
Laboratori: Introducció a Qt

Professors de IDI Q2 - 13/14

Llibreria Qt

- Va ser creada per Trolltech i és actualment mantinguda en Creative Commons a qt-project.org i per a usos comercials per Digia a qt.digia.com.
 - Per a plataformes Windows, Linux, Mac i Android
-

Llibreria Qt

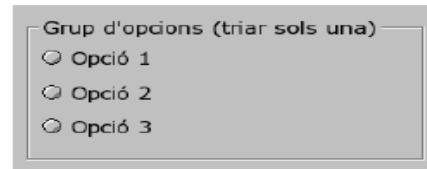
- Una llibreria en C++ per a dissenyar interfícies gràfiques d'usuari (GUI) en diferents plataformes.
- Proporciona diversos components atòmics (widgets) configurables.



(a)
Un
botó



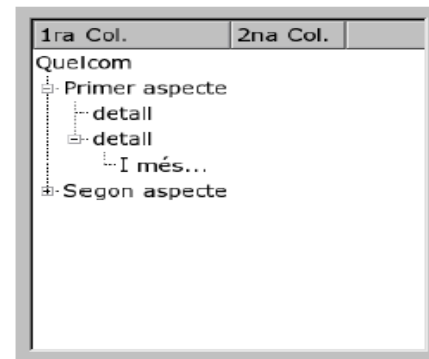
(b) Un "slider"



(c) "Radio buttons"



(d) Barra de "scroll"



(e) Llista jeràrquica

Llibreria Qt

Exemple: Hello Qt. Fitxer exemple.cpp

```
#include <QApplication>
#include <QPushButton>
int main(int argc, char **argv)
{
    QApplication a(argc,argv);
    QPushButton hello("Hello Qt!",0);
    hello.resize(100,30);
    hello.show();
    return a.exec();
}
```



Hello Qt!

Llibreria Qt

- Crear un fitxer `.pro` que conté la descripció del projecte que estem programant
 - Utilitzar les comandes `qmake` i `make`.
 - `qmake` genera el `Makefile` a partir del `.pro`
 - `make` compila i linca.
-

Compilar i linkar

- Crear un fitxer “*helloQT.pro*”

```
TEMPLATE = app
```

```
DEPENDPATH+=.
```

```
INCLUDEPATH+=.
```

```
#Input
```

```
SOURCES+=exemple.cpp
```

- Compilem i enllacem

```
qmake
```

```
make
```

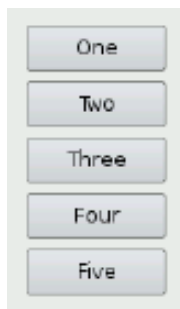
- Executable anomenat *helloQt* en el directori on estiguem.
- Executar-lo amb:
./helloQt

Llibreria Qt: Els Layouts

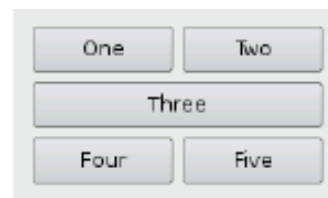
Els **layout** (disposicions) permeten organitzar els components visuals dintre de formularis i quadres de diàleg.



Horitzontal
(QHBoxLayout)



Vertical
(QVBoxLayout)



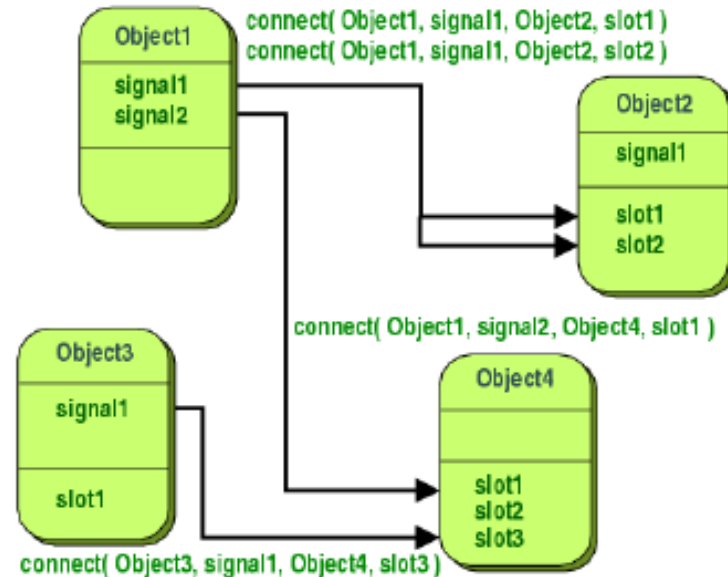
En graella
(QGridLayout)

Llibreria Qt: Signals i slots

- Per tal de connectar la interfície gràfica que dissenyem amb la nostra aplicació, caldrà connectar els elements gràfics Qt al nostre codi C++.
- Les connexions poden ser:
 - Alt nivell: associades als components
 - Baix nivell: events bàsics del computador
- **Signal:** Esdeveniment que succeeix durant l'execució de l'aplicació.
 - Ex: Clic sobre un widget...
- **Slot:** mètodes especials d'una classe que es poden connectar amb signals.

