

GUI

Modern cpp Programming lecture 7



GUI

- Graphical User Interface
 - A system of interactive visual components for computer software
 - *A form of user interface that allows users to interact with devices through graphical icons and audio indicator (Wikipedia)*
 - Users can manipulate the program by various interaction devices
 - Keyboard, Mouse (Desktop computer)
 - Touchpad, Trackpad (Laptop)
 - Multi-touch screen, virtual keyboard (Smartphone, Tablet PC)

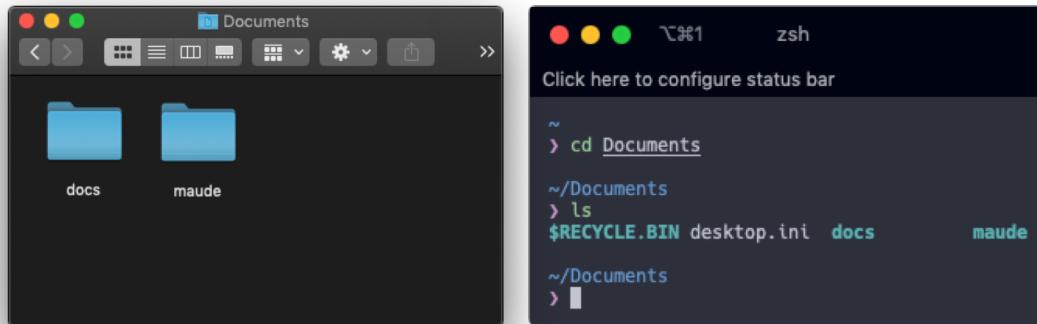
GUI vs CLI

GUI vs CLI

GUI	CLI
Graphical User interface	Command Line interface
Even a beginner can easily handle	User should have good knowledge of commands (Better for the experts)
Requires more memory as it contains a lot of graphical components	Does not require much memory
Slower	Faster
More flexible	Not much flexible

GUI vs CLI

- For example...
 - Most OSs provides both Graphical & Command-line ways to manage file system
 - macOS & Linux
 - Finder vs. Terminal



- Windows
 - Window File explorer vs. Command Prompt

GUI

- Most of the programs you use...
- are GUI program!!



Office program



Game



Messenger



zoom

Online conferencing program

- Therefore you must learn how to implement GUI!!

Implementing GUI program

- How?
- Almost all of modern programming languages provides GUI Library
 - Java
 - Swing, Applet...
 - cpp
 - **Qt**, WinAPI, wxWidgets
 - C#
 - .Net (dot net)

Implementing GUI Program

- In nowadays...
 - Using the combination of html / css / javascript is recommended
 - Why?
 - Cross-platform
 - Lots of packages / libraries
 - Easy to implement functionalities
 - Easy to design your program
- However, in this class...
 - We're learn the GUI framework for cpp!!

Qt

- Open source GUI framework for cpp
- Developed in 1990 by *Eirik Chambe-Eng & Haavard Nord*
- Cross-platform
 - Windows, macOS, Linux, Android, iOS...
- Many famous programs were developed using Qt
 - Google Earth, Mathematica, Bitcoin Core, Telegram...
- Also provides Qt for python (not that useful...)

Qt development

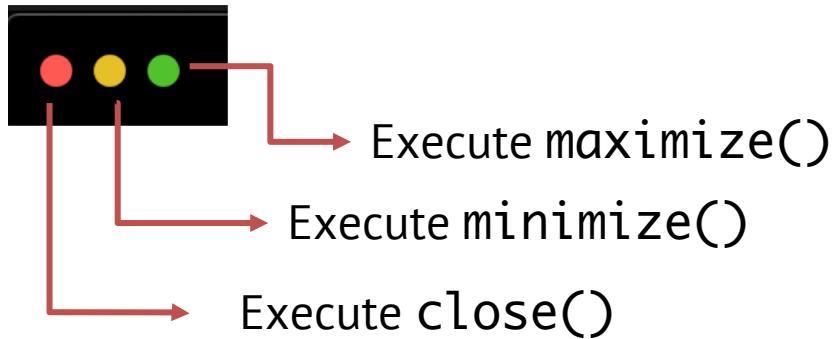
- Qt creator
 - A cross-platform IDE (integrated development environment)
 - macOS, Linux, Windows
 - Supports convenient Qt program development
 - Includes *Qt Designer*
 - Integrated GUI layout builder
 - Users can rapidly design and build widgets and dialogs
 - Previewed immediately (convenient!!)

Qt Classes

- Qt is implemented using OOP
 - Every module in Qt is *Class*
 - Each module can interact with other module by using *methods*
 - You can find Qt Classes info in the [doc](#)
 - ***Signals & Slots***
 - The way of sending & receiving messages btw Qt objects (modules)
 - Central feature of Qt

Signals & Slots

- Communications btw objects
- For example...

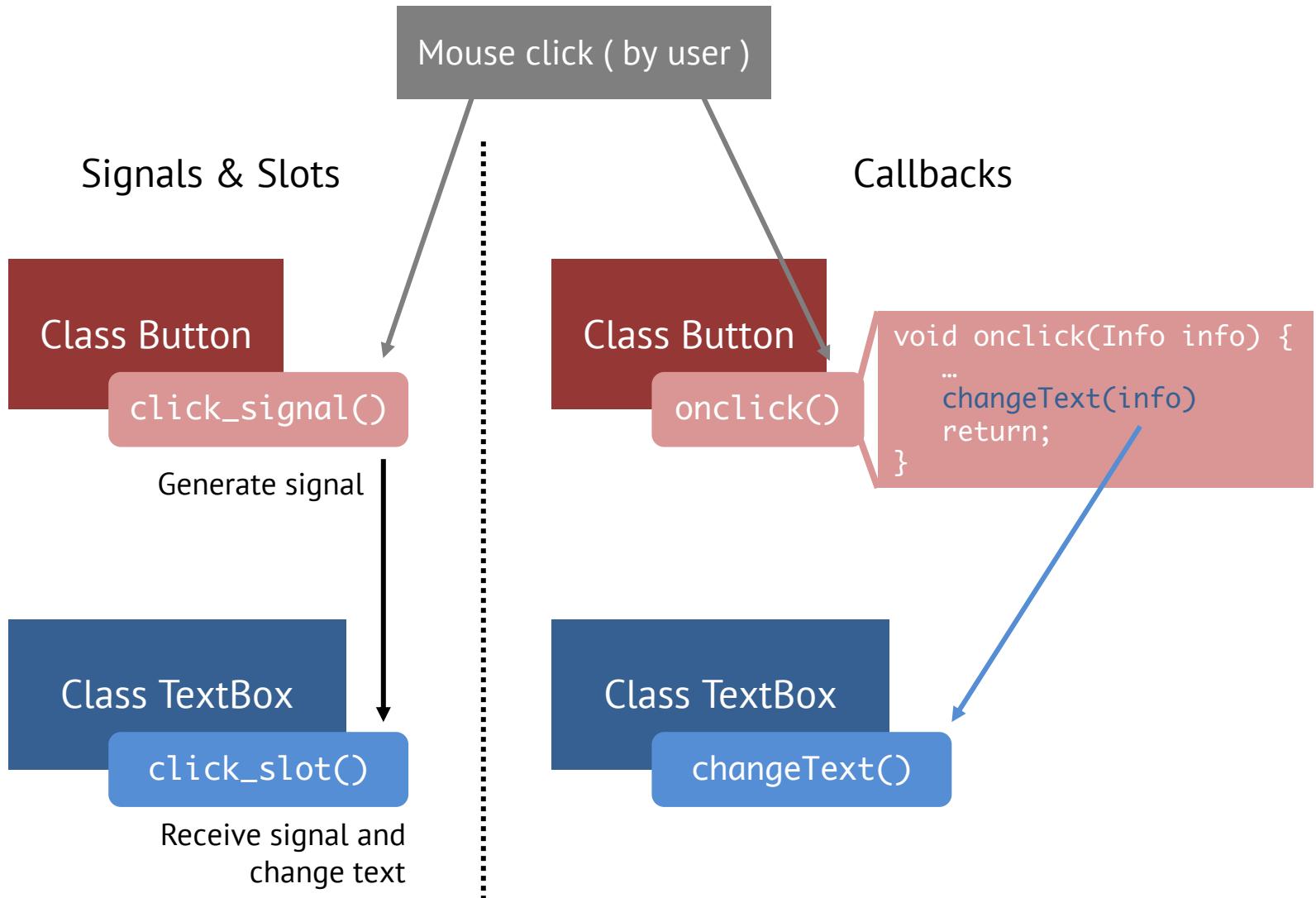


```
Class Window {  
private:  
    int tab_num  
    string current_url  
    ...  
  
public:  
    maximize();  
    minimize();  
    close();  
};
```

- Signals & Slots is one of the ways to implement the connection
 - Different to other frameworks that usually use *callbacks*

