**الفصل الثامن عشر**

**Programming ADO.NET Objects**

تمنحك مجموعة البيانات الوسائل البرمجيّة اللازمة للتعامل مع بياناتها.. تعال نتعرّف على أهمّ الكائنات التي ستحتاجها في هذا المضمار.

* [**التعامل مع الجداول:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\18-%20ADO.NET%20برمجة%20كائنات\1-%20التعامل%20مع%20الجداول.htm)
* [**التعامل مع العلاقات والقيود**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\18-%20ADO.NET%20برمجة%20كائنات\2-%20التعامل%20مع%20العلاقات%20والقيود.htm)**:**
* [**التعامل مع العُروض**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\18-%20ADO.NET%20برمجة%20كائنات\3-%20التعامل%20مع%20العُروض.htm)**:**
* [**التعامل مع الارتباطات**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\18-%20ADO.NET%20برمجة%20كائنات\4-%20التعامل%20مع%20الارتباطات.htm)**:**
* [**مجموعات البيانات محدّدة النوع**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\18-%20ADO.NET%20برمجة%20كائنات\5-%20مجموعات%20البيانات%20محدّدة%20النوع.htm)**:**
* [**مشروع مجموعة البيانات:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\18-%20ADO.NET%20برمجة%20كائنات\6-%20مشروع%20مجموعة%20البيانات.htm)
* [**تحديث مجموعات البيانات**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات.htm)**:**
* [**إنشاء مجموعات بيانات خاصّة**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\18-%20ADO.NET%20برمجة%20كائنات\8-%20إنشاء%20مجموعات%20بيانات%20خاصّة.htm)**:**

**التعامل مع الجداول**

هناك كائنات هامّة، تمدّك بالوسائل والخصائص اللازمة للتعامل مع الجداول، وهي:

* [**مجموعة الجداول DataTableCollection:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\1-%20التعامل%20مع%20الجداول\1-%20مجموعة%20الجداول.htm)
* [**كائن الجدول DataTable:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\1-%20التعامل%20مع%20الجداول\2-%20كائن%20الجدول.htm)
* [**كائن الصفّ DataRow:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\1-%20التعامل%20مع%20الجداول\3-%20كائن%20الصفّ.htm)
* [**كائن العمود DataColumn:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\1-%20التعامل%20مع%20الجداول\4-%20كائن%20العمود.htm)

كما أنّ هذان موضوعان هامّان عند التعامل مع الجداول:

* [**تطبيق دوال التجميع Aggregate Functions على بعض الأعمدة:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\1-%20التعامل%20مع%20الجداول\5-%20تطبيق%20دوال%20التجميع%20على%20بعض%20الأعمدة.htm)
* [**صيغة العمود:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\1-%20التعامل%20مع%20الجداول\6-%20صيغة%20العمود.htm)

**مجموعة الجداول DataTableCollection:**

وهي تحتوي على الجداول في صورة كائنات من النوع DataTable.. ويمكن الحصول على هذه المجموعة باستخدام خاصيّة Tables الخاصّة بمجموعة البيانات.

والكود التالي يطبع كلّ أسماء الجداول الموجودة في مجموعة البيانات المسمّاة DsBooks1:

**Dim tbl As System.Data.DataTable**

**For Each tbl In DsBooks1.Tables**

**Console.WriteLine(tbl.TableName)**

**Next**

وبخلاف خاصيّتي Count و Item (والأخيرة تتعامل مع الجدول باسمه موضوعا بين علامتي تنصيص، أو باستخدام رقمه في المجموعة كمعامل)، تمتلك مجموعة الجداول الوسائل التالية:

|  |  |
| --- | --- |
| Add | تضيف جدولا لجداول مجموعة البيانات.. ولها ثلاث صيغ:  الأولى بدون معاملات، وهي تنشئ جدولا باسم افتراضيّ وتضيفه لمجموعة البيانات..  والثانية تستقبل منك معاملا نصّيّا، هو اسم الجدول الذي يتمّ إنشاؤه..  وكلتا هاتين الصيغتين تعيد كائنا من النوع DataTable يمثّل الجدول الذي تمّ إنشاؤه.  أمّا الصيغة الثالثة، فهي تسمح لك بإضافة جدول موجود بالفعل، حيث تستقبل منك معاملا، هو عبارة عن كائن من النوع DataTable، يمثّل الجدول الذي تريد إضافته لمجموعة البيانات. |
| AddRange | تستقبل منك هذه الوسيلة مصفوفة من النوع DataTable، تحتوي على الجداول التي تريد إضافتها لنهاية مجموعة الجداول. |
| CanRemove | تعيد هذه الوسيلة True إذا كنت تستطيع أن تحذف من مجموعة الجداول، الجدول الذي يمثّله متغيّر من النوع DataTable المرسل لها كمعامل. |
| Remove | تحذف هذه الوسيلة الجدول المرسل لها كمعامل (سواء أكان هذا المعامل نصّا يمثّل اسم الجدول، أو كائنا من النوع DataTable). |
| RemoveAt | مماثلة للوسيلة السابقة، ولكنّها تستقبل كمعامل، رقم الجدول في مجموعة الجداول. |
| Clear | تحذف كلّ الجداول الموجودة في مجموعة البيانات. |
| Contains | تقبل هذه الوسيلة نصّا يمثّل اسم أحد الجداول، للتأكّد ممّا إذا كان موجودا في مجموعة البيانات أم لا. |
| CopyTo | لنسخ الجداول من مجموعة الجداول إلى مصفوفة. |
| IndexOf | تستقبل هذه الوسيلة معاملا عبارة عن كائن جدول DataTable، وتعيد رقمه في مجموعة الجداول. |

**كائن الجدول DataTable:**

يُتيح لك هذا الكائن التعامل مع أحد جداول مجموعة البيانات.. ويتمّ تعريفه كالتالي:

**Dim T As DataTable**

**T = DsBooks1.Tables(0)**

حيث عرّفنا كائن جدول، ووضعنا به الجدول الأوّل في مجموعة الجداول.

