Pemrograman Qt 2 – Mendayagunakan QGroupBox dan QHBoxLayout untuk Membentuk Kolom dalam QDialog

Bismillahirrahmanirrahim.

Tulisan ini adalah bentuk PDF dari http://malsasa.wordpress.com/2013/07/05/pemrograman-qt-2-mendayagunakan-qgroupbox-dan-qhboxlayout-untuk-membentuk-kolom-dalam-qdialog/.

Setelah kegiatan ngite kemarin, ternyata saya melanjutkan lagi dengan memodifikasi program yang sudah ada. Modifikasi yang saya lakukan adalah mengatur tampilan item-item agar terbagi jadi 2 kolom (atau lebih) dalam satu dialog. Saya juga mengusahakan tampilan agar ada spasi antara dua kolom. Modifikasi ini dilandasi rasa ingin tahu saya akan bisa atau tidaknya Qt membuat program GUI yang benar-benar sesuai keinginan saya. Ternyata bisa dan itu mudah. Saya pun ingin mencatat apa yang telah saya temukan dengan Qt ini di posting-posting mendatang. Semoga ini bermanfaat untuk kaum muslimin sekalian.

TIPS LATIHAN MEMROGRAM

1. Jangan ragu untuk Ctrl+N alias membuat proyek baru.
2. Gunakan F1 di dalam Qt Creator untuk mencari kode-kode yang berhubungan dengan salah satu kelas. Misalnya, ketika tidak tahu bisa atau tidaknya QFrame melakukan method addWidget dan bagaimana penulisan kodenya, cari saja laman dokumentasi QFrame. Lalu cukup Ctrl+F > addwidget di dalam laman tersebut.
3. Cara membaca dokumentasi internal Qt Creator adalah fokuskan penglihatan pada kode-kode yang paling penting saja. Misalnya, saya mencari bisa atau tidaknya suatu layout diatur ukuran margin-nya. Gampang, cari saja QLayout lalu Ctrl+F > margin. Di situ pasti kita temukan salah satu method penting milik QLayout: void setContentsMargins (int left, int top, int right, int bottom). Fokus pada kode ini, pada isi argumennya (itu lho, isi dari ()). Qt itu terdokumentasi dengan jelas, sampai nama fungsinya saja sudah jadi dokumentasi sendiri. Artinya setContentsMargins jelas adalah mengatur margin untuk konten dari QLayout. Nah, int left dan int top ini jelas-jelas untuk mengatur ukuran secara piksel sisi kiri dan sisi atas, begitu pula untuk right dan bottom. Nanti penulisan kodenya adalah objekQLayout.setContentsMargins(10, 10, 10 10); kalau kita ingin berikan setiap sisi 10 piksel ruang kosong. Seperti inilah cara membaca dokumentasi pemrograman di Qt (dan insya Allah sama untuk dokumentasi API lainnya).
4. Googling sedikit atau seperlunya saja, karena dokumentasi internal Qt Creator sudah sangat lengkap. Hematlah bandwidth Anda.
5. Tip dari semua ahli pemrograman: ambil kode orang lalu modifikasi dan lihat hasilnya. Ulangi.

Wujud Program 1

Kode Program

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include

Dialog::Dialog()
{
//DEKLARASI LAYOUT
QVBoxLayout *tinyLayout = new QVBoxLayout;
QVBoxLayout *tonoLayout = new QVBoxLayout;
QHBoxLayout *mainLayout = new QHBoxLayout;
QGroupBox *gbkiri  = new QGroupBox("KIRI");
QGroupBox *gbkanan = new QGroupBox("KANAN");

//DEKLARASI BUTTON
QPushButton *tombol = new QPushButton("TOMBOL");
QPushButton *timbil = new QPushButton("TIMBIL");
QPushButton *tumbul = new QPushButton("TUMBUL");
QPushButton *tembel = new QPushButton("TEMBEL");

//MEMASUKKAN TOMBOL KE DALAM LAYOUT TINY
tinyLayout->addWidget(tombol);
tinyLayout->addWidget(timbil);

//MENGATUR UKURAN MARGIN KIRI KANAN ATAS BAWAH UNTUK LAYOUT
//tinyLayout->setContentsMargins(1,1,55,1);

//MEMASUKKAN LAYOUT TINY KE DALAM GROUPBOX KIRI
gbkiri->setLayout(tinyLayout);

//MEMASUKKAN TOMBOL KE DALAM LAYOUT TONO
tonoLayout->addWidget(tumbul);
tonoLayout->addWidget(tembel);

//MEMASUKKAN LAYOUT TONO KE DALAM GROUPBOX KANAN
gbkanan->setLayout(tonoLayout);

