YAPILAN İŞ : Staj Projesi içinde Kullanılabilecek öğelerden biri olan StackView örneği oluşturma

İlk olarak QML dosyalarında yazılan komutlar anlatılıp ardından hangi sırayla işlemler yapıldığı anlatılacaktır.

```
ApplicationWindow {
    width: 640
    height: 480
    visible: true
    title: qsTr("Hello World")
    header: ToolBar {
         RowLayout {
                     anchors.fill: parent
                     ToolButton {
                         text: qsTr("‹")
                         onClicked: mystackview.pop()
                     }
                     Label {
                         text: "StackView Example"
                         elide: Label.ElideRight
                         horizontalAlignment: Qt.AlignHCenter
                         verticalAlignment: Qt.AlignVCenter
                         Layout.fillWidth: true
                     }
                 }
    }
    StackView {
        anchors{
            left: parent.left
            right: parent.right
            bottom: parent.bottom
            top: header.bottom
        }
        id: mystackview
        initialItem:starting_page

    }
    Component {
        id:starting_page
        StartingPage {}
    }
    Component {
        id:secong_page
        SecondPage{}
    }
    Component {
        id:third_page
         ThirdPage{}
    }
    Component {
        id:fourth_page
        FourthPage{}
    }
    function load_page(page){
        switch(page){
        case 'PAGE 1':
            mystackview.push(starting_page);
            break;
        case 'PAGE 2':
            mystackview.push(secong_page);
            break;
        case 'PAGE 3':
            mystackview.push(third_page);
            break;
        case 'PAGE 4':
            mystackview.push(fourth_page);
            break;
        }
    }
}

```

GÖRSEL 1 -> Main.qml

NOT 1 : Programı yazmaya başlarken ilk olarak StackView fonksiyonunu yazmakla başlanır. Burada her bir stack için farklı bir QML dosyası olacağı için bunları Component olarak ekledik. Ayrıca bu örnekte program Starting_page ile başladığı için initialItem olarak o yazıldı. İstenildiği takdirde örneğe göre düzenlenilebilir.

NOT 2: ToolBar kısmı programda bir stack içine girildiği zaman o stacktan çıkmak için kullanılmıştır. Bu kısım QT Creator içindeki Help kısmından alınıp düzenlenmiştir. Buradaki Allignment komutları hizalamak için kullanılmıştır. FillWidth ise tüm genişliği kullanmak için yazılmıştır.

NOT 3: load_page fonksiyonu bir butona basıldığında o stack'e girilmesini sağlar. Buradaki en önemli nokta ise case 'de yazılan Page isimleri ile starting_page kısmında göreceğimiz butonlarda yazılan Page isimlerinin aynı olmasıdır. Aksi taktirde butona basıldığında isimler tutmadığı için bu fonksiyon işlemez ve stack içine girilemez.

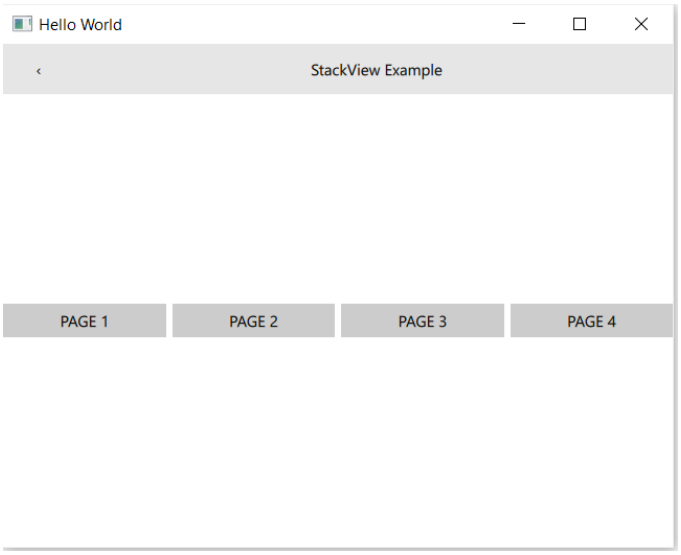
```
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Layouts 1.3

Item {
    RowLayout {
        anchors.centerIn: parent
        width: parent.width
        Button{
            text: 'PAGE 1'
            Layout.fillWidth: true
            onClicked: {
                load_page(text);
            }
        }
        Button{
            text: 'PAGE 2'
            Layout.fillWidth: true
            onClicked: {
                load_page(text);
            }
        }
        Button{
            text: 'PAGE 3'
            Layout.fillWidth: true
            onClicked: {
                load_page(text);
            }
        }
        Button{
            text: 'PAGE 4'
            Layout.fillWidth: true
            onClicked: {
                load_page(text);
            }
        }

    }
}
```

GÖRSEL 2 -> Starting_page. qml

NOT 1 : Burada RowLayout butonlar sıralı bir şekilde bir satırda bulunması istenildiğinden dolayı kullanılmıştır. Projeye göre değiştirilebilir bir komuttur. Zorunlu değildir.

