استخدام قواعد البيانات مع تطبيقات WPF

بقلم: محمد سامر أبو سلو

Samer.selo@yahoo.com

Samerselo2005@hotmail.com

ملاحظة هامة قبل البدء

حتى تستفيد من المواضيع في هذا الكتيب حول استخدام قواعد البيانات مع تطبيقات wpf بالشكل الأمثل يجب أن يكون لديك معرفة ببرمجة قواعد البيانات وأساسيات WPF و XAML

القسم الأول – مدخل إلى قواعد البيانات

أنشئ مشروعا جديدا من نوع WPF Application وسمه WpfDataEntery وأضف للمشروع قاعدة البيانات المرفقة مع الأمثلة وأنشئ DataSet بالاعتماد على الجدول Customer وذلك بالطريقة الاعتيادية عند إضافة مصدر بيانات جديد وبذلك نكون قد أضفنا CustomerDatSet للمشروع. ثم افتح قائمة View ثم other Window ثم اختر الأمر document outline ثم عدل كود xaml الخاص بالنافذة ليصبح كما يلي كي نقوم بتقسيم الشبكة إلى أربعة أقسام

```
<Window x:Class="Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="294" Width="525">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="46*" />
            <RowDefinition Height="216*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="138*" />
            <ColumnDefinition Width="140*" />
        </Grid.ColumnDefinitions>
    </Grid>
</Window>
```

ثم أضف أربعة أزرار للنافذة لاحظ انه بعد تفعيل document outline أنه أصبح بإمكانك سحب وإفلات التحكمات على النافذة تماما كما نفعل في تطبيقات Windows Forms ثم غير خصائص الأربعة أزرار ليتم ضبط خاصية Content لها إلى Delete و Add و Revert و Save و خاصية Name إلى btnDelete و btnAdd و btnRevert و btnSave ثم أضف أربعة أزرار أخرى من أجل الانتقال السابق والتالي والأول والأخير واجعل خاصية name كالتالي btnFirst و btnPrevious و btnNext و btnLast

أضف stackpanel عدد 2 إلى القسمين السفليين من الـ Grid وغير حجمهما ليملأ كل واحد كامل مساحة قسم الشبكة الموجود فيه ثم أضف أربعة تحكمات label لـ StackPanel اليساري و أربعة تحكمات TextBox لليميني ثم اضبط خاصية height لصناديق النصوص إلى 28 لمساواة ارتفاعها مع تحكمات Label ثم اضبط خاصية Width لجميع تحكمات Labelو TextBox إلى Auto

اضبط قيمة Content لـ Label العلوي إلى CustomerID واجعل خاصية name لـ TextBox العلوي txtCustomerID كما اضبط خاصية IsReadOnly له إلى True ثم اضبط خاصية Content لبقية تحكمات Lable إلى Last Name و First Name و City و خاصية name لبقية تحكمات TextBox إلى txtLastName و txtFirstName و txtCity

من أجل ربط صناديق النصوص بالبيانات افتح قائمة Data ثم اختر Show Data Sources انتقل إلى كود xaml و أضف النص التالي الذي يحدد كيفية ربط الخاصية text لصندوق النصوص txtCustomerID بالحقل CustomerID كما يلي Text="{Binding Path=CustomerID, Mode=OneWay}" وقمنا بتحديد OneWay لأنه للقراءة فقط

أضف الخاصية text لـ txtLastName كما يلي Text="{Binding Path=LastName} وقد أزلنا القيمة oneWay لأنه للقراءة والكتابة كرر نفس العملية بالنسبة لصندوقي النصوص الباقيين لربطها مع الحقول المناسبة وبذلك يصبح كود xaml لصناديق النصوص لديك مشابها للتالي

```
<TextBox Height="28" Name="txtCustomerID" Width="Auto" IsReadOnly="True"
                 Text="{Binding Path=CustomerID, Mode=OneWay}" />
<TextBox Height="28" Name="txtLastName" Width="Auto"
                 Text="{Binding Path=LastName}"/>
<TextBox Height="28" Name="txtFirstName" Width="Auto"
                 Text="{Binding Path=FirstName}"/>
<TextBox Height="28" Name="txtCity" Width="Auto"
                 Text="{Binding Path=City}"/>
```

