

# PyQt ile İş Uygulaması Geliştirimi

Ümit Öztosun

[umit@ly.com.tr](mailto:umit@ly.com.tr)

# Sunum Amacı

PyQt kullanarak kapsamlı, büyük ölçekli, profesyonel uygulamaların geliştirilebileceğini göstermek ve örneklemek.

# Sunum Planı

- **PyQt = Python + Qt**
  - Neden Python?
  - Neden Qt?
- **PyQt**
  - Genel Özellikler
  - Lisanslama
  - Kurulum
- **(Py)Qt Temelleri**
  - Olay Döngüsü (Eventloop)
  - Qt Designer
  - Sinyal ve Slot'lar
  - Yerleşimler (Layouts)
  - Modeller (Models)
- **Ağ Programlama**
  - Senkron - Asenkron
  - Twisted Kütüphanesi
  - Qt – Twisted Entegrasyonu
  - GUI Model Verisinin Sunucudan Alınması
- **Örnek Uygulamalar**
  - İş Yazılımı: Dia
  - Uygulama Geliştirme Platformu: MOCOP
- **Sorular**

# PyQt = Python + Qt

- Python dili kullanarak Qt kütüphanesi kullanılabilmesi için hazırlanmış bir bağdır (binding)
- Qt'nin çalıştığı tüm platformlarda çalışır: Linux, Windows, MacOS/X, Maemo (Internet Cihazları, Nokia N810 [OS2008])
- PyQt4: Qt 4.x
- PyQt3: Qt <= 3.x

# Neden Python?

- **Çok Kısaca**

- Temiz, okunaklı dil sözdizimi, yazması ve okuması kolay
- Açık kaynak kodlu, her türlü kullanıma uygun lisans
- Kolay öğrenilebilir
- Küçük betik programlama ihtiyaçlarından çok geniş yazılım projelerine kadar ölçeklenebilme
- Nesneye yönelimli; çok yüksek seviyeli dinamik veri tipleri sunmakta
- Çok kolaylıkla başka dillerle genişletilebilir (C, C++, Java; Swig, Boost.Python, Sip)
- Çok geniş bir standart kütüphane ile gelir (batteries included); Zip dosyası yaratmaktan, XML oluşturmaya, işleri alt süreçlere bölmekten, SHA1 hashleri oluşturmaya kadar pek çok modül standarttır.
- Çok geniş üçüncü parti kütüphane desteği: Qt, Gtk+, Resim İşleme, VLC, SDL, nümerik analiz ve daha pek çok kütüphane
- Platform bağımsız

# Kimler Python Kullanıyor?

- Pardus (Pisi, Mudur, Yalı)
- Redhat
- Google
- YouTube
- Industrial Light and Magic

# Neden Qt?

- Platform bağımsız (UNIX türevleri, Windows, MacOS/X, Embedded Systems)
- Temiz, net, nesneye yönelimli, iyi dokümante edilmiş API
- Sadece görsel kullanıcı arabirimleri için değil, pek çok kavram için platform bağımsız bir arabirim sunuyor: XML, threads, IPC, Ağ Programlama, WebKit, Test Altyapısı...
- Hem açık kaynak hem de ticari uygulama geliştirmeye uygun lisanslama
- Grafiksel işlemlerde yüksek performans
- Kendini ispatlamış referanslara sahip

# Kimler Qt Kullanıyor?

- KDE
- Google (Google Earth)
- Lucas Film Ltd.
- Skype
- Volvo Mobility
- Adobe (Photoshop Elements)



# PyQt

- Genel Özellikler
  - C++ kütüphanesi olan Qt'nin Python bağı
  - SIP isimli, C++ kütüphanelerinin Python'dan kullanılabilmesini sağlayan bir araç ile hazırlanıyor. SIP, Qt için özel olarak geliştirilmiş ama kullanımı Qt ile sınırlı değil
  - Lisanslaması genel olarak Qt'yi takip ediyor, açık kaynak kod geliştirimi için uygun
  - Yaygın olarak kullanılan Linux dağıtımları ile beraber geliyor
  - Qt API'sini hemen hemen aynen takip ediyor

# PyQt (Qt) Temelleri

- Eventloop (Olay Çevrimi):
  - Qt'nin olayları (event) beklediği, gerçekleşen olaylara göre gerekli işlemleri yaptığı, bir Qt uygulamasının sonuna kadar varlığını koruyan döngü
  - Olay temelli programlama yaklaşımı benimseyen diğer kütüphanelerde de benzer yapılar mevcuttur
  - Bize etkisi nedir?
    - Çok uzun bir işlemi olay çevrimi ile aynı iş parçacığı (thread) içerisinde gerçekleştirirsek, bu esnada gelen olayların işlenmesi gecikecektir. Bu da ekranın donması, kullanıcıya cevap verememesi gibi sonuçlara yol açar
    - Uzun süren işlemler işlemler için ayrı bir iş parçacığı (thread) kullanılabileceği gibi, asenkron (eşgüdüksüz) programlama ile de olay çevriminin aksaması engellenebilir.