NOT 2 : Bir önceki sayfada belirtildiği gibi Butonlar içinde yazılan onClicked komutunda load_page fonksiyonu çağırılır. Buradaki text button içinde yazılan text satırıdır. Burada text'in içinde yazılan label aynı zamanda switch-case yapısındaki case'lerden birini işaret etmektedir.

NOT 3 : Layout.fillWidth komutu true yada false olarak ayarlanabilir. Eğer true ise mümkün olan genişliğe göre ayarlanır. Yani normal koşullarda bu satır olmadığında butonun boyutu içinde yazan label'ın sığacağı kadar bir boyuttadır. Buna göre 4 butonda ayarlanır ve soldan başlayarak sıralanır. Geriye sağ tarafta boşluk bulunur. Ancak bu komut kullanıldığında RowLayout için tüm genişliği kullan dediğimiz için bu genişliği 4 ' e bölerek buton boyutlarını ayarlar. Boyutlandırmadaki bir diğer önemli husus ise buton içindeki label sığmalıdır.



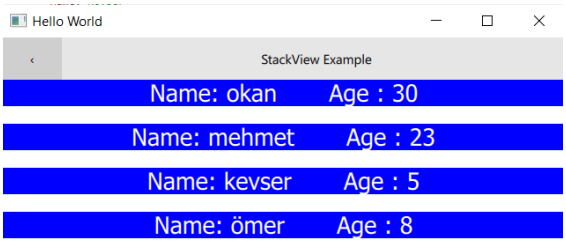
GÖRSEL 3 -> Program açıldığında (Program başlatıldığında starting page ile başlatılır.

```qml
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Layouts 1.3
Item {
    ListView{
        anchors.fill: parent
        width: parent.width
        height: parent.height
        spacing: 20
        model: ListModel{
            ListElement{
                name: 'okan'
                age:30
            }
            ListElement{
                name: 'mehmet'
                age: 23

            }
            ListElement{
                 name:'kevser'
                 age: 5

            }
            ListElement{
                name:'ömer'
                age: 8

            }
        }
        delegate: Rectangle{
            width: parent.width
            height: 30
            color: 'blue'
            Text {
                anchors.centerIn:  parent
                color: 'white'
                text: 'Name: ' + name + '       Age : ' + age
                font.pointSize: 16

            }
        }
    }
}
```

GÖRSEL 4 -> SecondPage.qml

NOT 1: Burada stack içini doldurmak için bir listview örneği yazılmıştır.

NOT 2: Burada delegate komutu ListView içindeki indexlere ulaşmak için kullanılmıştır. Bu indexlere ulaşıp Rectangle içinde belirlenen kurallara göre yazdırılır.

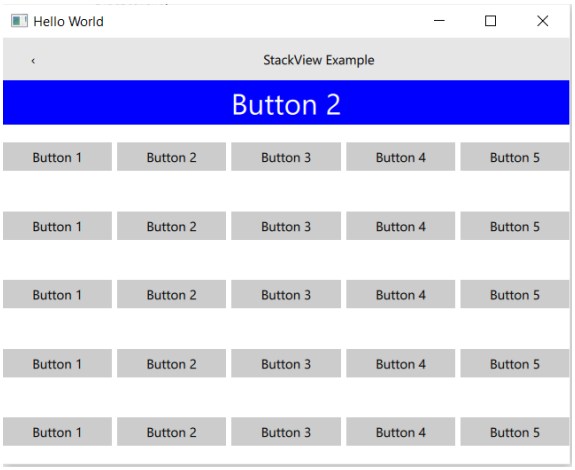


GÖRSEL 5 -> SecondPage programda görünüşü

```
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Layouts 1.3
Item {
    Rectangle{
        id: kare
        width: parent.width
        height: 50
        color: 'blue'
        Label {
            id : mytitle
            anchors.centerIn: parent
            color : 'white'
            font.pointSize: 20
        }
    }
    ColumnLayout {
        anchors {
            left: parent.left
            right: parent.right
            bottom: parent.bottom
            top: kare.bottom
        }
        spacing: 5
        Repeater {
           model :5
           RowLayout{
               Layout.fillHeight:true
               width: parent.width
               height: 10
               Repeater {
                   model : ListModel{
                       ListElement{
                           mytext:'Button 1'
                       }
                       ListElement{
                           mytext:'Button 2'
                       }
                       ListElement{
                           mytext:'Button 3'
                       }
                       ListElement{
                           mytext:'Button 4'
                       }
                       ListElement{
                           mytext:'Button 5'
                       }
                   }
                   Button{
                       Layout.fillWidth: true
                       text: mytext
                       onClicked: {
                           mytitle.text = text;
                       }
                   }
               }
           }
```

GÖRSEL 6 -> ThirdPage.qml

NOT 1 : Burada Stack doldurmak için daha önce yazılan Repeater örneği kullanılmıştır.