انقر على نافذة التطبيق بزر الفأرة اليميني ثم اختر View Code وأضف المتغيرات التالية في القسم العام للفئة في ملف الكود

```
Private CustomerData As New CustomerDataSet
Private taCust As New CustomerDataSetTableAdapters.CustomerTableAdapter
Private taManager As New CustomerDataSetTableAdapters.TableAdapterManager
```

أضف الكود التالي لحدث Loaded للنافذة

```
Me.taCust.Fill(Me.CustomerData.Customer)

Me.DataContext = Me.CustomerData.Customer
```

حيث قمنا بملئ taCust الذي هو عبارة عن محول البيانات بالبيانات باستخدام الطريقة Fill ثم ضبطنا قيمة الخاصية DataContext للنافذة إلى الجدول Customer فإن شغلت البرنامج هنا ستلاحظ ظهور البيانات على النافذة

عرف المتغير التالي على مستوى الفئة حيث يعتبر CollectionView في تطبيقات WPF بديلاً عن BindingSource في تطبيقات النوافذ

```
Private View As CollectionView
```

ومن أجل ربطه مع بياناتنا نستخدم الكود التالي في الحدث Loaded للنافذة بعد الكود الذي أدخلناه فيها سابقا

```
Me.View= CollectionViewSource.GetDefaultView(Me.CustomerData.Customer)
```

أجعل حدث النقر على أزرار التنقل كما في الكود التالي من أجل تنفيذ عمليات التنقل بين السجلات

```
    Private Sub btnFirst_Click(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles btnFirst.Click

        Me.View.MoveCurrentToFirst()
    End Sub

    Private Sub btnPrevious_Click(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles btnPrevious.Click

        If Me.View.CurrentPosition > 0 Then
            Me.View.MoveCurrentToPrevious()
        End If
    End Sub

    Private Sub btnNext_Click(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles btnNext.Click

        If Me.View.CurrentPosition < Me.View.Count - 1 Then
            Me.View.MoveCurrentToNext()
        End If
    End Sub

    Private Sub btnLast_Click(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles btnLast.Click

        Me.View.MoveCurrentToLast()
    End Sub
```

أدخل الكود التالي من أجل زر الحذف

```
    Private Sub btnDelete_Click(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles btnDelete.Click

        If Me.View.CurrentPosition > -1 Then
            Dim row = CType(Me.View.CurrentItem, System.Data.DataRowView).Row
            row.Delete()
        End If
    End Sub
```

حيث تأكدنا أولا أننا لسنا خارج النطاق قبل عملية الحذف ثم استخدمنا الدالة Ctype للحصول على السطر الحالي ثم قمنا بعملية الحذف

من أجل زر الإضافة أدخل الكود التالي

```
    Private Sub btnAdd_Click(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles btnAdd.Click

        Dim row = Me.CustomerData.Customer.NewCustomerRow
        row.LastName = "[New]"
        Me.CustomerData.Customer.AddCustomerRow(row)
        Me.View.MoveCurrentToLast()
    End Sub
```

حيث أنشانا سطرا جديدا وحددنا قيمة افتراضية لحقل الاسم الأخير بما أنه لا يمكن أن يكون فارغا حسب قاعدة البيانات ثم قمنا بإضافته للجدول Customer والسطر الأخير من اجل إظهار السجل الجديد

ومن أجل زر Revert أضف الكود التالي

```
    Private Sub btnRevert_Click(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles btnRevert.Click

        If Me.CustomerData.HasChanges Then
            Dim msg = "Are you sure that You want to loose all your changes?"
            If MessageBox.Show(msg, Me.Title, MessageBoxButton.YesNo) = _
                       MessageBoxResult.Yes Then

                Me.CustomerData.RejectChanges()
            End If
        End If
    End Sub
```

حيث أظهرنا رسالة للمستخدم ليؤكد أنه يرغب فعلا بإلغاء جميع التعديلات التي قام بها وإن وافق نستخدم الطريقة RejectChanges لإلغاء تلك التغييرات

أضف السطر التالي لحدث Loaded للنموذج لربط TableAdaptermanager مع الجدول المناسب

```
Me.taManager.CustomerTableAdapter = taCust
```

