Installazione Qt Creator

Nel 2023 su Debian si può installare il software Qt Creator con questo comando:

```
apt install qtcreator qmake6 build-essential libgl1-mesa-dev
```

Spesso, ma non sempre, Qt creator viene usato per compilare con il framework Qt

```
apt install clazy cmake g++ qt6-base-dev
```

I seguenti pacchetti sono raccomandati:

```
qtcreator-doc clang-tidy qt6-base-dev-tools  qt6-tools-dev-tools
qt6-declarative-dev-tools qt6-qmltooling-plugins gdb make qml-qt6
```

Infine, se si desidera usare la guida integrata (premendo F1 dopo aver selezionato una classe nell'IDE) si possono aggiungere anche i seguenti pacchetti:

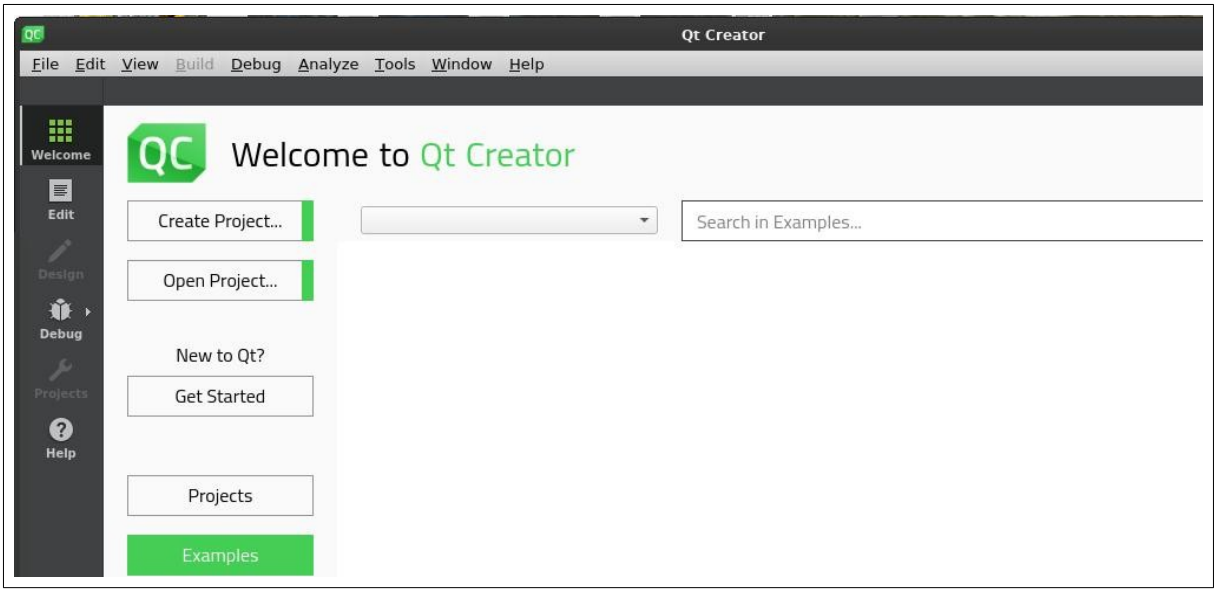
```
qt6-base-doc
qt6-tools-doc
```

Anche se non sono disponibili nella Debian stable, è facilissimo trovarli nella Debian testing.

Configurazione

Appena si avvia appare la finestra introduttiva.

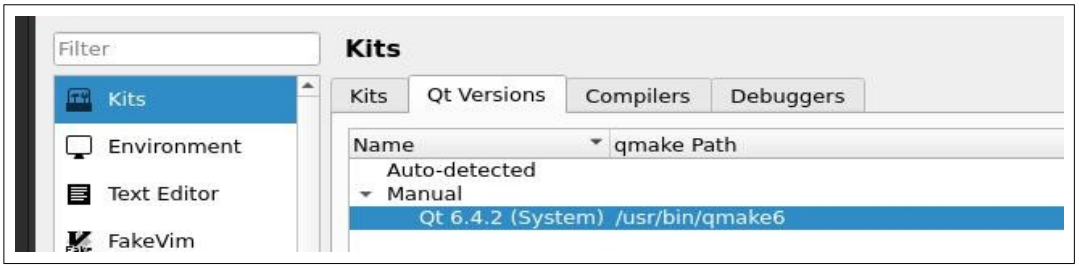
Per prima cosa potrebbe essere necessario **configurare** Qt Creator perché possa trovare il compilatore ed altri strumenti di compilazione, atrimenti sarebbe impossibile creare un progetto