//MEMASUKKAN KEDUA GROUPBOX KE DALAM LAYOUT HORIZONTAL
mainLayout->addWidget(gbkiri);
mainLayout->addWidget(gbkanan);

//MEMASUKKAN LAYOUT INDUK KE DALAM DIALOG TERTINGGI
setLayout(mainLayout);

setWindowTitle("Demo Kolom yang Dibentuk dengan Groupbox dan HBoxLayout");
this->setSizeGripEnabled(1);

}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include

class QDialog;

class Dialog : public QDialog
{
Q_OBJECT

public:
Dialog();

private:

};

#endif // MAINWINDOW_H
```

main.cpp

```
#include <QtGui/QApplication>
#include "mainwindow.h"

int main(int argc, char *argv[])
{
QApplication a(argc, argv);
Dialog w;
w.show();

return a.exec();
}
```

Analisis Kode Program 1

Sebelum kode, sebaiknya saya jelaskan logikanya dulu. Sangatlah mudah dipahami urut-urutannya.

1. Ada tombol.
2. Ada Vertical Box Layout.
3. Ada GroupBox. Namanya KANAN.
4. Tombol masuk ke Vertical Box Layout.
5. Layout masuk ke Groupbox.
6. Ulangi langkah 1 – 5. Bedanya, beri nama KIRI.
7. Ada Horizontal Box Layout.
8. Groupbox KANAN masuk ke Horizontal Box Layout.
9. GroupBox KIRI masuk ke Horizontal Box Layout juga.
10. Ada dialog.
11. Layout masuk dialog.
12. Jadilah program 1.

Nah, kode-kode di atas hanyalah menerjemahkan bentuk urut-urutan ini ke dalam perintah yang dipahami komputer. Jadilah program 1. Ingat, yang paling penting dalam pemrograman program 1 ini hanyalah mainwindow.cpp. Yang lainnya hampir-hampir tidak berbeda dengan pertama di-generate oleh Qt Creator. Yang paling sederhana justru mainwindow.h, karena bukannya ditambah isinya melainkan dihapusi sehingga jadi lowong. Oke, saya mulai catatkan kode-kode yang penting dari mainwindow.cpp.

```
Dialog::Dialog()
```

Apa ini? Ini adalah fungsi dasar mainwindow.cpp. Bedakan dengan fungsi utama di main.cpp. Fungsi ini adalah fungsi yang dipanggil kala Dialog dibuka. Jadi, apa pun perintah kita di dalamnya, akan ikut dijalankan ketika Dialog dipanggil. Sebagai pelengkap, nanti di dalam mainwindow.cpp bisa ada fungsi-fungsi lain selain Dialog ini. Misalnya, fungsi yang disandarkan kepada tombol, agar jika tombol diklik maka sesuatu yang ditulis dalam fungsi tersebut berjalan.

```
QVBoxLayout *tinyLayout = new QVBoxLayout;
QVBoxLayout *tonoLayout = new QVBoxLayout;
QHBoxLayout *mainLayout = new QHBoxLayout;
QGroupBox *gbkiri  = new QGroupBox("KIRI");
QGroupBox *gbkanan = new QGroupBox("KANAN");
```

Kode di atas ini tujuannya untuk membuat objek layout dan groupbox. Karena saya ingin membuat satu program dengan dua blok tombol, satu kanan satu kiri, maka saya harus punya satu layout horizontal. Maka dari itu objek dari QHBoxLayout saya buat sebagai mainLayout. QVBoxLayout saya turunkan jadi objek karena masing-masing blok harus memiliki layout-nya sendiri. Dan setiap blok, diangkut oleh QGroupBox. Perhatikan, kita bisa langsung menyematkan judul KANAN dan KIRI pada GroupBox dengan memasukkannya sebagai string dalam passing parameter.

```
tinyLayout->addWidget(tombol);
tinyLayout->addWidget(timbil);
```

Kode di atas ini adalah perwujudan dari logika nomor 4. Harus seperti ini karena nanti tombol yang empat dipecah 2 untuk KANAN dan 2 untuk KIRI. Ini langkah pertama.

```
gbkiri->setLayout(tinyLayout);
```

Kode di atas ini perwujudan dari logika nomor 5. Beginilah caranya memasukkan tombol/objek GUI lain ke dalam GroupBox. Yang dimasukkan adalah layout yang memuat mereka. Otomatis objek yang dimuat ikut masuk ke dalam GroupBox.