وسوف يكون كود زر الحفظ كما يلي

```
    Private Sub btnSave_Click(ByVal sender As System.Object, _
        ByVal e As System.Windows.RoutedEventArgs) Handles btnSave.Click

        Try
            If Me.CustomerData.HasChanges Then
                If Me.taManager.UpdateAll(Me.CustomerData) > 0 Then
                    MsgBox("Saved")
                End If
```

```
        End If
    Catch ex As Exception
        MsgBox(ex.ToString)
    End Try
End Sub
```

حيث استخدمنا حلقة Try لالتقاط أي خطأ يصدر عن قاعدة البيانات وتأكدنا من أنه توجد تعديلات كي نقوم بحفظها وذلك باستخدام الدالة HasChanges ثم استخدمنا الطريقة UpdateAll من أجل عملية الحفظ الفعلية

شغل البرنامج واختبره

القسم الثاني – التعامل مع ListBox

أنشئ مشروعا جديدا من النوع WPF Application وسمه ListBoxData ثم أضف قاعدة البيانات المرفقة الأمثلة إلى المشروع وفي معالج إنشاء مجموعة البيانات اختر الجدول Products وسم مجموعة البيانات ProductDataSet ثم انتقل إلى محرر الكود الخاص بالنافذة وأدخل في الفئة الخاصة بالنافذة الكود التالي من أجل تحميل البيانات

```
Private taProduct As New ProductDataSetTableAdapters.ProductTableAdapter
Private dsProduct As New ProductDataSet

Private Sub Window1_Loaded(ByVal sender As System.Object, _
    ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded

    Me.taProduct.Fill(Me.dsProduct.Product)

End Sub
```

عد إلى محرر النافذة وأضف ListBox إلى سطحها ثم عد إلى محرر الكود وأضف السطر التالي في بعد السطر الموجود في حدث Loaded للنموذج حيث نقوم بضبط قيمة الخاصية ItemsSourc e لصندوق القائمة إلى الجدول Product

```
Me.ListBox1.ItemsSource=Me.dsProduct.Product
```

ونقوم بضبط قيمة الخاصية DisplayMemberPath على name من أجل تحديد اسم العمود الذي نريد أن نظهر بياناته ولعمل ذلك أضف السطر التالي بعد السطر السابق

```
Me.ListBox1.DisplayMemberPath = "Name"
```

شغل التطبيق ولاحظ البيانات في صندوق القائمة

ومن أجل ترتيب البيانات الظاهرة في صندوق القائمة يمكننا استبدال قيمة الخاصية ItemsSource باستعلام لينك ولعمل ذلك استبدل السطر

```
Me.ListBox1.ItemsSource = Me.dsProduct.Product
```

بالسطر

```
Me.ListBox1.ItemsSource = From p In Me.dsProduct.Product Order By p.Name
```

انتقل إلى محرر النافذة وأضف صندوق نصوص أسفل صندوق القائمة حيث سنستخدمه لإظهار وصف المنتج ولعمل ذلك سنحتاج لربط كلا التحكمين مع جدول البيانات وأسهل طريقة في WPF لربط عدة تحكمات هي ضبط ما يسمى بـ DataContext للنافذة بحيث تستخدمها التحكمات افتراضيا

انتقل إلى محرر الكود واستبدل كافة الكود الخاص بالحدث Loaded للنافذة بالكود التالي

```
Me.taProduct.Fill(Me.dsProduct.Product)
Me.DataContext = From p In Me.dsProduct.Product Order By p.Name
```

عد إلى محرر xaml الخاص بالنافذة حيث سنقوم بربط البيانات من هناك حيث سنقوم بتعديل الكود الخاص بصندوق القائمة ليضبط قيمة الخاصيتين ItemsSource و DisplayMemberPath حيث سيصبح كود xaml الخاص بـ ListBox كما يلي مع العلم أن الخاصية

IsSynchronizedWithCurrnetItem تتيح مزامنة القيمة المختارة في صندوق القائمة مع السجل الحالي بحيث تتيح إظهار الوصف المناسب للمنتج المختار في صندوق النصوص

```
<ListBox Margin="12,12,12,120" Name="ListBox1"
         ItemsSource="{Binding Path={}}"
         DisplayMemberPath="Name"
         IsSynchronizedWithCurrentItem="True"/>
```

