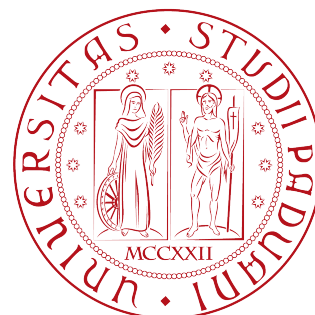


07/07/2014

# qDB

Progetto Programmazione ad oggetti

Alessandro Moretto  
MATRICOLA: 1049179





## Sommario

Introduzione e note di Sviluppo.....	2
Variabili di sviluppo .....	2
Logica del programma .....	2
Descrizione delle classi .....	2
Grafica del programma .....	3
Descrizione dei widget .....	4
MVC .....	4

## Introduzione e note di Sviluppo

QDB è un sistema minimale per la gestione di un piccolo database che permette di inserire, rimuovere, ricercare e modificare gli oggetti di cui il database è popolato. Inoltre il sistema permetterà il salvataggio e il caricamento dello stato del database al fine di evitare la perdita di dati importanti.

Gli oggetti che posso inserire possono essere Frigoriferi o Lavastoviglie con le loro caratteristiche tecniche.

Come pattern architetturale ho scelto MVC (Model View Controller) separando quindi la parte logica dalla parte grafica.

## Variabili di sviluppo

- Sistema Operativo: Windows 8.1 Professional, Ubuntu 12.04;
- **Compilatore://////////**
- Versione libreria Qt: 4.7.4 (32 bit), 4.8.0 (64 bit)

## Logica del programma

La “logica del programma” è quella parte di qDB deputata alla memorizzazione e all’elaborazione dei dati che vengono immessi dall’utente.

Questa parte è stata divisa in due sotto-problemi:

- 1) Definire un template di classe Container<K> i cui oggetti rappresentano un contenitore di oggetti di tipo K sui quali fare le varie operazioni;
- 2) Definire una gerarchia di classi composta da tre classi elettrodomestico (padre), Lavastoviglie e frigorifero che modellano una realtà di oggetti da gestire tramite il database. Definire inoltre una classe SmartElettrodomsticoPtr di puntatori smart alla classe base elettrodomestico (per poi popolare il contenitore con oggetti di SmartElettrodomsticoPtr).

## Descrizione delle classi

- Container<k>: è la parte centrale del progetto e determina la velocità di esecuzione delle varie operazioni e quindi è servito fare una scelta logica importante. La necessità principale è raggiungere il dato nella maniera più veloce possibile. Ho quindi optato per un contenitore basato su una struttura ad albero binario ordinato (Binary Search Tree) piuttosto che una lista dato mi permette di raggiungere un nodo con velocità media logaritmica al numero di oggetti presenti. L’albero non è bilanciato ma probabilisticamente con un numero alto di elementi (quando questa scelta farà la differenza) l’albero tende ad un bilanciamento perfetto essendo gli elementi inseriti “casualmente”. E nel peggior dei casi (inserimento ordinato di dati) avrò le prestazioni di una lista (ma ordinata => tempo medio  $O(\log_2 n)$ , tempo peggiore  $O(n)$  e tempo minore  $O(1)$ ). Inoltre ogni salvataggio e ripristino l’albero viene bilanciato (rendendo anche il tempo peggiore  $O(\log_2 n)$ ). Essendo che qDB non gestisce più database

