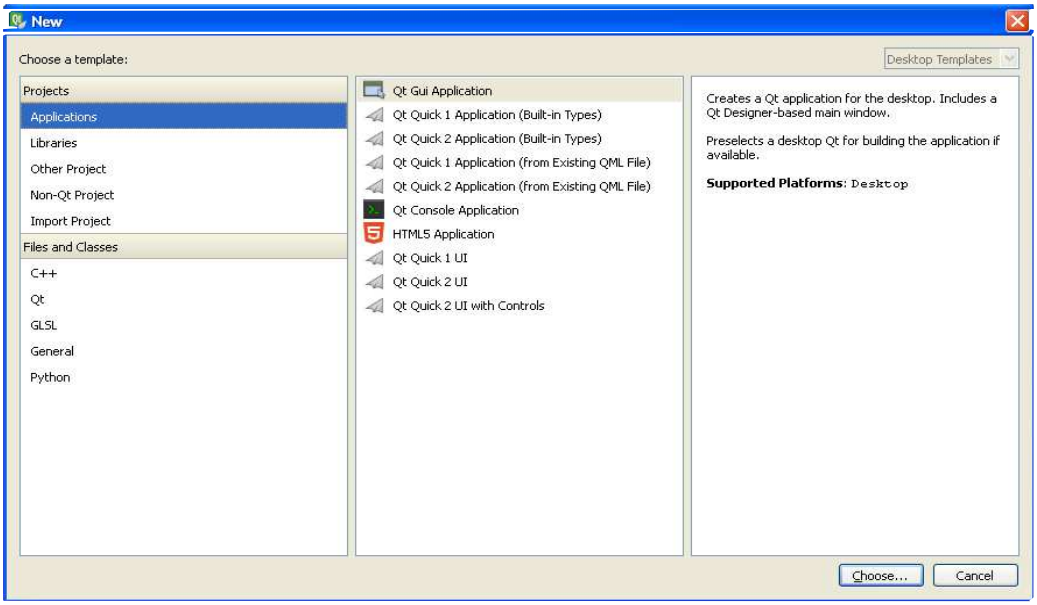
Fondamenti di Qt Creator

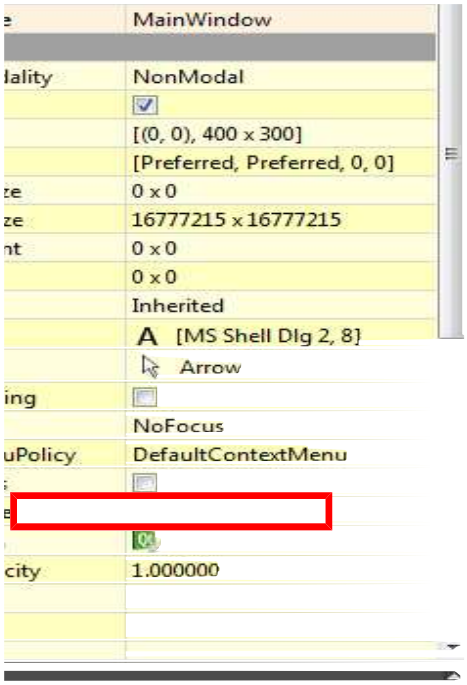
Come creare un programma

Aprire Qt Creator e selezionare Qt GUI Application e seguire le istruzioni.



Main Window

É la finestra principale, viene creata all'apertura del progetto, se si vuole cambiare il titolo alla finestra del programma nel pannello degli attributi si sceglie "Window title".



Label

Rappresenta del testo solitamente fisso oppure modificabile dal programma, quindi può essere

utilizzato come un controllo di output. Si trova nel container “display widgets”.

Per scrivere il suo contenuto si usa il metodo set Text.

Esempio:

```
ui->label->setText("ciao");
```

Per disegnare il bordo si va nel pannello delle proprietà e si sceglie frameShape.

Push Button

Il push button viene utilizzato per fare in modo che quando viene premuto vengano eseguite delle istruzioni date dal programmatore. Molto importante: quando viene inserito un nuovo controllo nella finestra (bottone , label, ecc) è una buona idea cambiare il nome del controllo per dargiene uno più significativo. La convenzione da noi adottata prevede che venga messo un prefisso, ad esempioo *btn* per i bottoni, *lbl* per le label, ecc. seguito da un nome significativo per l'applicazione. In questo modo il codice sarà più leggibile e autoesplicativo. Per cambiare il nome si clicca il tasto destro e si sceglie *Change Object name* (fig1) . In fig2 viene invece descritto come passare alla scrittura delle istruzioni dopo aver cliccato il bottone; tutto questo lo si fa cliccando il tasto destro e andando su *Go to slot...* e scegliendo la voce *clicked()*, questo solitamente perchè noi vorremo associare delle azioni alla pressione del bottone. Se invece avessimo voluto associare le azioni a qualche altro evento (il doppio click, il rilascio del pulsante, ecc.) avremmo scelto un altro slot.