```
mainLayout->addWidget(gbkiri);
mainLayout->addWidget(gbkanan);
```

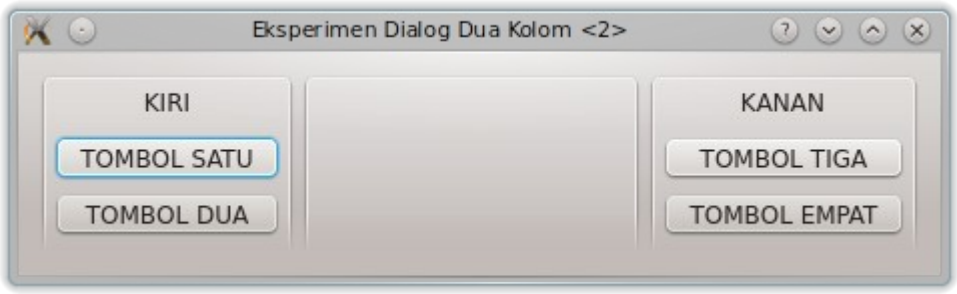
Kode di atas ini perwujudan dari logika nomor 8 dan 9. Ingat, tadi mainLayout berformat Horizontal. Dan format horizontal inilah yang penting untuk membentuk kolom KIRI dan KANAN. Enaknya layout horizontal ini, kita cukup addWidget ke dalamnya berapa kali pun, dengan widget apa saja (dalam program saya ini, GroupBox), maka pasti otomatis terbentuk kolom ke arah kanan. Karena itulah dia dinamakan Horizontal (Horizontal Box layout). Ingat, otomatis hanya dengan layout.

```
setLayout(mainLayout);
```

Kode di atas ini adalah method. Kita tahu, semestinya suatu method di dalam pemrograman OOP harus disandarkan pada objek. Misalnya tadi, objeknya mainLayout methodnya addWidget maka bentuk kodenya mainLayout->addWidget(). Bagaimana dengan ini? Apakah beda? Tidak, ini sama.

Hanya saja, kelasnya tidak perlu dituliskan. Mengapa? Karena kalau ditulis mutlak seperti ini, berarti method ini disandarkan pada kelas tertinggi dalam program ini. Apakah itu? Itulah Dialog. Berarti, method ini berlaku kepada dialog. Apa yang diberlakukan? Ya, memasukkan mainLayout ke dalam Dialog. Inilah perwujudan logika nomor 11. Dengan ini, sudah sempurnalah kode kita dan jadilah dialog kita dengan dua kolom, masing-masingnya ada 2 tombol.

Wujud Program 2



Kode Program

Kalau yang kedua ini, sama dengan yang pertama tetapi hanya sedikit perbedaan. Jadi cukup satu berkas saja.

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QtGui>

Dialog::Dialog()
{
    QVBoxLayout *layoutKontainerKanan = new QVBoxLayout;
    QVBoxLayout *layoutKontainerKiri  = new QVBoxLayout;
    QVBoxLayout *layoutKontainerTengah = new QVBoxLayout;
    QHBoxLayout *mainLayout = new QHBoxLayout;

    QGroupBox *gbkiri = new QGroupBox(tr("KIRI"));
    QGroupBox *gbkanan = new QGroupBox(tr("KANAN"));
    QGroupBox *gbtengah = new QGroupBox;

    //EALAH, TERNYATA CUKUP DITULIS QSizePolicy::Fixed [BISA GANTI Fixed
DENGAN METHOD LAIN]
    //TIWAS DIGOLEKI LAN DIJAJAL SAMPEK MUMET
    //KATA KUNCI GOOGLE: QSpacerItem *spasi = new
QSpacerItem(55,55,Fixed,Fixed);
    //TIDAK PERLU DEKLARASI enum DI HEADER SAMA SEKALI
    QSpacerItem *spacer = new
QSpacerItem(155,1,QSizePolicy::Fixed,QSizePolicy::Fixed);

    QPushButton *butonsatu  = new QPushButton("TOMBOL SATU");
    QPushButton *butondua   = new QPushButton("TOMBOL DUA");
    QPushButton *butontiga  = new QPushButton("TOMBOL TIGA");
    QPushButton *butonempat = new QPushButton("TOMBOL EMPAT");

    //MENGISI OBJEK KE DALAM KONTAINER DAN GROUPBOX KANAN
    layoutKontainerKanan->addWidget(butonsatu);
    layoutKontainerKanan->addWidget(butondua);

    gbkiri->setLayout(layoutKontainerKanan);

    //TENGAAAAAAH
    layoutKontainerTengah->addSpacerItem(spacer);
    gbtengah->setLayout(layoutKontainerTengah);

    //MENGISI OBJEK KE DALAM KONTAINER DAN GROUPBOX KIRI
    layoutKontainerKiri->addWidget(butontiga);
    layoutKontainerKiri->addWidget(butonempat);

    gbkanan->setLayout(layoutKontainerKiri);

    mainLayout->addWidget(gbkiri);
    mainLayout->addWidget(gbtengah);
```

```
    mainLayout->addWidget(gbkanan);

    //FINISHING ALL LAYOUT
    setLayout(mainLayout);
    setWindowTitle("Eksperimen Dialog Dua Kolom");

}
```

