أبوبكر شرف الدين سويدان

DataSet, DataTable

تحت المجهر في

Visual Basic.net 2010

2013

نسخة إلكترونية غير منقحة

بسم اللّه الرحمن الرحيم

ولسوف يعطيك ربك فترضى

الإهداء

إلى أرواح شهداء بلادي..

جدول المحتويات

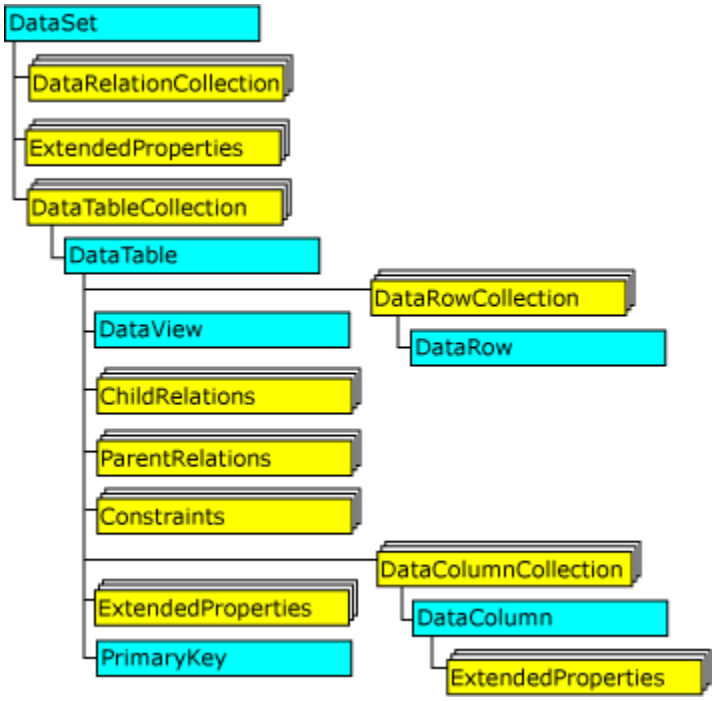
الباب الأول

DataSet

الكلاس DataSet

هي أماكن لتخزين جداول البيانات في الذاكرة، وهي مؤقتة، تفقد البيانات التي تحملها بانتهاء العملية. كل DataSet تحتوي على جدول بيانات DataTable أو أكثر وكذلك على علاقات DataRelation يمكن إنشاؤها بين الجداول، وعلى DataView. كل DataTable يحتوي على سجلات أو صفوف DataRows، وكل DataRow يحتوي على مجموعة من الأعمدة أو الحقول DataColumns. وهذا الكلاس يتبع مباشرة لفضاء الأسماء System.Data.

والشكل التالي يوضح تركيبة الـDataSet:



مصدر مخطط الـDataSet هو: http://msdn.microsoft.com/en-us/library/zb0sdh0b%28v=vs.100%29.aspx

ملاحظات:

إنشاء DataSet

نستطيع إنشاء DataSet جديدة من خلال استدعاء "منشيء الـDataSets"، أو ما يسمى بالإنجليزية DataSet Constructor، وبشكل اختياري؛ يمكننا إعطاؤها وصفاً ، وفي حال ترك هذا الخيار يتم إعطاؤها اسماً افتراضياً وهو: "NewDataSet".

```
Dim customerOrders As DataSet = New DataSet("CustomerOrders")
```

أو:

```
Dim customerOrders As New DataSet("CustomerOrders")
MsgBox(customerOrders.DataSetName, MsgBoxStyle.Information, "DataSet Name is:")
```

في الكود السابق تم إنشاء DataSet بالاسم CustomerOrders، وفي الكود التالي يتم إنشاء DataSet بالاسم الافتراضي:

```
Dim customerOrders As New DataSet
```

ولنجرب هذا الكود أيضاً:

```
Dim customerOrders As New DataSet
MsgBox(customerOrders.DataSetName, MsgBoxStyle.Information, "DataSet Name is:")
```

أهم خصائص الـDataSet:

الخاصية DataSetName:

ومن خلالها يمكن ضبط / استرجاع اسم الـDataSet الحالية.

```
Dim dataSet As DataSet
dataSet = New DataSet("SuppliersProducts")
MsgBox(dataSet.DataSetName, MsgBoxStyle.Information, "DataSet Name is:")
```

الخاصية IsInitialized:

وترجع قيمة منطقية تحدد ما إذا تم تهيئة الـDataSet أم لا.

```
Dim dataSet As DataSet
dataSet = New DataSet("SuppliersProducts")
MsgBox(dataSet.IsInitialized, MsgBoxStyle.Information, "IsInitialized")
```

ملاحظات:

الخاصية Tables:

ومن خلالها يمكن استرجاع تجمع الـDataTables التابعة لها. أو اختيار جدول محدد عن طريق اسمه أو رقم تسلسله. سأتطرق لها في أمثلة الفقرات التالية، ولاحقاً في باب DataTable إن شاء اللّه.

أهم وظائف الـDataSet:

الوظيفة AcceptChanges:

وتقوم بتطبيق كافة التغييرات التي طرأت على الـDataSet ومحتوياتها، منذ آخر مرة تم استدعاء هذه الوظيفة.

```
Private Sub AcceptChanges()
   Dim myDataSet As DataSet
   myDataSet = new DataSet()

   ' Not shown: methods to fill the DataSet with data.
   Dim t As DataTable

   t = myDataSet.Tables("Suppliers")

   ' Add a DataRow to a table.
   Dim myRow As DataRow
   myRow = t.NewRow()
   myRow("CompanyID") = "NWTRADECO"
   myRow("CompanyName") = "NortWest Trade Company"

   ' Add the row.
   t.Rows.Add( myRow )

   ' Calling AcceptChanges on the DataSet causes AcceptChanges to be
   ' called on all subordinate objects.
   myDataSet.AcceptChanges()
End Sub
```

الوظيفة Clear:

وتقوم بمسح كافة بيانات الـDataSet وذلك من خلال مسح كافة السجلات في كافة الجداول التي تتبع هذه الـDataSet.

```
dataSet.Clear()
```

الوظيفة Clone:

وتقوم بنسخ تركيبة الـDataSet بما في ذلك تركيبات الجداول والعلاقات، دون نسخ البيانات التي تحملها.

ملاحظات:

.....

.....