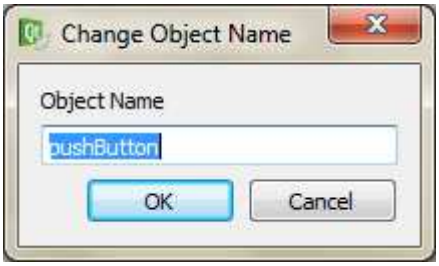


Figura 1.

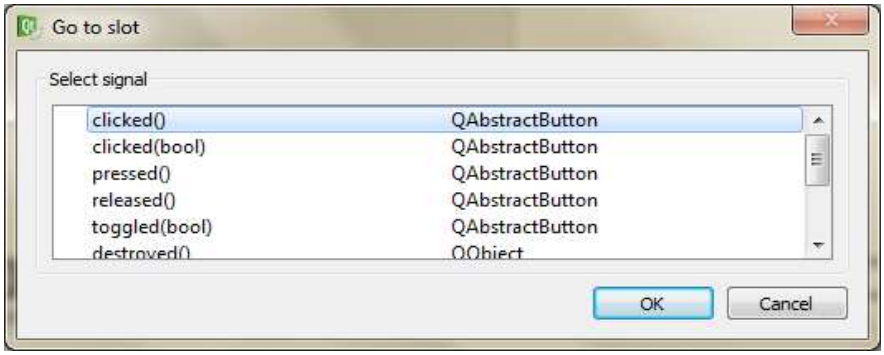


Figura 2.

Line Edit

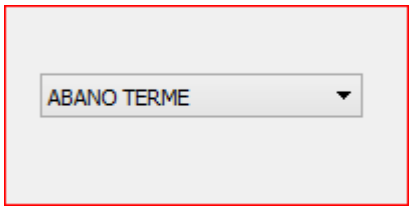
Il line edit + un controllo di input che viene usato per permettere all'utente di inserire un testo libero, di una lunghezza limitata, poiché si estende su una sola linea. Si trova nel container “Input Widgets”

Per leggere in valore contenuto al suo interno si usa il metodo

```
ui->lineEdit->text();
```

Combo Box

Permette all’utente di scegliere da un elenco di una serie di valori, l’utente seleziona un valore tra quelli in elenco. Si trova nel container “Buttons”.



Solitamente viene usato come campo di input, andando a leggere da programma quello che l'utente a selezionato in esecuzione.

Ci sono due metodi che possono essere usati per leggere la scelta dell'utente: il primo permette di leggere il testo contenuto e va usato in questo modo:

```
QString comune;
comune = ui->cmbComuni->currentText();
```

Nella stringa comune si troverà il valore scelto come come lo vede l'utente. L'altro metodo permette invece di leggere la posizione della scelta dell'utente, considerando che la prima voce del menù si trova in posizione 0, la seconda in posizione 1 e via così

```
int comune;
comune = ui->cmbComuni->currentIndex();
```

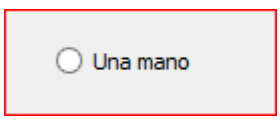
Dopo questa chiamata l'intero comune conterrà al suo interno l'indice della scelta fatta. Quale dei due metodi usare? Dipende da quello che si vuole fare, non esiste una regola: se è più comodo sapere il valore della scelta fatta dall'utente si sceglierà il primo, se invece interessa la posizione della scelta dell'utente si sceglierà il secondo.

Per inserire i valori nel menù si può procedere in due modi. Il primo prevede di usare l'interfaccia di QT Creator e fare doppio click sul combo che si vuole riempire: apparirà una finestra di immissione che permette di inserire i valori uno a uno. Se i valori sono pochi e non cambiano questo modo è adeguato, altrimenti lo si può fare da programma usando il metodo addItem in questo modo:

```
ui->cmbComuni->addItem(“Primo”);
ui->cmbComuni->addItem(“Secondo”);
ui->cmbComuni->addItem(“Terzo”);
```

Radio Button

Come il combo box è un controllo grafico che consente all'utente di effettuare una scelta singola in un insieme di opzioni predefinite. Si trova nel container “Buttons”.



Può essere attivato attraverso da codice tramite l'istruzione

```
ui->radProva->setChecked(true);
```

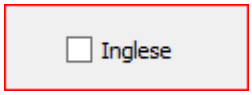
anche se molto più spesso viene usato come campo di input, per verificare se è selezionato o meno con il metodo isChecked:

```
if (ui->radProva->isChecked())
```

```
ui->lblProva->setText("Sono attivo");
```

Check Box

Ha un uso e una forma simile a quella del redio button, solo che permette di scegliere o meno un'opzione indipendentemente dal comportamento di altri check box presenti nella medesima finestra. Si trova nel container “Buttons”.