واستبدل كود صندوق النصوص بالكود التالي من أجل ربطه مع الحقل Description

```
<TextBox Text="{Binding Path=Description}" Height="95"
         Margin="16,0,14,12" Name="TextBox1" VerticalAlignment="Bottom" />
```

يمكننا الاستفادة أيضا من ميزة في WPF هي إمكانية صنع قوالب للتحكمات Templates ولتجربة ذلك أزل صندوق النصوص من على النافذة حيث سنجعله يظهر داخل صندوق القائمة وسع الآن صندوق القائمة ليملأ النموذج ثم عدل كود xaml لصندوق القائمة ليصبح كالتالي حتى يمكننا إضافة التحكمات داخل صندوق القائمة لاحظ حذف الخاصية DisplayMemberPath

```
<ListBox Margin="12" Name="ListBox1"
         ItemsSource="{Binding Path={}}"
         IsSynchronizedWithCurrentItem="True">


</ListBox>
```

داخل كود صندوق القائمة وقبل قسم إغلاقها </ListBox> أضف الكود التالي الذي يعرف قالب بيانات DataTemplate داخل صندوق القائمة ونضيف بداخله StackPanel باتجاه أفقي كي يساعدنا في ترتيب التحكمات بداخل صندوق القائمة لاحظ هنا أن جميع التحكمات في WPF هي مستوعبات Containers يمكن وضع تحكمات بداخلها

```
<ListBox.ItemTemplate>
    <DataTemplate>
        <StackPanel Orientation="Horizontal">

        </StackPanel>
    </DataTemplate>
</ListBox.ItemTemplate>
```

ضع داخل الـ StackPanel تحكمان Label بحيث يكونان مربوطين مع العمودين name و Description كما يلي

```
<Label Width="100" Content="{Binding Path=Name}"></Label>
<Label Width="200" Content="{Binding Path=Description}"></Label>
```

شغل المشروع واختبره – لاحظ ظهور عمودين للبيانات بداخل صندوق القائمة وفي الحقيقة بشكل عام يمكنك وضع أي تحكم بداخل صندوق القائمة مثل الأزرار أو صناديق النصوص أو مربعات الاختيار ... الخ

القسم الثالث – التعامل مع ComboBox

أنشئ مشروعا جديدا وسمه WpfLookup_VB ثم أضف مصدر بيانات جديد باستخدام المعالج واستخدم قاعدة البيانات المرفقة مع الأمثلة واختر الجدول Orders وسم الـ Dataset بالاسم OrdersDataSet ثم أضف مصدر بيانات جديد واختر الجدول Customer بحيث تحدد الحقول CustomerID و FirstName و LastName فقط وسم الـ DataSet بالاسم CustomerLookupDataSet

افتح CustomerLookupDataSet ثم انقر بزر الفأرة اليميني على CustomerTableAdapter ثم اختر Configure وأضف قسم Order By للاستعلام الموجود في الصفحة بحيث يصبح الاستعلام كما يلي

SELECT CustomerID, LastName, FirstName FROM Customer ORDER BY LastName, FirstName

ثم اضغط Finish

أغلق محرر مجموعة البيانات ثم اذهب إلى قائمة View ثم اختر إظهار نافذة Document Outline التي تساعدنا في الانتقال عبر xaml