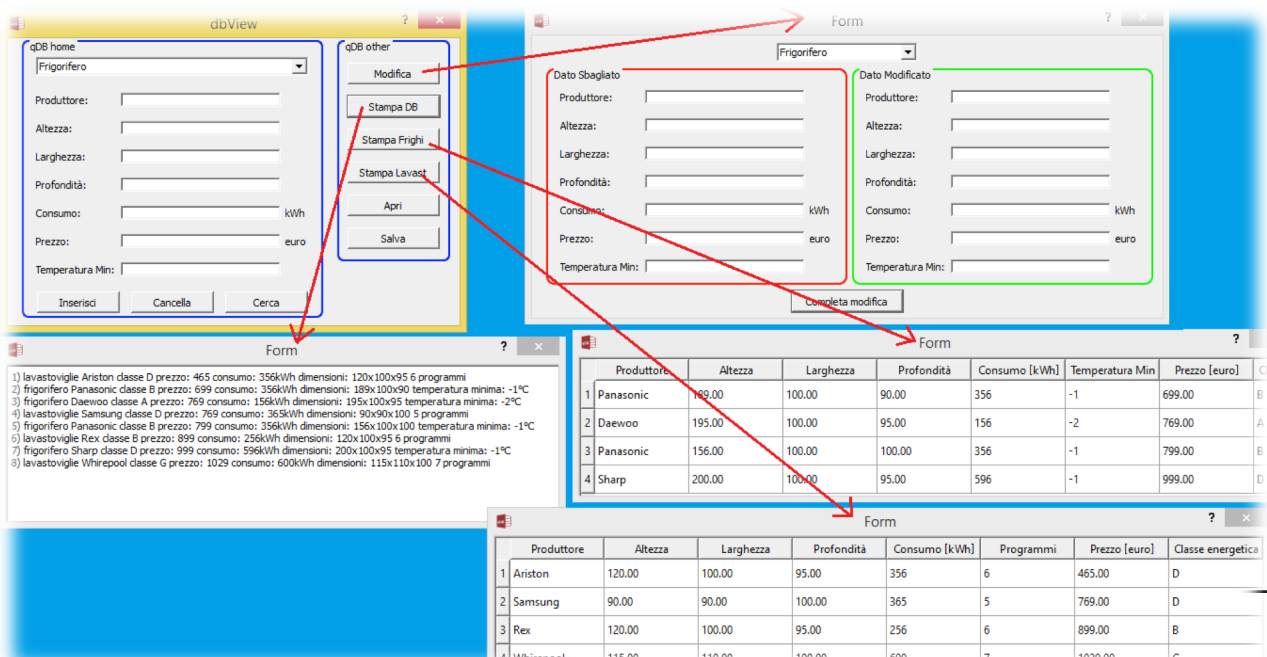
contemporaneamente, la struttura non prevede gestione controllata della memoria che provocherebbe uno spreco di velocità e memoria inutile (per quanto piccolo). Il Container, come già detto, è un template, ciò ci permette di utilizzarlo con qualsiasi tipo, ovviamente dovrà disporre di distruttore, assegnazione, costruttore di copia, operatori di uguaglianza e maggioranza.

Per il container è a disposizione anche un const\_iterator che permette di scorrere il contenitore ma non modificarlo (voglio lasciare tale compito alla funzione modifica perché modificare un valore “sul posto” comporterebbe ad un disordine della struttura).

- **Elettrodomestico**: è la classe astratta radice della gerarchia utilizzata. Contiene le caratteristiche di un generico elettrodomestico ovvero nome produttore, prezzo, altezza, larghezza, profondità ed infine consumo. La classe è dotata anche di un metodo virtuale puro (**virtual char classeRisparmioEnergetico() const**) che rende la classe astratta, ciò è necessario perché la classe risparmio energetico non si può sapere di un generico elettrodomestico ma cambia di tipo in tipo.
- **Lavastoviglie**: è una delle due classi figlie di elettrodomestico e aggiunge la caratteristica numero di programmi. Questa classe ridefinisce il la classe virtuale pura della radice della gerarchia (**virtual char classeRisparmioEnergetico() const**) e ne calcola la classe secondo i suoi parametri.
- **Frigorifero**: è una delle due classi figlie di elettrodomestico e aggiunge la caratteristica temperatura minima. Questa classe ridefinisce il la classe virtuale pura della radice della gerarchia (**virtual char classeRisparmioEnergetico() const**) e ne calcola la classe secondo i suoi parametri.
- **SmartElettrodomesticoPtr**: è la classe smart (che popolerà il container) che avrà come campo dati un puntatore polimorfo alla radice della gerarchia (elettrodomestico).