.....

```
Dim dataSet As DataSet
dataSet = New DataSet("SuppliersProducts")
MsgBox(dataSet.DataSetName, MsgBoxStyle.Information, "DataSet Name is:")

Dim ClonedDataSet As DataSet
ClonedDataSet = dataSet.Clone()
MsgBox(ClonedDataSet.DataSetName, MsgBoxStyle.Information, "ClonedDataSet Name is:")
```

ولنجرب الكود بدون إعطاء اسم للـDataSet:

```
Dim dataSet As DataSet
dataSet = New DataSet
MsgBox(dataSet.DataSetName, MsgBoxStyle.Information, "DataSet Name is:")

Dim ClonedDataSet As DataSet
ClonedDataSet = dataSet.Clone()
MsgBox(ClonedDataSet.DataSetName, MsgBoxStyle.Information, "ClonedDataSet Name is:")
```

الوظيفة Copy:

وتقوم بنسخ تركيبة الـDataSet بما في ذلك تركيبات الجداول والعلاقات مع كافة البيانات التي تحملها.

```
Dim dataSet As DataSet
dataSet = New DataSet("SuppliersProducts")
MsgBox(dataSet.DataSetName, MsgBoxStyle.Information, "DataSet Name is:")

Dim CopiedDataSet As DataSet
ClonedDataSet = dataSet.Copy()
MsgBox(CopiedDataSet.DataSetName, MsgBoxStyle.Information, "CopiedDataSet Name is:")
```

ولنجرب الكود بدون إعطاء اسم للـDataSet:

```
Dim dataSet As DataSet
dataSet = New DataSet
MsgBox(dataSet.DataSetName, MsgBoxStyle.Information, "DataSet Name is:")

Dim CopiedDataSet As DataSet
ClonedDataSet = dataSet.Copy()
MsgBox(CopiedDataSet.DataSetName, MsgBoxStyle.Information, "CopiedDataSet Name is:")
```

ملاحظات:

الوظيفة GetChanges:

تأخذ نسخة من الـDataSet والتي تم إجراء عمليات تعديل وتغيير عليها منذ أن تم تعبئتها بالبيانات أو منذ آخر مرة تم تطبيق الوظيفة AcceptChanges، وتخزنها في DataSet جديدة.

وسأتطرق للمزيد في موضوع: معالجة الحزم، لاحقاً إن شاء اللّه.

الوظيفة HasChanges:

وترجع قيمة منطقية (True/False) تحدد ما إذا تم تغيير في محتويات الـDataSet، والتغييرات تشمل إضافة صفوف جديدة، أو تعديل صفوف موجودة، أو حذف لبعض أو كل الصفوف.

```
If dataSet.HasChanges = True Then
     MsgBox("There are some changes")
Else
     MsgBox("The DataSet was not changed")
End If
```

الوظيفة Merge(DataSet):

وتقوم بدمج الـDataSet وتركيبتها كاملةً مع تركيبة الـDataSet الحالية.

ولكي نفهمها نكتب:

```
Dim FirstDataSet As New DataSet("FirstDataSet")
Dim FirstTable As New DataTable("FirstTable")
```

وهنا قمنا بإنشاء DataSet وDataTable، الخطوة التالية هي ضم الجدول إلى FirstDataSet:

```
FirstDataSet.Tables.Add(FirstTable)
```

والآن: FirstDataSet تحتوي على جدول واحد هو FirstTable، ويمكننا التحقق من ذلك عن طريق كتابة:

```
MsgBox(FirstDataSet.Tables(0).TableName)
```

والآن: نقوم بإنشاء DataSet جديدة باسم SecondDataSet فارغة ولا تحتوي على أي جدول:

```
Dim SecondDataSet As New DataSet("SecondDataSet")
```

ملاحظات:

.....

.....

.....

ونقوم بدمج محتويات الـFirstDataSet مع محتويات الـSecondDataSet:

```
SecondDataSet.Merge(FirstDataSet)
```

وهنا تمت عملية الدمج، فعند كتابة:

```
MsgBox(SecondDataSet.Tables(0).TableName)
```

يعطينا نفس اسم الجدول الموجود في FirstDataSet، وهذا دليل على صحة العملية.

الوظيفة Merge(DataTable):

سأشرحها لاحقاً عند الحديث عن الـDataTable إن شاء اللّه.

الوظيفة Merge(DataSet, Boolean):

وتقوم بدمج الـDataSet وتركيبتها كاملةً مع تركيبة الـDataSet الحالية، مع قيمة منطقية تحدد ما إذا كانت عملية الدمج تشمل التغييرات أم لا.

```
SecondDataSet.Merge(FirstDataSet, False)
```

هنا تتم عملية الدمج بحيث لا تشمل التغييرات.

```
SecondDataSet.Merge(FirstDataSet, True)
```

هنا تتم عملية الدمج بحيث تشمل التغييرات.

الوظيفة RejectChanges:

وتقوم بإلغاء كافة التغييرات التي طرأت على الـDataSet منذ تعبئتها بالبيانات لأول مرة، أو منذ آخر مرة تم استدعاء الطريقة AcceptChanges فيها.

سأتكلم بمزيد من التفصيل عن الوظيفتين (AcceptChanges, RejectChanges) لاحقاً في موضوع: معالجة الحزم إن شاء اللّه.

ملاحظات:

الوظيفة Reset:

وتقوم بمسح محتويات الـDataSet وإرجاعها إلى حالتها السابقة.

```
Dim FirstDataSet As New DataSet("FirstDataSet")
Dim FirstTable As New DataTable("FirstTable")
FirstDataSet.Tables.Add(FirstTable)
MsgBox(FirstDataSet.Tables(0).TableName)

Dim SecondDataSet As New DataSet("SecondDataSet")
SecondDataSet.Merge(FirstDataSet)
MsgBox(SecondDataSet.Tables(0).TableName)

SecondDataSet.Reset()
MsgBox(SecondDataSet.Tables(0).TableName, , "after reset is called")
```

عند تنفيذ الكود السابق، وبعد استدعاء الوظيفة Reset يظهر خطأ يقول: Cannot find table 0 ، ويعني أن الـSecondDataset أرجعت إلى سابق عهدها فارغة.

ملاحظات:

الخلاصة

عرفنا في هذا الباب ما هي الـDataSet، وكيفية إنشائها وتعديل خصائصها والاستفادة من وظائفها.

وكما تعلمون، فإن الـDataSet هي كائن مؤقت، منفصل عن قواعد البيانات الفعلية، ولكن يمكننا تعبئتها بالبيانات، وتطبيق التغييرات الطارئة عليها من وإلى قاعدة البيانات نفسها من خلال أدوات أخرى سأتكلم عنها لاحقاً في نهاية هذه المذكرة إن شاء اللّه.

مازالت هناك الكثير من الخصائص الأخرى بالطبع، وكذلك وظائف أخرى تفيد المتقدمين في التعامل مع قواعد البيانات في لغة فجول بيسك 2010، ولكنني آثرت الحديث عن أهم تلك الخصائص والوظائف، مراعياً أن المذكرة موجهة للمبتدئين أمثالي.

ذيّلتُ كل صفحة من صفحات هذا الباب بمساحة فارغة لكتابة ملاحظاتكم، وكذلك لتدوين الأخطاء التي ربما وقعتُ فيها نتيجة فهمي المحدود، وقلة مصادري. فأرجو منكم إرسال تعليقاتكم وملاحظاتكم وتصحيحاتكم إليّ عن طريق بريدي الإلكتروني:

abubakerswedan@yahoo.com

كي أتمكن من تصحيح المعلومات في هذه المذكرة وتوفير نسخة جديدة منقحة ومزيدة قدر الإمكان.