عدل كود Grid في Xaml من أجل تقسيم الشبكة إلى أربعة أقسام بحيث يصبح كما يلي

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="110*" />
        <ColumnDefinition Width="168*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="222*" />
        <RowDefinition Height="40*" />
    </Grid.RowDefinitions>
</Grid>
```

أدرج StackPanel في كل قسم من القسمين العلويين للشبكة واجعل كل واحد منهما يملأ كامل مساحة قسم من أقسام الشبكة ثم أضف زرين في القسم اليميني السفلي للشبكة بحيث تكون خاصية name لهما btnSave و btnAdd و خاصية Contenet لهما Add و Save ثم أضف أربعة تحكمات Label في الـ stackPanel العلوي اليميني واضبط خاصية Width لها إلى Auto ثم اضبط خاصية Content لها بحيث تكون من الأعلى للأسفل Order ID و Customer و Order Date و Ship Date

أدرج في الـ StackPanel العلوي اليميني تحكم TextBox ثم تحكم ComboBox ثم تحكمي TextBox ثم اضبط خاصية height لهذه التحكمات إلى 28 كي نضبط ارتفاعها بشكل موازي لارتفاع تحكمات Label في القسم المقابل واضبط خاصية Width لهذه التحكمات أيضا إلى Auto ثم اختر الـ Grid واضبط الخاصية Margin إلى 6 كما قم بضبط الخاصية Margin لجميع تحكمات Label و textbox و ComboBox إلى 3

عدل كود xaml لصندوق النصوص الأول كي يتم ربطه مع الحقل OrderID فيصبح كما يلي

```
<TextBox Height="28" Name="TextBox1" Width="Auto" Margin="3"
         Text="{Binding Path=OrderID, Mode=OneWay}" IsReadOnly="True"/>
```

ثم عدل كود xaml لصندوقي النصوص الآخرين كي يتم ربطهما مع الحقلين OrderDate و ShipDate فيصبح كما يلي

```
<TextBox Height="28" Name="TextBox2" Width="Auto" Margin="3"
         Text="{Binding Path=OrderDate}"/>
<TextBox Height="28" Name="TextBox3" Width="Auto" Margin="3"
         Text="{Binding Path=ShipDate}"/>
```

اضبط خصائص الـ Combobox لربط قيمة الإظهار مع الحقل LastName و القيمة المختارة مع CustomerID

```
<ComboBox Height="28" Name="ComboBox1" Width="Auto" Margin="3"
          DisplayMemberPath="LastName" SelectedValuePath="CustomerID"
          SelectedValue="{Binding Path=CustomerID}"/>
```

انتقل إلى محرر الكود الخاص بالنافذة وأضف المتغيرات التالية العامة على مستوى النموذج كي نستخدمها للتعامل مع قواعد البيانات

```
Private OrderData As New OrdersDataSet
Private taOrder As New OrdersDataSetTableAdapters.OrdersTableAdapter
Private taManager As New OrdersDataSetTableAdapters.TableAdapterManager
Private CustomerLookup As New CustomerLookupDataSet.CustomerDataTable
```

أضف الكود التالي في الحدث loaded الخاص بالنافذة كي يتم ربط البيانات مع النموذج وإظهارها

```
Dim taCustomer As New CustomerLookupDataSetTableAdapters.CustomerTableAdapter
taCustomer.Fill(Me.CustomerLookup)

Me.taOrder.Fill(Me.OrderData.Orders)
Me.taManager.OrdersTableAdapter = Me.taOrder

Me.DataContext = Me.OrderData.Orders
Me.ComboBox1.ItemsSource = Me.CustomerLookup
```

حيث قمنا بتعريف TableAdapterManager للجدول Customer واستخدمناه لملئ CustometLookup بالبيانات بواسطة الطريقة Fill وبنفس الأسلوب قمنا بملئ OdrerData بالبيانات باستخدام taOrder ثم قمنا بضبط OrdersTableAdapter العائد لـ taManager ليساوي taOrder ثم ربطنا الـ DataContext الخاصة بالنافذة مع الجدول orders كي نعلم جميع التحكمات الموجودة عليها بأن الخصائص المرتبطة بقاعدة البيانات هي عائدة للجدول Orders ثم ضبطنا خاصية itemsSource العائدة لـ ComboBox مع CustomerLookup من أجل إظهار البيانات بشكل صحيح فيه

عد إلى محرر النافذة وانقر نقرا مزدوجا على الزر Add ليتم إضافة معالج لحدث النقر على الزر ونقلك لمحرر الكود ثم أضف السطر التالي مع المتغيرات التي عرفناها في القسم العام للفئة

```
Private View As CollectionView
```

وفي نهاية الكود الموجود في الحدث Loaded للنافذة أضف الكود التالي