Aprire il menù **Edit** > Preferences…

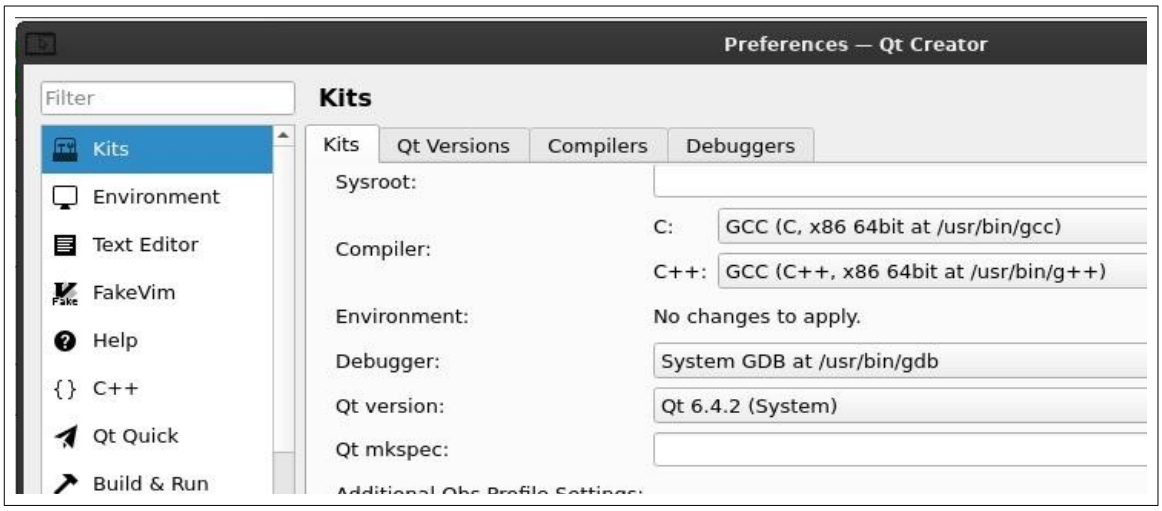
Configurazione testata su Debian: aprire la scheda Kits > Qt version >**Add**

e verificare di aver selezionato questo percorso: `/usr/bin/qmake6`



Configurazione testata su Debian: aprire la scheda Kits > Kits

Scendere in basso e verificare di aver selezionato la versione Qt 6.x.y (quella disponibile)



Dimostrazione d'uso di Qt Creator

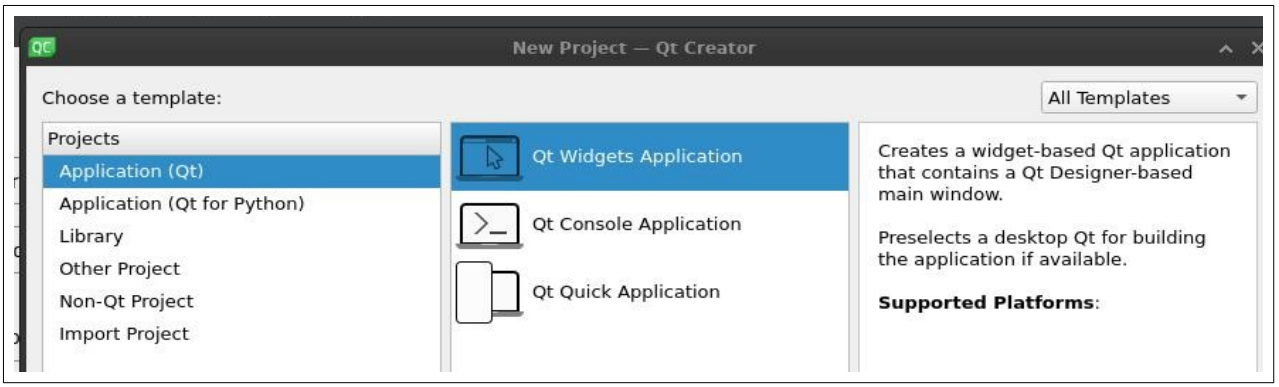
La finestra introduttiva presenta una barra verticale a sinistra (mode selector) che permette di scegliere tra:
Welcome, Edit, Debug, Projects, Help.
Provare ad aprirle tutte almeno una volta...
Nell'area di benvenuto ci sono tre linguette: Get Started, Projects, Examples, Community

Esempio guidato

Vedremo come creare un nuovo progetto e come compilarlo.

