Fabio Falzoi
Tech lead

Integrazione C++ QML

ISIS Gobetti Volta
GENNAIO 2021

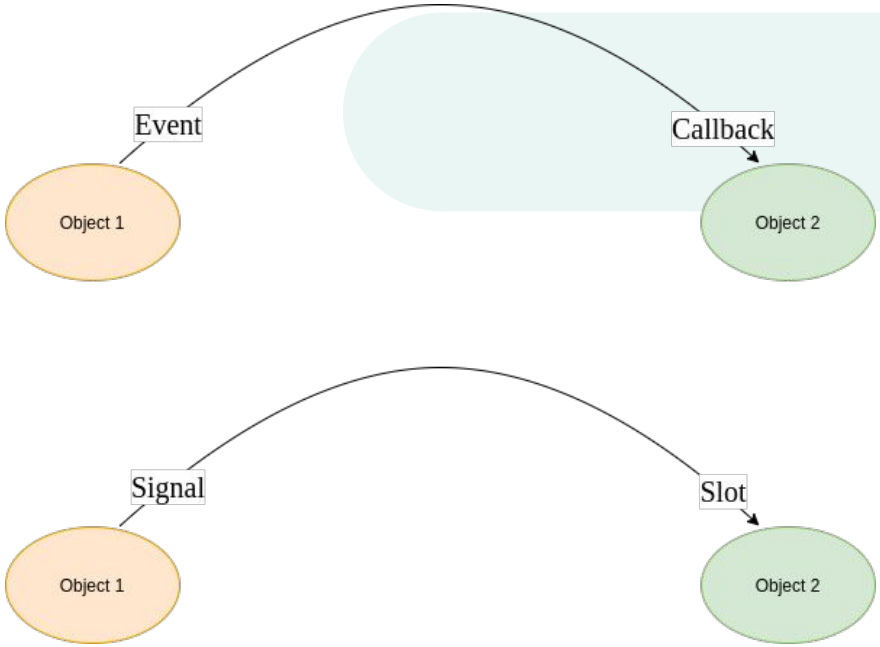
develer

LEZIONE 3 - PROGRAMMA

- Comunicazione UI (QML) -> C++
- Comunicazione C++ -> UI (QML)

Recap

- Una porzione di codice da eseguire per reagire ad un *evento*
- In Qt, gli eventi e le callback sono implementati mediante il meccanismo dei *signal* e degli *slot*



develer

C++ to QML

- Come esporre una proprietà di una classe C++ al QML?

```
// myclass.h

class MyClass : public QObject
{
   Q_OBJECT

   Q_PROPERTY(QString myproperty READ currentMyProperty NOTIFY myPropertyChanged)

public:
   explicit MyClass(QObject *parent = nullptr);

   QString currentMyProperty();

signals:
   void myPropertyChanged();

private:
   QString myProperty;
};
```

C++ to QML

- Si utilizza la macro *Q_PROPERTY* per dichiarare tipo e nome della property
- *QString myproperty*

```
// myclass.h

class MyClass : public QObject
{
   Q_OBJECT

   Q_PROPERTY(QString myproperty READ currentMyProperty NOTIFY myPropertyChanged)

public:
   explicit MyClass(QObject *parent = nullptr);

   QString currentMyProperty();

signals:
   void myPropertyChanged();

private:
   QString myProperty;
};
```


develer

C++ to QML

- Si “aggancia” una funzione membro da chiamare ad ogni lettura della property da QML
- *READ currentMyProperty*

```
// myclass.h

class MyClass : public QObject
{
   Q_OBJECT

   Q_PROPERTY(QString myproperty READ currentMyProperty NOTIFY myPropertyChanged)

public:
   explicit MyClass(QObject *parent = nullptr);

   QString currentMyProperty();

signals:
   void myPropertyChanged();

private:
   QString myProperty;
};
```

C++ to QML

- Si “aggancia” un segnale da emettere ad ogni modifica della proprietà (in C++), in modo che l’interfaccia sia forzata ad aggiornarsi
- *NOTIFY myPropertyChanged*

```
// myclass.h

class MyClass : public QObject
{
   Q_OBJECT

   Q_PROPERTY(QString myproperty READ currentMyProperty NOTIFY myPropertyChanged)

public:
   explicit MyClass(QObject *parent = nullptr);

   QString currentMyProperty();

signals:
   void myPropertyChanged();

private:
   QString myProperty;
};
```


develer

- L'implementazione del costruttore fornisce il valore iniziale della proprietà
- L'implementazione di *currentMyProperty* ritorna il valore corrente

```
// myclass.cpp

MyClass::MyClass(QObject *parent) : QObject(parent)
{
    myProperty = "initial value";
}

QString MyClass::currentMyProperty() {
    return myProperty;
}
```