هل أنتم مستعدون الآن للتعرف على DataTable ودراسته تحت المجهر؟

عن نفسي، سأعد فنجان قهوة ، وأفعل..

ملاحظات:

الباب الثاني

DataTable

الكلاس DataTable

ويمثل جدولاً واحداً ضمن الـDataSet. وكل جدول لا بد أن تتوفر فيه الكلاسات: DataColumn وDataRow، فللجدول حقول (أعمدة Columns) وسجلات (صفوف Rows). وهذا الكلاس يتبع مباشرة لفضاء الأسماء System.Data.

إنشاء جدول Creating a DataTable

هذا الكلاس يتبع مباشرة لفضاء الأسماء System.Data، ولإنشاء جدول **نقوم بتعريف كائن DataTable**، وبشكل اختياري يمكننا منحه اسماً:

```
Dim UnNamedTable As New DataTable
```

في السطر السابق تم تعريف كائن يمثل جدولاً دون أن نسميه، وفي السطر التالي، نقوم بتعريف كائن يمثل جدولاً باسم Customers:

```
Dim Customers As New DataTable("Customers")
```

- إذا لم نحدد اسماً للجدول، يقوم ADO.NET بتسميته Table1، والجدول الذي يليه Table2 وهكذا.
- بعد إنشاء كائن الجدول، يمكننا تعديل قيمة الخاصية TableName لإعطائه اسماً أو تعديله، كما يمكننا التعامل مع بقية الخصائص والوظائف والأحداث. وكذلك يتم **إنشاء كلاس يمثل تجمعاً للجداول DataTableCollection وضم الجدول الجديد له**، وهذا الكلاس يتبع فضاء الأسماء System.Data.

إضافة DataTable إلى DatatSet

يضاف الجدول DataTable المنشأ حديثاً إلى الـDataSet بالصورة التالية:

```
Dim EmployeesDataSet As New DataSet("Employees DataSet")
Dim PersonalInfoTable As New DataTable

EmployeesDataSet.Tables.Add(PersonalInfoTable)
```

وذلك من خلال الوظيفة Add التابعة للخاصية Tables التابعة للـDataSet.

ملاحظات:

.....

.....

.....

أهم خصائص هذا الكلاس:

الخاصية TableName:

ومن خلالها يمكن ضبط / استرجاع اسم الـDataTable.

```
Dim FirstTable As New DataTable
MsgBox(FirstTable.TableName)
FirstTable.TableName = "Nationalities"
MsgBox(FirstTable.TableName)
```

الخاصية Columns:

ومن خلالها يمكن استرجاع تجمع الأعمدة أو الحقول التابعة للـDataTable.

```
Private Sub PrintValues(ByVal table As DataTable)
    Dim row As DataRow
    Dim column As DataColumn
    For Each row in table.Rows
       For Each column In table.Columns
          Msgbox(row(column))
       Next
    Next
End Sub
```

وستتضح الصورة أكثر بعد الحديث عن كل من : DataColumn و DataRow.

الخاصية DataSet:

ومن خلالها يمكن استرجاع الـDataSet التي يتبعها هذا الـDataTable.

```
Dim EmployeesDataSet As New DataSet("Employees DataSet")
Dim PersonalInfoTable As New DataTable("Personal Info")

EmployeesDataSet.Tables.Add(PersonalInfoTable)
MsgBox(PersonalInfoTable.DataSet.DataSetName)
```

الخاصية PrimaryKey:

ومن خلالها يمكن ضبط / استرجاع مصفوفة من الأعمدة أو الحقول التي تعمل كمفاتيح أساسية للـDataTable. وسأشرح عنها عند الحديث عن DataColumn.

الخاصية Rows:

ومن خلالها يمكن استرجاع تجمع الصفوف التي تتبع هذا الـDataTable. وسأشرح عنها عند الحديث عن DataRow.

ملاحظات:

.....

.....

.....

أهم الوظائف Methodes التي تتبع الكلاس DataTable:

قبل أن أعرض على حضراتكم هذه الوظائف، أود أن أوضح أن بعضها يحتاج إلى المزيد من الشرح لاحقاً في مواضيع هذا الكتاب، إما لأن هناك بعض المعلومات الناقصة (إلى حد الآن) أو أن الوقت المناسب لشرحها لم يحن بعد.

الوظيفة Clear:

وتقوم بمسح محتويات الجدول من البيانات فقط، ولا تحذف تركيبته.

```
PersonalInfoTable.Clear()
```

الوظيفة Clone:

وتقوم بنسخ تركيبة الجدول دون البيانات.

```
Dim PersonalInfoTable As New DataTable
PersonalInfoTable.TableName = "Original Table"

Dim ClonedTable As New DataTable
ClonedTable = PersonalInfoTable.Clone()
MsgBox(ClonedTable.TableName)
```

الوظيفة Copy:

وتقوم بنسخ تركيبة ومحتويات الجدول إلى جدول آخر .

```
Dim PersonalInfoTable As New DataTable
PersonalInfoTable.TableName = "Original Table"

Dim CopiedTable As New DataTable
CopiedTable = PersonalInfoTable.Copy()
MsgBox(CopiedTable.TableName)
```

الوظيفة ImportRow:

وتقوم بنسخ DataRow إلى جدول DataTable مع الاحتفاظ بإعداداته السابقة والقيم التي يحملها. وسأشرح عنها عند الحديث عن DataRow.

الوظيفة Merge(DataTable):

وتقوم بدمج الجدول المعطى مع الجدول الحالي. سأشرحها لاحقاً في استعراض العمليات على الجداول.

ملاحظات:

الوظيفة Merge(DataTable, Boolean):

وتقوم بدمج الجدول المعطى مع الجدول الحالي مع إعطاء الإمكانية لحفظ التغييرات من عدمه. . سأشرحها لاحقاً في استعراض العمليات على الجداول.

الوظيفة NewRow:

وتقوم بإنشاء DataRow بنفس تركيبة الجدول. بحيث تراعى أنواع البيانات وأحجامها وإعداداتها لكل عمود.

الوظيفة Reset:

وتقوم بإعادة ضبط الجدول إلى حالته القديمة.

الوظيفة Select:

وتقوم باسترجاع مصفوفة تمثل كافة الـDataRows بالجدول.

الوظيفة Select(String):

وتقوم باسترجاع مصفوفة تمثل الـDataRows التي تخضع للمعايير المحددة.

الوظيفة Select(String, String):

وتقوم باسترجاع مصفوفة تمثل الـDataRows التي تخضع للمعايير المحددة بترتيب محدد.

الوظيفة Select(String, String, DataViewRowState):

وتقوم باسترجاع مصفوفة تمثل الـDataRows التي تخضع للمعايير المحددة بترتيب يطابق الحالة المحددة.