L'interfaccia IDE

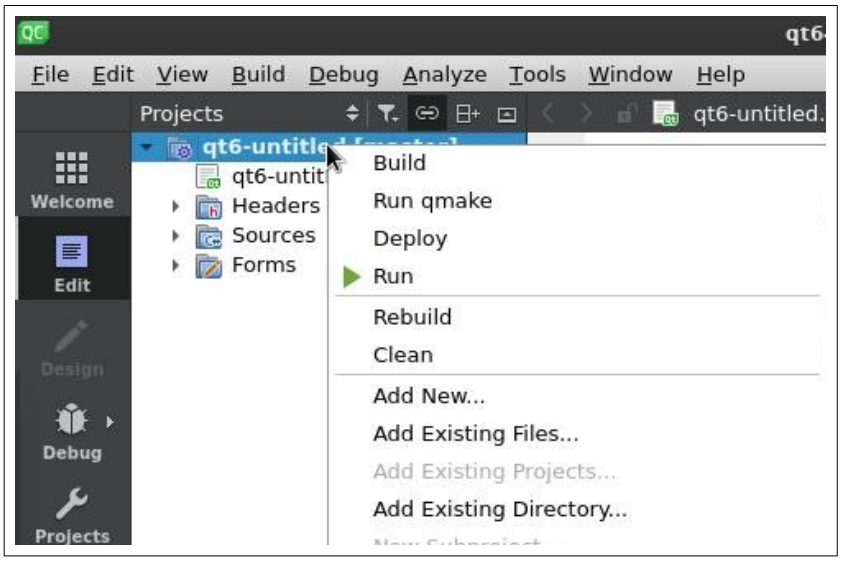
Dal menù File > New file or project... > Application
In questo esempio si userà il template Qt Widget



1. Una sequenza di scelte guidate aiuterà a configurare il progetto:

- **Location**: digitare nome e percorso di una cartella dove a sua volta verrà creata una nuova cartella.
- **Build system**: qmake (oppure cmake)
- **Classes information**: scegliere dal menù a tendina la classe "base" (per ora Qwidget). In alternativa ci sono QDialog e QMainWindow. Quest'ultima crea una finestra completa, dotata di tutti gli optionals. Quando una finestra madre ha una finestra figlia (ad es. dialog) la figlia può bloccare oppure non bloccare il genitore (setWindowModality "Modal" o "nonModal").
- **Translation** (omesso)
- **Kit selection**: selezionare ambiente desktop generico (ma potrebbe richiedere l’installazione e configurazione di particolari librerie di sviluppo Qt)
- **Project management**: si può attivare subito git

2. La cartella superiore è quella del progetto e cliccandoci col tasto destro del mouse si potrebbero eseguire le seguenti azioni:



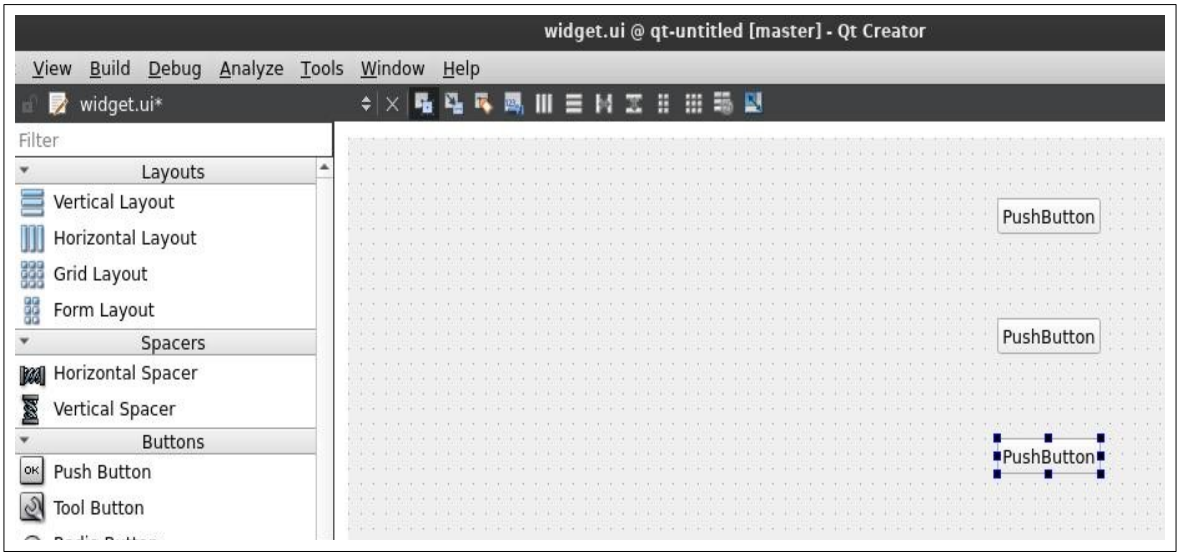
- creare un nuovo file: add new...
- impostare il tipo di compilazione: Set Build Configuration (Release o Debug)
- avviare la compilazione del progetto: Build Project
- impostare il progetto da eseguire tra i diversi progetti aperti: Set Run Configuration
- deploy (omesso)

3. Aprire con un doppio click il file **widget.ui** dalla cartella Forms. Questo file assomiglia ad una pagina web, ma viene interpretato e rappresentato mediante un'interfaccia grafica che nasconde il codice sorgente (che verrà gestito automaticamente). L'area dell'editor mostra, da sinistra a destra, tre nuove aree di lavoro:
   - la colonna dei widgets divisi per gruppi (con il tasto destro si può passare ad icon view)
   - l'anteprima del form (al centro) oppure il sorgente
   - l'albero gerarchico degli oggetti (a destra), con sotto la casella "editor delle proprietà" dell'oggetto selezionato.
   - Al di sopra di queste tre aree è presente la barra degli strumenti (toolbar), dove sono mostrate 12 pulsanti: 4 modalità di editing, 7 modalità per il layout e una per il ridimensionamento

4. Provare a passare il mouse sopra le 4 diverse modalità di editing (le prime 4 icone)
5. In questo momento c'è un unico oggetto presente nell'area a destra, detta "albero degli oggetti", ed è Widget, il contenitore del form, mostrato come un rettangolo grigio (dotato di griglia a quadretti). Nell'area sottostante ("editor della proprietà") si nota che tale oggetto appartiene alla classe QWidget e anche come questa erediti alcune proprietà (objectName) dalla classe QObject. Ogni gruppo di proprietà è raggruppata per colore a seconda della classe di appartenenza.

Costruire un form (GUI)

6. Per visualizzare l'anteprima selezionare il menù Tools > form editor > preview oppure premere CTRL + ALT + R



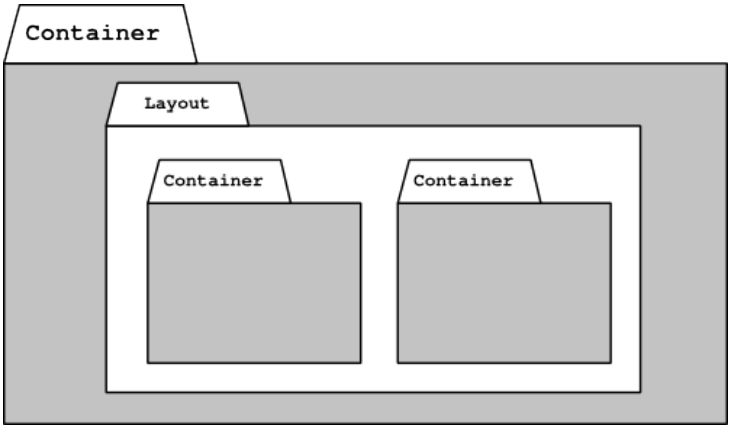
7. Trascinare tre "**Push Button**" (QPushButton) dalla colonna dei widget verso l'anteprima del form. Essi verranno automaticamente aggiunti anche all'albero degli oggetti in alto a sinistra.
8. Allo stesso modo trascinare anche un "**List Widget**" (QListWidget).
9. Domanda: da quante classi eredita le proprie proprietà l'oggetto di tipo QPushButton? Per rispondere basta leggere il contenuto della casella "editor delle proprietà"...
10. Ognuno dei precedenti oggetti viene creato assegnandogli un nome predefinito (ad esempio `pushButton_2`) e mostra un testo predefinito (PushButton). Tutte queste proprietà si possono modificare facendo doppio click sugli oggetti o nella casella "editor delle proprietà".
11. Modificare le proprietà degli oggetti seguendo lo schema della seguente tabella.

