QT Eğitimi - 2 :Yerleşim (Layout)

Önder Arslan

<onder@xcoders.net>

Sürüm 1.0

Özet

Bu belge qt öğrenimi için bir dizi şeklinde hazırlanmıştır. C++ bilmek ön koşuldur.

Giriş

QT eğitimine yerleşim (layout) 'lerle devam ediyoruz. Aslında bunu anlatmanın sanırım en kolay yolu beş-on tane QPushButton oluşturup bunları bir şekilde yerleştirmek olmalı ama biz her zaman olduğu gibi zor yolu seçeceğiz. Her örnekte biraz daha fazla kontrol kullanıp bunların da birkaç özelliğini öğrenmek ufkumuzun açılmasına biraz daha yardımcı olucak diye düşünüyorum. Önceki derste kullandığım kontroller ve özellikleri konusunda fazla açıklama yapmayacağım. Anlayamadığınız yerde birinci derse dönebilirsiniz.

Yerlesim

QT4 'te yerlesim (layout) sistemi oldukça güçlü ve kolay. Kontrollerinizi yerleştirmek için kullanabileceğiniz üç tane yerleşim sınıfı (layout classes) var : yatay , dikey ve ızgara yerleşim sınıfları (horizontal, vertica, grid layout classes). Yine adeti bozmadan sınıf bildirimimizle başlıyoruz :

```
/*--- form.h ---*/

#ifndef FORM_H
#define FORM_H

#include <QtGui>

class Form : public QDialog
{
   Q_OBJECT
   public:
      QPushButton *btnMesaj;
      QPushButton *btnAc;
      QPushButton *btnTemizle;
      QPushButton *btnCik;

      QLabel *lblMesaj;
      QLabel *lblAc;
      QLabel *lblTemizle;

      QTextEdit *yazi;

      QDirModel *model;
      QTreeView *treeDosyalar;
```

```
    Form();

  public slots:
    void mesajGoster()
    {
      QMessageBox::information(this, "Mesaj", "Layer 'lar o kadar da zor degilmis
:)");
    }

    void dosyaAc()
    {
      QString dosyaYolu = QFileDialog::getOpenFileName(this);
      QFile dosya(dosyaYolu);
      dosya.open(QFile::ReadOnly | QFile::Text);
      QTextStream ts(&dosya);
      yazi->setPlainText(ts.readAll());
    }

    void treeAc(const QModelIndex qm)
    {
      QDirModel *yol= new QDirModel();
      QFile dosya_(yol->filePath(qm));
      dosya_.open(QFile::ReadOnly | QFile::Text);
      QTextStream ts_(&dosya_);
      yazi->setPlainText(ts_.readAll());
    }

};

#endif
```

form.h:

İlk olarak yine başlık (header file) dosyamıza include ön işlemci komutuyla gerekli modülleri ekliyoruz. **QtGui** modülü qt 'deki görsel kontrolleri barındırdığı için "merhaba dünya" örneğinde olduğu gibi tüm kontrolleri tek tek include önişlemci komutuyla dahil etmemize gerek kalmıyor. Hemen altında form sınıfımızı tanımlıyoruz ve public türetmesi ile **QDialog** 'dan türetiyoruz. QDialog kullanıcı arabirimimizin tabanı diyebiliriz. Kontrollerimizi bunun üstünde oluşturuyoruz, windows 'ta form, java 'da jframe gibi ;) Hemen altında Q_OBJECT makromuzu ve yapıcı fonksiyon (constructer) ile kullanıcağımız kontrollerin bildirimlerini yapıyoruz : **QPushButton**, **QLabel**, **QTextEdit**, **QTreeView**. public slots 'a butonlar tıklandığı zaman ve treeview 'dan bir text dosyası üzerinde çift tıklandığı zaman çalışacak fonksiyonları tanımlıyoruz. mesajGoster() fonksiyonu bir önceki derse gönderme olarak orda : QMessageBox 'ın kullanımını unutmadık değil mi? dosyaAc() fonksiyonu btnAc QPushButton 'ına tıklanınca çalışacak fonksiyon adından belli olduğu gibi bir text dosyası açmaya yarıyor. Bu fonksiyon içinde **QFileDialog** kontolü yardımıyla seçtiğimiz text dosyasını bir **QString** değişkene atıyoruz. Daha sonra bu text dosyası için "dosya" adında **QFile** nesnesi oluşturuyoruz. QFile sınıfı bize dosyaya yazma ve dosyadan okuma yapabilmemiz için bir arayüz sağlıyor. QFile 'ın open fonksiyonuyla dosyamızı salt okunur ve text modunda açıyoruz. **QTextStream** nesnesi oluşturarak **QTextEdit** 'in setPlainText fonksiyonu ile açılan dosyadaki yazıları görüntülüyoruz. treeAc(const QModelIndex qm) fonksiyonu 'da dosyaAc() fonksiyonuyla aynı işi yapıyor. treeAc fonksiyonu dosyanın yol bilgisini dosyaAc fonksiyonundan farklı olarak bir **QDirModel** nesnesi ile yerel dosya sistemine erişerek **QTreeView** 'de seçilen text dosyasının yolunu QFile nesnesine geçiyor.

```
/*--- form.cpp ---*/
```

```
#include "form.h"

Form::Form()
{
   resize(600,600);
   QGridLayout *gLayout= new QGridLayout();

   lblMesaj = new QLabel("Mesaj goster :");
   gLayout->addWidget(lblMesaj,0,0);
   lblAc = new QLabel("Dosya Ac :");
   gLayout->addWidget(lblAc,1,0);
   lblTemizle = new QLabel("Temizle :");
   gLayout->addWidget(lblTemizle,2,0);

   btnMesaj = new QPushButton("Mesaj Goster");
   gLayout->addWidget(btnMesaj,0,1);
   btnAc = new QPushButton("Dosya Ac");
   gLayout->addWidget(btnAc,1,1);
   btnTemizle = new QPushButton("Temizle");
   gLayout->addWidget(btnTemizle,2,1);

   treeDosyalar = new QTreeView();
   model = new QDirModel();
   treeDosyalar->setModel(model);
   gLayout->addWidget(treeDosyalar,0,2,14,1);

   yazi = new QTextEdit();
   yazi->setPlainText("Temizle butonuna basarsan bu yazi silinir :)");
   yazi->resize(550,300);
   gLayout->addWidget(yazi,15,0,1,3);

   QHBoxLayout *hLayout = new QHBoxLayout();
   btnCik = new QPushButton("Cik Git");
   hLayout->addStretch(10);
   hLayout->addWidget(btnCik,Qt::AlignRight);

   QVBoxLayout *aLayout = new  QVBoxLayout();

   aLayout->addLayout(gLayout);
   aLayout->addLayout(hLayout);

   setLayout(aLayout);

   connect(btnMesaj ,SIGNAL(clicked()),this,SLOT(mesajGoster()));
   connect(btnAc ,SIGNAL(clicked()),this,SLOT(dosyaAc()));
   connect(btnTemizle ,SIGNAL(clicked()),yazi,SLOT(clear()));
   connect(btnCik ,SIGNAL(clicked()),this,SLOT(reject()));
   connect(treeDosyalar, SIGNAL(doubleClicked(QModelIndex )),this, SLOT(treeAc(const
QModelIndex )));

}
```