ملاحظات:

إضافة أعمدة (حقول) للجدول Adding Columns to DataTable

الحمد للّه، وصلنا إلى الجزء المثير والممتع! إذ أنه بعد تعريف الكائن الذي يمثل DataTable، سيكون ذلك الكائن فارغاً، وبالتالي لا نستفيد منه شيئاً! والخطوة القادمة هي بالتأكيد إنشاء الحقول أو Data Columns.

يحتوي فضاء الأسماء System.Data أيضاً على كلاس يسمى **DataColumn** والذي يمثل حقلاً واحداً في تركيبة الجدول **Table Schema**.

ولإضافة حقل إلى الجدول، نقوم بتعريف كائن يمثل الجدول:

```
Dim Customers As New DataTable("Customers")
```

ثم نعرف كائناً آخر يمثل الحقل، **وأنبه هنا إلى أن الحد الأدنى لتعريف الحقل هو توفير اسم الحقل ونوع البيانات** التي يحملها:

```
Dim CustomerID As New DataColumn("ID", GetType(Long))
```

ففي السطر السابق، تم تعريف الكائن CustomerID على أنه كائن يمثل حقلاً اسمه ID ونوع بياناته Long.

بقيت خطوة أخيرة، وهي ضم الكائن الجديد للجدول:

```
Customers.Columns.Add(CustomerID)
```

وفي السطر السابق، تم استخدام الوظيفة Add التابعة لتجمع الحقول Columns لإضافة الحقل الجديد إلى الجدول. ونفعل ذلك لكل عمود جديد.

أو بطريقة أخرى، وهي باستخدام الوظيفة Add التابعة لتجمع الحقول Columns لإضافة حقول جديدة **دون** تعريف كائنات تمثل كل الحقول:

```
Dim Customers As New DataTable("Customers")
Customers.Columns.Add("ID", GetType(Long))
Customers.Columns.Add("CustomerName", GetType(String))
Customers.Columns.Add("BirthDay", GetType(Date))
```

ملاحظات:

خصائص الكلاس DataColumn

الكود السابق يوضح طريقة إنشاء الحقول بأبسط الطرق، وبالحد الأدنى من المعلومات المتعلقة بالحقول. ولكن هناك الكثير من الخصائص الأخرى التي تجعل من إنشاء الحقول وضمها للجدول أكثر احترافية وإتقاناً، أذكر منها على سبيل المثال:

الخاصية AllowDBNull:

وتحمل قيمة منطقية Boolean، وتحدد ما إذا كان من المسموح أن لا يحمل هذا الحقل أية قيمة، أي قيمته = Null.

```
Dim column As DataColumn
column = New DataColumn("ID", System.Type.GetType("System.Int32"))
column.AllowDBNull = True
```

أو

```
Dim column As DataColumn
column = New DataColumn("ID", System.Type.GetType("System.Int32"))
column.AllowDBNull = False
```

ويمكن استعمال الكلاس System.DBNull لاختبار القيمة قبل تخزينها لاحقاً في هذا الحقل (تلقائياً: كل الحقول تسمح بالقيمة Null):

```
If (DBNull.Value.Equals(FieldValue)) Then
    Do something
End If
```

وكذلك توجد دالة في فجول بيسك وظيفتها فحص قيم المتغيرات:

```
If (IsDBNull(FieldValue)) Then
    Do something
End If
```

الخصائص AutoIncrement، وAutoIncrementSeed وAutoIncrementStep:

تدير هذه الخصائص عملية الزيادة التلقائية لقيم الحقول التي تحمل هذه الميزة، فعند تطبيقها، يتم زيادة القيم تلقائياً عند إضافة سجلات جديدة بمقادير تتحدد من خلال الخاصيتين AutoIncrementSeed وAutoIncrementStep. وبشكل تلقائي فإن الخاصية AutoIncrement مضبوطة على الوضع False، وبالتالي فإن هذه الآلية لا تعمل.

```
Dim column As DataColumn = New DataColumn
column.DataType = System.Type.GetType("System.Int32")
With column
     .AutoIncrement = True
     .AutoIncrementSeed = 1000
     .AutoIncrementStep = 10
End With
```

ملاحظات:

.....

.....

.....

الخاصية ColumnName:

وهي خاصية نصية تمثل اسم الحقل. ويتم تحديدها وضبطها وقت إنشاء الحقل.

```
Dim column As New DataColumn
With column
       .ColumnName = "CustomerName"
       .AllowDBNull = True
End With
MsgBox(column.ColumnName)
```

الخاصية DataType:

وتمثل نوع البيانات التي سيحملها الحقل. ويتم تحديدها وضبطها وقت إنشاء الحقل، وإذا ما تم إضافة البيانات، لا يمكن تعديل نوعها.

```
Dim column As New DataColumn
With column
       .ColumnName = "FirstName"
       .DataType = System.Type.GetType("System.String")
End With
MsgBox(column.DataType.ToString)
```

الخاصية DefaultValue:

وتمثل القيمة الافتراضية للحقل، ويمكن لأي حقل أن يحمل قيمة افتراضية، ويتم تخزين هذه القيمة كلما تم إنشاء سجل جديد، ويمكن تعديلها لاحقاً دون أية مشكلة.

```
Dim column As New DataColumn
With column
       .ColumnName = "FirstName"
       .DataType = System.Type.GetType("System.String")
       .DefaultValue = "None"
End With
MsgBox(column.DefaultValue)
```

الخاصية MaxLength:

وهي خاصة بالحقول التي تحمل قيماً نصية، وتحدد طول النص الذي سيحمله هذا الحقل، وتلقائياً: قيمته هي 1-، بمعنى أنه لا حد لطول النص.

```
Dim column As New DataColumn
With column
       .ColumnName = "FirstName"
       .DataType = System.Type.GetType("System.String")
       .MaxLength = 20
       .DefaultValue = "None"
End With
```

ملاحظات:

الخاصية ReadOnly:

حقول القراءة فقط لا يمكن تعديل قيمها في أي سجل قد تمت إضافته فعلاً للجدول، وتلقائياً: جميع قيم الحقول قابلة للتعديل.

```
Dim column As New DataColumn
With column
       .ColumnName = "FirstName"
       .DataType = System.Type.GetType("System.String")
       .ReadOnly = True
       .DefaultValue = "None"
End With
```

الخاصية Unique:

وهي خاصية منطقية، إذا ضبطت بالقيمة True، فإن الحقل لا يسمح بحمل قيمتين متساويتين، وكذلك في هذا النوع من الحقول لا يمكن تخزين قيمة Null.

```
Dim column As New DataColumn
With column
       .ColumnName = "ID"
       .DataType = System.Type.GetType("System.Int32")
       .Unique = True
End With
```

وهناك **خاصية** تتبع الكلاس **DataTable** وهي **PrimaryKey**، ومن خلالها – كما أسلفت - يمكن ضبط / استرجاع مصفوفة من الأعمدة أو الحقول والتي تعمل كمفاتيح أساسية للـDataTable:

```
Dim Customers As New DataTable("Customers")
Customers.Columns.Add("ID", GetType(Long))
Customers.Columns.Add("CustomerName", GetType(String))
Customers.Columns.Add("BirthDay", GetType(Date))

Customers.PrimaryKey = {Customer.Columns("ID")}
```

عند إضافة DataColumn للجدول DataTable، يتم إضافته إلى ما يعرف بـColumns Collection أو تجمع الأعمدة (الحقول). هذه الحقول لا تحمل أية قيم إلى حد الآن، وإنما يتم إضافة البيانات عن طريق ما يعرف بـRows Collection أو تجمع الصفوف والذي يتبع الكائن DataTable كما سنرى لاحقاً إن شاء اللّه.