Signals & Slots

- *Signals & Slots vs. Callback*



Signals & Slots

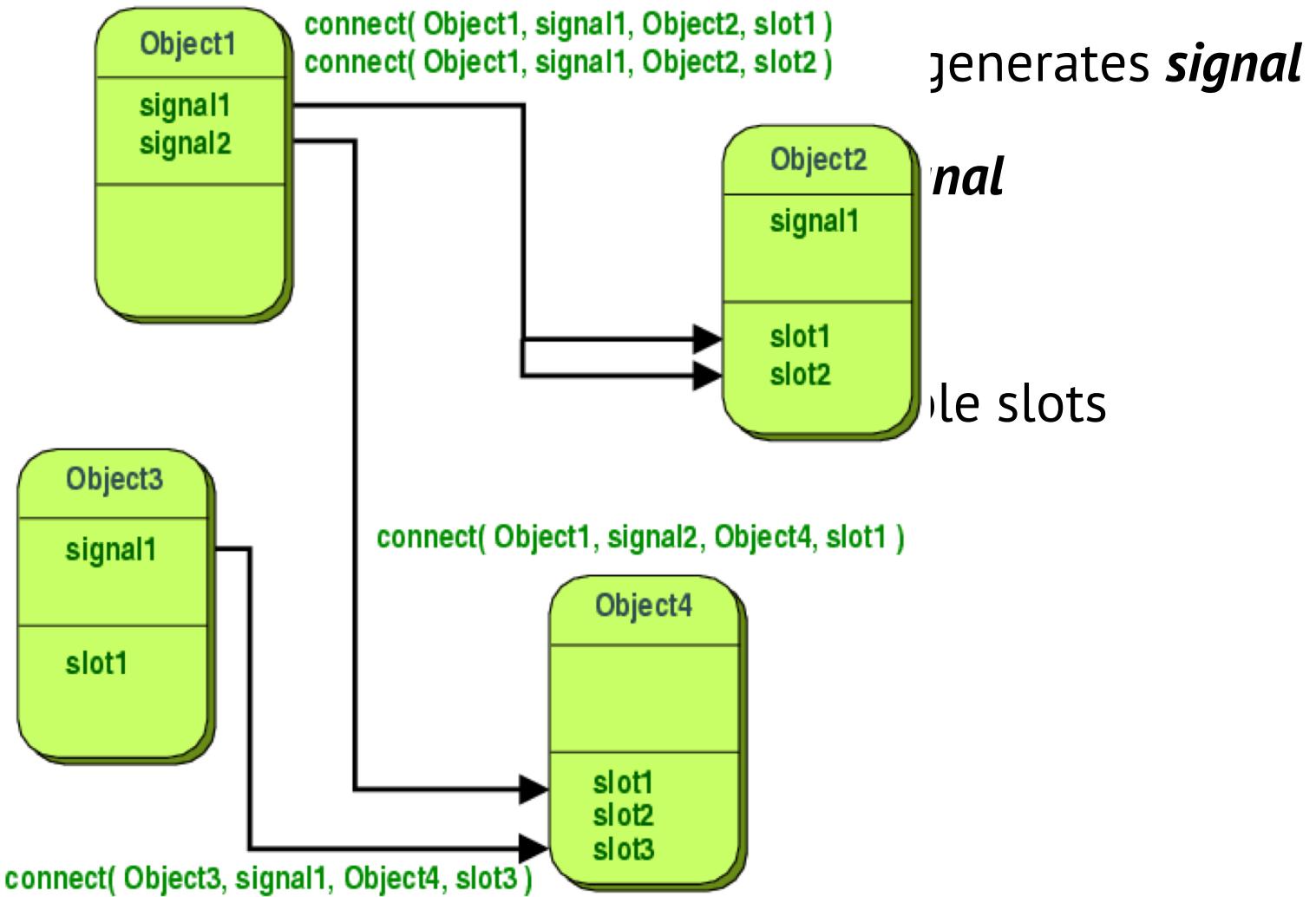
- Event-Driven
- If an event occurs in a module, the module generates ***signal***
- Then the target ***slot(s)*** response(s) to the ***signal***
- You should *connect* signals & slots
- multiple signals can be connected to multiple slots

Signals & Slots

- Event-Driven
- If an event occurs in a module, the module generates ***signal***
- Then the target ***slot(s)*** response(s) to the ***signal***
- You should *connect* signals & slots
- multiple signals can be connected to multiple slots

Signals & Slots

- Event-Driven
- If an
- Ther
- You
- mult

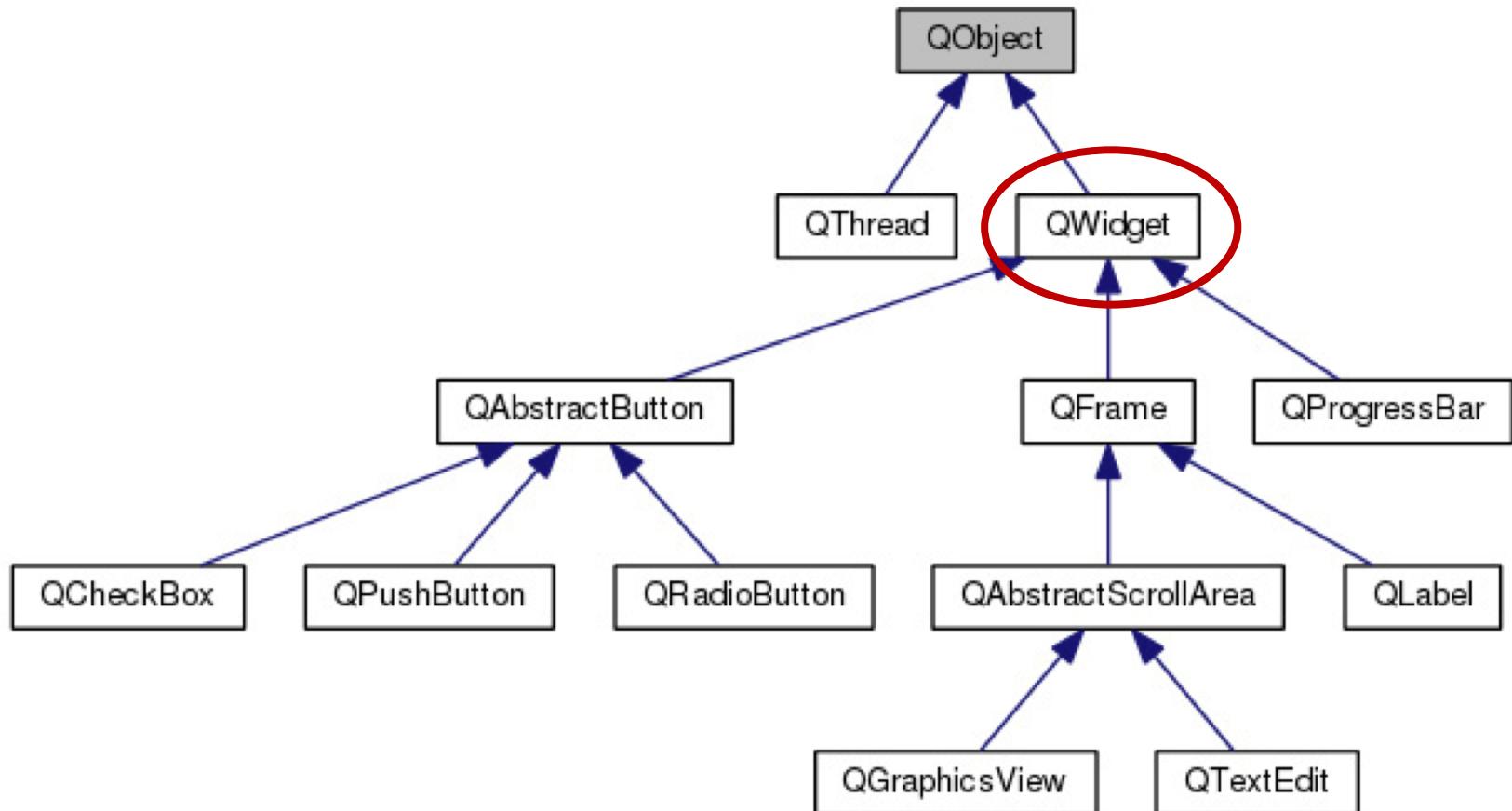


QObject class

- The heart of the Qt Object Model
 - The base class of almost all Qt classes & widgets
 - Defines the functions for basic Qt features & ***inherit them***
 - destroyed()
 - Basic functions to support signals & slots
 - connect() function for signals & slots
 - And else...

QObject class

- QObject hierarchy



QWidget class

- The core of Qt user interface
- Receives user input
 - mouse, keyboard, and other...
- Paints a representation of itself on the screen
- Locating widgets
 - Clipped by its parent & by the widgets in front of it