Cukup perhatikan saja kode di atas pada baris-baris yang saya hightlight. Cuma itu perbedaannya dengan program sebelumnya. Hasil kode di atas adalah satu kolom baru di tengah-tengah blok KANAN dan KIRI. Apa yang spesial di program kedua ini?

```
QSpacerItem *spacer = new
QSpacerItem(155,1,QSizePolicy::Fixed,QSizePolicy::Fixed);
```

Jika kita belum mengenal yang namanya kelas, sebaiknya memulai di sini. Cukup tahu dulu saja kalau QSpacerItem ini dinamakan kelas. Nah, spacer di sini dinamakan objek (anak kelas). Di dalam pemrograman OOP mana pun, sebuah kelas yang sedang membentuk objek, bisa menerima parameter-parameter tertentu dengan melalui passing parameter. Di mana letak passing parameter itu? Sebelumnya, pahami dulu yang namanya pembuatan objek itu:

```
NamaKelas *namaobjek = new Namakelas(passingparameter);
```

Kode di atas adalah bentuk umum deklarasi pembuatan objek (penurunan kelas). Anda mau lari ke mana pun dari kode Qt framework (atau misal bahasa pemrograman OOP mana saja), tidak akan bebas dari deklarasi macam ini. Jika sudah mengerti keberadaan deklarasi objek ini, maka sekarang waktunya membahas bentuk umum deklarasi objek khusus untuk QSpacerItem saja. Ingat, khusus QSpacerItem saja karena setiap kelas dalam Qt memiliki model deklarasinya sendiri-sendiri.

```
QSpacerItem *objectName = new QSpacerItem( int w, int h,
QSizePolicy::Policy hPolicy = QSizePolicy::Minimum, QSizePolicy::Policy
vPolicy = QSizePolicy::Minimum );
```

Perhatikan selalu passing parameter, yakni isi dari tanda kurung. Itulah bentuk umum dari deklarasi objek khusus QSpacerItem. Apa gunanya? Banyak sekali. Di antaranya, dengan mengganti tulisan int w dengan angka 155, maka objek saya bernama spacer pasti memiliki jarak lebar horizontal 155 piksel. Sangat berguna, bukan? Yang penting lagi, perhatikan kode QSizePolicy::Policy hPolicy = QSizePolicy::Minimum. Kode ini bisa kita ganti dengan QSizepolicy::Fixed begitu saja. Mengapa? Karena di dalam dokumentasi asli Qt, disebutkan kode-kode yang bisa dipakai untuk dipasang bersama QSizePolicy di antaranya Fixed, Minimum, Maximum, dan lain-lain. Jadi, bisa pula diganti dengan QSizePolicy::Minimum atau QSizePolicy::Maximum dan yang lainnya. Baca dokumentasi internal Qt Creator pada bab QSizePolicy Class Reference. Seperti demikianlah cara mempergunakan kode di dalam dokumentasi tersebut. Yap, saya sendiri sudah merasakan bingungnya tetapi akhirnya bisa memakainya dan di sini saya catatkan. Oya, kita belum terlalu perlu untuk tahu detail apa maksudnya Fixed, Maximum, Minimum, dan lain-lain. Cukup bisa memakainya dahulu saja. Oleh karena itulah, kode deklarasi saya menjadi:

```
QSpacerItem *spacer = new
QSpacerItem(155,1,QSizePolicy::Fixed,QSizePolicy::Fixed);
```

persis seperti di atas. Nah, kemudian, sama seperti sebelumnya. Masukkan spacer ini (sebagai objek, sebagai widget yang bisa mengisi layout) ke dalam layout. Kodenya:

```
layoutKontainerTengah->addSpacerItem(spacer);
```