وتوجد صيغة أخرى تسمح لك بتعريف جدول جديد بالاسم الذي تريده، كالتالي:

**Dim T As New DataTable("جدول جديد ")**

**DsBooks1.Tables.Add(T)**

حيث استخدمنا الوسيلة Add لإضافة هذا الجدول لمجموعة الجداول.

**ويمتلك كائن الجدول الخصائص التالية:**

|  |  |
| --- | --- |
| لو جعلت هذه الخاصيّة True، فستتمّ مراعاة حالة الحروف (صغيرة Small أو كبيرة Capital) عند مقارنة النصوص في هذا الجدول.. أمّا لو جعلتها False، فلن تتمّ مراعاة حالة الأحرف. | CaseSensitive |
| تعيد مجموعة العلاقات المتولّدة من هذا الجدول (التي يدخل فيها كجدول رئيسيّ Master). | [ParentRelations](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassparentrelationstopic.htm) |
| تعيد مجموعة العلاقات الواصلة لهذا الجدول (التي يدخل فيها كجدول ثانويّ Detail). | ChildRelations |
| تعيد مجموعة الأعمدة (الحقول) الموجودة في هذا الجدول. | [Columns](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclasscolumnstopic.htm) |
| تعيد مجموعة الصفوف (السجلات) الموجودة في هذا الجدول.. والجملة التالية تقرأ قيمة الخانة الواقعة في الصفّ الأوّل والعمود الأوّل في جدول الكتب:  **Dim T As DataTable = DsBooks1.Tables("Books")**  **MsgBox( T.Rows(0)(0) )**  لا تجعل الصيغة **Rows(0)(0)** تربكك، فهي اختصار للصيغة التالية:  **MsgBox( T.Rows.Item(0).Item(0) )**  وإذا كان الأمر لم يتّضح بعد، دعنا نكتب نفس الصيغة ولكن مع بعض الثرثرة.. إنّ خاصيّة Item الخاصّة بكائن الصفّ DataRow تعيد كائن أعمدة DataColumn يحتوي على الخانات الموجودة في الصفّ المطلوب:  **Dim C As DataColumn = T.Rows.Item(0)**  أمّا خاصيّة Item الخاصّة بكائن الأعمدة، فهي تعيد قيمة الخانة الموجودة في العمود المطلوب:  **MsgBox( C.Item(0).Item(0) )** | [Rows](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassrowstopic.htm) |
| تعيد مجموعة القيود (الشروط) الموجودة في هذا الجدول. | [Constraints](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassconstraintstopic.htm) |
| تعيد كائن مجموعة بيانات DataSet Object، يشير إلى مجموعة البيانات التي ينتمي إليها هذا الجدول. | [DataSet](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassdatasettopic.htm) |
| تعيد هذه الخاصيّة كائن العرض View الذي يمثّل البيانات التي سيتمّ عرضها للمستخدم من هذا الجدول (راجع موضوع العروض Views في [أوّل فصول قواعد البيانات](file:///C:\Users\Administrator\Desktop\vb.net\15-%20تركيب%20قواعد%20البيانات\4-%20إنشاء%20قواعد%20بيانات\3-%20أهمّ%20عناصر%20الرابطة\2-%20العروض.htm)).. والكود التالي يؤدّي لعرض أسماء المؤلّفين الذين تبدأ أسماؤهم بالحروف من الألف إلى التاء فقط:  **Dim T As DataTable = DsBooks1.Tables("Books")**  **T.DefaultView.RowFilter = " Book < 'ث' "**  ويمكن ترتيب السجلات المعروضة، عن طريق خاصيّة Sort.. انظر كيف نرتّب الكتب المعروضة تنازليّا عن طريق اسم الكتاب ورقم المؤلّف:  **T.DefaultView.Sort = "Book, AuthorID DESC"**  وسنتحدّث عن كائن العرض View بالتفصيل لاحقا. | [DefaultView](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassdefaultviewtopic.htm) |
| تسمح لك هذه الخاصيّة بقراءة أو تغيير اسم الجدول. | [TableName](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclasstablenametopic.htm) |
| تمثّل هذه الخاصيّة النصّ الذي سيتمّ عرضه كعنوان للجدول في أدوات واجهة المستخدم (مثل جدول البيانات DataGrid). | [DisplayExpression](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassdisplayexpressiontopic.htm) |
| تعيد هذه الخاصيّة مجموعة الخصائص PropertyCollection الإضافيّة للجدول.. نعم.. يمكنك إضافة الخصائص التي تروق لك إلى كائن الجدول.. والمثال التالي يريك كيف تضيف خاصيّة اسمها Password لجدول الكتب، وتضع فيها القيمة "كلمة المرور":  **Dim T As DataTable = DsBooks1.Tables("Books")**  **Dim P As PropertyCollection**  **P = T.** [**ExtendedProperties**](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassextendedpropertiestopic.htm)  **P.Add ("Password", "كلمة المرور")**  وللتعامل مع هذه الخاصيّة استخدم كودا كالتالي:  **P.Item ("Password") = "تغيّرت"**  **Dim X As Integer = P.Item ("Password")** | [ExtendedProperties](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassextendedpropertiestopic.htm) |
| تعيد True إذا كانت هناك أيّة أخطاء في أيّ صفّ في أيّ جدول في مجموعة البيانات التي ينتمي إليها الجدول (هذه الأخطاء سيتمّ اكتشافها عند محاولة حفظ السجلات من مجموعة البيانات إلى قاعدة البيانات، كما سنرى لاحقا). | [HasErrors](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclasshaserrorstopic.htm) |
| هذه الخاصيّة عبارة عن كائن من النوع CultureInfo، يمثّل اللغة (الثقافة) التي تريد التعامل بها لمقارنة النصوص.. ولقد تعرّفنا على هذا الكائن من قبل في [فصل أدوات ويندوز](file:///C:\Users\Administrator\Desktop\01-%20أساسيّات%20اللغة\06-%20Basic%20Controls%20الأدوات%20الأساسيّة\1-%20مربّع%20النصّ\6-%20تغيير%20لغة%20لوحة%20المفاتيح.htm)، حين استخدمناه لتغيير لغة لوحة المفاتيح أثناء شرحنا لمربّع النصّ. | [Locale](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclasslocaletopic.htm) |
| تسمح لك هذه الخاصّة بتحديد السعة المبدئيّة (أصغر حجم) للجدول بالبايت. | [MinimumCapacity](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassminimumcapacitytopic.htm) |
| هذه الخاصيّة عبارة عن مصفوفة من كائن الأعمدة DataColumn التي تريد استخدامها كمفتاح أساسي للجدول.. أعرف أنّك ستتعجّب من ذلك، لأنّنا اتفقنا أنّ لكلّ جدول مفتاح أساسيّ وحيد فقط.. نعم.. ما زالت هذه هي القاعدة.. ولكنّ الذي لا تعرفه هو أنّ بإمكانك جعل عمودين (أو أكثر) معا يمثّلان المفتاح الرئيسيّ للجدول، وذلك عندما لا يوجد عمود منفرد يصلح لتمييز سجلات الجدول بدون التباس.. في هذه الحالة يمكن تكرار قيم العمود الواحد، ولكن لا يمكن تكرار زوج القيم في العمودين.. مثلا، يمكن أن يحتوي أحد العمودين على رقم 5 مكرّرا، والعمود الثاني على الاسم "علي" مكرّرا، ولكن لا يسمح أن يتكرّر وجود صفّ به قيمتا هذين العمودين (5، "علي").. يسمح فقط بمثل هذه الأزواج: (1، "علي").. (5، "عمر").. (5، "نجوى").. بحيث يكون كلّ زوج منها متفرّدا ومميّزا للسجلّ الذي يوجد به. | [PrimaryKey](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatatableclassprimarykeytopic.htm) |

**كما يمتلك كائن الجدول الوسائل التالية:**

|  |  |
| --- | --- |
| تؤدّي هذه الوسيلة لقبول التغييرات التي أجراها المستخدم على الجدول منذ أن تمّ تحميله من قاعدة البيانات أو منذ آخر مرّة تمّ حفظه فيها.. ليس معنى هذا أنّ هذه التغييرات سيتمّ حفظها لقاعدة البيانات، ولكن سيتمّ النظر إليها على أنّها البيانات الأصليّة للسجلات، ولن يمكنك التراجع عنها (ما لم تُعِد تحميل البيانات من قاعدة البيانات من جديد).. [وسيتّضح لك هذا بالتفصيل لاحقا](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\7-%20تحديث%20مجموعات%20البيانات.htm). | AcceptChanges |
| استخدم هذه الوسيلة لرفض كلّ التغييرات التي أجراها المستخدم على سجلات الجدول، بحيث يعود للحالة التي كان عليها عند تحميله من قاعدة البيانات (أو عند آخر استخدام للوسيلة AcceptChanges). | RejectChanges |
| استخدم هذه الوسيلة لإعادة الجدول لقيمه الافتراضيّة قبل إدخال أيّ تعديلات به. | Reset |
| استخدم الوسيلة BeginInit لمنع المستخدم من التعامل مع الجدول أثناء وضعك للقيم الابتدائيّة له.. وبعد أن تنتهي من ذلك، استخدم الوسيلة EndInit للسماح للمستخدم بالتعامل مع الجدول. | BeginInit EndInit |
| الوسلة BeginLoadData مماثلة للوسيلة BeginInit، ولكنّها تمنع ADO.NET من إرسال الإشعارات Notifications ومن إصلاح الفهارس Index Maintenance ومن التحقّق من قواعد الصحّة Constraints أثناء تحميلك لبيانات الجدول.. وبعد أن تنتهي من ذلك، استخدم الوسيلة EndLoadData، للسماح لـ ADO.NET بالعودة لمزاولة ما نُهيَِ عنه. | BeginLoadData  EndLoadData |
| تمحو كلّ البيانات الموجودة في الجدول. | Clear |
| تستنسخ بنية الجدول بكلّ ما فيه من مخطّطات Schemas وقيود Constraints.. ولكنّ الجدول الناتج يكون فارغا. | Clone |
| تنسخ الجدول ببنيته وبياناته، لتنتج جدولا مماثلا له تماما. | Copy |
| تعيد جدولا جديدا، يحتوي فقط على الصفوف التي تمّ تعديلها في الجدول الحاليّ أو الصفوف التي تمّت إضافتها إليه أو حذفها منه منذ أن تمّ تحميله (أو منذ آخر استدعاء للوسيلة AcceptChanges). | GetChanges |
| تعيد مصفوفة تحتوي على الصفوف DataRow التي حدثت بها أخطاء عند محاولة حفظ الجدول في قاعدة البيانات. | GetErrors |
| استخدم هذه الوسيلة لتنسخ إلى الجدول، السجلّ DaraRow المرسل لها كمعامل، بكلّ بياناته وخصائصه. | ImportRow |
| أرسل لهذه الوسيلة مصفوفة تحتوي على قيم أحد السجلات.. ستبحث الوسيلة عن الحقل الذي يحتوي على القيمة المناظرة للمفتاح الأساسيّ في المصفوفة، فإذا وجدته، تقوم بنسخ باقي القيم من المصفوفة إلى باقي حقول السجلّ.. وإذا لم تجدْه، تقوم بإنشاء سجلّ جديد وتضع بحقوله قيم المصفوفة.  ويوجد معامل ثانٍ لهذه المصفوفة، لو جعلته True فسيتمّ قبول التغييرات التي أدخلت على الحقل (بحيث لا يمكن التراجع عنها بعد ذلك).  وتعيد هذه الوسيلة كائن صفّ DataRow يشير للصفّ الذي تمّ تعديله أو إنشاؤه. | LoadDataRow |