```
Me.View= CollectionViewSource.GetDefaultView(Me.OrderData.Orders)
```

ثم انتقل لحدث Click الخاص بالزر Add وأضف الكود التالي الذي يمثل كود إضافة سطر جديد

```
Dim row = Me.OrderData.Orders.NewOrdersRow
row.CustomerID = 0
Me.OrderData.Orders.AddOrdersRow(row)
Me.View.MoveCurrentToLast()
```

أنشئ معالج لحدث النقر على زر الحفظ وأدخل فيه الكود التالي

```
Try
    If Me.OrderData.HasChanges Then
        If Me.taManager.UpdateAll(Me.OrderData) > 0 Then
            MsgBox("Saved")
        End If
    End If
```

```
Catch ex As Exception
    MsgBox(ex.ToString)
End Try
```

انتقل إلى محرر xaml للنافذة واذهب إلى كود ComboBox الذي سنقوم بتعديله حتى يقوم بإظهار حقلي الاسم الأخير والاسم الأول في البداية احذف الخاصية DisplayMemberPath="LastName" حيث سنضيف ItemsTemplate بدلا عنها التي تمكننا من إضافة التحكمات بداخله لإظهار الاسم الأخير والاسم الأول عدل كود الـComboBox ليصبح كما يلي

```
<ComboBox Height="28" Name="ComboBox1" Width="Auto" Margin="3"
          SelectedValuePath="CustomerID"
          SelectedValue="{Binding Path=CustomerID}">
    <ItemsControl.ItemTemplate>
        <DataTemplate>
            <StackPanel Orientation="Horizontal">
                <TextBlock Name="LastName"
                           Text="{Binding Path=LastName}"/>
                <TextBlock Width="5">,</TextBlock>
                <TextBlock Name="FirstName"
                           Text="{Binding Path=FirstName}"/>
            </StackPanel>
       </DataTemplate>
    </ItemsControl.ItemTemplate>
</ComboBox>
```

حيث أضفنا itemTemplate وحددنا أنها من نوع DataTemplate ثم أضفنا بداخلها تحكم StackPanel بترتيب أفقي الذي سيحتوي على تحكماتنا المضافة ويتحكم بترتيبها ثم أضفنا ثلاثة تحكمات TextBlock داخله الأول تم ربطه مع LastName و الأخير مع FirstName والوسطي وضعنا فيه فاصلة فقط وطريقة الربط هي باستخدام Binding بنفس الطريقة التي استخدمناها مع صناديق النصوص

شغل المشروع واختبره

القسم الرابع – التحقق من البيانات

سنرى في هذا القسم كيف يمكننا التحقق من البيانات واستكشاف الأخطاء فيها حيث توجد ثلاث خطوات أساسية لإظهار رسائل التحقق

في البداية يجب على الغرض Object الذي يحمل البيانات في النموذج أن يحقق الواجهة IDataErrorInfo وإن كنت تستخدم DataSets فهي تقوم بذلك من أجلك عندها يمكننا كتابة أكواد التحقق من البيانات وإظهار رسائل الأخطاء اعتمادا على حاجة المشروع الذي نقوم بتنفيذه. والأمر الثاني الذي يجب علينا عمله هو ضبط الـ Binding الخاص بالتحكمات بأننا نريد إظهار إخبار مرئي عندما يفشل التحقق ونقوم بذلك بضبط الخاصية ValidatesOnDataErrors إلى True وفي الوضع الافتراضي يحيط WPF التحكم بإطار أحمر ولكنه لا يظهر أية رسالة والمرحلة الأخيرة هي ضبط Validation.ErrorTemplate للتحكم كي نتمكن من إظهار الرسائل على التحكمات وجعلها تبدو تماما كما نريد

أنشئ مشروعا جديدا من النوع WPF Application وسمه WPFDataValidation ويتوجب عليك استخدام قاعدة البيانات المرفقة مع الأمثلة حيث سنقوم بكتابة كود للتحقق من أن الحقل LastName في الجدول Customer غير فارغ قبل الحفظ

من قائمة Data اختر الأمر Add New Data Source لإضافة DataSet للمشروع باختيار الجدول Customer وسمها CustomerDataSet ثم افتح CustomerDataSet وانقر بزر الفأرة اليميني على الجدول Customer ثم اختر الأمر View Code ليرينا Partial Class Code وفي كود الفئة الجزئية CustomerDataTable أضف الدالة التالية للتحقق من أن LastName غير فارغ