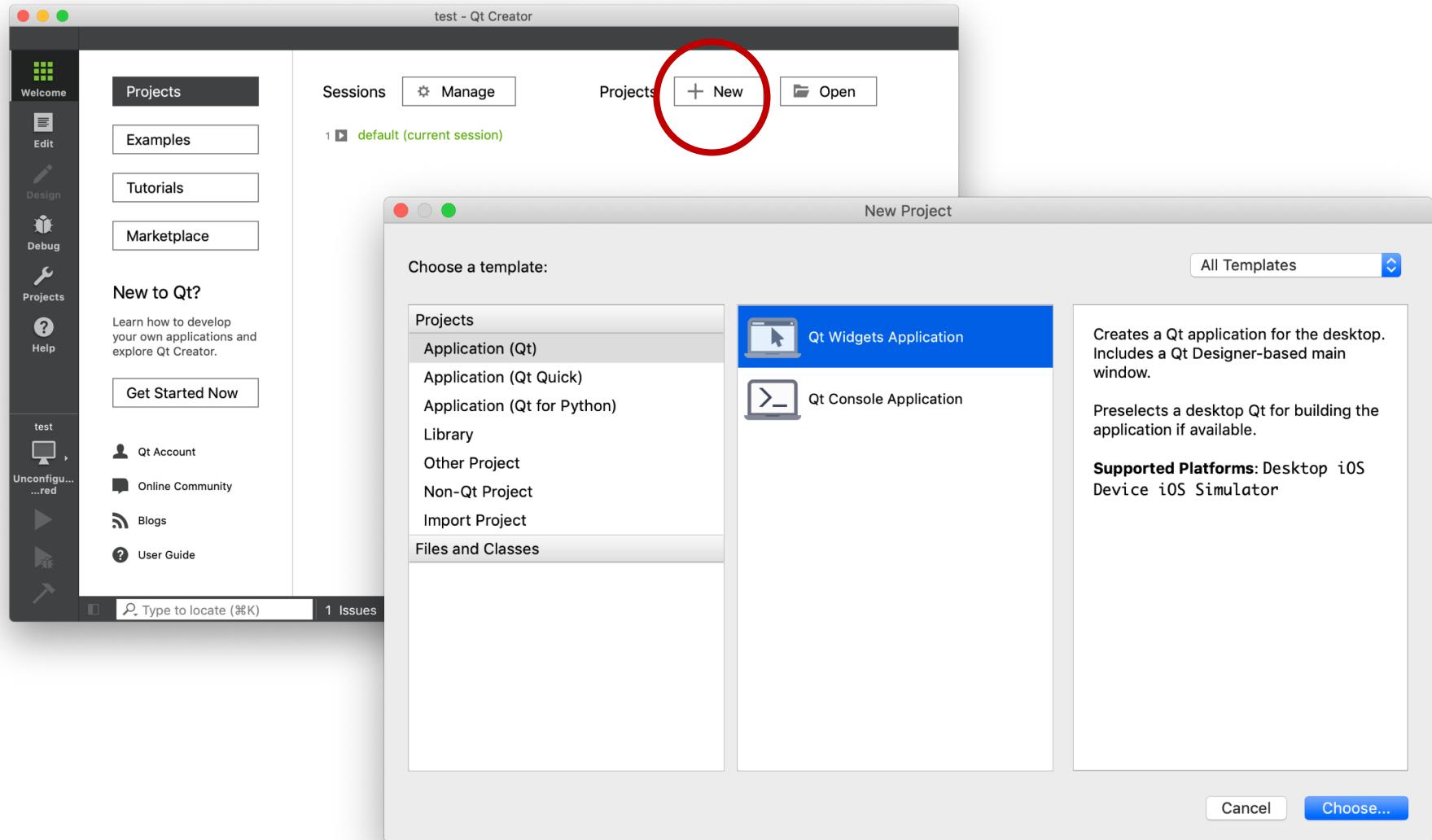
Qt Tutorial

Preparation

- Install Qt & Qt Creator
 - We'll use Qt 5.12.9.
 - Refer the [manual](#) for the installation