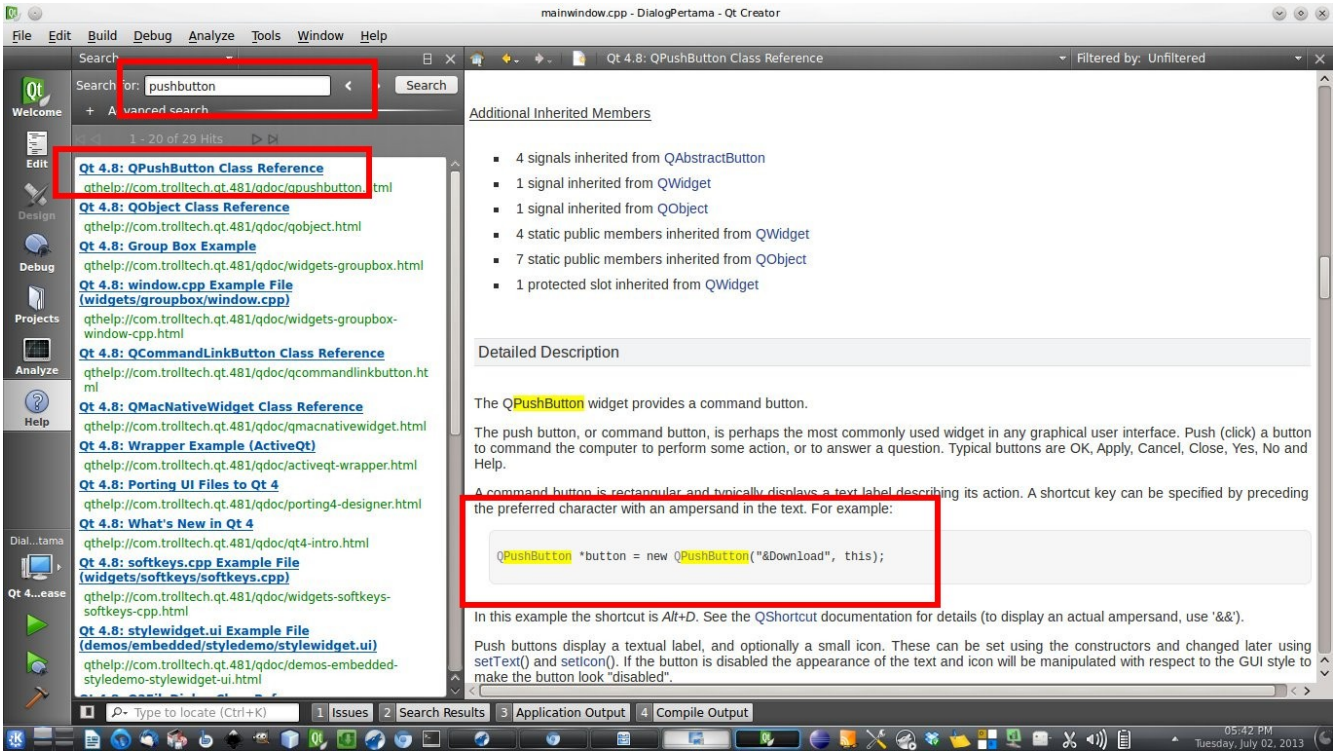
Kita perlakukan objek spacer ini sama seperti objek-objek lainnya (seperti PushButton juga) yakni dimasukkan ke dalam layout. Lalu, layout dimasukkan ke dalam GroupBox. Kodenya:

```
gbtengah->setLayout(layoutKontainerTengah);
```

Akhirnya selesai kodenya dan jadilah program kedua.

Rangkuman

1. Kelas-kelas Qt yang kita pergunakan dalam kedua program awal ini adalah:
   - QDialog
   - QVBoxLayout
   - QHBoxLayout
   - QGroupBox
   - QPushButton
   - QSpacerItem
2. Deklarasi pembuatan objek dari kelas di dalam Qt secara umum adalah:
3. NamaKelas *namaObjek = new NamaKelas(passing parameter);
4. Contoh deklarasi pembuatan objek dari QSpacerItem adalah:
5. QSpacerItem *spacer = new QSpacerItem(155, 1, QSizePolicy::Fixed, QSizePolicy::Fixed);
6. Belajar pemrograman C++ dalam Qt framework bisa dilakukan cukup dengan membaca dokumentasi Qt dalam Qt Creator saja. Tentu dengan sedikit googling.



Pembuatan kode program GUI di dalam Qt dilakukan secara berurutan menurut aturan yang berlaku. Widget dimasukkan ke dalam layout, lalu layout dimasukkan ke dalam kontainer tertinggi (bisa Dialog bisa Window). Atau, widget dimasukkan ke dalam layout, layout dimasukkan ke dalam groupbox, lalu groupbox dimasukkan dalam layout tertinggi, lalu layout tertinggi itu dimasukkan ke dalam kontainer tertinggi (Dialog atau Window).