# İlk Örnek

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
from PyQt4 import Qt

# Öncelikle QApplication nesnemizi oluşturalım
app = Qt.QApplication(sys.argv)

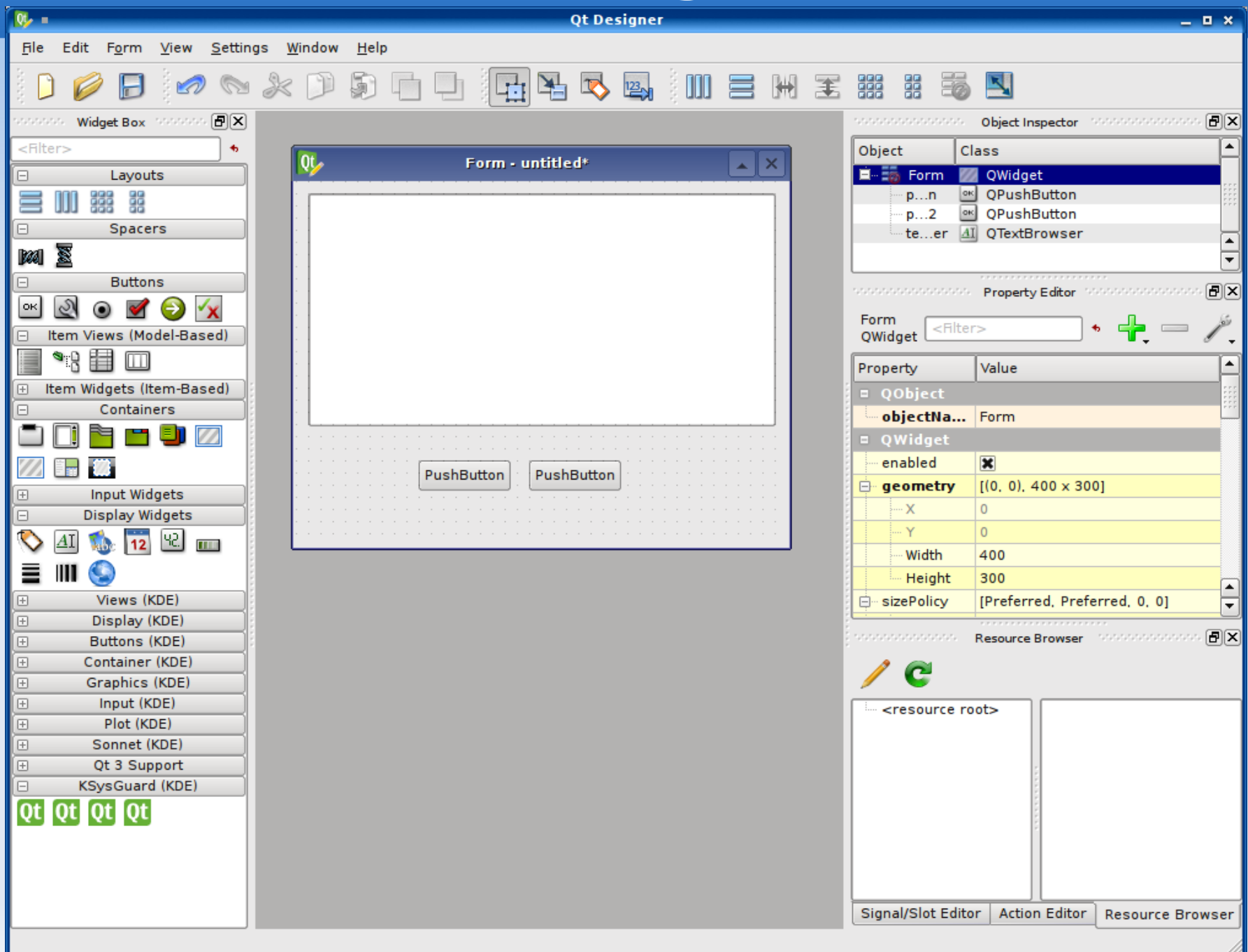
# Basit bir etiket ile mesajımızı dünyaya gösterelim
hello = Qt.QLabel(u"<h2>Merhaba Dünya</h2>")