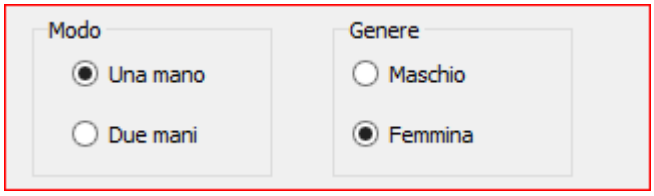


Sia per leggere che per scrivere si usano gli stesi metodi dei radio button

```
ui->checkBox->isChecked() //per leggere se è selezionato o no
ui->checkBox->setChecked()//per selezionarlo
```

Group Box

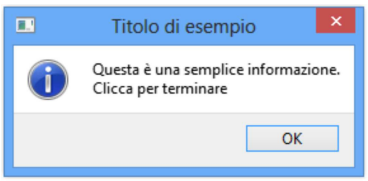
Il gruppo box è un controllo di tipo container, cioè serve per contenere al suo interno altri controlli (bottoni, label, ecc.). Nei nostri programmi è usato per separare gruppi di radio button in modo che non si influenzino a vicenda. Si trova nel container “Containers”



Message Box

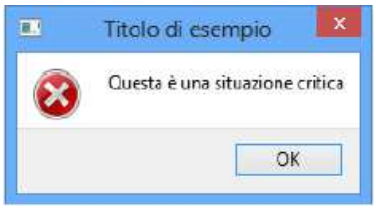
Sono delle finestre predefinite che vengono utilizzate per mostrare semplici messaggi all'utente o per chiedergli cosa vuole fare in particolari situazioni. Sono “modali”, cioè acquisiscono il focus dell'applicazione e non lo rilasciano finché non vengono chiuse, in modo da non permettere all'utente di ignorarle.

```
QMessageBox::information(this,"Titolo di esempio","Questa è una semplice
informazione.\nClicca per terminare");
```

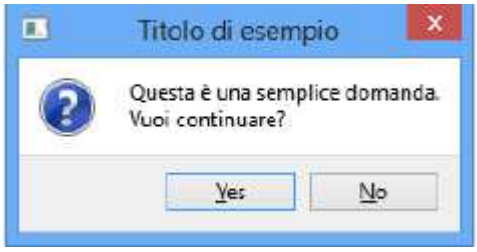



```
QMessageBox ::warning(this,"Titolo di esempio","Questo è un avvertimento");
```

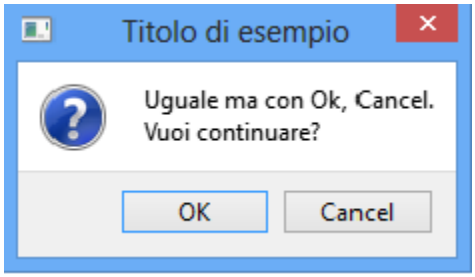

```
QMessageBox ::critical(this,"Titolo di esempio","Questa è una situazione
critica");
```




```
int ret = QMessageBox::question(this,"Titolo di esempio", "Questa è una semplice
domanda.\nVuoi continuare?");
if (ret == QmessageBox::Yes)
  QMessageBox::information(this,"Titolo di esempio","Hai premuto Si");
else
  QMessageBox::information(this,"Titolo di esempio","Hai premuto No");
```




```
int ret = QMessageBox::question(this,"Titolo di esempio","Uguale ma con Ok,
Cancel.\nVuoi continuare?",QMessageBox::Ok,QMessageBox::Cancel);
if (ret == QmessageBox::Ok)
  QMessageBox::information(this,"Titolo di esempio","Hai premuto Ok");
else
  QMessageBox::information(this,"Titolo di esempio","Hai premuto Cancel");
```

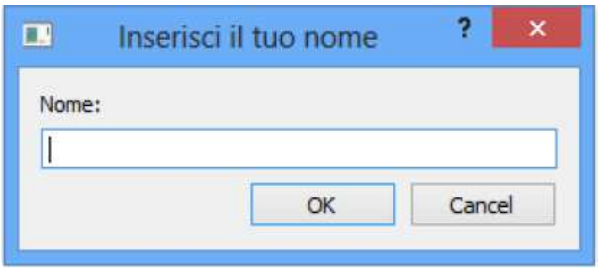



```
int ret = QMessageBox::question(this,"Titolo di esempio","Uguale ma con tre
pulsanti...",QMessageBox::Save | QMessageBox::Discard | QMessageBox::Cancel );
if (ret == QMessageBox::Save)
    QMessageBox::information(this,"Titolo di esempio","Hai premuto Save");
else if (ret == QMessageBox::Discard)
    QMessageBox::information(this,"Titolo di esempio","Hai premuto Discard");
else
    QMessageBox::information(this,"Titolo di esempio","Hai premuto Cancel");
```