| تنشئ صفّا جديدا له نفس تركيب صفوف الجدول، وتعيده لك، لكن دون إضافته للجدول (عليك أن تضيفه أنت بنفسك). | NewRow |
| تعيد مصفوفة تحتوي على كلّ صفوف الجدول.  وهناك صيغة لهذه الدالة، تستقبل معاملا نصّيا يمثّل الشرط الذي يحجب أن تحقّقه السجلات المعادة.. والجملة التالية تعيد كلّ الكتب التي تبدأ بحروف تسبق حرف الثاء في الترتيب الأبجديّ:  **Dim R As DataRow**  **R = T.Select (" Book < 'ث' ")**  وهناك صيغة ثالثة، تتيح لك اختيار الترتيب الذي ستظهر به هذه الصفوف.. انظر كيف نعرض أسماء الكتب السابق اختيارها مرتّبة تنازليّا على حسب ترتيبها الأبجديّ:  **R = T.Select ("Book < 'ث' ", "Book DESC")**  لاحظ أنّ معامل الترتيب يتكوّن من شقّين: اسم العمود الذي ترتّب على أساسه، ونوع الترتيب (تصاعديّ ASC ـ وهو الافتراضيّ ـ أو تنازليّ DESC).  وتوجد صيغة رابعة، تسمح لك بتحديد حالة السجلّ، من حيث كونه مضافا أو معدّلا أو محذوفا أو لم يتغيّر... إلخ.. [وسنتعرّف على هذه الحالات لاحقا](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\7-%20تحديث%20مجموعات%20البيانات\6-%20حالات%20السجلّ.htm). | Select |
| تحدّد الحقول التي ينطبق عليها مواصفات (شروط) معيّنة، وتجري عليها إحدى دوال التجميع Aggregate Function.. [وسنفرد لهذا الموضوع عنوانا مستقلا عمّا قليل](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\5-%20تطبيق%20دوال%20التجميع%20على%20بعض%20الأعمدة.htm). | Compute |

**هذا، ويمتلك كائن الجدول الأحداث التالية:**

|  |  |
| --- | --- |
| يحدث بعد أن تتغيّر إحدى القيم في أحد أعمدة الجدول.. ويمتلك المعامل e لهذا الحدث الخصائص التالية:  العمود Column ـ الصفّ Row ـ القيمة المقترحة (التي تمّ إدخالها) ProposedValue. | ColumnChanged |
| مماثل للحدث السابق، ولكنّه يحدث عند محاولة إجراء التغيير في أحد أعمدة الجدول (قبل حدوث التغيير). | ColumnChanging |
| تحدث بعد أن تتغيّر إحدى القيم في أحد سجلات الجدول.. ولا أرى فارقا بينه وبين الحدث ColumnChanged، فكلاهما يحدث عند تغيّر قيمة أحد خانات الجدول، اللهمّ إلا إنّ المعامل e يمنحك خاصيّة Action التي تخبرك بنوع التغيير الذي حدث للصفّ (إضافة Add ـ حذف Delete ـ تغيير Change ـ حفظ التغييرات Commit ـ تراجع عن التغيير RollBack ـ لا تغيير Nothing). | RowChanged |
| مماثل للحدث السابق، ولكنّه يحدث عند محاولة إجراء التغيير في أحد سجلات الجدول (قبل حدوث التغيير). | RowChanging |
| يحدث بعد حذف أحد سجلات الجدول.. وله نفس خصائص المعامل e الموجودة للحدث RowChanged. | RowDeleted |
| مماثل للحدث السابق، ولكنّه يحدث عند محاولة حذف أحد سجلات الجدول (قبل إتمام الحذف). | RowDeleting |

هنا سيبرز السؤال: كيف نستخدم هذه الأحداث؟

بسيطة.. عرّف متغيّر الجدول على مستوى النموذج بكلمة WithEvents، وبهذا سيظهر اسمه في القائمة العلويّة اليسرى لمحرّر الكود، وستظهر أحداثه في القائمة العلويّة اليمنى:

**Dim WithEvents T As DataTable**

وفي حدث تحميل النموذج، ضع الجدول الذي تراقب أحداثه في هذا المتغيّر:

**T = DsBooks1.Tables("Books")**

الآن يمكنك أن تعرض رسالة للمستخدم تخبره بنوع التغيير الذي حدث في السجلّ كالتالي:

**Private Sub T\_RowChanged(ByVal sender As Object, ByVal e As \_**

**Data.DataRowChangeEventArgs) Handles T.RowChanged**

**MsgBox(e.Action.ToString)**

**End Sub**

**كائن الصفّ DataRow:**

يمثّل أحد صفوف الجدول.. والكود التالي يعرّف صفّا جديدا ويضيفه لجدول الكتب:

**Dim R As DataRow = DsBooks1.Tables("Books").NewRow**

**DsBooks1.Tables("Books").Rows.Add(R)**

**ويمتلك السجلّ الخصائص التالية:**

|  |  |
| --- | --- |
| تعيد True إذا كانت هناك أخطاء قد حدثت عند محاولة حفظ السجلّ. | HasErrors |
| طبعا هذه الخاصيّة غنيّة عن التعريف.. وهي تقبل معاملا، يحدّد العمود الذي تريد قراءة بياناته في هذا السجلّ.. انظر كيف نقرأ اسم المؤلّف الموجود في الصفّ الثالث في جدول المؤلّفين (لا تنسَ أنّ الصفّ الأوّل هو رقم صفر):  **X = DsBooks1.Tables("Authors").Rows(2).Item("Author")**  أوباختصار**:**  **X = DsBooks1.Tables("Authors").Rows(2)("Author")** | Item |
| تقرأ أو تغيّر كلّ خانات هذا السجلّ من خلال وضع القيم في مصفوفة. | ItemArray |
| تقرأ أو تغيّر النصّ الذي يصف الخطأ الذي حدث في هذا الصفّ. | RowError |
| تقرأ حالة السجلّ من حيث كونه مضافا أو محذوفا أو معدّلا... إلخ، [كما سنرى لاحقا](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\7-%20تحديث%20مجموعات%20البيانات\6-%20حالات%20السجلّ.htm). | RowState |
| تعيد كائن الجدول الذي يحتوي على هذا السجلّ.. فإذا لم يكن هذا السجلّ منتميا لجدول حتى الآن، فإنّها تعيد كائن جدول، يماثل هذا السجلّ (له نفس تركيب سجلاته: نفس الحقول والقيم). | Table |

**كما يمتلك السجلّ الوسائل التالية:**

|  |  |
| --- | --- |
| تحفظ التغييرات التي حدثت بالسجلّ، بحيث لا يمكن التراجع عنها بعد ذلك، إلا إذا أعاد المستخدم تحميل الجدول من قاعدة البيانات. | AcceptChanges |
| تتراجع عن التغييرات التي حدثت بالسجلّ، بحيث يعود لآخر وضع تمّ حفظه عليه. | RejectChanges |
| تبدأ الوسيلة BeginEdit عمليّة تحرير السجلّ.. إنّ هذا يوقف إطلاق الأحداث الخاصّة بمجموعة البيانات، ويوقف عمليّات التحقّق من صحّة البيانات المدخلة، إلى أن يتمّ استدعاء الوسيلة EndEdit التي تنهي عمليّة التحرير، أو الوسيلة CancelEdit التي تلغي عمليّة التحرير وتعيد للسجلّ قيمه الأصليّة.. كما يتمّ إنهاء عمليّة التحرير إذا قام المستخدم بحفظ البيانات إلى قاعدة البيانات.  جدير بنا ذكرا، أنّ الوسيلة BeginEdit تُستدعى آليّا عندما يحاول المستخدم تحرير البيانات في الأدوات المرتبطة بمجموعة البيانات Data-bound Controls. | BeginEdit  EndEdit  CancelEdit |
| تعيد نصّا يصف الخطأ الذي حدث في أحد أعمدة هذا السجلّ. | GetColumnError |
| تسمح لك بوضع نصّ يصف الخطأ الذي حدث في أحد أعمدة هذا السجلّ. | SetColumnError |
| تعيد مصفوفة تحتوي على الأعمدة التي بها أخطاء في هذا السجلّ. | GetColumnsInError |
| تمحو كلّ المعلومات التي تشير إلى حدوث أخطاء في السجلّ (لا يعني هذا إجراء أيّ تعديل في البيانات.. فقط سيبدو أنّ السجلّ قد فقد ذاكرته بخصوص الأخطاء التي حدثت، و... "عفا الله عمّا سلف"). | ClearErrors |
| تحذف السجلّ. | Delete |
| تعيد مصفوفة بها كلّ السجلات المرتبطة بعلاقة بهذا السجلّ في جداول أخرى. | GetChildRows |
| تعيد السجلّ الذي يرتبط به هذا السجلّ بعلاقة (تعيد السجلّ الموجود في الجدول الرئيسيّ). | GetParentRow |
| تعيد مصفوفة تحتوي على كلّ السجلات الرئيسيّة من الجداول الأخرى، المرتبطة بعلاقة بهذا السجلّ. | GetParentRows |
| تعيد True إذا كانت فارغة تلك الخانة المحدّدة بالسجلّ الحالي والعمود المرسل كمعامل. | IsNull |
| تسمح لك بتحديد قيم جديدة للسجلّ الرئيسيّ في الجدول الذي يرتبط معه هذا السجلّ بعلاقة. | SetParentRow |