```
Private Sub CheckLastName(ByVal row As CustomerRow)
    If row.IsNull("LastName") OrElse row.LastName = "" Then
        row.SetColumnError(Me.LastNameColumn, "Please eneter a last name")
    Else
        row.SetColumnError(Me.LastNameColumn, "")
    End If
End Sub
```

أضف إجراء معالجة الحدث Column Changed لـ CustomerDataTable من أجل التحقق من قيمة LastName واجعل كوده يبدو كما يلي

```
Private Sub CustomerDataTable_ColumnChanged(ByVal sender As Object, _
           ByVal e As System.Data.DataColumnChangeEventArgs) _
           Handles Me.ColumnChanged

    If e.Column Is LastNameColumn Then
        Me.CheckLastName(CType(e.Row, CustomerRow))
    End If
End Sub
```

ثم أضف إجراء لمعالجة الحدث TableNewRow لـ CustomerDataTable وسيكون كوده كما يلي

```
Private Sub CustomerDataTable_TableNewRow(ByVal sender As Object, _
            ByVal e As System.Data.DataTableNewRowEventArgs) _
            Handles Me.TableNewRow

    Me.CheckLastName(CType(e.Row, CustomerRow))
End Sub
```

عد إلى محرر تصميم نافذة المشروع كي نقوم بتصميم الواجهة وعدل كود xaml لديك كما في الكود التالي الذي سيشكل واجهة المشروع حيث تم تصميم الواجهة بنفس الطريقة التي اتبعناها سابقا في الأقسام السابقة

```
<Window x:Class="Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="300" Width="300">
    <Grid Margin="6">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="95*" />
            <ColumnDefinition Width="183*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="207*" />
            <RowDefinition Height="55*" />
        </Grid.RowDefinitions>
        <StackPanel Name="StackPanel1">
            <Label Height="28" Name="Label1" Width="Auto" Margin="3">
                  Last Name</Label>
            <Label Height="28" Name="Label2" Width="Auto" Margin="3">
                   First Name</Label>
            <Label Height="28" Name="Label3" Width="Auto"
                   Margin="3">Address</Label>
            <Label Height="28" Name="Label4" Width="Auto"
                   Margin="3">City</Label>
            <Label Height="28" Name="Label5" Width="Auto"
                   Margin="3">State</Label>
            <Label Height="28" Name="Label6" Width="Auto" Margin="3">
                   Zip</Label>
        </StackPanel>
        <StackPanel Grid.Column="1" Name="StackPanel2">
            <TextBox Height="28" Name="TextBox1" Width="Auto" Margin="3"
                     Text="{Binding Path=LastName}"/>
            <TextBox Height="28" Name="TextBox2" Width="Auto" Margin="3"
                     Text="{Binding Path=FirstName}"/>
            <TextBox Height="28" Name="TextBox3" Width="Auto" Margin="3"
                     Text="{Binding Path=Address}"/>
            <TextBox Height="28" Name="TextBox4" Width="Auto" Margin="3"
                    Text="{Binding Path=City}" />
            <TextBox Height="28" Name="TextBox5" Width="Auto" Margin="3"
                     Text="{Binding Path=State}"/>
            <TextBox Height="28" Name="TextBox6" Width="Auto" Margin="3"
                     Text="{Binding Path=ZIP}"/>
        </StackPanel>
        <Button Grid.Column="1" Grid.Row="1" Margin="13,20,0,12"
           Name="btnAdd" HorizontalAlignment="Left" Width="75">Add</Button>
        <Button Grid.Column="1" Grid.Row="1" HorizontalAlignment="Right"
            Margin="0,20,6,12" Name="btnSave" Width="75">Save</Button>
    </Grid>
</Window>
```

ومن أجل عملية التحقق من البيانات يجب علينا إضافة خاصية إضافية لـ Binding هي ValidatesOnDataErrors و ضبطها إلى القيمة True وذلك لجميع التحكمات TextBox في كود xaml السابق مثال