ملاحظات:

إنشاء صفوف جديدة Creating New Rows

إن عملية إنشاء الجداول Creating DataTables وأعمدتها DataColumns هي عملية أساسية قبل المضي قدماً للخطوة اللاحقة وهي: عملية إضافة البيانات للجدول أو "إنشاء سجلات (صفوف Rows) جديدة".

وكما تعلمنا سابقاً، فإننا إذ ننشئ الأعمدة (الحقول DataColumns) إنما ننشئ قوالب لا نهائية من حيث العدد، يمكننا حفظ البيانات فيها بالصورة التي نريد.

والآن، أود أن أحدثكم عن الكلاس DataRow..

الكلاس DataRow

ويمثل صفاً كاملاً من الـDataColumns التي يتكون منها الـDataTables. آخذاً في الاعتبار نوع وحجم كل عمود DataColumn في الجدول.

أهم خصائص الكلاس DataRow

الخاصية Item(DataColumn):

ومن خلالها يتم ضبط / استرجاع القيمة المخزنة بحقل محدد.

```
Dim Customers As New DataTable("Customers")
Customers.Columns.Add("ID", GetType(Long))
Customers.Columns.Add("CustomerName", GetType(String))
Customers.Columns.Add("RegistrationDate", GetType(Date))
Customers.PrimaryKey = {Customers.Columns("ID")}

Dim NewRow As DataRow
NewRow = Customers.NewRow
With NewRow
       .Item("ID") = 1
       .Item("CustomerName") = "Abubaker Swedan"
       .Item("RegistrationDate") = Now.Date
End With

Customers.Rows.Add(NewRow)
MsgBox(Customers.Rows(0).Item("CustomerName"))
MsgBox(Customers.Rows(0).Item("RegistrationDate"))
```

ملاحظات:

الخاصية Item(Int32):

ومن خلالها يتم ضبط / استرجاع القيمة المخزنة بحقل محدد بدلالة رقم فهرسه.

```
Dim Customers As New DataTable("Customers")
Customers.Columns.Add("ID", GetType(Long))
Customers.Columns.Add("CustomerName", GetType(String))
Customers.Columns.Add("RegistrationDate", GetType(Date))
Customers.PrimaryKey = {Customers.Columns("ID")}

Dim NewRow As DataRow
NewRow = Customers.NewRow
With NewRow
       .Item(0) = 1
       .Item(1) = "Abubaker Swedan"
       .Item(2) = Now.Date
End With

Customers.Rows.Add(NewRow)
MsgBox(Customers.Rows(0).Item(1))
MsgBox(Customers.Rows(0).Item(2))
```

الخاصية Item(String):

ومن خلالها يتم ضبط / استرجاع القيمة المخزنة بحقل محدد بدلالة اسمه.

```
Dim Customers As New DataTable("Customers")
Customers.Columns.Add("ID", GetType(Long))
Customers.Columns.Add("CustomerName", GetType(String))
Customers.Columns.Add("RegistrationDate", GetType(Date))
Customers.PrimaryKey = {Customers.Columns("ID")}

Dim NewRow As DataRow
NewRow = Customers.NewRow
With NewRow
       .Item("ID") = 1
       .Item("CustomerName") = "Abubaker Swedan"
       .Item("RegistrationDate") = Now.Date
End With

Customers.Rows.Add(NewRow)
MsgBox(Customers.Rows(0).Item("CustomerName"))
MsgBox(Customers.Rows(0).Item("RegistrationDate"))
```

ملاحظات:

الخاصية Table:

ومن خلالها يتم استرجاع اسم الجدول DataTable الذي ينتمي إليه الصف.

```
Dim Customers As New DataTable("Customers")
Customers.Columns.Add("ID", GetType(Long))
Customers.Columns.Add("CustomerName", GetType(String))
Customers.Columns.Add("RegistrationDate", GetType(Date))
Customers.PrimaryKey = {Customers.Columns("ID")}

Dim NewRow As DataRow
NewRow = Customers.NewRow
With NewRow
       .Item("ID") = 1
       .Item("CustomerName") = "Abubaker Swedan"
       .Item("RegistrationDate") = Now.Date
End With

Customers.Rows.Add(NewRow)
MsgBox(NewRow.Table.TableName)
```

أما أهم وأشهر الوظائف Methodes التي تتبع الكلاس DataRow فهي:

الوظيفة Delete:

وتقوم بحذف الـDataRow.

الوظائف التالية سنفهمها أكثر عند الحديث عن العلاقات بين الجداول DataRelation لاحقاً إن شاء اللّه:

الوظيفة GetChildRows(DataRelation):

وتقوم باسترجاع الصفوف (الأبناء) للـDataRow اعتماداً على DataRelation.

الوظيفة GetChildRows(DataRelation):

وتقوم باسترجاع الـRows (الأبناء) للـDataRow اعتماداً على قيمة الخاصية RelationName التابعة للكلاس DataRelation.

الوظيفة GetParentRow(DataRelation):

ملاحظات:

.....

.....

.....

وتقوم باسترجاع الصف (الأب) للـDataRow اعتماداً على الـDataRelation المحددة.

الوظيفة GetParentRow(String):

وتقوم باسترجاع الصف (الأب) للـDataRow اعتماداً على قيمة الخاصية RelationName التابعة للكلاس DataRelation.

الوظيفة IsNull(DataColumn):

وتحدد ما إذا كانت القيمة التي يحملها الـDataColumn هي Null.

```
If NewRow.IsNull(ColumnObject) Then
	MsgBox("Empty Column")
Else
	MsgBox("it is not null value")
End If
```

الوظيفة IsNull(Int32):

وتحدد ما إذا كانت القيمة التي يحملها الـDataColumn (بدلالة فهرسه) هي Null.

```
If NewRow.IsNull(1) Then
	MsgBox("Empty Column")
Else
	MsgBox("it is not null value")
End If
```

الوظيفة IsNull(String):

وتحدد ما إذا كانت القيمة التي يحملها الـDataColumn (بدلالة اسمه) هي Null.

```
If NewRow.IsNull("CustomerName") Then
	MsgBox("Empty Column")
Else
	MsgBox("it is not null value")
End If
```

يدعم الكلاس DataTable الطريقة NewRow لتوليد صف جديد يخضع لإعدادات الـDataColumns، من حيث نوع الحقل وطوله وخصائصه الأخرى وذلك لكل حقل من الحقول التي يحتوي عليها DataTable.