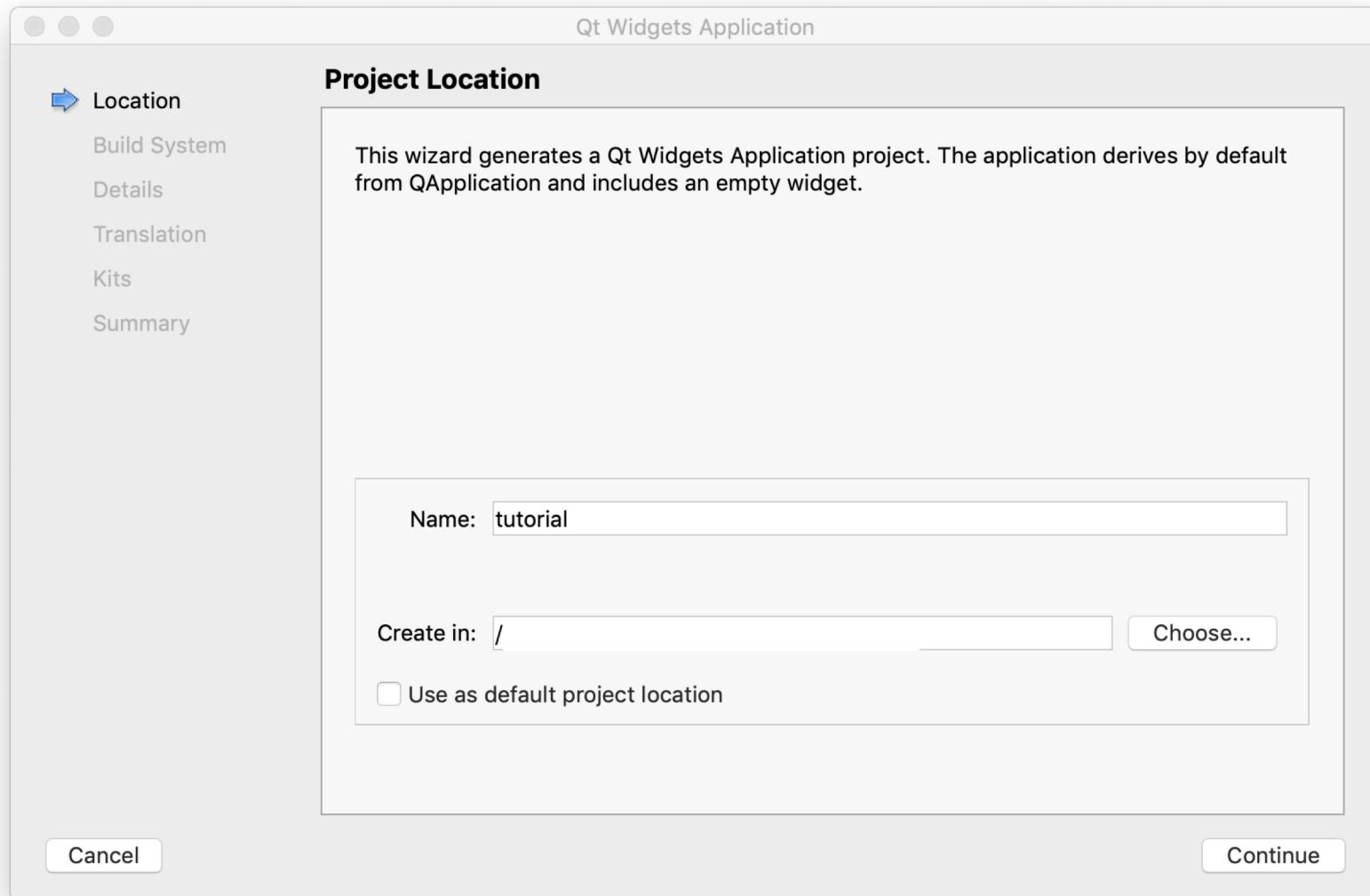
Create new project

- Click **+New** project button and select **Qt Widgets Application**



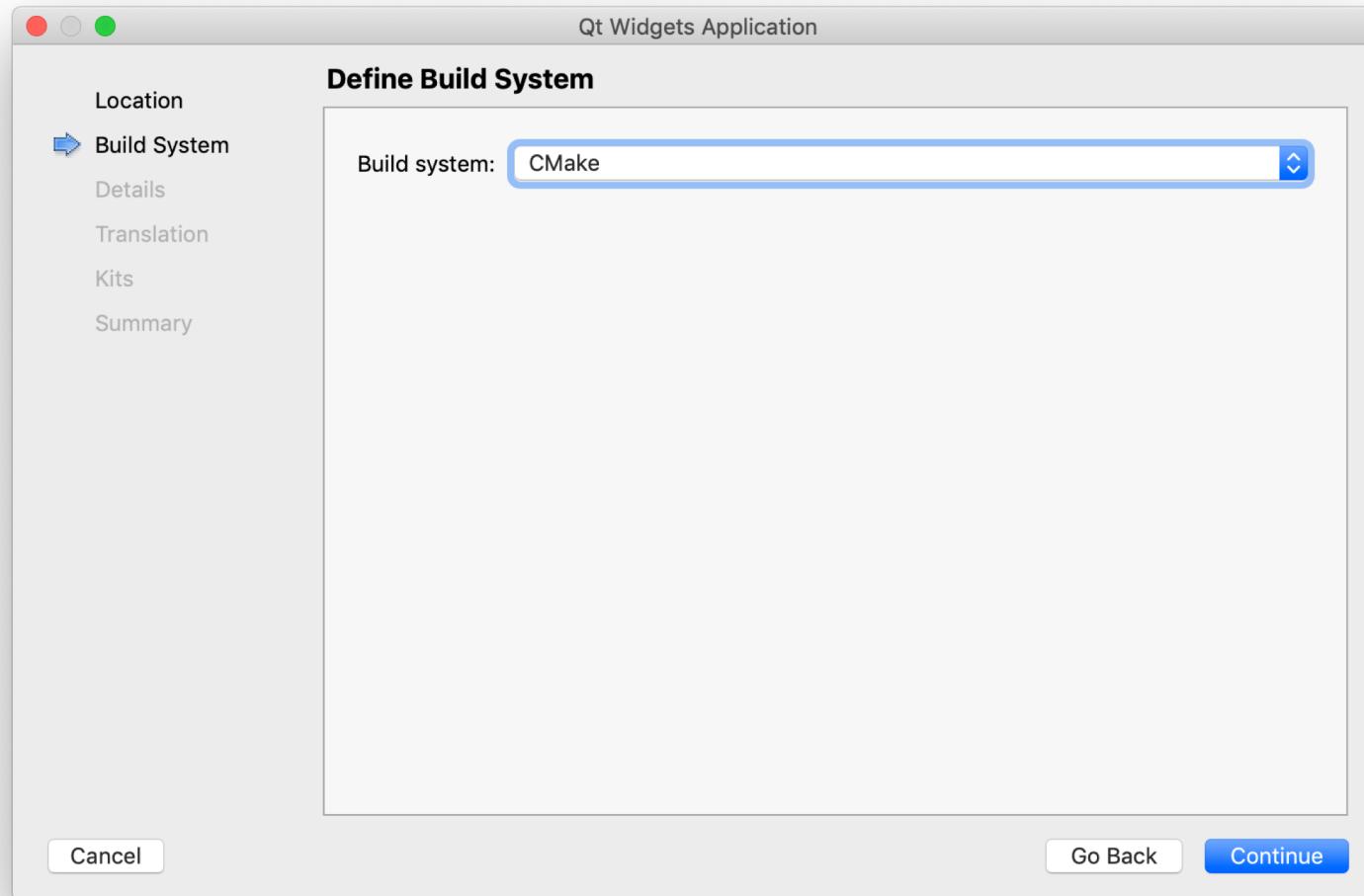
Create new project

- Set proper project path (location) and continue



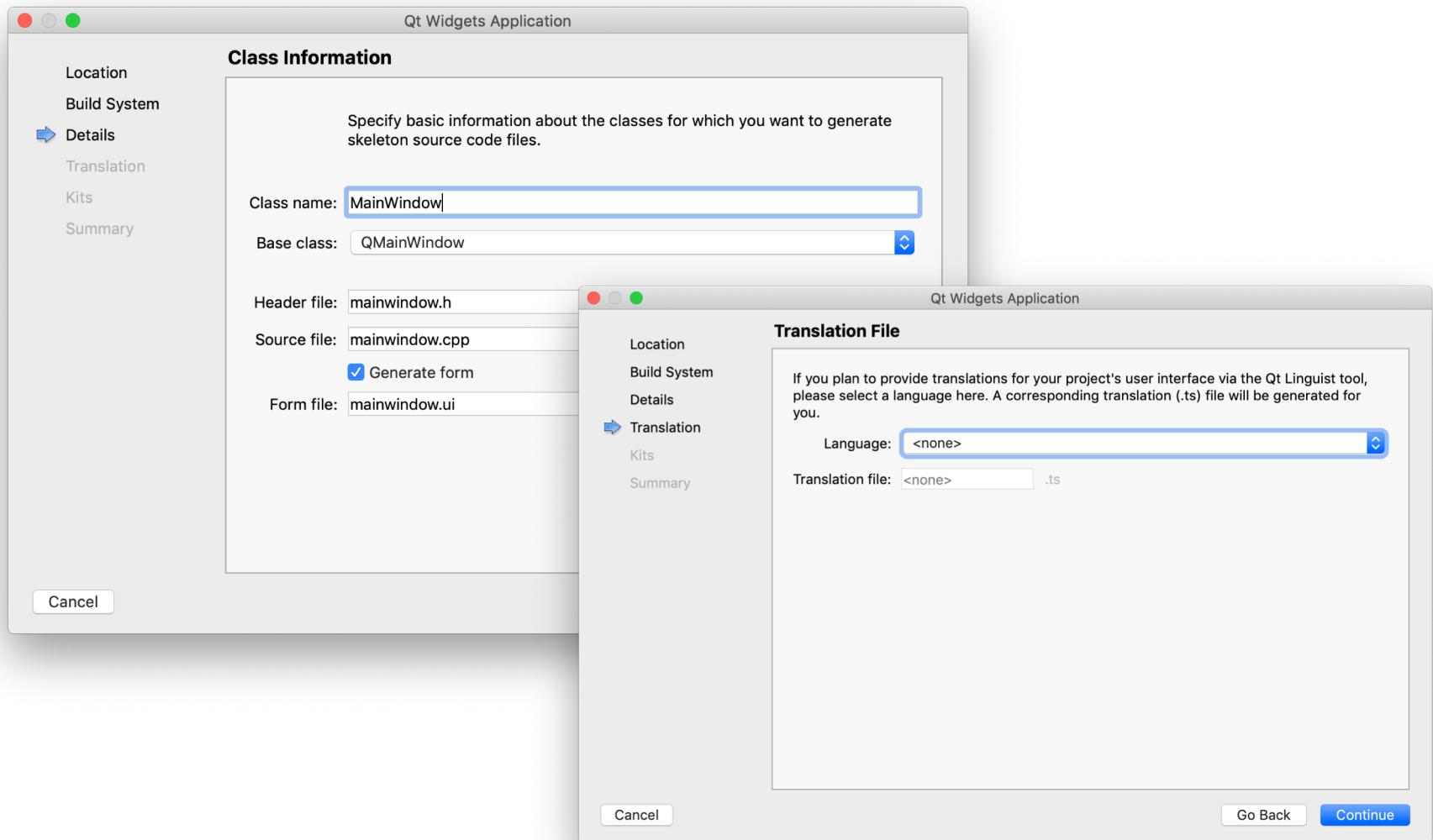
Create new project

- Select *cmake* as build system (maybe different in Windows)