# Widgetımızı göstermemiz gerekli
hello.show()

# Eventloop'u başlatıyoruz
app.exec_()
```



# Qt Designer



# Qt Designer, devam

- Görsel tasarım aracı
- Sadece ekran (form) tasarlamakta kullanılmaz, herhangi bir özel görsel öge (widget) geliştiriminde kullanılabilir
- Tasarımlarda kullanılan resim ve ikon dosyaları Qt Resource adı verilen dosyalarda tanımlanabilir
- Qt'nin temelini oluşturan sinyal ve slot bağlantıları yapılabilir

# Sinyal ve Slot'lar

- Olay tabanlı Qt altyapısında nesnelerin birbirine mesaj gönderebilmesini sağlar
- A nesnenin oluşturabileceği "sinyal" bir B nesnesin "slot"una bağlanır
- A nesnesi üzerinde ilgili olay meydana gelince (örneğin 'clicked'), B nesnesinin slotunda tanımlı olan işlemler gerçekleştirilir
- Sinyal ve slot yapısı kolay ve esnek bir şekilde PyQt ile kullanılabilir

# Sinyal/Slot Örneği

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
from PyQt4 import Qt

def btnClicked():
    Qt.QMessageBox.information(None, u'Uyarı', u'"Tıklayınız" tuşuna bastınız!')

if __name__ == '__main__':
    # Öncelikle QApplication nesnemizi oluşturalım
    app = Qt.QApplication(sys.argv)

    # Tuşumuz ve sinyal slot bağlantısı
    btn = Qt.QPushButton(u'Tıklayınız!')
    Qt.QObject.connect(btn, Qt.SIGNAL('clicked()'), btnClicked)

    # Widgetımızı göstermemiz gerekli
    btn.show()

    # Uygulamayı başlatalım
    app.exec_()
```

# Sadece Qt Designer Kullanarak Sinyal/Slot Örneği

The screenshot shows the Qt Designer interface with a central canvas titled "Form - signalslot\_justdesigner.ui". The canvas contains a horizontal slider widget and a label widget. A red arrow points from the slider to the Signal/Slot Editor panel on the right.

**Object Inspector:**

Object	Class
Form	QWidget
horizontalSlider	QSlider
label	QLabel
lcdNumber	QLCDNumber