Llibreria Qt: Signals i slots

Els signals i els slots s'utilitzen per a la comunicació entre objectes. Qualsevol classe que hereti de QObject (o de les seves subclasses), pot contenir signals i slots.

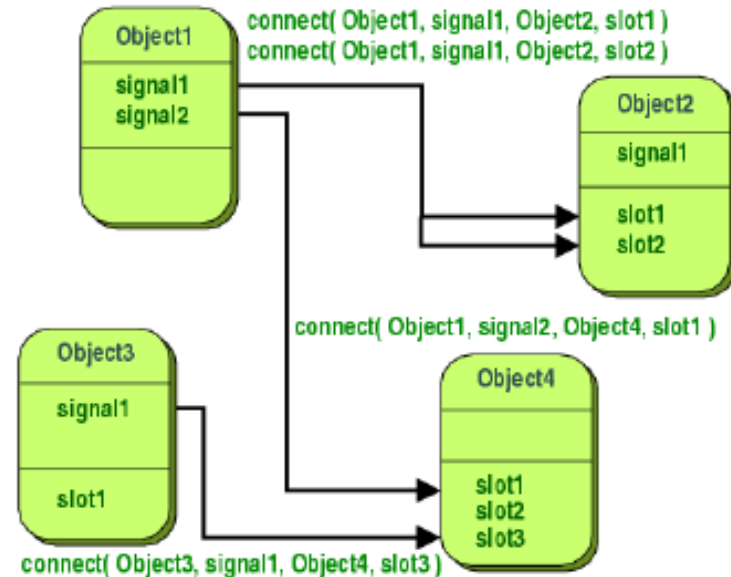


Tant signals com slots són mètodes.

Llibreria Qt: Signals i slots

Els signals es llancen quan es produeix un esdeveniment en l'aplicació.

El seu codi està preprogramat i no es pot canviar.



Els slots s'executen quan es produeix un signal. El seu codi pot ser implementat per nosaltres.

Podem fer que un signal es llanci també des del codi ([emit](#)).

Llibreria Qt: Signals i slots

Com es connecten Signals i Slots? **exemple.cpp**

```
#include <QApplication>
#include <QPushButton>
#include <QFrame>
int main(int argc, char **argv)
{   QApplication a(argc, argv);
    QFrame F(0, NULL);
    QPushButton *hello=new QPushButton("Hello Qt!", &F);
a.connect(hello, SIGNAL(clicked()), &F, SLOT(close()));
    F.show();
    return a.exec();
}
```

Podeu trobar un exemple més complet a:

</assig/idi/sessions/S1-IntroQt-Designer>

Llibreria Qt: Signals i slots

- La informació que circula entre signals i slots viatja a través dels paràmetres.
 - Els slots tenen paràmetres que venen carregats de dades, les que envia el signal.
 - Pot haver més d'un slot connectat a un mateix signal, de manera que quan s'emeti un signal, s'executaran tots els seus slots; no podrem saber, però, en quin ordre.
-

Llibreria Qt: Signals i slots

- En el directori

`/assig/idi/sessions/S1-IntroQt-Designer`

trobareu un fitxer `lab0.pro` i un `lab0.cpp`

- `lab0.pro` serveix per a descriure com és el vostre projecte: els fitxers que el componen, les llibreries que cal lincar...
 - Podeu executar-lo fent `./lab0`
-

Llibreria Qt

- En una aplicació complexa caldrà crear les nostres pròpies classes derivades de les de Qt per a programar els slots que calguin. Podem derivar de:
- `QObject` (per a objectes no gràfics)
- `QWidget` o les seves derivades (per a dissenyar nous components gràfics amb noves funcionalitats)

Example: MyLabel.h

```
#include <QLabel>

class MyLabel: public QLabel
{
    Q_OBJECT      ←----- IMPORTANT
public:
    MyLabel(QWidget *parent);
    ~MyLabel();

    public slots:    ←----- IMPORTANT

    void changeToRed();
    void changeToBlue();

    signals:         ←----- IMPORTANT

    void exempleSignal();
};
```

Els slots els implementarem a
MyLabel.cpp

Els signals no els implementem
però es poden llençar en
qualsevol punt del codi cridant a
la funció:

emit nom_signal(paràmetres)

Llibreria Qt

Compilem la classe MyLabel

No és codi C++.

Necessita ésser preprocessat amb el meta-object compiler (MOC):

Ho fa automàticament el Makefile si treballem amb .pro

Llibreria Qt

- Més informació on-line usant la comanda:
assistant&