## Grafica del programma

La grafica di qDB è stata sviluppata utilizzando le librerie Qt 4.7.4. L'interfaccia è la seguente:



The screenshot displays the qDB application interface, which includes several windows for managing appliance data:

- dbView**: A window showing a list of appliances with their details. It includes a dropdown menu for selecting the appliance type (Frigorifero, Lavastoviglie) and buttons for adding, deleting, and searching for records.
- Form (Frigorifero)**: A window for editing or adding data for a refrigerator. It includes input fields for product name, price, height, width, depth, consumption, and temperature minimum. It also has a table for displaying the data.
- Form (Lavastoviglie)**: A window for editing or adding data for a dishwasher. It includes input fields for product name, price, height, width, depth, consumption, and number of programs. It also has a table for displaying the data.

The tables in the Form windows show the following data:

Prodotto	Altezza	Larghezza	Profondità	Consumo [kWh]	Temperatura Min	Prezzo [euro]	Classe energetica
1 Panasonic	169.00	100.00	90.00	356	-1	699.00	B
2 Daewoo	195.00	100.00	95.00	156	-2	769.00	A
3 Panasonic	156.00	100.00	100.00	356	-1	799.00	B
4 Sharp	200.00	100.00	95.00	596	-1	999.00	D

Prodotto	Altezza	Larghezza	Profondità	Consumo [kWh]	Programmi	Prezzo [euro]	Classe energetica
1 Ariston	120.00	100.00	95.00	356	6	465.00	D
2 Samsung	90.00	90.00	100.00	365	5	769.00	D
3 Rex	120.00	100.00	95.00	256	6	899.00	B
4 Whirepool	115.00	110.00	100.00	600	7	1029.00	G

## Descrizione dei widget

- **loadFieldElettro**: deriva da `QWidget`. È un widget composto da due `qLineEdit` e due `qLabel` sovrapposti e attraverso uno slot (`void showField(int value)`) si può scegliere quale accoppiata mostrare (coppia per 'Temperatura' o coppia per 'Numero Programmi').
- **dbView**: deriva da `qDialog`. È la dialog principale nella quale si può inserire, cancellare e cercare dati. Si può inoltre con un click aprire la dialog di modifica, stampare tutto il database, stampare tutte e sole le lavastoviglie o i frigoriferi ed infine salvare o ripristinare la nostra base di dati.
- **editForm**: deriva da `qDialog`. È la dialog che serve per modificare i dati già inseriti nel database. A sinistra si trova la zona di inserimento dei dati da sostituire precedentemente inseriti, mentre a destra il dato corretto da sostituire a quello vecchio.
- **outFrigoView**: deriva da `qDialog`. È la dialog che mostra in una tabella tutti i frigoriferi inseriti nel database.
- **outLavastView**: deriva da `qDialog`. È la dialog che mostra in una tabella tutte le lavastoviglie inserite nel database.
- **outputForm**: deriva da `qDialog`. È una dialog con un semplice `QTextEdit` nel quale viene stampato l'intero database in formato testuale.

## MVC

Come già detto, come pattern architetturale ho scelto MVC (Model View Controller). Ho deciso di attuare una netta ed evidente separazione con tre classi:

- **Model**: `dbModel`;
- **View**: `dbView` (già descritta sopra);
- **Controller**: `dbController`.

Questa scelta mi permette di dividere completamente la parte logica da quella grafica rendendo molto più facile una futura manutenzione del software.

Ciò che permette la 'connessione' tra parte logica e grafica è il controller che attraverso un sistema di signal e slot permette l'interazione tra le parti.