Input Box

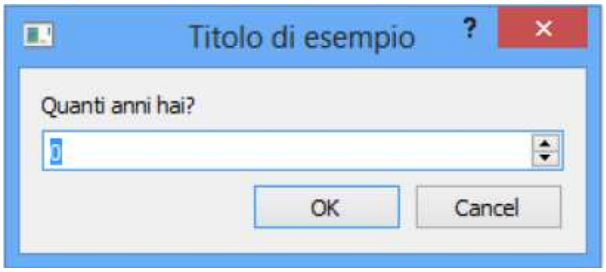
Per leggere stringhe di testo:

```
QString name = QInputDialog::getText(this,"Inserisci il tuo nome","Nome:");
if (!name.isEmpty())
    QMessageBox::information(this,"Titolo","Ciao "+name+" benvenuto.");
```



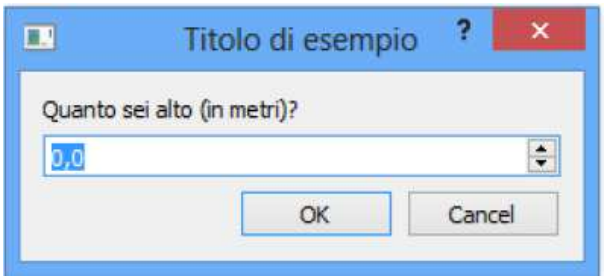
Per leggere un numero intero:

```
int iret = QInputDialog::getInt(this,"Titolo di esempio","Quanti anni hai?");
if (iret!=0)
   QMessageBox::information(this,"Titolo","Hai inserito "+QString::number(iret)
+" anni");
```



Per leggere un numero double:

```
double altezza = QInputDialog::getDouble(this,"Titolo di esempio","Quanto sei
alto (in metri)?");
if (altezza!=0)
    QMessageBox::information(this,"Titolo","Sei alto "+QString::number(altezza)
+" metri");
```



List Box

Per creare una ListBox selezionare il control **List Widget** dal pannetto *Item Widget (Item-Based)* e trascinarlo nell'area grafica della finestra principale.

Ora supponiamo che la variabile si chiami **listElenco**

- Riga corrente selezionata:
  int idx = ui->listElenco->currentRow();
  idx è l'indice della riga selezionata, oppure -1 se nessuna riga è selezionata
- Aggiungere una riga in fondo alla listBox
  ui->listElenco->addItem("nuova riga");
- Inserire una nuova riga in posizione indicata da idx:
  ui->listElenco->insertItem(idx,"riga da inserire");
- Rimuovere una riga alla posizione indicata da idx:
  ui->listElenco->model()->removeRow(idx);
- Cancellare tutte le righe:
  ui->listElenco->clear();
- Numero di righe della listBox:
  int n = ui->listElenco->count();
- Ottenere il testo della riga corrente:
  Qstring testo = ui->listElenco->currentItem()->text();
- Modificare il testo della riga corrente:
  ui->listElenco->currentItem()->setText("nuovo testo");

Intercettare gli eventi:

- Quando si cambia la riga selezionata nella listBox:
  itemSelectionChanged()

Come gestire le stringhe

Generalmente tutto il testo che appare in un' applicazione a finestra è una stringa, anche se apparentemente può sembrare un numero. Spesso c'è quindi la necessità di fare una conversione da stringa a numero e viceversa.

Conversione da stringa a numero:

Se s è una stringa allora si applica il metodo toInt(o altri a seconda del tipo). Spesso la stringa è il contenuto di un oggetto dell'interfaccia quindi un esempio realistico potrebbe essere:

```
ui->label->text().toInt();
```

Conversione da numero a stringa:

Se n è una variabile numerica allora si usa il metodo statico Qstring::number(n); Spesso ci serve mostrare il numero all'interno di un ogetto dell'interfaccia quindi un esempio realistico potrebbe essere :

```
ui->label->setText(QString::number(n));
```

Come concatenare due stringhe

L'operatore di concatenazione di due strighe è il "+"

Esempio:

```
QString a = "Ciao";
QString b = "Mondo";
QString c = a + b ;
```

Dopo queste istruzioni la variabile c conterrà la stringa "CiaoMondo"

Esempio:

```
QString a = "Ciao";
QString b = "Mondo";
int n = 3;
QString c = a + QString::number(n) + b ;
```

Dopo queste istruzioni la variabile c conterrà la stringa "Ciao3Mondo"