Create new project

- Use default Class information & Translation settings



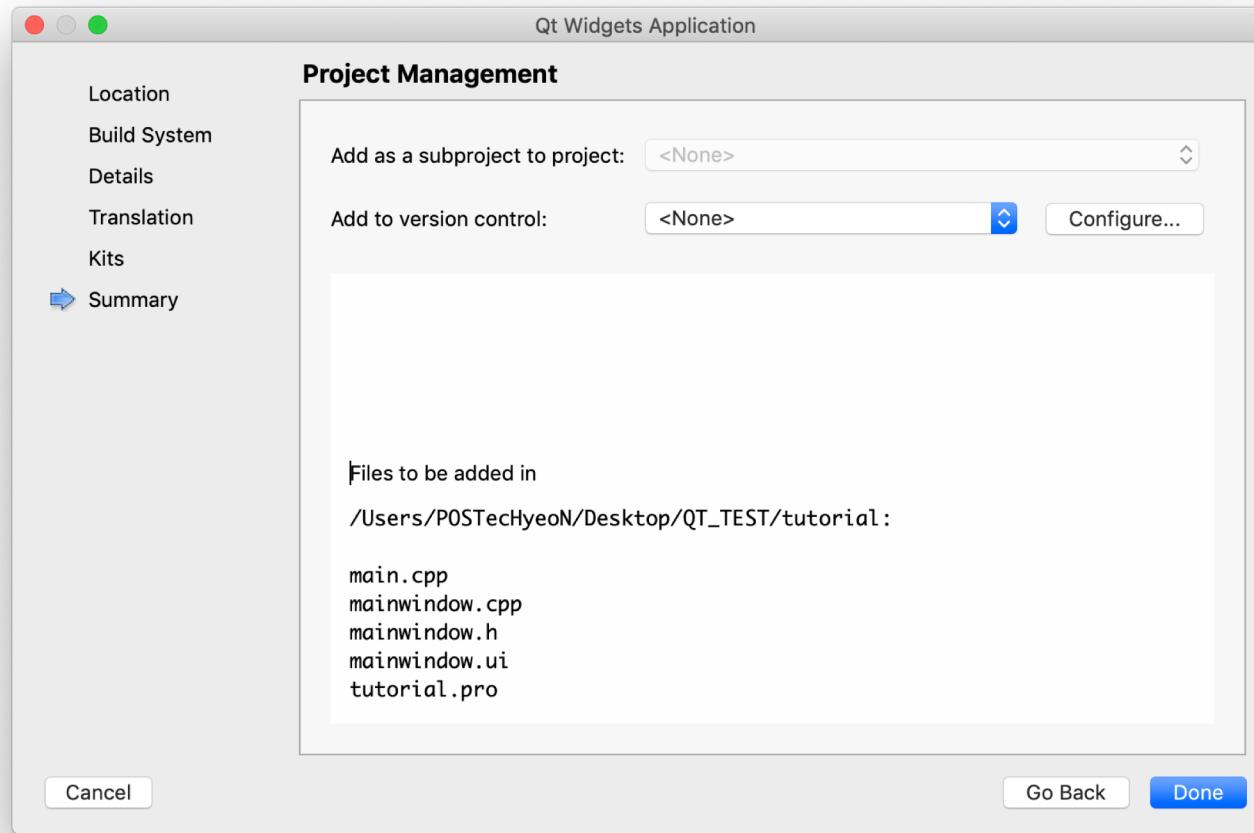
Create new project

- Select proper kit for the **Desktop & your compiler** and continue



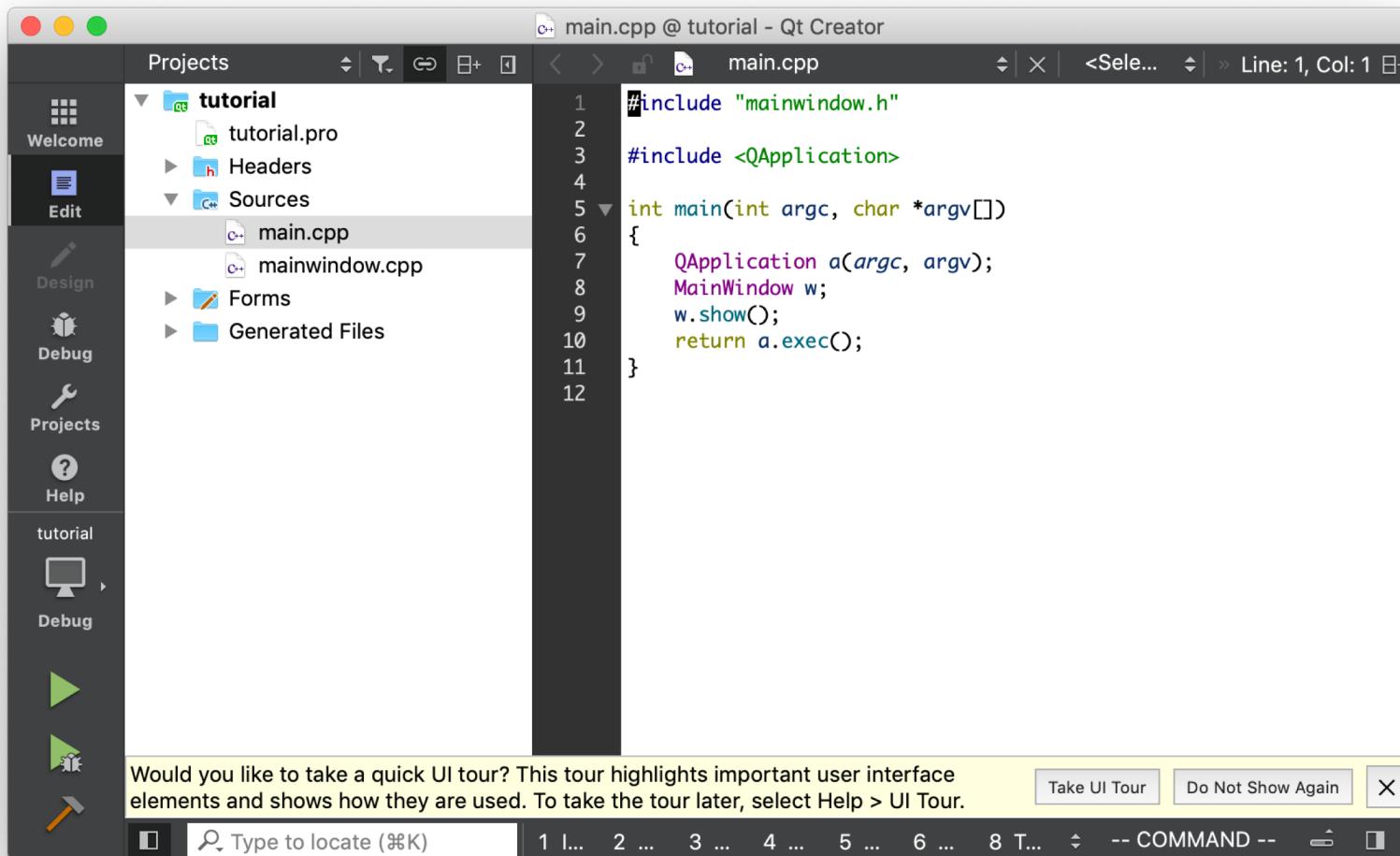
Create new project

- Use default project management setting



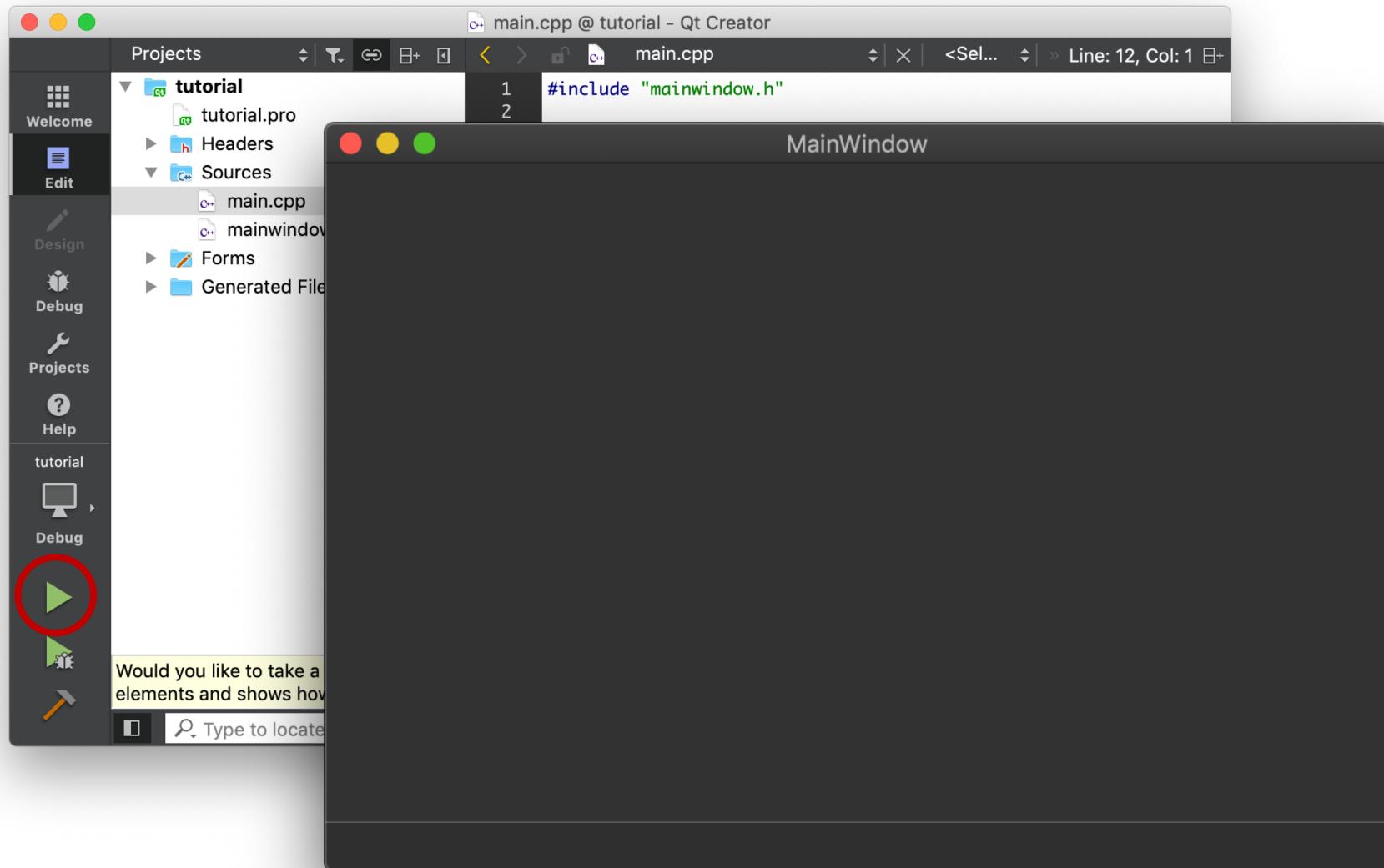
Create new project

- Qt creator should look like this:



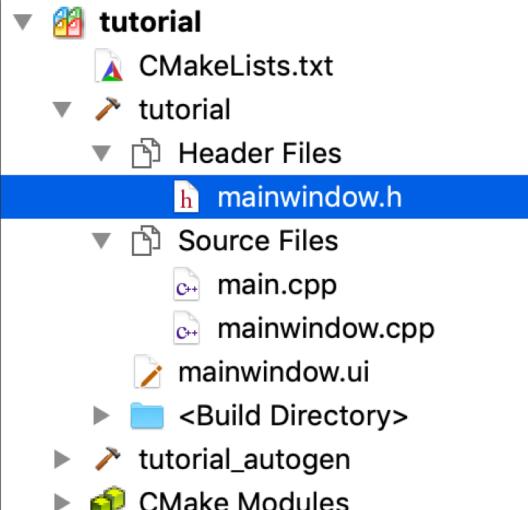
Create new project

- Press compiler button: then MainWindow appears



Create new project

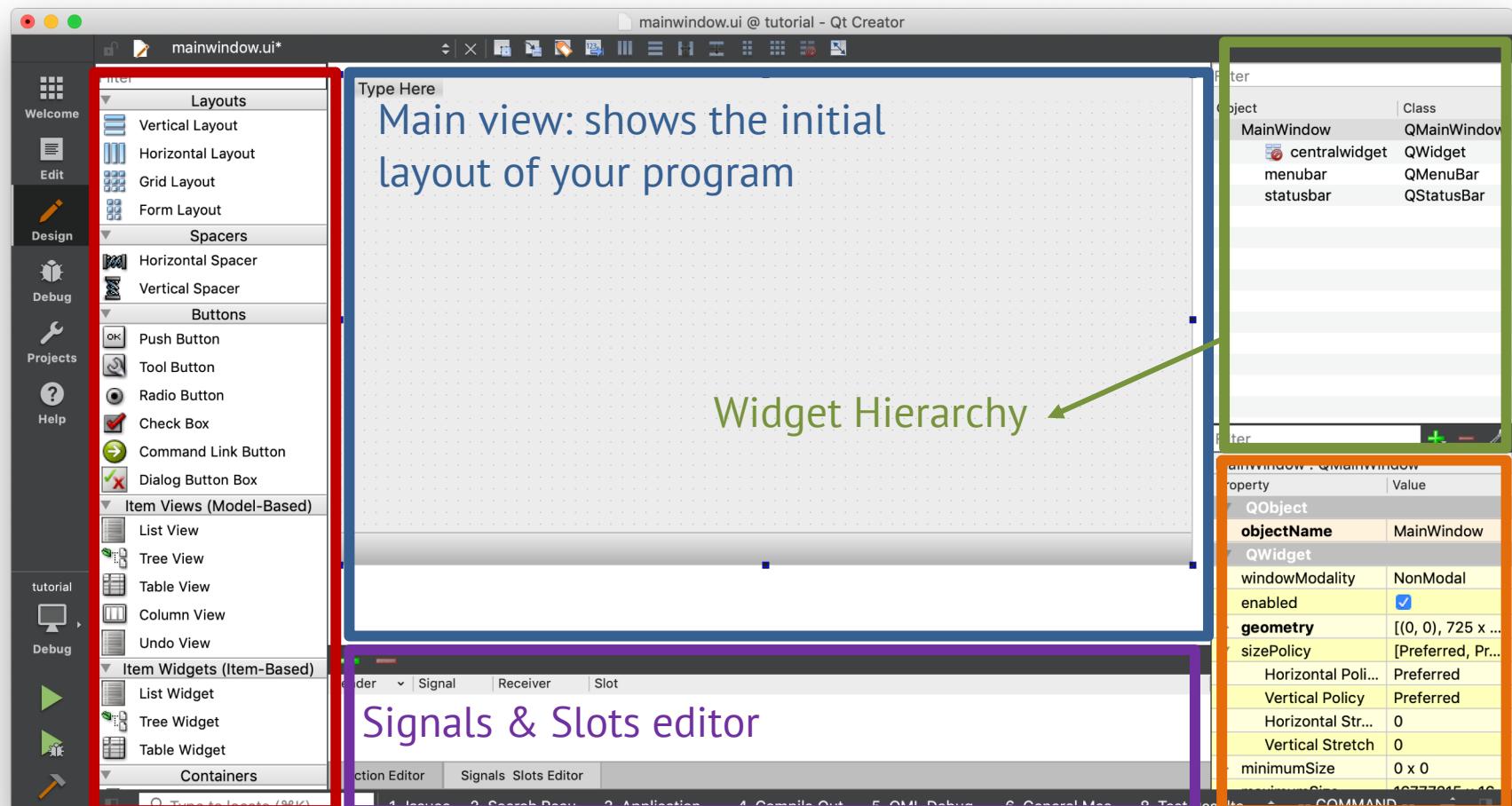
- File description
 - CMakeLists.txt
 - Project configuration file
 - mainwindow.h
 - mainwindow.cpp
 - Defines the behavior & characteristic of mainWindow
 - Inherits QMainWindow (**which inherits QWidget**)
 - mainwindow.ui
 - Defines the layout of mainWindow
 - You can modify it in **Design** tab
 - main.cpp
 - Open / close mainWindow / Application



```
#include "mainwindow.h"  
  
#include <QApplication>  
  
int main(int argc, char *argv[])  
{  
    QApplication a(argc, argv);  
    MainWindow w;  
    w.show();  
    return a.exec();  
}
```