ولفتح صف (سجل DataRow) جديد في الجدول:

```
Dim TheNewRow As DataRow = TableName.NewRow()
```

ملاحظات:

بعد تنفيذ السطر البرمجي السابق، يتم وبشكل تلقائي إنشاء صف بيانات جديد بحيث يراعى فيه خصائص كل حقل، ويتم وضع القيمة Null في كافة الحقول، عدا تلك التي لها قيم افتراضية Default Values، أو التي خاصية AutoIncrement لها = True.

ضبط قيم الحقول في الـDataRow الجديد

يتضمن الكلاس DataRow خاصية مهمة وهي Item، والتي تعطي إمكانية الوصول إلى كل DataColumn معرّف في الـDataTable اعتماداً على اسم الحقل أو رقم ترتيبه (ابتداء من الصفر) أو متغير يمثل الـDataColumn، و السنّة المتبعة هي الاعتماد على اسم الحقل تجنباً لأخطاء غير مقصودة، وتوضيحاً للكود، مع الأخذ في الاعتبار خصائص كل حقل على حدة.

```
Dim TheNewRow As DataRow = TableName.NewRow()
TheNewRow.Item("ID")= 100
TheNewRow.Item(0)= 100
Dim SpecificRow As DataColumn = TableName.Columns(0)
TheNewRow.Item(SpecificRow)= 100
```

في الكود السابق تم ضبط قيمة أول حقل في الجدول (ID) من خلال اسم الحقل مرة، ورقم ترتيبه مرة ، ومن خلال مرجع يمثل الحقل المطلوب مرة ثالثة.

الخاصية Item هي الخاصية التلقائية للكلاس DataRow، ولذلك يمكن الاستغناء عن ذكرها وكتابة:

```
Dim TheNewRow As DataRow = TableName.NewRow()
TheNewRow("ID")= 100
```

كما يمكن استعمال علامة التعجب (!) لتشير إلى اسم الحقل (لا تدعم رقم فهرس الحقل) في الجدول:

```
Dim Customers As New DataTable("Customers")
Customers.Columns.Add("ID", GetType(Long))
Customers.Columns.Add("CustomerName", GetType(String))
Customers.Columns.Add("RegistrationDate", GetType(Date))
Customers.PrimaryKey = {Customers.Columns("ID")}
Dim NewRow As DataRow
NewRow = Customers.NewRow
NewRow!ID = 1
NewRow!CustomerName = "Abubaker Swedan"
NewRow!RegistrationDate = Now.Date
Customers.Rows.Add(NewRow)
```

ملاحظات:

حفظ البيانات في الجدول DataTable

بعد ضبط وتعيين القيم في السجل الجديد، تكون الخطوة اللاحقة هي ضم هذا السجل إلى الجدول DataTable، مستعملين الطريقة Rows.Add. التابعة للـDataTable:

```
Dim TheNewRow As DataRow = TableName.NewRow()

TheNewRow.Item("ID")= 100
TableName.Rows.Add(TheNewRow)
```

كما يمكن استعمال طريقة أخرى، وهي بتنفيذ الطريقة Rows.Add وإرسال قيم الحقول كبارامترات:

```
Dim TheNewRow As DataRow = TableName.NewRow()
TableName.Rows.Add(100, “Abubaker”, “Tripoli”, …)
```

وأي طريقة اخترنا، فإن الوظيفة Add تختبر القيم المرسلة من حيث توافقها مع تركيبة الجدول وحقوله، وترفض القيم التي لا تتماشى مع القواعد.

ملاحظات:

الوصول إلى الصف المطلوب في جدول DataTable

أو بصيغة أخرى، طريقة تحديد صف معين أو مجموعة من الصفوف في جدول لإجراء عمليات عليها. وتوجد عدة طرق لاسترجاع الصفوف المطلوبة منها:

أ) استرجاع الصفوف بدلالة قيم الأعمدة

وذلك باستخدام الوظيفة Select Method، التابعة للـDataTable

```
Dim foundRows() As Data.DataRow
foundRows = DataSet1.Tables("Customers").Select("LastName = 'Swedan'")
```

ففي هذا الكود تم تعريف مصفوفة لتجمع بداخلها الصفوف المسترجعة من عملية البحث باستخدام الطريقة Select، والتي مررنا لها قيمة الحقل LastName. فإن تم العثور على السجلات المطلوبة فإنها تخزن في تلك المصفوفة.

ويمكن معرفة عدد الصفوف المسترجعة من خلال الكود التالي:

```
Dim RowsCount As Integer
RowsCount = foundRows.Count
```

ويمكن الوصول إلى حقول كل صف بالكيفية التالية:

```
Dim FirstColValue As Integer
FirstColValue = foundRows(0).Item("OrderID")
```

ب) استرجاع الصفوف بدلالة قيمة الحقل المفتاحي Primary Key Value

وفي هذه الحالة نستخدم الطريقة **Find** التابعة للتجمع **DataRowCollection** ونرسل لها القيمة المفتاحية **كبارامتر**:

```
Dim s As String = "primaryKeyValue"
Dim foundRow As DataRow = dataset1.Tables("Customers").Rows.Find(s)

If foundRow IsNot Nothing Then
    MsgBox(foundRow(1).ToString())
Else
    MsgBox("A row with the primary key of " & s & " could not be found")
End If
```

ملاحظات:

.....

.....

.....

تعديل البيانات في الجدول DataTable

لتعديل بيانات أي صف في جدول DataTable، يلزمنا كبداية **تحديد الصف المطلوب تعديل بياناته**، ومن ثم إسناد القيم المناسبة لكل حقل من حقوله.

ولمعرفة الصف المطلوب تعديله، نستخدم الطريقة **Select Method** التابعة لـDataTable بالصورة التالية:

```
Dim customerRow() As Data.DataRow
customerRow = DataSet1.Tables("Customers").Select("CustomerID = 107")

customerRow(0)("CompanyName") = "New Company Name"
customerRow(0)("City") = "Tripoli"
```

ففي الكود السابق، الحقل CustomerID هو الحقل المفتاحي للجدول Primary Key، وبالتالي عند استخدام الطريقة Select وتمرير رقم الزبون فإن النتيجة هي إرجاع مصفوفة customerRow() تحتوي على صف واحد فقط، إذ أن رقم الزبون لا يتكرر. ثم استعملنا الرقم (0) للتعبير عن أول صف مسترجع في المصفوفة، وعدلنا قيم الحقول المقصودة.

بالإمكان تعديل الكود السابق والخاص بضبط القيم إلى:

```
customerRow(0).Item("CompanyName") = "New Company Name"
customerRow(0).Item("City") = "Tripoli"
```

ولكن كما قلتُ سابقاً أن **الخاصية Item هي الخاصية التلقائية للكلاس DataRow**، ولذلك يمكن الاستغناء عن ذكرها.