**كائن العمود DataColumn:**

يمكنك تعريف عمود وإضافته لجدول الكتب كالتالي:

**Dim C As New DataColumn()**

**DsBooks1.Tables("Books").Columns.Add(C)**

لاحظ أنّ هذا العمود سيكتسب الاسم الافتراضيّ Column1 عند إضافته للجدول.. إنّنا لم نختر اسما للعمود ولم نحدّد أيّا من خصائصه.. عامّة، هناك صيغ كثيرة للتعريف، ترسل لها المعاملات اللازمة لتحديد اسم العمود ونوع بياناته بل وصيغة العمود (كأن يكون حاصل ضرب عمودين آخرين أو ما شابه [كما سنرى لاحقا](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\6-%20صيغة%20العمود.htm)).

**تعال نتعرّف على خصائص العمود:**

|  |  |
| --- | --- |
| إذا جعلت قيمة هذه الخاصيّة True فسيُسمح للمستخدم بترك أيّ خانة في هذا العمود فارغة. | AllowDBNull |
| إذا جعلت قيمة هذه الخاصيّة True، فسيصبح هذا العمود ترقيما تلقائيّا للسجلات. | AutoIncrement |
| تقرأ أو تغيّر أصغر رقم يبدأ به الترقيم في العمود الذي يعمل كترقيم تلقائيّ للجدول. | AutoIncrementSeed |
| تحدّد مقدار الزيادة في عمود الترقيم التلقائيّ. | AutoIncrementStep |
| عنوان العمود. | Caption |
| اسم العمود. | ColumnName |
| نوع بيانات العمود. | DataType |
| القيمة الافتراضيّة لخانات العمود. | DefaultValue |
| صيغة العمود.. [وسنتعرّف عليها لاحقا](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\6-%20صيغة%20العمود.htm). | Expression |
| مماثلة [لتلك الخاصّة بكائن الجدول](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\2-%20كائن%20الجدول.htm). | ExtendedProperties |
| أقصى طول (بالبايت) لبيانات خانات العمود (لن تفيد مع البيانات الرقميّة). | MaxLength |
| ترتيب العمود في مجموعة الأعمدة. | Ordinal |
| إذا جعلت قيمة هذه الخاصيّة True فلن يتمكّن المستخدم من تغيير قيمة الخانة بعد إضافتها للعمود. | ReadOnly |
| تعيد كائن الجدول الذي ينتمي إليه هذا العمود. | Table |
| إذا جعلت قيمة هذه الخاصيّة True فلن يسمح بتكرار قيم خانات هذا العمود. | Unique |

**تطبيق دوال التجميع Aggregate Functions على بعض الأعمدة:**

افترض أنّم تريد حساب عدد الكتب التي تبدأ بحروف تسبق حرف النون في جدول الكتب.. يمكنك أن تفعل ذلك عن طريق الوسيلة Compute الخاصّة بكائن الجدول، وذلك كالتالي:

**Dim C As Integer**

**C = DsBooks1.Tables("Books").Compute("Count(Book)", "Book < 'ن' ")**

حيث أرسلنا في المعامل الأوّل العمليّة الحسابيّة، وفي المعامل الثاني الشرط الذي نريد أن تحقّقه السجلات التي ستتمّ عليها العمليّة الحسابيّة.. إنّ للمعامل الثاني أهمّيّة جليلة، فهو يماثل مقطع WHERE في جملة SQL.. انظر مثلا كيف نحسب عدد الكتب التي ألّفها المؤلّف رقم 4:

**C = DsBooks1.Tables("Books").Compute("Count(Book)", "ID = 4")**

يبدو هذا جميلا، ولكن للأسف: لا يمكنك إجراء أيّ عمليّة على أكثر من عمود مباشرة بهذه الوسيلة.. افترض أنّك تريد ـ لسبب ما ـ حساب مجموع خانات عمود ناتج عن ضرب العمودين ID و AuthorID في جدول الكتب.. لن تصلح الجملة التالية لفعل ذلك:

**DsBooks1.Tables("Books").Compute("Sum(ID\*AuthorID)", "")**

إذن ما العمل؟

الحلّ في هذه الحالة، هو إنشاء عمود جديد في الجدول، واستخدام خاصيّة "الصيغة" Expression الخاصّة به لتكون قيم خاناته هي حاصل ضرب العمودين المذكورين.. كما يمكن إرسال الصيغة للمعامل الثالث لمنشئ العمود عند تعريفه.. بعد هذا احسب مجموع خانات هذا العمود:

**Dim Col As DataColumn**

**Col = New DataColumn("Temp", GetType(Integer), "ID \* AuthorID")**

**Dim T As DataTable = DsBooks1.Tables("Books")**

**T.Columns.Add(Col)**

**MsgBox(T.Compute("Sum(Temp)", ""))**

ولكن انتبه: إذا كنت تعرض جدول الكتب في جدول بيانات DataGrid، فسيظهر به العمود الجديد وسيراه المستخدم، وهو ما قد لا تريده.. في هذه الحالة يجمل بك أن تخفي هذا العمود، عن طريق أحد الحلّين التاليين:

1- إنشاء جدول مؤقّت، تنسخ إليه جدول الكتب، ثمّ تضيف إليه العمود الاحتياطيّ وتجرى عليه الحسابات، دون حتّى أن تضيف هذا الجدول المؤقّت لمجموعة البيانات.. تعال نرى ذلك.. إنّه نفس الكود السابق، ما عدا أنّنا سنضع في المتغيّر T نسخة من جدول الكتب، بدلا من وضع جدول الكتب نفسه:

**Dim Col As New DataColumn("Temp", GetType(Integer), \_**

**"ID \* AuthorID")**

**Dim T As DataTable = DsBooks1.Tables("Books").Copy**

**T.Columns.Add(Col)**

**MsgBox(T.Compute("Sum(Temp)", ""))**

ولكن يا خَسارة: الحلو لا يكتمل!.. العيب الخطير في هذه الطريقة يظهر عندما يكون حجم الجدول كبيرا، حيث قد تستهلك عمليّة نسخ الجدول وقتا ملموسا.

2- إضافة العمود المؤقّت لجدول الكتب، ولكن جعله خفيّا Hidden.. واضح أنّ هذه هي الطريقة الأبسط.. ولكن كيف نُخفي العمود؟

لكي نفعل ذلك، سنستخدم المعامل الرابع في حدث الإنشاء عند تعريف العمود، كالتالي:

**Dim Col As New DataColumn("Temp", GetType(Integer), \_**

**"ID \* AuthorID", MappingType.Hidden)**

الآن لن يتمّ عرض هذا العمود للمستخدم.

ولكن ما زال أمامنا بعض العمل، فلو تمّ استدعاء الكود الذي يضيف العمود المؤقّت لجدول الكتب أكثر من مرّة، فسيحدث خطأ، نتيجة لأنّك تحاول إضافة عمود له نفس اسم عمود موجود سابقا.. إنّ هذا يستدعي أن نستخدم الوسيلة Contains الخاصّة بمجموعة أعمدة الجدول Columns للتأكّد من وجود العمود قبل محاولة إضافته.. هذه طريقة.. ولكنّ الأفضل، هو حذف العمود المؤقّت بعد الانتهاء من استخدامه، وذلك حتّى لا يتمّ حفظه في قاعدة البيانات لو قام المستخدم بتحديثها.. وإلا فإنّ عليك أن تتأكّد من عدم وجود هذا العمود في الجدول قبل حفظه في قاعدة البيانات، فإن كان موجودا، فعليك أن تحذفه أولا قبل إجراء عمليّة التحديث.

مشكلةٌ أخرى يجب أن تضعها في اعتبارك، هي تلك التي ستحدث لو كان ناتج عمليّة الحساب عَدَمًا DBNull (بسبب أنّ الجدول فارغ مثلا).. في هذه الحالة لو حاولت أن تحوّل هذه النتيجة إلى نصّ لعرضها في الرسالة فسيحدث خطأ.. لهذا فلا بدّ أن تتأكّد أولا من هذه الحالة.. عامّةً ها هو ذا الكود:

**Dim Col As New DataColumn("Temp", GetType(Integer), \_**

**"ID \* AuthorID", MappingType.Hidden)**

**Dim T As DataTable = DsBooks1.Tables("Books")**

**If T.Columns.Contains("Temp") = False Then**

**T.Columns.Add(Col)**

**Dim C As Object = T.Compute("Sum(Temp)", "")**

**T.Columns.Remove(Col)**

**If C.GetType Is GetType(DBNull) Then C = 0**

**MsgBox(C.ToString)**

**End** **If**

لاحظ أنّ الشرط:

**If T.Columns.Contains("Temp") = False Then**

غير ضروريّ، لأنّنا نحذف العمود مباشرة بعد انتهاء استخدامنا له:

**T.Columns.Remove(Col)**

هذا إلا إذا كان الجدول الأصليّ في قاعدة البيانات يحتوي على عمود اسمه Temp!.. ولهذا فمن الأفضل أن تستخدم اسما غير مألوف لهذا العمود المؤقّت، وليكن TempAkroponnous!

طيب.. فلنكتب الكود في أعمّ صيغة ممكنة له: لو كان العمود Temp موجودا في الجدول، سنحاول إضافة عمود اسمه Temp1، فإذا كان موجودا، فسنحاول إضافة العمود Temp2.... وهكذا..

هذا هو الكود:

**Dim ColName As String = "Temp"**

**Dim N As Integer**

**Dim Col As DataColumn**

**Dim T As DataTable = DsBooks1.Tables("Books")**

**' سنستخدم جملة تكراريّة مفتوحة لتجربة اسم عمود غير مستخدم**

**Do**

**Col = New DataColumn(ColName, GetType(Integer), \_**

**"ID \* AuthorID", MappingType.Hidden)**

**If T.Columns.Contains(ColName) Then**

**N += 1**

**ColName = "Temp" & Trim(N)**

**Else**

**T.Columns.Add(Col)**

**Dim C As Object = T.Compute("Sum("&ColName &")", "")**

**T.Columns.Remove(Col)**

**If C.GetType Is GetType(DBNull) Then C = 0**

**MsgBox(C.ToString)**

**' الجملة التالية ضروريّة للخروج من الجملة التكراريّة اللانهائيّة**

**Exit Do**

**End** **If**

**Loop**

**صيغة العمود:**

رأينا كيف يمكن استخدام خاصيّة [Expression](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\5-%20تطبيق%20دوال%20التجميع%20على%20بعض%20الأعمدة.htm) الخاصّة بكائن العمود لجعل خانات العمود تحتوي على ناتج عمليّة حسابيّة معيّنة، في الواقع ليست هذه هي الوظيفة الوحيدة لهذه الخاصيّة، إذ يمكن استخدامها كشرط اختبار، لعرض سجلات الجدول التي تحقّق مواصفات معينة (وبهذا تقوم هذه الخاصيّة بنفس وظيفة خاصيّة [DefaultView.RowFilter](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\2-%20كائن%20الجدول.htm)).