form.cpp

İlk olarak resize fonksiyonu ile QDialog 'un genişlik ve yüksekliğini ayarlıyoruz - Tamamen keyfi :) - . Bir **QGridLayout** nesnesi (gLayout) oluşturuyoruz. Izgara yerleşimi (Grid layout) adından da anlaşıldığı üzere yerleşiminizi bölünmüş kareler (yada dikdörgtenler ;)) yardımıyla yapmanızı sağlıyor. Oldukça kullanışlı ve kolay. Grid yerleşimine kontrolleri(widget) **addWidget** fonksiyonu ile yerleştiriyoruz. addWidgetin aldığı değerler sırasıyla : eklenecek kontrol ismi (widget name), yerleştirilecek satır(row), yerleştirilecek kolon(column), içerdiği satır (rowspan), içerdiği kolon (column span) ve hizalama(alignment). Örneğin gLayout->addWidget(yazi,15,0,1,3); satırında QTextEdit nesnesi 15 inci satır, 0 ncı kolona , 1 satır ve 3 kolon 'u kapsayacak şekilde ekliyoruz. gLayout->addWidget(lblAc,1,0); satırında addWidget'in sadece satır ve kolon bilgileri isteyen overload 'unu kullandığımı söylememe gerek yok sanırım ;) btnCik (çıkış düğmesi) 'ni eklemek için bir yatay yerlesim nesnesi (**QHBoxLayout**) oluşturuyoruz. QHboxLayout 'u kontrolleri yatay yerleştirmek için kullanıyoruz. hLayout->addWidget(btnCik,Qt::AlignRight); satırıyla çıkış düğmesini sağa hizalı olarak ekliyoruz. Bundan önceki satırda addStretch fonksiyonunu çağırdığıma dikkat edin, gerginlik(stretch) ayarını yapmassanız yerleştirdiğiniz düğme kolon genişliğinde görünür. Dikkat ederseniz tüm yerleşim nesnelerine bir kontrol eklemek addWidget fonksiyonuyla ekliyoruz. Oluşturduğumuz QGridLayout ve QHBoxLayout 'u alt alta yerleştirmek için yatay yerleşim (**QVBoxLayout**) nesnesi oluşturuyoruz. Yalnız ekleyeceğimiz nesneler birer yerleşim(layout) olduğundan bu sefer **addLayout** fonksiyonunu kullanıyoruz. Son olarak setLayout fonksiyonu ile QDialog 'umuzun yerleşimi olarak tüm layoutlarımızı barındıran aLayout -QVBoxLayout- nesnemizi ayarlıyoruz.SIGNAl ve SLOT 'lar için birinci derse bakabilirsiniz ;)

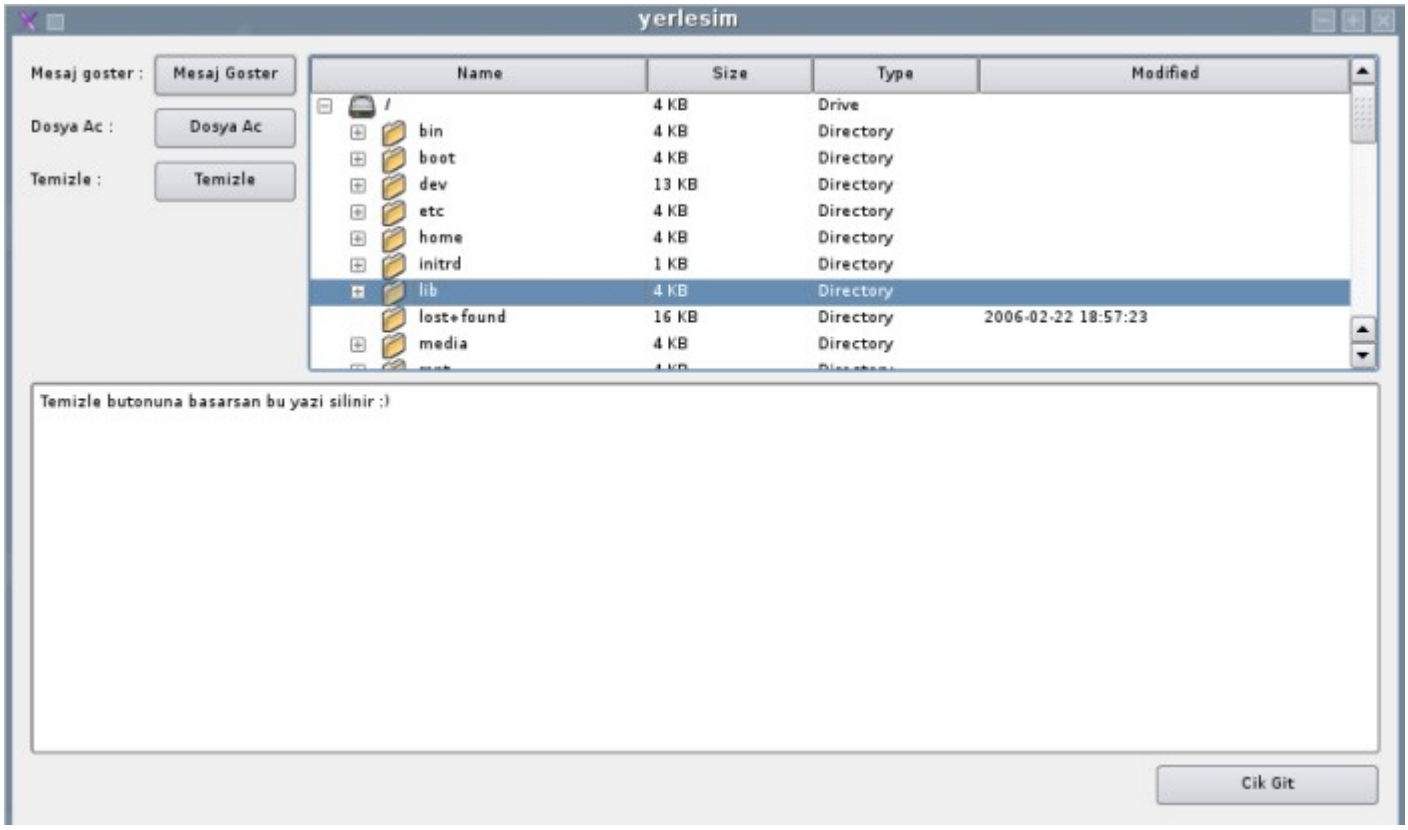
```
/*--- main.cpp ---*/

#include "form.h"

#include <QApplication>


int main(int argc, char *argv[])
{
   QApplication app
(argc,argv);
   Form form;
   form.show();
   return app.exec();
}
```

main.cpp

main.cpp kaynak kod dosyası içinde önceden açıklanmayan hiçbir kod parçası yok ;)

Derleme ve Çalıştırma

Yazdığınız programı (eğer yazdıysanız) çalıştırdıysanız linux altında şöyle bir ekran görüntüsü ile karşılaşmalısınız :



Kısaca çalışmasına bakarsak : Mesaj Göster yazan düğmeyi tıklarsanız mesaj görürsünüz. Dosya aç butonuyla yada ağaç yardımıyla dosya seçebilirsiniz. Temizleye basarak yazıları silebilir çık düğmesiyle de çıkabilirsiniz :) Hepsi bu.

Son Söz

Yazının başında bahsettiğim gibi yerleşimleri (layout) öğrenmek için fazla fazla kodlar yazdık :) Kodları fazla şişirmemek için hata kontrolü yapmadım. Yani metin dosyası dışında bir dosyayı açmaya çalışmayın :) Açmaya çalışırsanız ne olacağı konusunda bir fikrim yok .

Telif Hakkı ve Lisans

Feragatname