainwindow.ui  x86Untitled.pro.user
nvidia@tegra-ubuntu:~/x86Untitled$ qmake
nvidia@tegra-ubuntu:~/x86Untitled$ ls
main.cpp      mainwindow.ui  x86Untitled.pro.user
mainwindow.cpp Makefile
mainwindow.h  x86Untitled.pro
```

QT CREATOR 跨平台編譯

- 步驟4:輸入make 指令後會生成執行檔

```
nvidia@tegra-ubuntu:~/x86Untitled$ make
```

- 步驟5:執行執行檔，即會產生視窗。

```
-lQt5Widgets -lQt5Gui -lQt5Core -lGL -lpthread  
nvidia@tegra-ubuntu:~/x86Untitled$ ./x86Untitled  
Could not initialize OpenGL for RasterGLSurface, reverting to RasterSurface.  
█
```