UI modification

- Enter Design mode by double-clicking mainwindow.ui

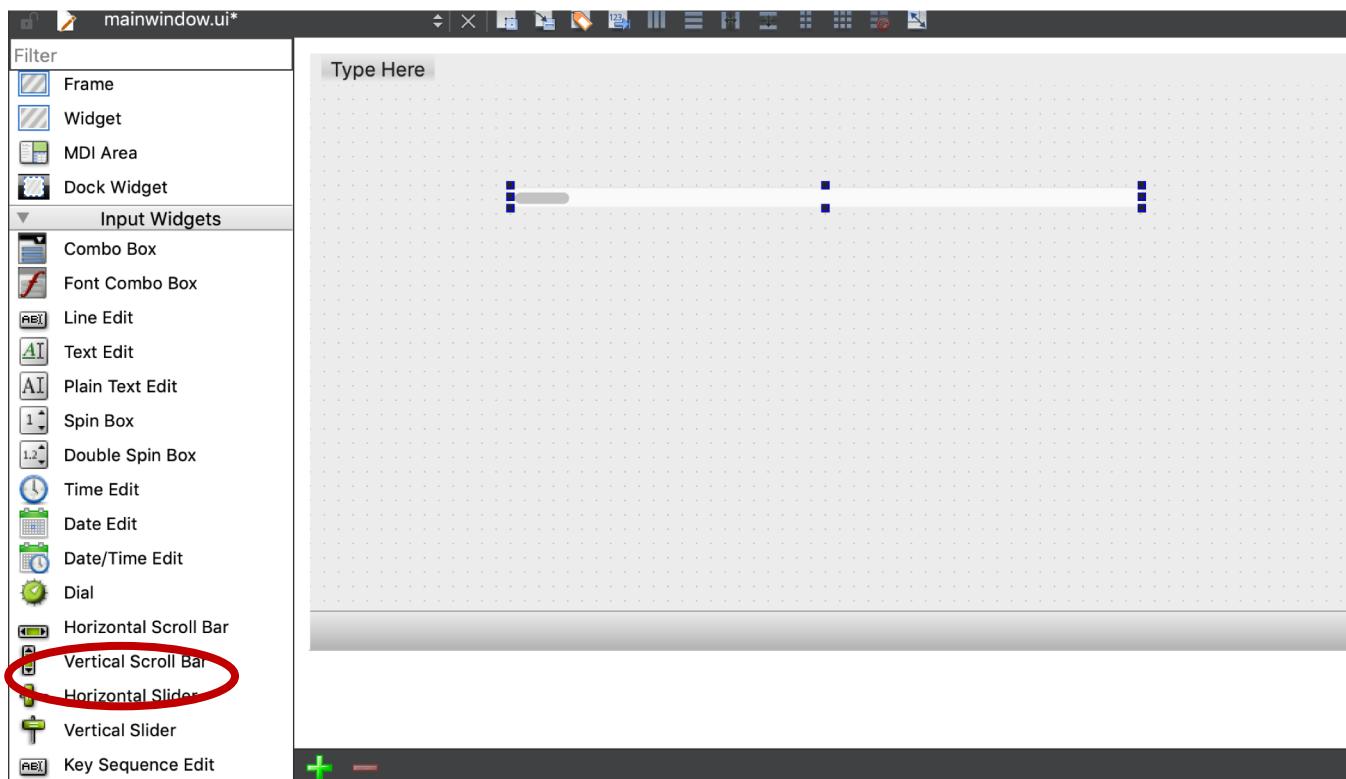


QWidget Library: You can drag & drop any widget from this menu

The info / property of currently selected widget

UI modification

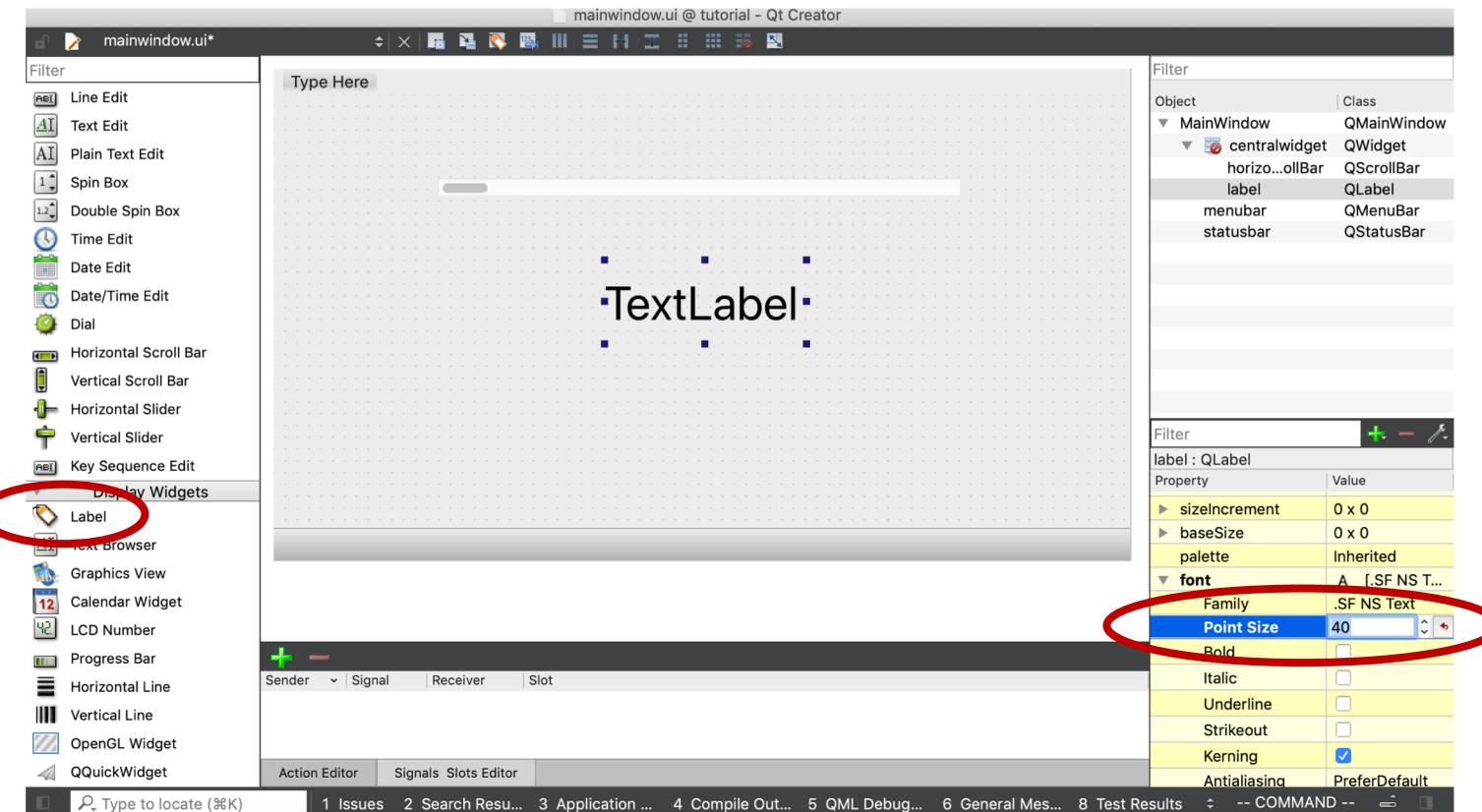
- Add Horizontal slide (Drag & Drop)
 - Also modify the location & size