```
Text="{Binding Path=ZIP, ValidatesOnDataErrors=True}
```

انتقل إلى محرر الكود الخاص بالنافذة وأدخل الكود التالي الخاص بملئ البيانات حيث تمت كتابته بنفس الأسلوب المتبع في الأقسام السابقة

```
Class Window1
    Private CustomerData As New CustomerDataSet
    Private taCust As New CustomerDataSetTableAdapters.CustomerTableAdapter
    Private taManager As New CustomerDataSetTableAdapters.TableAdapterManager
    Private View As CollectionView
```

```
    Private Sub Window1_Loaded(ByVal sender As System.Object, _
            ByVal e As System.Windows.RoutedEventArgs) Handles MyBase.Loaded

        Me.taCust.Fill(Me.CustomerData.Customer)
        Me.taManager.CustomerTableAdapter = taCust
        Me.DataContext = Me.CustomerData.Customer
        Me.View = _
    CType(CollectionViewSource.GetDefaultView(Me.CustomerData.Customer), _
   CollectionView)


    End Sub
End Class
```

شغل المشروع وتأكد من رؤية البيانات قم بمسح البيانات من الحقل LastName وانتقل للحقل الذي يليه ستلاحظ أن صندوق النصوص الفارغ قد تمت إحاطته باللون الأحمر دلالة على فشل التحقق من البيانات

أضف إجراء معالجة النقر على الزر Add وضع فيه الكود التالي

```
If Not Me.CustomerData.HasErrors Then
    Dim row = Me.CustomerData.Customer.NewCustomerRow()
    row.LastName = ""
    Me.CustomerData.Customer.AddCustomerRow(row)
    Me.View.MoveCurrentToLast()
End If
```

أضف إجراء لمعالجة حدث النقر على الزر Save وضع فيه الكود التالي

```
Try
    If Me.CustomerData.HasErrors Then
        MsgBox("Please correct the errors first")
    Else
        If Me.CustomerData.HasChanges Then
            If Me.taManager.UpdateAll(Me.CustomerData) > 0 Then
                MsgBox("Saved.")
            End If
        End If
    End If
Catch ex As Exception
    MsgBox(ex.ToString)
End Try
```

لاحظ في زري الإضافة والحفظ أننا استخدمنا الطريقة HasErrors العائدة لـ CuastomerData كي نتحقق من عدم وجود أخطاء في البيانات قبل المتابعة

افتح الملف Application.xaml واجعل الكود فيه يبدو كما يلي من أجل إنشاء ValidationTemplate

```
<Application x:Class="Application"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    StartupUri="Window1.xaml">
    <Application.Resources>
        <Style x:Key="myErrorTemplate" TargetType="Control">
            <Setter Property="Validation.ErrorTemplate">
                <Setter.Value>
                    <ControlTemplate>
```

```
                        <DockPanel LastChildFill="True">
                            <TextBlock DockPanel.Dock="Right"
                               Foreground="Red"
                               FontSize="14pt"
                               Margin="-15,0,0,0" FontWeight="Bold">*
                            </TextBlock>
                            <Border BorderBrush="Red" BorderThickness="1">
                                <AdornedElementPlaceholder Name="myControl"/>
                            </Border>
                        </DockPanel>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
            <Style.Triggers>
                <Trigger Property="Validation.HasError" Value="true">
                    <Setter Property="ToolTip"
                  Value="{Binding RelativeSource={x:Static RelativeSource.Self},
                        Path=(Validation.Errors)[0].ErrorContent}"/>
                </Trigger>
            </Style.Triggers>
        </Style>
       <Style TargetType="TextBox" BasedOn="{StaticResource myErrorTemplate}" />
      <Style TargetType="CheckBox" BasedOn="{StaticResource myErrorTemplate}" />
      <Style TargetType="ComboBox" BasedOn="{StaticResource myErrorTemplate}" />
    </Application.Resources>
</Application>
```

حيث ضبطنا الخاصية Setter لـ Validation.ErrorTemplate من أجل تحديد أن هذه الـ Control Template هي خاصة بوقت حدوث خطأ وقسم Style.Triggers يقوم بإظهار ToolTip على التحكم عند حدوث خطأ