oggetto	proprietà	valore
primo pulsante	`objectName`	`addButton`
	`text`	`Aggiungi...`
secondo pulsante	`objectName`	`deleteButton`
	`text`	`Rimuovi`
terzo pulsante	`objectName`	`clearButton`
	`text`	`Pulisci`

12. Per visualizzare l'anteprima selezionare il menù Tools > form editor > preview... Provare ad ingrandire la finestra a tutto schermo per dimostrare che il layout è fisso e non si adatta bene alle dimensioni dello schermo.
13. Selezionando l'oggetto contenitore Widget (dall'albero degli oggetti) e premere il pulsante "Lay Out in a grid" dalla barra degli strumenti. Si può annullare usando il menù Edit > undo, oppure CTRL + Z. A conseguenza di questa azione:
    - si modificherà l'icona dell'albero
    - si allargheranno i pulsanti

- si renderà disponibile un nuovo pulsante nella barra degli strumenti (break layout).

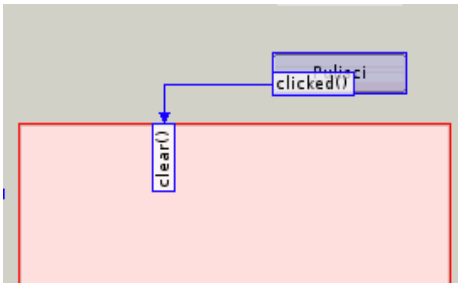
14. Per fare altre prove si può applicare uno spaziatore (trascinandolo da sinistra) e selezionare di nuovo "Lay Out in a grid" dalla barra degli strumenti. Widget è un Container (come QGroupBox e QTabWidget).



All'interno di un **Container** si possono applicare diversi tipi di Layout: verticale, orizzontale e a griglia. Dentro il **Layout** si trovano altri Container (altri widget, considerati figli del precedente Container) che verranno disposti secondo il layout. Dentro ogni Container (figlio) si possono applicare altri layout per disporre altri container, e così via... Quando il form viene costruito in QtDesigner, prima si applicano i Container (sia genitori che figli) e solo dopo il Layout. Quando il codice viene scritto a mano invece, gli ultimi ad essere inseriti sono i Container figli.

Collegare signal e slot

15. Selezionare "Edit signals/slots" dalla barra degli strumenti.
16. Nell'anteprima del form, trascinare il mouse dal pulsante "Pulisci" verso la "List Widget" e collegare il signal **clicked()** allo slot **clear()**. Il risultato di questa operazione dovrebbe essere visibile anche in basso, nella casella **Signals_Slots Editor**.



17. Tornare nuovamente alla modalità "Edit widget" usando la barra degli strumenti.
18. Cliccare con il tasto destro del mouse sul pulsante "aggiungi" e selezionare "Go to slot..."
19. Selezionare dalla finestra di dialogo che comparirà il signal **clicked()**. Questo signal verrà collegato con un nuovo slot (un metodo) generato automaticamente.
    Si passa all'editor del file **widget.cpp** per inserire il codice desiderato in tale slot (un metodo).

```
/** questo metodo apre una finestra di dialogo per l'input del testo */
void Widget::on_addButton_clicked()
{
    QString mioTesto = QInputDialog::getText(this,"Inserimento","Testo:");
    if (!mioTesto.isEmpty())
        ui->listWidget->addItem(mioTesto);
}
```

La linea rossa a sinistra indica le modifiche non ancora salvate.