- Compile it and check the slider...nothing happens while moving it

UI modification

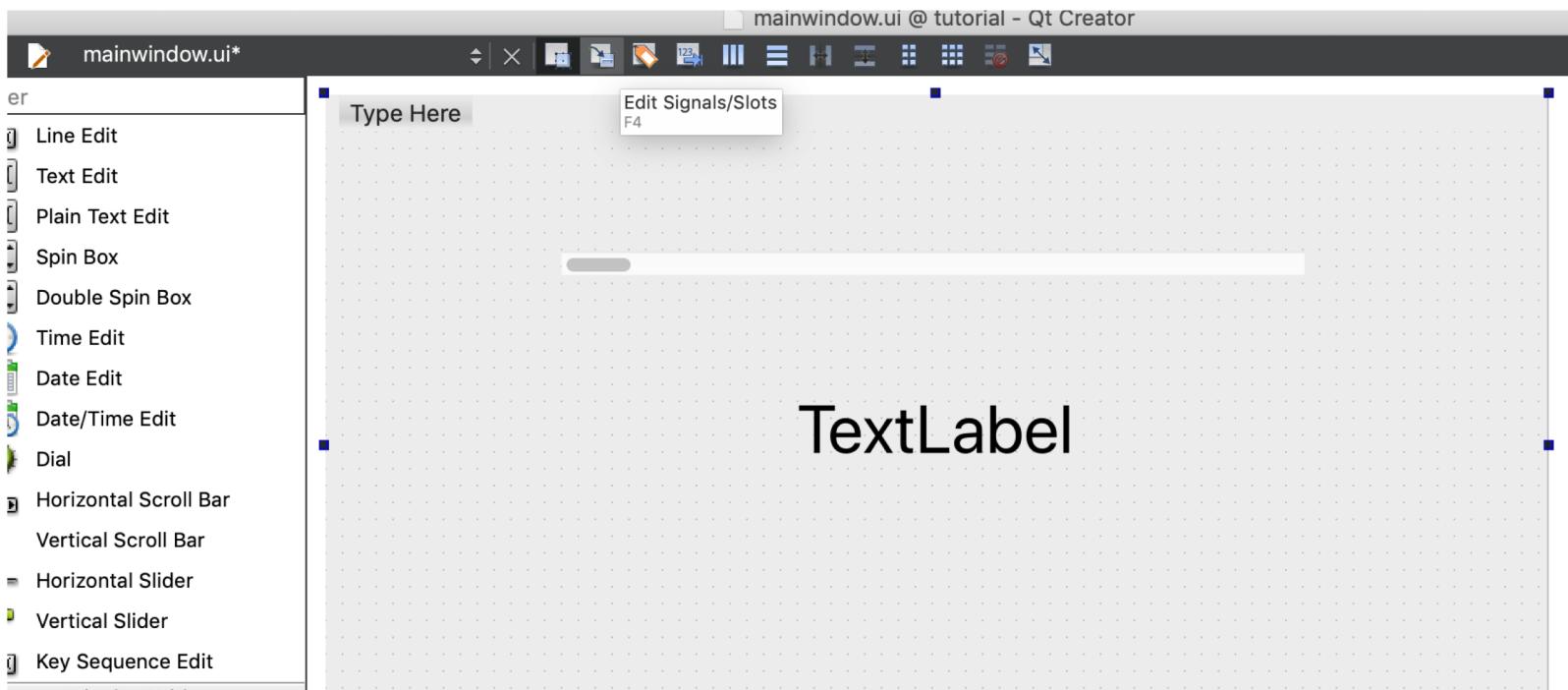
- Add text label (Label) and set its font size to 40



- Compile and check the difference!!