**Property Editor:**

horizontalSlider  
QSlider

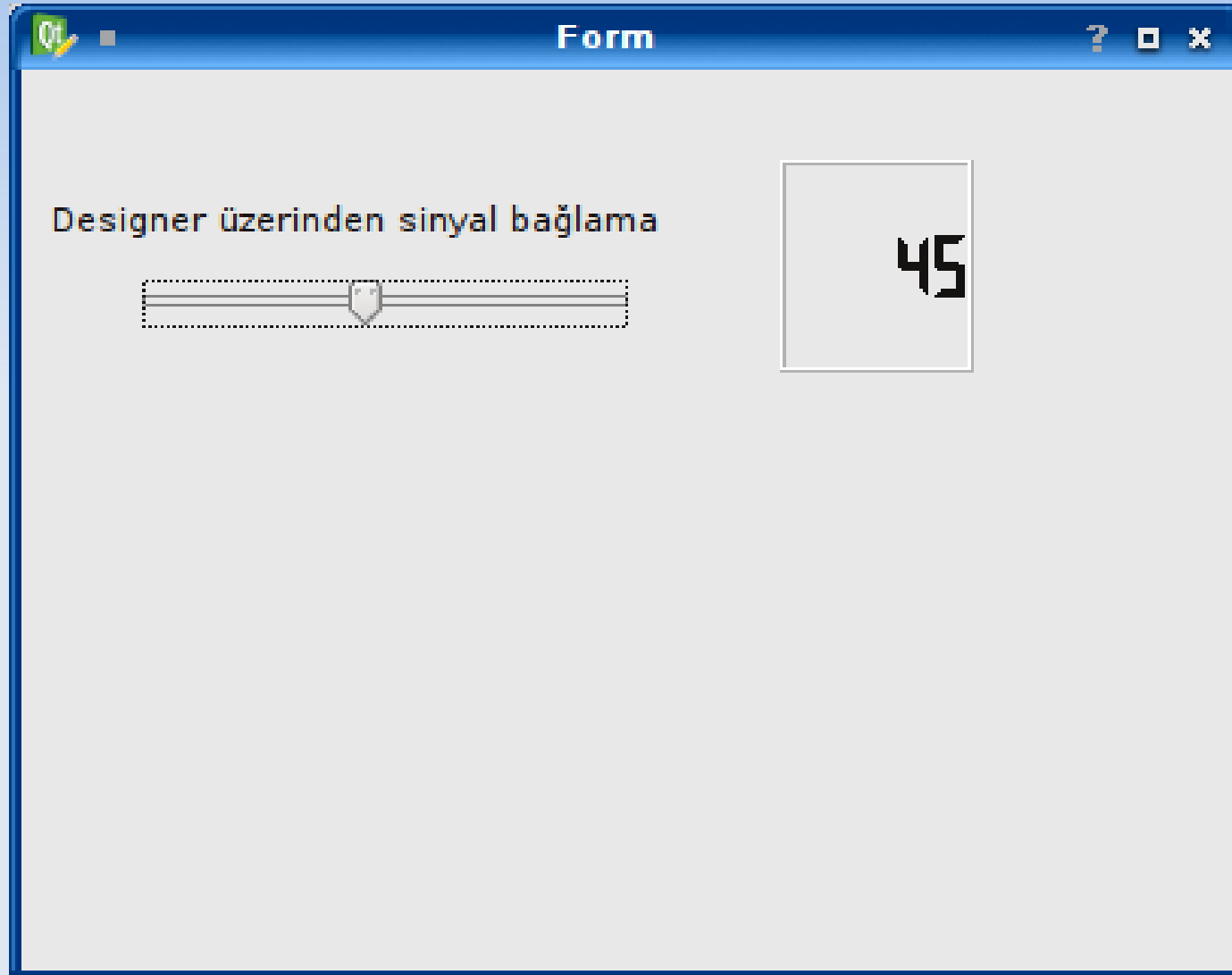
Property	Value
locale	Turkish, Turkey
QAbstractSlider	
minimum	0
maximum	99
singleStep	1
pageStep	10
value	0
sliderPosit...	0
tracking	<input checked="" type="checkbox"/>
orientation	Horizontal
invertedAp...	<input type="checkbox"/>

**Signal/Slot Editor:**

Sender	Signal	Receiver	Slot
horizontalSlider	valueChanged(int)	lcdNumber	display(int)



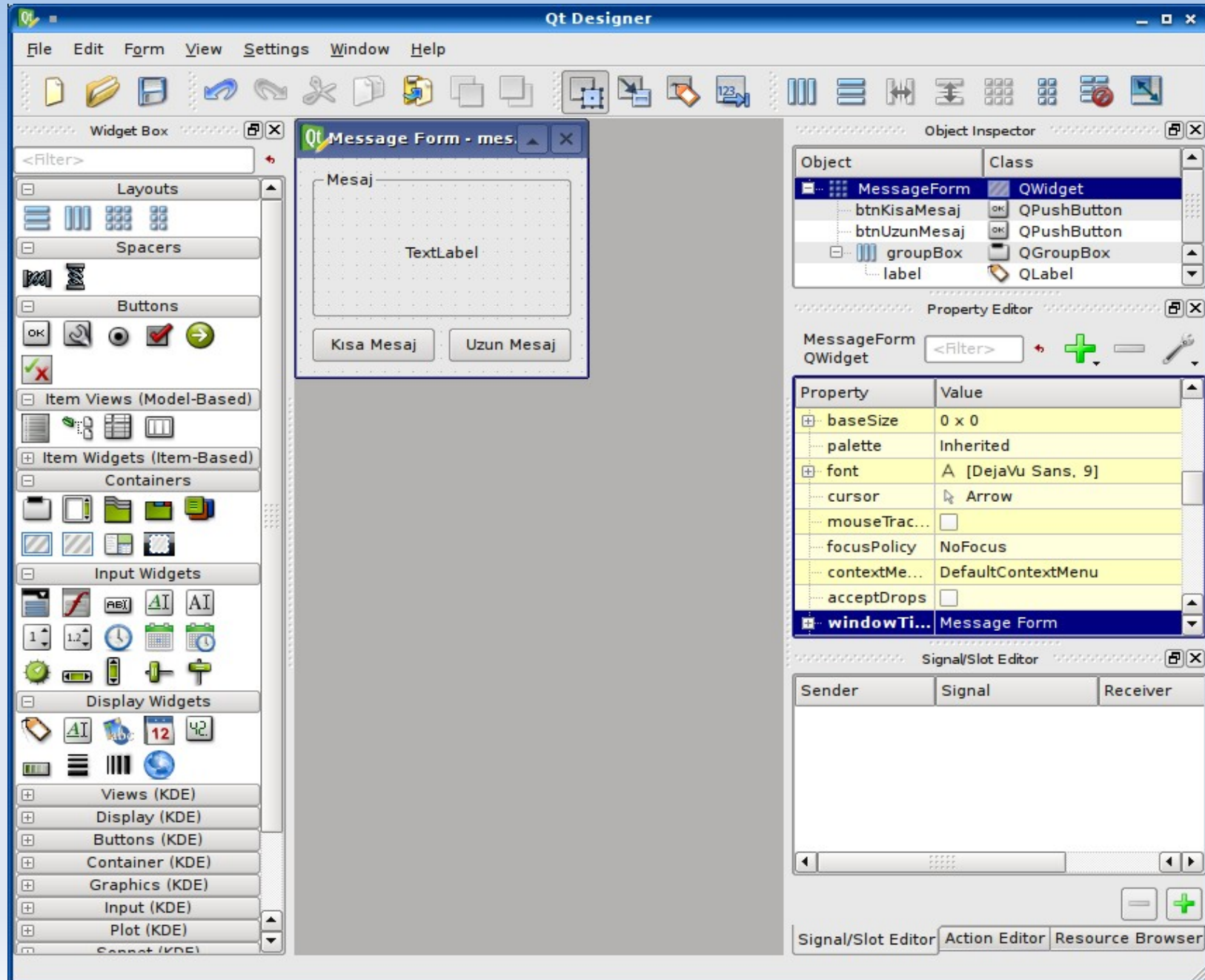
# Sadece Qt Designer Kullanarak Sinyal/Slot Örneği



# \*.ui Dosyaları

- Qt Designer tarafından oluşturulan grafiksel arabirim tasarımlarını XML formatında saklarlar
- Ui dosyaları istenirse *uic* modülü ile dinamik olarak yüklenebilir, istenirse de *pyuic(4)* isimli araç kullanarak python kodu üretilebilir
- İster dinamik yükleme ile isterse de hazırlanan python dosyaları kullanarak kendi kodumuzu inşa edebiliriz
- Genellikle, *pyuic* tarafından üretilen kodda değişiklik yapılmaz, tüm kodlama ayrı bir sınıf üzerinde yapılır

# Ui Dosyası Kullanan Örnek



# Ui Dosyası Kullanan Örnek, devam

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# message1.py

import sys
from PyQt4.Qt import *
from ui_messageform import Ui_MessageForm

class Message(QWidget, Ui_MessageForm):
    def __init__(self):
        QWidget.__init__(self)
        # Ui_MessageForm içindeki GUI tanımlarını mevcut nesne
        # ile birleştirelim
        self.setupUi(self)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = Message()
    form.show()
    app.exec_()
```

# Ui Dosyası Kullanan Örnek, devam



# Ui Dosyası Kullanan Örnek, Gelişmiş Sürüm

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# message2.py

import sys
from PyQt4.Qt import *
from ui_messageform import Ui_MessageForm

class Message(QWidget, Ui_MessageForm):
    def __init__(self):
        QWidget.__init__(self)
        self.setupUi(self)
        # Sinyal bağlantıları
        self.connect(self.btnKisaMesaj, SIGNAL('clicked()'), self.btnKisaMesaj_clicked)
        self.connect(self.btnUzunMesaj, SIGNAL('clicked()'), self.btnUzunMesaj_clicked)

    def btnKisaMesaj_clicked(self):
        self.label.setText(u'Kısa')

    def btnUzunMesaj_clicked(self):
        self.label.setText(u'Bu oldukça uzun bir mesajdır, hatta daha uzatabiliriz')

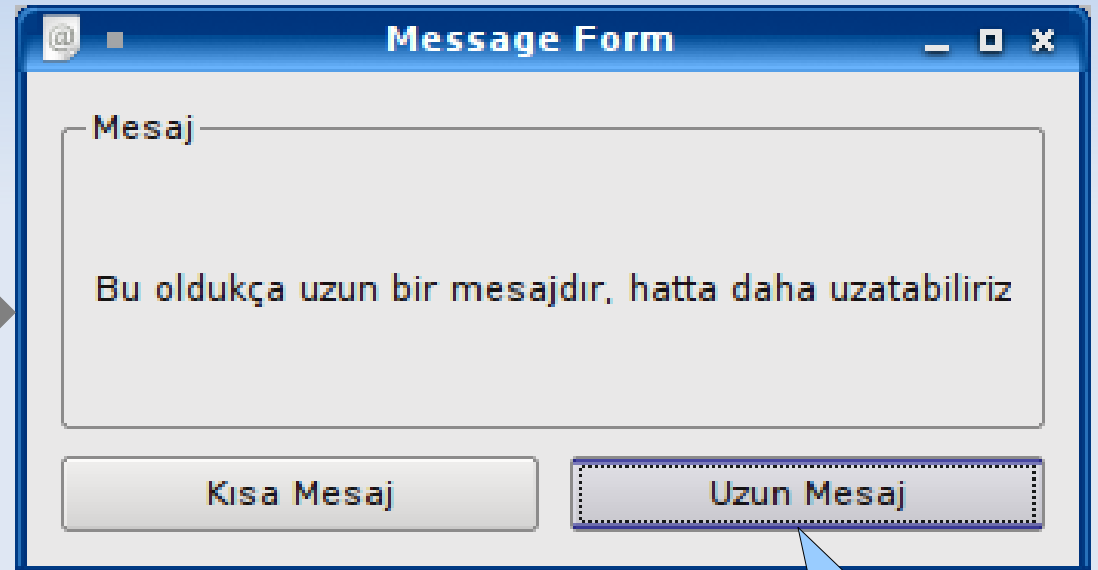
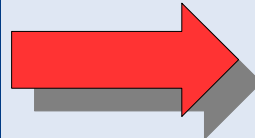
if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = Message()
    form.show()
    app.exec_()
```

# Yerleşim (Layout) Kullanmanın Avantajları



A screenshot of a Windows-style window titled "Message For" with standard minimize, maximize, and close buttons. Inside the window, there is a text area labeled "Mesaj" containing the word "Kısa". Below the text area are two buttons: "Kısa Mesaj" and "Uzun Mesaj". The "Kısa Mesaj" button is highlighted with a blue dashed border.

Kısa Mesaj  
Tuşuna  
basıldı



A screenshot of a Windows-style window titled "Message Form" with standard minimize, maximize, and close buttons. Inside the window, the text area labeled "Mesaj" now contains the text "Bu oldukça uzun bir mesajdır, hatta daha uzatabiliriz". Below the text area are two buttons: "Kısa Mesaj" and "Uzun Mesaj". The "Uzun Mesaj" button is highlighted with a blue dashed border.

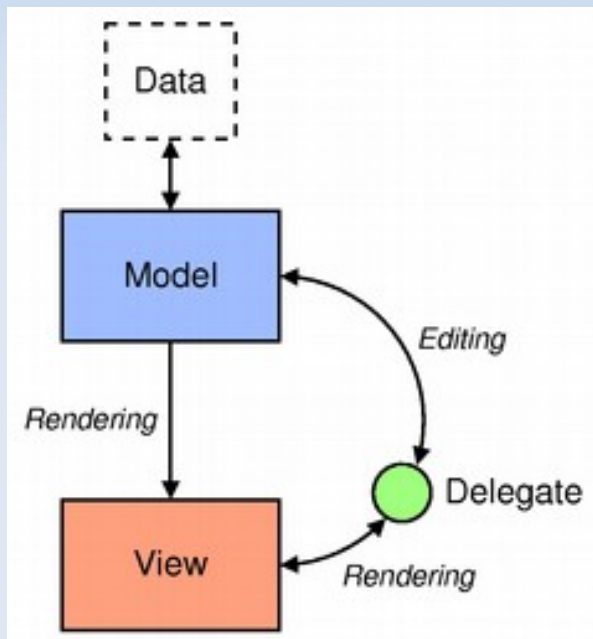
Uzun Mesaj tuşuna  
basıldı ve form  
otomatik olarak büyüdü

# Neden Yerleşim (Layout) Kullanılmalı?

- Kullanıcılarınızın platformundan, font tipi ve font boyu tercihlerinden etkilenmemek için
- Uygulamanızın metinsel içeriği değişse bile ekran tasarımlarınızın bozulmaması için
- Uygulamanız farklı dillere çevrildiği zaman tasarım bütünlüğünü korumak için
- Formlarınızın mantıklı öntanımlı boyları olması için
- Otomatik olarak pencere büyütme/küçültme sorununu çözmek için
- Dinamik olarak görsel parçacık (widget) ekleme çıkarma sonrası ekranın kendisini ayarlaması için