20. Nello stesso file sorgente, aggiungere in alto: `#include <QInputDialog>`

21. Compilare (Build), cliccando sul pulsante del martello in basso a sinistra
22. Sulla barra di stato ci sono quattro pannelli, l'ultimo a destra mostra l'output della compilazione.
23. Provando ad eseguire (Run) si possono testare solo i due pulsanti (Aggiungi e Pulisci)
24. Ora bisogna ripetere i passi seguiti per il pulsante Aggiungi, anche per il pulsante Rimuovi. Se necessario cliccare dove indicato al punto n.5
25. Si passa ancora all'editor del file sorgente per inserire il seguente codice:

```
/** questo metodo elimina un elemento */
void Widget::on_deleteButton_clicked()
{
    foreach(QListWidgetItem *item, ui->listWidget->selectedItems())
        delete item;
}
```

26. Compilare ed eseguire di nuovo per provare anche il pulsante Rimuovi
27. L'editing del codice non è finito, perché si può notare che il pulsante Rimuovi non deve essere cliccabile se non c'è nulla da rimuovere...
28. Premendo F4 si passa all'header (widget.h) dove si aggiunge, nella sezione `private slots`, il seguente codice:

```
void updateDeleteEnabled();
```

29. Premendo di nuovo F4 si torna al codice sorgente (widget.cpp) dove si crea un nuovo metodo

```
/** questo metodo abilita Rimuovi solo quando ho selezionato un elemento */
void Widget::updateDeleteEnabled()
{
ui->deleteButton->setEnabled(ui->listWidget->selectedItems().count()!=0);
}
```

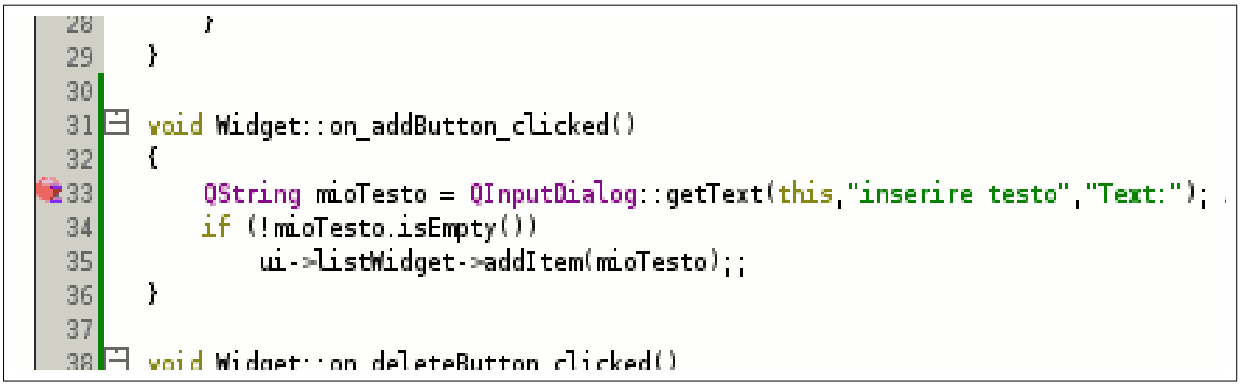
30. infine aggiungere l'istruzione `connect` sotto `ui->setupUi(this);`

```
connect(ui->listWidget->selectionModel(),
      SIGNAL(selectionChanged(QItemSelection,QItemSelection)),
      this, SLOT(updateDeleteEnabled()));
updateDeleteEnabled();
```

31. connect è necessario per collegare il SIGNAL di selezione del mouse all'ultimo metodo (SLOT) appena creato. Infine ri-compilare e ri-eseguire.

Errori e Debug

32. Per esercizio, provare a togliere un punto e virgola per vedere come sono segnalati gli errori. L'errore viene segnalato ben prima della compilazione, sottolineandolo in rosso, come su LibreOffice...Ricordarsi di rimettere quanto si è cancellato...
33. Dopo aver compilato il programma senza errori, inserire un breakpoint facendo un click con il mouse a sinistra della riga numero 33 come in figura:



34. Avviare la sessione di debug (ricompilando e riseseguendo il programma)
35. Questa volta, oltre alla solita finestra, apparirà anche l'area dei messaggi di debug.
36. Cliccando sul programma il pulsante "Aggiungi..." si arriva al punto di breackpoint. Sono mostrate le chiamate di ogni funzione e le eventuali variabili in "watch list".