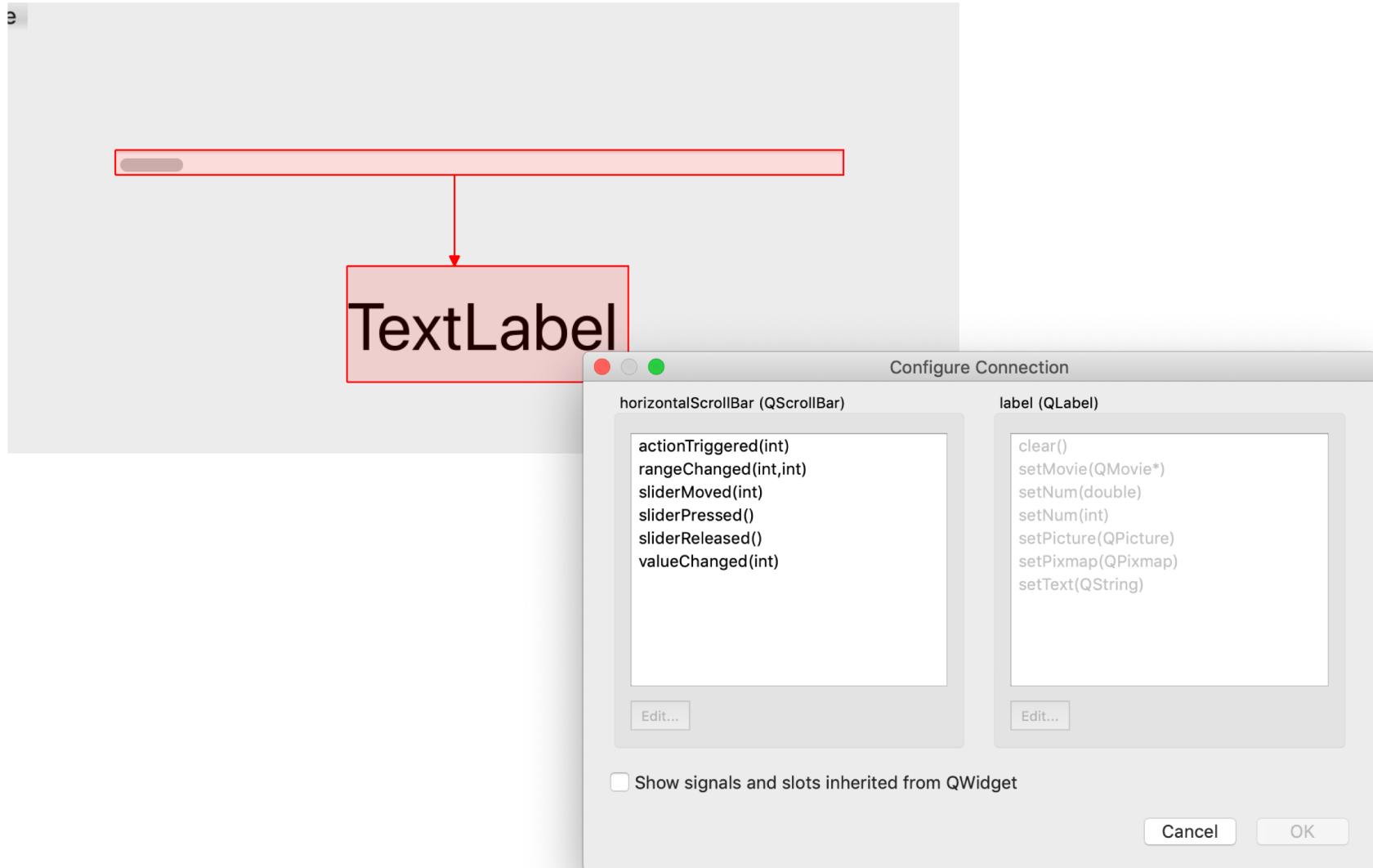
Add Signals & Slots

- Select ***Edit Signals/Slots*** button to enter editing mode



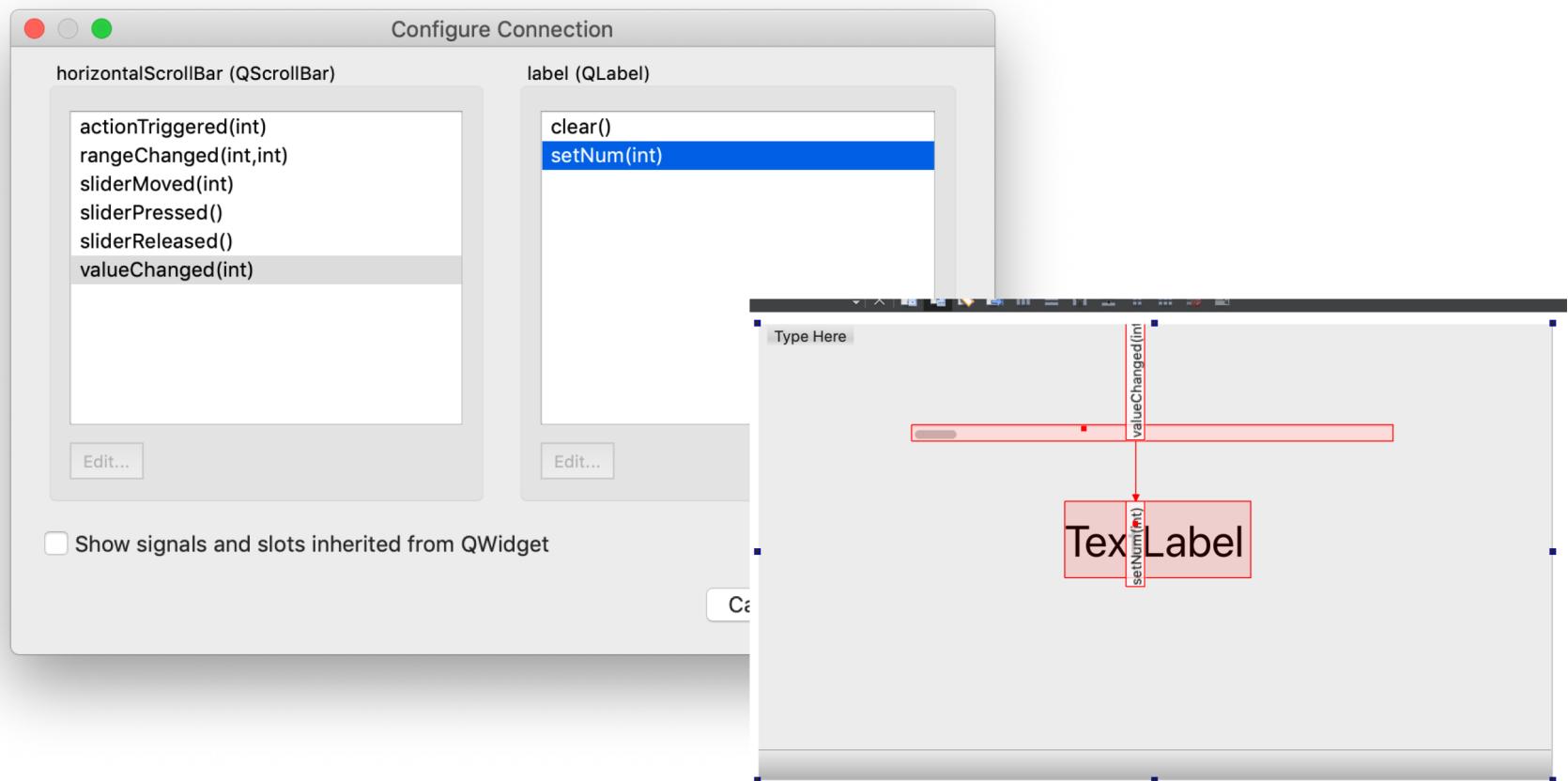
Add Signals & Slots

- Click **slider** and drag & drop it to **TextLabel**



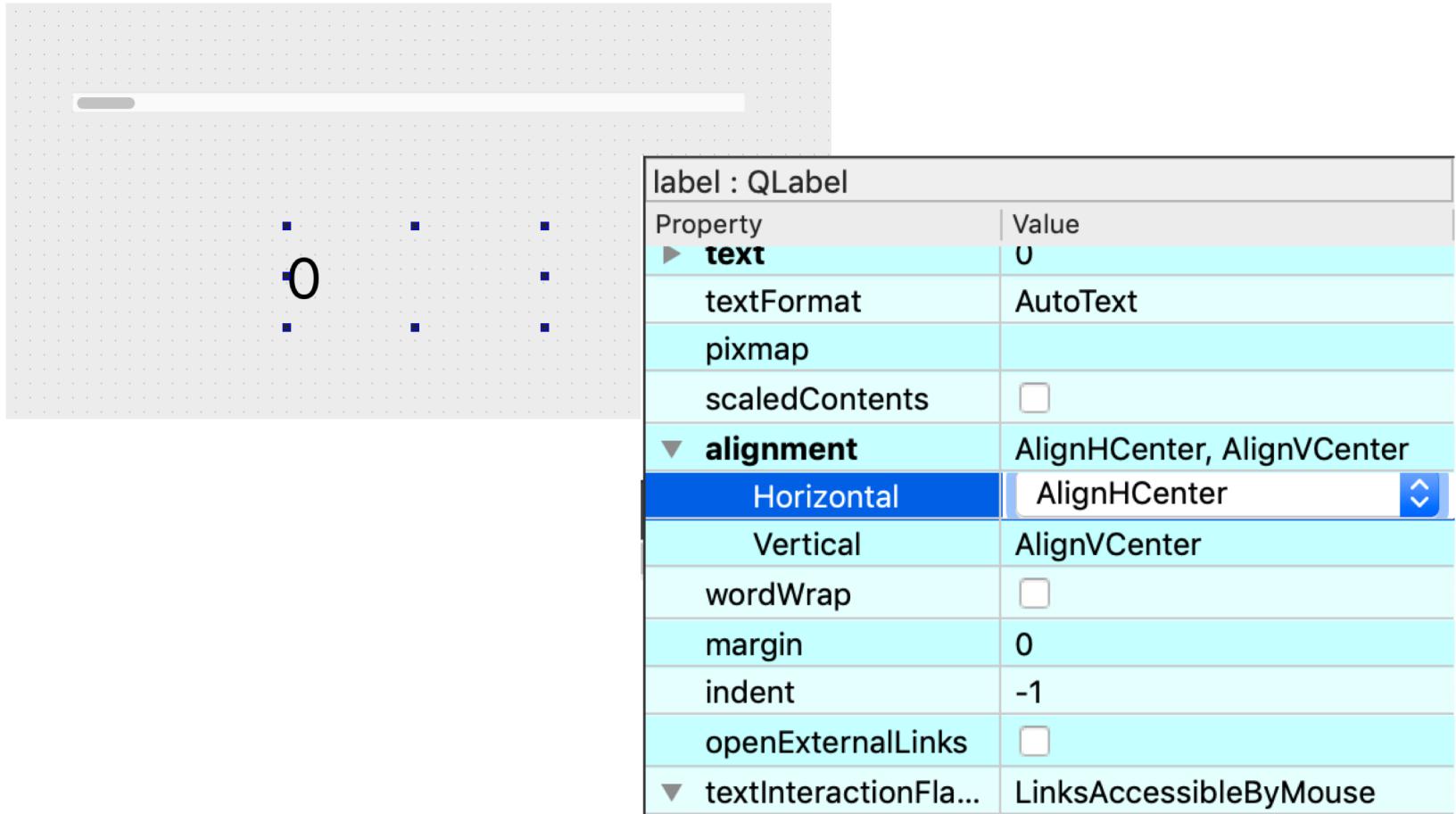
Add Signals & Slots

- Select proper signal & slot
 - signal : valueChanged()
 - slot: setNum()



Add Signals & Slots

- SetTextLabel's initial value to “0” and set center alignment

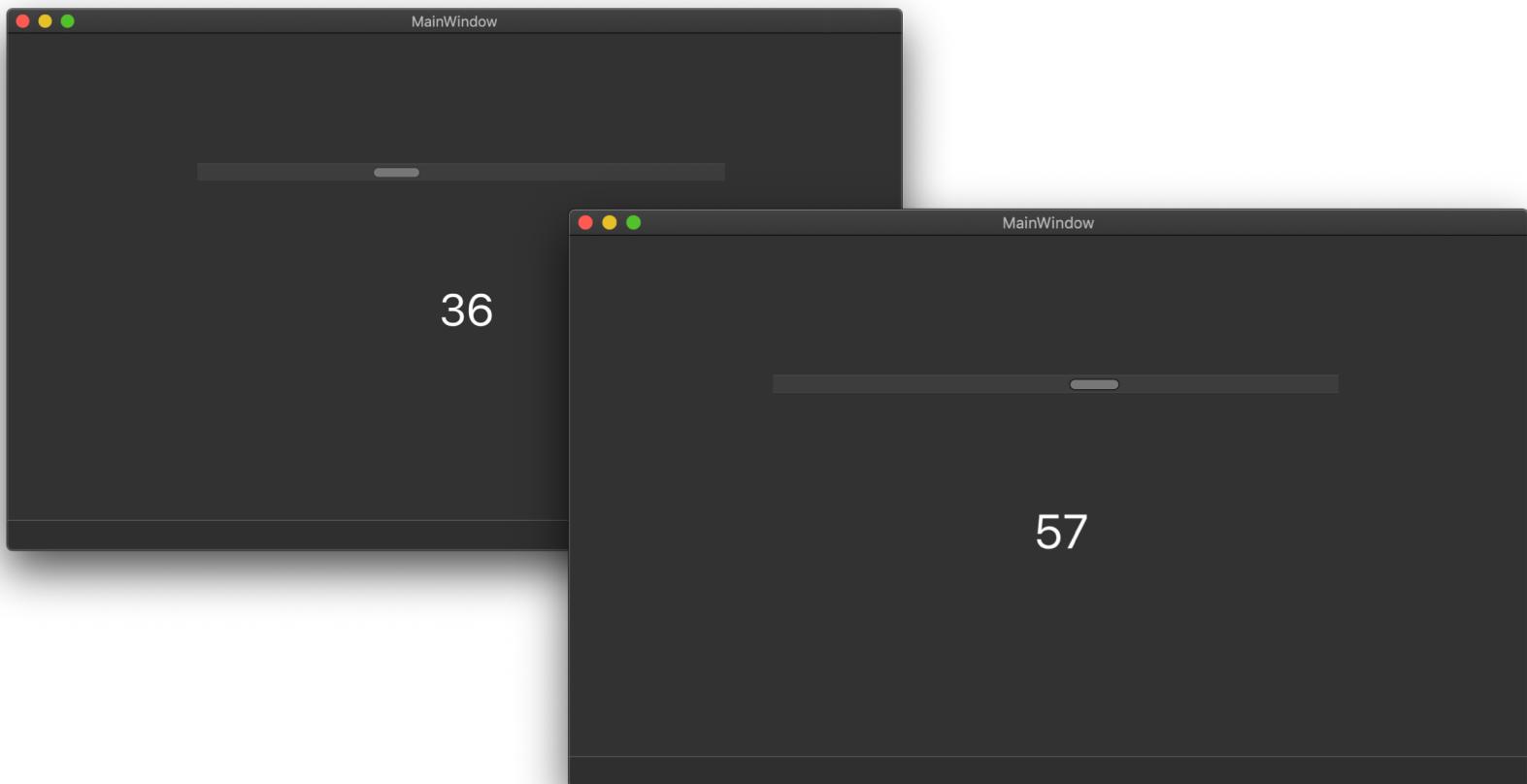


The screenshot shows the Qt Designer interface. On the left is a gray workspace containing a single QLabel widget centered on a dotted grid. The label displays the text "0". To the right of the workspace is a properties dialog for the label. The dialog has a title bar "label : QLabel" and a table of properties.

Property	Value
▶ text	0
textFormat	AutoText
pixmap	
scaledContents	<input type="checkbox"/>
▼ alignment	AlignHCenter, AlignVCenter
Horizontal	AlignHCenter
Vertical	AlignVCenter
wordWrap	<input type="checkbox"/>
margin	0
indent	-1
openExternalLinks	<input type="checkbox"/>
▼ textInteractionF...	LinksAccessibleByMouse

Add Signals & Slots

- Compile it!!



ASSN4

Order 66



Order 66

- Preparing...

Thank you!!

contact: hj@hcil.snu.ac.kr