وفي حالة أننا نعلم رقم ترتيب الصف المطلوب تعديله، بدلالة رقم الجدول في الـDataSet نكتب:

```
DataSet1.Tables(0).Rows(4).Item(1) = "New Company Name"
DataSet1.Tables(0).Rows(4).Item(2) = "Tripoli"
```

أو بدلالة اسم الجدول:

```
DataSet1.Tables("Customers").Rows(4).Item(1) = "New Company Name"
DataSet1.Tables("Customers").Rows(4).Item(2) = "Tripoli"
```

ملاحظات:

عرض بيانات الجدول Viewing Data in a DataTable

يمكننا الوصول لبيانات الجدول من خلال استعمال التجمعات Rows وColumns الخاصة بالـDataTable. وكذلك استعمال الطريقة **Select Method** لاسترجاع مجموعة من السجلات التي تخضع لشروط معينة وترتيب معين وحالة صف معينة.

بالإضافة إلى استعمال الطريقة **Find Method** التابعة للتجمع DataRowCollection للبحث عن سجل معين عن طريق مفتاحه الأساسي.

استعمال الطريقة **Select Method** يرجع مجموعة من كائنات الـDataRow ، تنطبق عليها شروطنا، وتقبل بارامترات تمثل:

- صيغة الفرز (الشروط) Filter expression.
- طريقة الترتيب Sort expression.
- حالة كل صف DataViewRowState.

صيغة الفرز تحدد الصفوف المسترجعة بناء على قيم الحقول DataColumn مثلاً "LastName = "Swedan.

وطريقة الترتيب تحدد من خلال جملة Sql مثلاً ORDER BY LastName DESC.

أما حالة كل صف فسنتعرف عليها لاحقاً إن شاء اللّه في موضوع: حالة الصف Row State.

ملاحظات:

حذف البيانات من جدول Removing Data from DataTable

يمكن حذف الصفوف من الجدول DataTable باستخدام:

- الطريقة Remove التابعة للتجمع DataTable.Rows.
- الطريقة RemoveAt التابعة للتجمع DataTable.Rows.

والطريقة Remove Method تأخذ متغيراً يمثل صفاً في الجدول:

```
Dim SelectedRow As DataRow = SelectedTable.Rows(0)
SelectedTable.Rows.Remove(SelectedRow)
```

أما الطريقة RemoveAt Method فتأخذ رقم فهرس الصف مباشرة:

```
SelectedTable.Rows.RemoveAt(0)
```

وكل الطرق تؤدي إلى **مكة**.

ويمكن حذف جميع الصفوف بأمر واحد، وذلك باستخدام الطريقة Clear Method التابعة للكائن DataTable.Rows:

```
SelectedTable.Rows.Clear()
```

ملاحظات:

معالجة الحزم / الدفعات Batch Processing

جميع العمليات التي قمنا بها سابقاً هي عمليات مباشرة تتم على سجلات الجدول وحقوله، وقد تكون مناسبة في حالات العمل على سجل واحد. ولكن عند التعامل مع الجداول الكبيرة والرغبة في إجراء العمليات على عدد غير محدد من السجلات فالطريقة لا بد وأن تختلف.

ADO.NET يوفر مزايا جديدة بحيث يمكننا عمل تغييرات على عدة سجلات، وبعدها نقرر هل نطبق هذه التغييرات أم نتجاهلها! وهذه الطريقة تسمى: **معالجة الحزم أو الدفعات**.

وللاستفادة من هذه الطريقة في المعالجة، ببساطة نقوم بإحداث التغييرات التي نريد، وعندما نكون جاهزين لتطبيق هذه التغيرات:

- نستخدم الطريقة **AcceptChanges Method** التابعة للـDataTable ليتم تطبيق التغييرات وتحديث بيانات الجدول.
- أو نستخدم الطريقة **RejectChanges Method** لرفض التغييرات غير المحفوظة والحفاظ على بيانات الجدول كما هي دون تغيير.

يمكن تطبيق هاتين الطريقتين حتى على الـDataRows. فكل DataRow يدعم هاتين الطريقتين، ولكن استخدام AcceptChanges أو RejectChanges على الجدول بأكمله **أفضل** من استعمالهما مع كل صف على حدة، لأنه سيتم الدوران على كافة الصفوف التي حدث بها التغيير وتطبيقها (في حال استعمال AcceptChanges) أو رفض التغييرات (في حال استعمال RejectChanges).

```
SelectedTable.AcceptChanges()
SelectedTable.RejectChanges()
```

ملاحظات:

حالة الصف Row State

في أثناء إجراء التغييرات على الصف، يقوم ADO.NET بحفظ النسخة الأصلية والنسخة المعدلة لكافة الحقول التي يحتويها هذا الصف، وكذلك بمراقبة وتحديد الصفوف المضافة و/أو المحذوفة من الجدول، بحيث يمكن أن نعود للنسخة الأصلية حين الحاجة. يفعل ADO.NET كل ذلك من خلال حفظ حالة كل حقل لكل الصفوف ويتم تحديث قيمة الخاصية **DataRow.RowState** لكل صف على حدة، والتي تحتمل إحدى القيم التالية:

- **DataRowState.Detached**: وهي الحالة الافتراضية لأي صف لم يتم إضافته بعد إلى DataTable.
- **DataRowState.Added**: وهي حالة كل صف تم إضافته إلى DataTable ولكن لم تأكيد الإضافة. بحيث لو استعملنا RejectChanges يتم حذفها فوراً.
- **DataRowState.UnChanged**: وهي الحالة الافتراضية لأي صف موجود مسبقاً بالجدول ولم تتم عليه أي عملية تعديل منذ آخر مرة تم فيها استدعاء الطريقة AcceptChanges Method. وهي الحالة الافتراضية أيضاً لكافة السجلات التي تم إنشاؤها من خلال الطريقة NewRow.
- **DataRowState.Deleted**: الصفوف المحذوفة لا يتم إزالتها فعلياً من الجدول إلى أن يتم استدعاء الطريقة AcceptChanges، بل يتم تأشيرها على أنها ستحذف من خلال هذه الخاصية.
- **DataRowState.Modified**: أي صف تم تعديل حقوله بأي طريقة يؤشر على أنه Modified.

ففي كل مرة نقوم بإضافة أو تعديل صف، يتم تحديث حالته فوراً، على عكس عملية الحذف من خلال Rows.Remove و Rows.RemoveAt، فإنه يحدث التفاف حول نظام تتبع حالات السجلات هذا، ولا يتم تحديث الحالة على نفس المنوال، لأنه يتم حذف الصفوف مباشرة، ولا يجعلها تخضع لنظام معالجة الحزم.