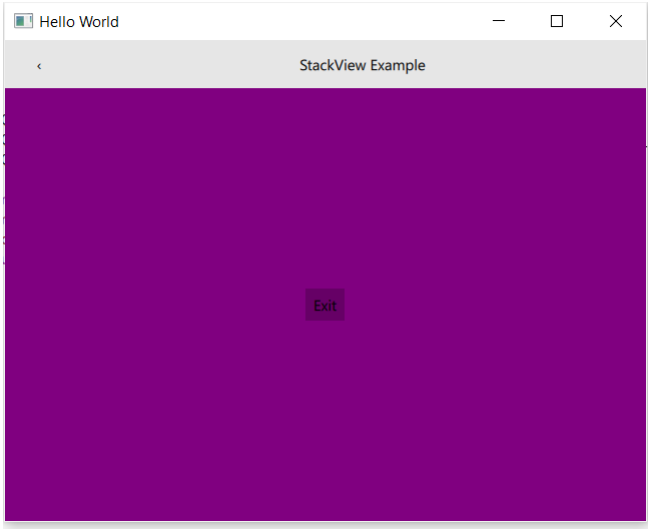


GÖRSEL 7->ThirdPage programda görünüşü

```qml
import QtQuick 2.9
import QtQuick.Controls 2.5
import QtQuick.Layouts 1.3
Item {
    Rectangle {
        anchors.fill: parent
        color: 'purple'
        Button{
            anchors.centerIn: parent
            text: 'Exit'

            onClicked: {
                Qt.quit();
            }
        }
    }
}

```

GÖRSEL 8-> FourthPage.qml

NOT 1: Burada stack doldurmak için tüm ekranı doldurcak şekilde bir rectangle oluşturulup arka plan rengi değiştirilip Exit butonu koyulmuştur.



GÖRSEL 9 -> FourthPage programda görünüşü

PROGRAM OLUŞTURULMASI

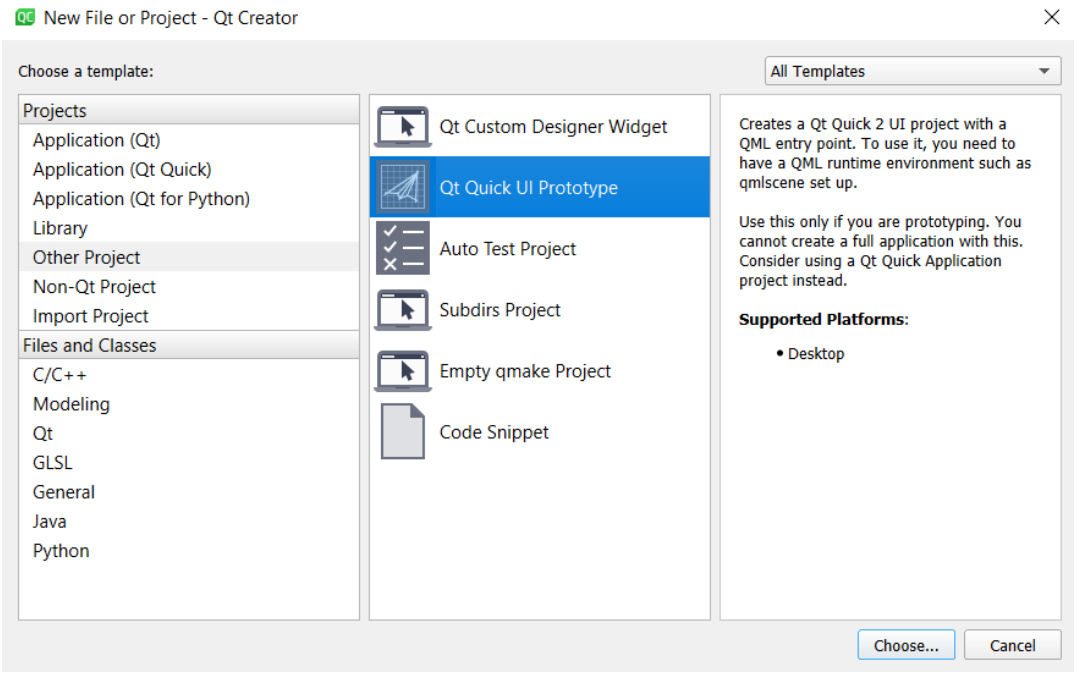
1. İlk olarak Qt Quick Application-Empty projesi oluşturulmalıdır.
2. Daha sonra main.qml sayfası içinde StackView oluşturulmalıdır. Burada Componentler eklenmeli ve initialItem belirtilmelidir.
3. Daha sonra Page'ler arasında gezinti yapabilmek için ToolBar eklenmelidir. Buradaki mantık Stack'ın bir özelliği olan pop ve push mantığıdır. ToolBarda geri gitmek için stackte pop yapılır. Butonla girmek için push yapılır.
4. Ardından Stack için kaç tane öğe oluşturmak istiyorsak o kadar QML File (Qt Quick 2) dosyası oluşturulmalıdır.
5. Oluşturulan bu sayfalar Stack'ın içindeki Componentlere Görsel 1 'deki gibi eklenmelidir.
6. StartingPage sayfasında buton düzeni oluşturulmalıdır.
7. Buton düzeni oluşturulduktan sonra tekrar main.qml sayfasında oluşturulan sayfalara erişim sağlamak için örnek olarak yazılmış olan load_page gibi bir fonksiyon oluşturulmalıdır. Buradaki önemli husus notlarda belirtilmiştir.
8. Bu fonksiyon yazıldıktan sonra startingPage'deki butonlarda fonksiyon çağrımı yapılmalıdır.
9. Bu işlem yapıldıktan sonra artık geri kalan sayfalarda ne yapılmak isteniyorsa onlar eklenmelidir.

NOT 1 : Burada QML File (Qt Quick 2)oluşturulmaya özen gösterilmelidir. Eğer main gibi Empty dosyası oluşturulursa bunu İtem olarak algılamaz. Ayrı bir proje gibi oluşturulur ve entegrasyonu zorlaşır. Bu projede bu yapılmaya çalışılmış ancak başarılamamıştır.

NOT 2 : Bu projede Buton 1 e basıldığında starting_page ' e dönmesi komutu verilmiştir. Boş bırakılmıştır.

YAPILAN İŞ : Staj Projesi içinde Kullanılabilecek öğelerden biri olan Dijital Sat örneği oluşturma

Bu örnekte diğer yazılan programlar gibi QT Quick Application- Empty yerine QT Quick UI Prototype çeşidiyle proje oluşturulmuştur.



GÖRSEL 10-> Proje Oluşturma

```
import QtQuick 2.12

Item {
    width: 800
    height: 600
    Column {
        anchors.centerIn: parent
        Text {
            id: text1
            font {
                family: "Comic Sans MS"
                pixelSize: 80
            }
            anchors.horizontalCenter: parent.horizontalCenter
        }
        Text {
            id: text2
            font {
                family: "Comic Sans MS"
                pixelSize: 40
            }
            anchors.horizontalCenter: parent.horizontalCenter
        }
    }
    Timer {
        interval: 500
        running: true
        repeat: true

        onTriggered: {
            var date = new Date()
            text1.text = date.toLocaleTimeString(Qt.locale("en_US"), "hh:mm:ss ap")
            text2.text = date.toLocaleDateString(Qt.locale("en_US"))
        }
    }
}
```

GÖRSEL 11 -> Digital Saat Projesi Kodları

Program Oluşturulması

1. Bu projede ilk olarak ekran boyutu ayarlanıp 2 adet Column oluşturulmuştur. Bunlar merkezlenmiştir.
2. Burada diğer örneklere ekstra olarak Text kısımlarında yazı boyutu ve tipi değiştirilmiştir.
3. Daha sonra Timer kullanılarak saat ve gün ayarlanmıştır.
   NOT 1 : İnterval Triggerlar arasındaki aralığı milisaniye cinsinden ayarlanması için kullanılmıştır.
   NOT 2 : Running saatin çalışması için gereklidir ve değerinin true olması gerekir.
   NOT 3 : Repeat komutu saatin tekrar tekrar çalışmasını sağlar. Yani tetikleyici 500 ms sonra tekrar çalışmasını kontrol eder. Bu değer false olursa program açıldığında saat hangi değerde ise o değerde kalır ve çalışmaz .
   NOT 4 : onTriggered içinde ise saat ve tarih yazdırılır. Ayrıca Date fonksiyonu çağrılır ve bu fonksiyon ile yazdırma yapılır.