# Modeller (Models, Qt Model/View Architecture)



- Popüler MVC (Model - View - Controller) yaklaşımını gerçekleştirmeye yarayan nesnelerdir.
- Model: Uygulamada kullanılan nesnedir
- View: Ekran ve rapor benzeri yerlerdeki nesne gösterimidir
- Controller: Kullanıcının veri ile etkileşimini kontrol eden bileşendir
- Qt Model/View yapısında bu kavramlar şu şekilde eşleşir
  - Model: Model
  - View: View
  - Controller: Delegate

# Salt Okunur Tablo Örneği

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# tablemodel.py

import re, os, sys
from PyQt4.Qt import *
from ui_tablemodel import Ui_TableModel

class TableModelWidget(QWidget, Ui_TableModel):
    def __init__(self, *args):
        QWidget.__init__(self, *args)
        self.setupUi(self)
        self.connect(self.btnDoldur, SIGNAL('clicked()'), self.fillTable)

    def fillTable(self):
        headerData = [u'Ad', u'Soyad', u'Adres']
        tableData = [
            [u'Ahmet', u'Yılmaz', u'Ankara'],
            [u'Deniz', u'Cengiz', u'Bursa'],
            [u'Doruk', u'Keskin', u'Antalya'],
            [u'İsmail', u'Derin', u'İstanbul'],
            [u'Arif', u'Özlemmez', u'İstanbul'],
        ]
        model = TableModel(tableData, headerData, self)
        self.tableView.setModel(model)
```

# Salt Okunur Tablo Örneği, devam

```
class TableModel(QAbstractTableModel):
    def __init__(self, tableData, headerData, parent=None, *args):
        QAbstractTableModel.__init__(self, parent, *args)
        self._tableData = tableData
        self._headerData = headerData

    def rowCount(self, parent):
        return len(self._tableData)

    def columnCount(self, parent):
        return len(self._tableData[0])

    def data(self, index, role):
        if not index.isValid():
            return QVariant()
        elif role != Qt.DisplayRole:
            return QVariant()
        return QVariant(self._tableData[index.row()][index.column()])

    def headerData(self, col, orientation, role):
        if orientation == Qt.Horizontal and role == Qt.DisplayRole:
            return QVariant(self._headerData[col])
        return QVariant()

if __name__ == "__main__":
    app = QApplication(sys.argv)
    w = TableModelWidget()
    w.show()
    app.exec_()
```

Yeniden gerçekleştirilmiş  
Qt Metodları



# Salt Okunur Tablo Örneđi, devam



The image shows a Windows application window titled "Form". Inside the window, there is a table with three columns: "Ad", "Soyad", and "Adres". The table contains five rows of data. Below the table, there is a button labeled "Tabloyu Doldur".

Ad	Soyad	Adres
Ahmet	Yilmaz	Ankara
Deniz	Cengiz	Bursa
Doruk	Keskin	Antalya
İsmail	Derin	İstanbul
Arif	Özlemez	İstanbul

Tabloyu Doldur

# Ağ Programlama

- İki temel yaklaşım mevcut:
  - Senkron (eşgüdümlü – synchronous)
    - ```
networkObj = connectToServer()  
data = networkObj.getData()  
# Sunucudan veri gelene kadar bekler  
processServerData(data)
```
    - Çoklu program parçacıklı (multi-threaded) yapıya uygundur
  - Asenkron (eşgüdümsüz – asynchronous)
    - ```
networkObj = connectToServer()  
dataHandle = networkObj.getData()  
# Sunucuyu beklemez, çalışmaya devam eder  
dataHandle.addCallback(processServerData)  
doSomething()  
...
```
    - Multi-threaded yapıdan kaçınılabilir

# Ağ Programlama, devam

- Asenkron programlama için çok geniş ve gelişmiş bir kütüphane olan *Twisted* kullanılabilir
  - Çok geniş protokol desteği; TCP, UDP, SSL/TLS, multicast, Unix sockets, HTTP, NNTP, IMAP, SSH, IRC, FTP, DNS, ...
  - Asenkron programlamayı kolaylaştırıcı nesneleri içeriyor, *deferred* isimli soyutlama nesnesi kullanılıyor
  - Büyük projelerde kullanılıyor ve uzun yıllardır aktif olarak geliştiriliyor (Nasa, justin.tv, Apple, ...)
  - PyQt, PyGTK, PyGame gibi kütüphanelerin olay döngüleri ile entegrasyon imkânı mevcut

# Sunucu Örneği

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# simpleserver.py

from twisted.spread import pb
from twisted.internet import reactor

class Server(pb.Root):
    def remote_getTableData(self):
        print '+++ istemci veri istedi'
        return [
            [u'Ahmet', u'Yılmaz', u'Ankara'],
            [u'Deniz', u'Cengiz', u'Bursa'],
            [u'Doruk', u'Keskin', u'Antalya'],
            [u'İsmail', u'Derin', u'İstanbul'],
            [u'Arif', u'Özlemmez', u'İstanbul'],
        ]

if __name__ == '__main__':
    serverFactory = pb.PBServerFactory(Server())
    reactor.listenTCP(3333, serverFactory)
    print '+++ Sunucu başlatıldı'
    reactor.run()
```