ولحل هذه المشكلة، نستعمل الطريقة Delete Method التابعة للـDataRow عوضاً عن Rows.Remove و/أو Rows.RemoveAt، فهي لا تقوم بحذف الصفوف، بل بتغيير حالتها إلى Deleted، وبالتالي نستطيع تأكيد الحذف من خلال الطريقة AcceptChanges، أو التراجع عن الحذف من خلال RejectChanges.

```
TheRow.Delete()
```

عند استدعاء الطريقة AcceptChanges سواء من خلال DataSet أو DataTable أو DataRow فإن يتم حذف كافة السجلات التي حالتها = Deleted، وتطبيق التعديلات التي حصلت على السجلات الأخرى بحيث تحل القيم الجديدة Current Values محل القيم الأصلية Original Values.

أما عند استدعاء الطريقة RejectChanges، فإنه يتم حذف كافة السجلات التي حالتها = Added، وتعطى باقي السجلات الحالة Unchanged، وتلغى التعديلات التي حصلت على السجلات الأخرى بحيث تحل القيم الأصلية Original Values محل القيم الجديدة Current Values.

ملاحظات:

نُسخ الصف Row Versions

عند إجراء التعديلات والتغييرات سواء إضافة أو حذف أو تعديل في بيانات الصفوف، فإن ADO.NET يحتفظ بنسخ متعددة لكل صف حتى ولو تم تغيير قيمة حقل واحد فقط.

DataRowVersion

ويحتمل إحدى القيم التالية:

- **DataRowVersion.Original**: يعني الاحتفاظ بالقيم الأصلية للحقول منذ آخر استدعاء للطريقة AcceptChanges، ولا يشمل الصفوف الجديدة.
- **DataRowVersion.Porposed**: يعني أن قيمة حقل ما تغيرت، ولكن لم يتم تأكيد التغيير. هذه النسخة لا تتوفر حتى يتم البدء في عملية تعديل قيمة الحقل. بعد تأكيد التغييرات، تتحول القيمة إلى Original.
- **DataRowVersion.Current**: يعني أن التعديل جارٍ الآن، وعند تأكيده، تتحول القيمة إلى Original.
- **DataRowVersion.Default**: يعني النسخة التلقائية للصف. فالنسخة التلقائية لـ Added وModified وUnchanged هي Current، والنسخة التلقائية لـDeleted هي Original، والنسخة التلقائية لـDetached هي Proposed.

حيث Added وModified وUnchanged وDeleted و Detached هي حالات الصفوف.

التحقق من صحة التغييرات Validating Changes

قلتُ في وقت سابق أنه عند الرغبة في حفظ البيانات في الجدول، فإن الوظيفة Add تختبر القيم المرسلة من حيث توافقها مع تركيبة الجدول وحقوله، وترفض القيم التي لا تتماشى مع القواعد.

وفي الحقيقة، فإن الاعتماد على الطريقة Add وحدها لمعرفة القيم الصحيحة من الخاطئة ليس كافياً، إذ مازال بإمكان المستخدم إدخال قيم غير صحيحة، ولا ننس الحكمة التي تقول: لا تثق في مدخلات المستخدم!.

وكمثال على المدخلات الخاطئة إدخال القيمة 5,261 في حقل من نوع System.Int32 والخاص بحفظ عمر شخص. أو إدخال قيمة نصية تتجاوز الحد الأقصى للنص في الحقل. ولذا يجب كتابة أكواد تتحقق من صحة المدخلات لتكون في المدى المرغوب.

ملاحظات:

فحص التغييرات الطارئة على محتويات جدول

قلتُ سابقاً أنه عند استعمال نظام معالجة الحزم، وفي حال حدث تغيير في الجدول، لا يتم تطبيق التعديلات إلا عند استدعاء الطريقة AcceptChanges Method. وبالتالي فالتغييرات تبقى معلقة pending حتى تطبيقها أو تجاهلها وإرجاع الجدول إلى سابق عهده (إلى حالته عند استدعاء AcceptChanges آخر مرة). وقلتُ أن عملية تتبع الصفوف تتم عن طريق:

- كل صف يحتوي على "RowState" وهي إحدى الحالات: Detached وAdded وModified وDeleted وUnchanged.
- كل صف تم تغيير محتوياته يتضمن نُسخاً مختلفة للصفوف: Original وProposed وCurrent وDefault.

معرفة ما إذا كان هناك تغيير في الصفوف

هناك طريقة Method تتبع الـDataSet هي: HasChanges، ذات قيمة منطقية، إذا ساوت True فإن هناك تغيير حدث على بعض الصفوف أو كلها. وفي تلك الحالة يمكن استدعاء الطريقة GetChanges التابعة للـDataSet و/أو للـDataTable لاسترجاع مصفوفة تحتوي على الصفوف المتغيرة، وهذه المصفوفة هي بمثابة DataSet أو DataTable مملوء بالصفوف المتغيرة فقط. ومن نافلة القول أنه يجب استدعاء الطريقة GetChanges **قبل** تطبيق التغييرات باستخدام الطريقة AcceptChanges، وإلا فلن نتحصل على أي صف!.

أ) استرجاع الصفوف المتغيرة من DataSet

وهنا نقوم بإنشاء DataSet وملئها بالصفوف المتغيرة:

```
Dim changedRecords As DataSet = dataset1.GetChanges()
```

ب) استرجاع الصفوف المتغيرة من DataTable

وهنا نقوم بإنشاء DataTable وملئه بالصفوف المتغيرة:

```
Dim changedRecordsTable As DataTable = Customers.GetChanges()
```

ج) استرجاع الصفوف المتغيرة بحسب نوعية التغيير (النسخة)

وهنا نرسل نوع الإصدارة إلى الطريقة GetChanges كبارامتر:

```
Dim addedRecords As DataSet = dataset1.GetChanges(DataRowState.Added)
```

ملاحظات:

الخلاصة

تعرفنا في الباب الثاني على الكلاس DataTable، وعرفنا كيفية إنشائه وإنشاء أعمدته، وكذلك كيفية إنشاء الصفوف الجديدة، وعرفنا الكثير عن خصائص ووظائف كل كلاس على حدة.

وكل ما قمنا به ما هو إلا الخطوة الأولى، تليها خطوات أخرى سنتابعها في ما تبقى من الكتاب من أجزاء إن شاء اللّه.

أملى أن تستفيدوا مما قدمتُ إلى حد الآن، ورغبتي ملحة في أن ترسلوا لي ملاحظاتكم وتصحيحاتكم وإضافاتكم على بريدي الإلكتروني abubakerswedan@yahoo.com، حتى أخرج نسخاً منقحة ومزيدة إن شاء اللّه.

النسخة التي بين أيديكم الآن، هي جزء من كتاب أعده يتحدث عن كل ما يتعلق ببرمجة قواعد البيانات في لغة فجول بيسك، وسأضع النسخة كاملة للتنزيل المجاني فور انتهائي من كتابتها وتنقيحها وإخراجها بصورة جيدة إن شاء اللّه.

أتمنى للجميع التوفيق.

ملاحظات: