Framework Qt

POuL: Politecnico Open unix Labs

26 marzo 2012

1. Nella puntata precedente
   - C++
   - Qt

```
class Padre {
    ....
};

class Figlio : public Padre
{
public:
    Figlio(int value);
    ~Figlio();

    int get_value();
    void set_value(int value);

private:
    float mean(int a, int b);

    int value;
};
```

- Concetto di classe
- `private, public`
- Ereditarietà
- Costruttore e distruttore

Qt

```
class MyObject : public QObject
{
    Q_OBJECT

signals:
    void my_signal(int, int);
slots:
    void my_slot(int, int);
    void on_signal_do(QFile &file);

public:
    MyObject(QObject *parent = 0);
    ~MyObject();

    void my_function(QString, int);
};
```

- Eridaritarietà da QObject
- Q_OBJECT
- signals, slots

2. Qt Creator
   - Qt Designer
   - Demo

Qt Designer

- In passato Qt forniva uno strumento separato per la creazione di GUI chiamato Qt Designer. Oggi è direttamente integrato nell'IDE Qt Creator.
- Esistono ancora dei plugin per poter utilizzare questo potente strumento in Visual Studio oppure in Eclipse.

DEMO

3. Visualizzazione Dati
    - Visualizzazione dati
    - Item-Based vs Model-Based
    - Item-Based
    - Model-Based
    - Model-View-Delegate
    - Demo

Visualizzazione dati

Esistono principalmente 3 tipi di Widget per visualizzare un insieme di dati

- Lista (`QListView`, `QListWidget`)
- Tabella (`QTableView`, `QTableWidget`)
- Albero (`QTreeView`, `QTreeWidget`)
- (Su più colonne `QColumnView`)

Item-Based vs Model-Based

Esistono due categorie di Widget per visualizzare i dati nelle modalità elencate prima.

- Item-Based
- Model-Based

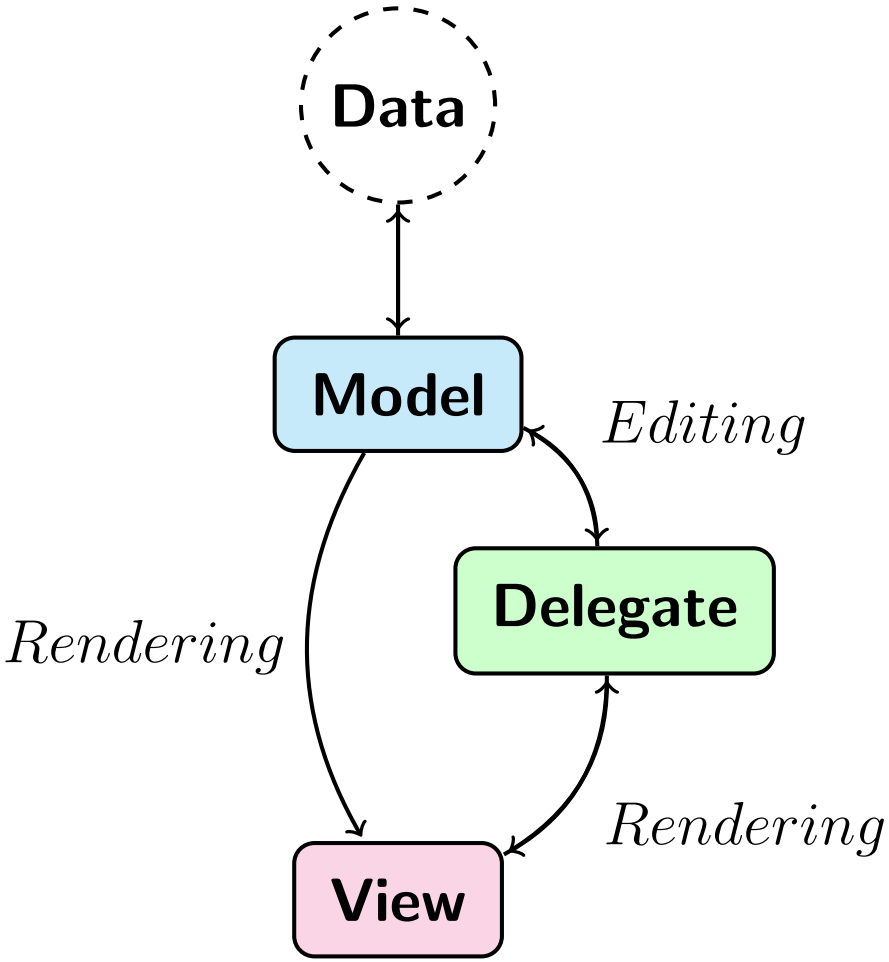
Item-Based

- I dati sono gestiti direttamente da ogni `QListWidget`, `QTableWidget` o `QTreeWidget`.
- Utile quando si devono visualizzare pochi dati in modo rapido.
- Non adatto a visualizzare dati provenienti da sorgenti di dati "esterne" (database, filesystem, ...) e in modo "creativo".

Model-Based

- Nelle classi Model-Based viene usato il pattern MVC per dividere le entità che gestiscono i dati (Model) da quelle che li visualizzano (View) a quelle che si occupano di catturare le interazioni dell'utente (Controlloller).
- Questo permette di avere una maggiore flessibilità nel gestire i dati e nel visualizzarli.
- Permette di separare il codice che legge i dati da quello che li visualizza (utile nel caso in cui leggere i dati può essere una cosa complessa).
- Qt utilizza una versione leggermente modificata del pattern MVC che ha chiamato Model-View-Delegate.

Model-View-Delegate



Model

Iteragisce con i dati fornendo un'interfaccia comune per leggerli e modificarli.

View

Visualizza in un determinato modo (lista, tabella, ...) i dati contenuti nel Model.

Delegate

Renderizza ogni singolo dato e si occupa dell'aggiornamento dei dati nel Model.

DEMO

4. Moduli QtNetwork e QtWebKit
    - QtNetwork
    - QtWebkit
    - Demo

QtNetwork

Modulo per la programmazione di rete. Fornisce un'interfaccia multipiattaforma per la l'accesso alla rete dai socket fino al protocollo `HTTP`.

Le principali funzionalità sono

- Socket `TCP`, `UDP` sia server che client (`QTcpSocket`, ...)
- Pieno supporto a `SSL` (tramite `OpenSSL`).
- Supporto protocollo `FTP`, `HTTP`
- Cache, Cookie, Proxy ...

QtWebKit

QtWebKit fornisce un motore di rendering web basato su WebKit (alla base di Google Chrome, Safari, ...).

Non stiamo parlando di un motore di rendering "giocattolo", sono supportate le seguenti funzionalità

- HTML5, CSS3, SVG, ...
- Javascript
- Web SQL Database, Web Storage
- Supporto per plugin esterni

DEMO