Twisted olay döngüsü  
burada başlatılıyor

# İstemci Örneği

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# simpleclient.py

from twisted.spread import pb
from twisted.internet import reactor

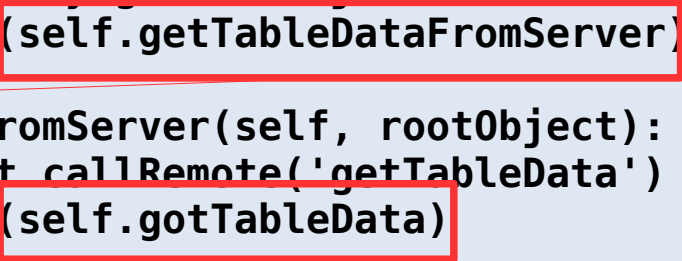
class Client(object):

    def connect(self):
        clientfactory = pb.PBClientFactory()
        reactor.connectTCP("localhost", 3333, clientfactory)
        d = clientfactory.getRootObject()
        d.addCallback(self.getTableDataFromServer)

    def getTableDataFromServer(self, rootObject):
        d = rootObject.callRemote('getTableData')
        d.addCallback(self.getTableData)

    def getTableData(self, result):
        print "Sunucudan gelen cevap: ", result

if __name__ == '__main__':
    Client().connect()
    reactor.run()
```





# Tablo Verisinin Sunucudan Alınması

```
import re, os, sys, qt4reactor
from PyQt4.Qt import *
from ui_tablemodel import Ui_TableModel
# twisted.internet.reactor importundan önce qt4reactor kurulmalı
app = QApplication(sys.argv)
qt4reactor.install(app) # Bu satır Twisted <-> Qt entegrasyonunu sağlar
from twisted.spread import pb
from twisted.internet import reactor

class TableModelWidget(QWidget, Ui_TableModel):
    def __init__(self, *args):
        QWidget.__init__(self, *args)
        self.setupUi(self)
        self.connect(self.btnDoldur, SIGNAL('clicked()'), self.connectToServer)

    def connectToServer(self):
        clientfactory = pb.PBClientFactory()
        reactor.connectTCP("localhost", 3333, clientfactory)
        d = clientfactory.getRootObject()
        d.addCallback(self.getTableDataFromServer)

    def getTableDataFromServer(self, rootObject):
        d = rootObject.callRemote('getTableData')
        d.addCallback(self.fillTable)

    def fillTable(self, tableData):
        headerData = [u'Ad', u'Soyad', u'Adres']
        model = TableModel(tableData, headerData, self)
        self.tableView.setModel(model)
```

# Örnek Uygulamalar

- DiA

- Farklı amaçlara yönelik uygulamalar için geliştirmiş olduğumuz platform
- Temel olarak Dia İş Uygulamasını çalıştırıyor
- Küçük bir istemci ve uygulama sunucudan oluşuyor; ekranlar da dahil olmak üzere her şey sunucudan alınıyor
- Tek bir istemci ile farklı uygulamalara bağlanmak mümkün

# Örnek Uygulamalar, devam

- MOCOP

- TÜbitak tarafından desteklenen projemiz
- Geliştirimi devam ediyor
- Uygulama geliştirmeye yönelik, platform bağımsız bir geliştirme platformu
- Önceki bilgi birikimimizi kullanarak daha önceki altyapılarımızda eksik gördüğümüz noktalara eğildik
- GPL lisansı ile uyumlu olarak da dağıtılacak

# Sorular

# Teşekkürler

- Referans olarak başvurulabilecek kaynaklar:
  - [www.python.org](http://www.python.org)
  - [www.riverbankcomputing.co.uk](http://www.riverbankcomputing.co.uk) (PyQt)
  - [www.twistedmatrix.com](http://www.twistedmatrix.com)
  - [www.qtsoftware.com](http://www.qtsoftware.com)
  - [mocop.ly.com.tr](http://mocop.ly.com.tr)
  - [www.dia.gen.tr](http://www.dia.gen.tr)