C++ to QML

- Nel *main.cpp* dobbiamo istanziare un oggetto di classe *MyClass* ed esportarlo al QML, per rendere “visibile” la sua proprietà
- Il nome dell’oggetto, in QML, sarà *myClass*

```cpp
// main.cpp

int main(int argc, char *argv[])
{
#if QT_VERSION < QT_VERSION_CHECK(6, 0, 0)
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
#endif

    QGuiApplication app(argc, argv);

    QQmlApplicationEngine engine;
    const QUrl url(QStringLiteral("qrc:/main.qml"));
    QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,
                     &app, [url](QObject *obj, const QUrl &objUrl) {
        if (!obj && url == objUrl)
            QCoreApplication::exit(-1);
    }, Qt::QueuedConnection);
    engine.load(url);

    MyClass mc;
    engine.rootContext()->setContextProperty("myClass", &mc);

    return app.exec();
}
```

C++ to QML

- In QML potremo riferirci alla proprietà *myProperty* dell'oggetto di classe *myClass* in questo modo

```
// main.qml

Text {
    text: myClass.myproperty
}
```

QML to C++

- Come possiamo comunicare valori dall'interfaccia al C++?
- Usiamo un TextInput come esempio: vogliamo passare il test contenuto al C++, per aggiornare *myProperty*
- In QML, possiamo usare del codice JavaScript per inviare il nuovo valore ogni volta che il TextInput subisce un editing

```
// main.qml

TextInput {
    onEditingFinished: {
        myClass.updateMyProperty(this.text)
    }
}
```


develer

QML to C++

- *updateMyProperty* è uno slot pubblico che riceve il testo dalla UI

```
// myclass.h

class MyClass : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QString myproperty READ currentMyProperty NOTIFY myPropertyChanged)
public:
    explicit MyClass(QObject *parent = nullptr);

    QString currentMyProperty();

signals:
    void myPropertyChanged();

public slots:
    void updateMyProperty(QString newProperty);

private:
    QString myProperty;
};
```

QML to C++

- *updateMyProperty* è uno slot pubblico che riceve il testo dalla UI
- Nell'implementazione devo emettere il segnale *myPropertyChanged* per notificare la UI che il valore è cambiato!

```
// myclass.cpp

MyClass::MyClass(QObject *parent) : QObject(parent)
{
    myProperty = "initial value";
}

QString MyClass::currentMyProperty() {
    return myProperty;
}

void MyClass::updateMyProperty(QString newProperty) {
    myProperty = newProperty;
    emit myPropertyChanged();
}
```

Message Board

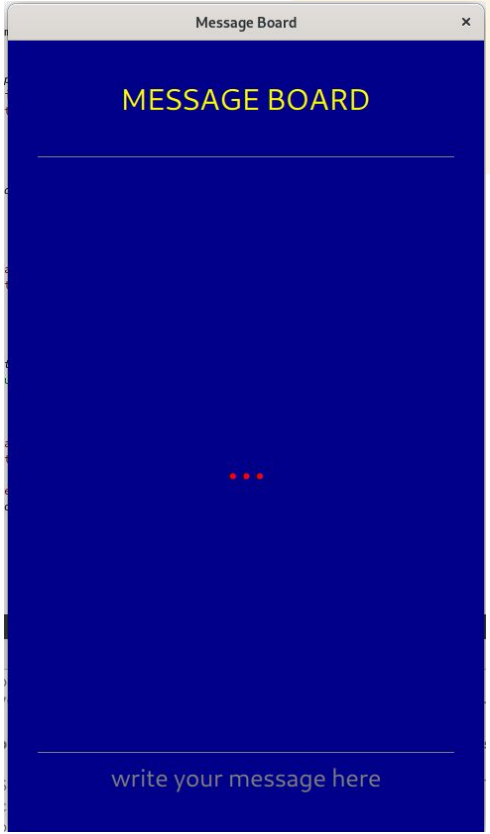
1. Implementate una message board dove ogni messaggio, scritto in basso nell'apposito spazio di input, compare nella board al centro

develer

Message Board

Interfaccia grafica

- Finestra 480x800, sfondo *dark blue*
- Titolo a 40 pixel dal bordo superiore
- Prima linea separazione a 40 pixel dal titolo
- Messaggio centrale "..." a 275 pixel dal bordo superiore della finestra
- Seconda linea separazione a 80 pixel dal bordo inferiore della finestra
- TextInput a 40 pixel dal bordo inferiore della finestra



develer

CONTACTS

Fabio Falzoi

fabio@develer.com

Twitter: @Pippolo84

www.develer.com