وأعتقد أنّك تحتاج لأن تتعرّف على المعاملات والدوالّ التي يمكنك استخدامها لتكوين صيغة العمليّة الحسابيّة أو شرط العرض:

|  |  |
| --- | --- |
| AND  OR  NOT | لإجراء العمليّات المنطقيّة.. مثال:  Book < 'م' And AuthorID = 4 |
| IN  LIKE | يمكنك استخدامهما بنفس الطريقة التي تعرّفنا عليها عند شرح جمل SQL في [الفصل قبل السابق](file:///C:\Users\Administrator\Desktop\vb.net\16-%20SQL%20لغة%20الاستعلام%20المركّبة\3-%20استعلامات%20التحديد\3-%20حيث\2-%20معاملات%20المقارنة\4-%20يشبه.htm). |
| CONVERT | استخدم هذه الدالة للتحويل بين أنواع البيانات.. مثال:  **Col.Expression = "Convert(ID, 'System.Int32')"** |
| LEN | استخدم هذه الدالة للحصول على طول نصّ معيّن.. مثال:  **Col.Expression = "Len(Book)"**  لاحظ أنّك غير مجبر على قصر معامل هذه الدالة (والدوال التالية) على اسم أحد حقول الجدول.. إنّ تعبيرا كهذا متاح أيضا:  **Col.Expression = "Len(Book + AuthorID)"**  **ملحوظة:**  لا تحاول استخدام العلامة "&" لتشبيك النصوص، فهي غير متاحة هنا.. استخدم بدلا منها علامة الجمع "+". |
| ISNULL | هذه الدالة تعيد القيمة المرسلة في المعامل الثاني، إذا كانت الخانة الحاليّة في الحقل المحدّد في المعامل الأوّل فارغة.. مثال:  **Col.Expression = "IsNull(AuthorID, -1)"** |
| TRIM | استخدم هذه الدالة لإزالة المسافات من بداية ونهاية خانات الحقل المرسل لها كمعامل. |
| SUBSTRING | استخدم هذه الدالة للحصول على جزء من النصّ الموجود في خانات العمود المرسل لها كمعامل، بدءا من الحرف المذكور رقمه في المعامل الثاني، وبالطول المحدّد في المعامل الثالث.. (هذه الدالة تماثل دالة Mid في VB).. مثال:  **Col.Expression = "SUBSTRING(Book, 2, 8)"** |
| IIF | هذه الدالة مماثلة لدالة VB التي تحمل نفس الاسم.. ماذا؟.. لم نشرحها؟.. هلمّوا إذن:  تأخذ هذه الدالة ثلاث معاملات: الأوّل شرط سيتمّ التحقّق منه، والثاني هو القيمة المعادة إذا كان الشرط صحيحا، والثالث هو القيمة المعادة إذا كان الشرط خاطئا.. سلسة.. أليس كذلك؟  **Col.Expression = "IIF(LEN(Book)>10), 'طويل', 'قصير')"** |

ولا يتوّقف دور خاصيّة Expression على هذا، بل يمتدّ ليمنحك إمكانيّة رائعة، هي قدرتك على إجراء العمليّات الحسابيّة على الجدول الثانويّ المرتبط بالجدول الحاليّ بعلاقة، وذلك باستخدام كلمة Child.. والمثال التالي يضيف عمودا لجدول المؤلّفين، يعرض عدد الكتب التي ألّفها كلّ مؤلّف:

**Dim Col As New DataColumn("NoOfBooks", GetType(Integer))**

**Col.Expression = "Count(Child.AuthorID)"**

**DsBooks1.Tables("Authors").Columns.Add(Col)**

ولو كان الجدول يحتوي على أكثر من حقل يرتبط كلّ منها بعلاقة مختلفة بجدول آخر، فيمكنك إرسال اسم العلاقة كمعامل لتعبير Child، على غرار:

**Col.Expression = "Count(Child(AuthorsBooks).AuthorID)"**

**التعامل مع العلاقات Relations والقيود Constrains**

هناك كائنات هامّة، تمدّك بالوسائل والخصائص اللازمة للتعامل مع العلاقات والقيود، وهي:

* [**مجموعة العلاقات Relations Collection:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\2-%20التعامل%20مع%20العلاقات%20والقيود\1-%20مجموعة%20العلاقات.htm)
* [**كائن العلاقة DataRelation:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\2-%20التعامل%20مع%20العلاقات%20والقيود\2-%20كائن%20العلاقة.htm)
* [**كائن قيد التفرّد UniqueConstraint:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\2-%20التعامل%20مع%20العلاقات%20والقيود\3-%20كائن%20قيد%20التفرّد.htm)
* [**كائن القيود الثانويّة ForeignKeyConstraint:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\2-%20التعامل%20مع%20العلاقات%20والقيود\4-%20كائن%20القيود%20الثانويّة.htm)

**مجموعة العلاقات Relations Collection:**

وهي تحتوي على العَلاقات بين الجداول في صورة كائنات من النوع DataRelation.. والكود التالي يطبع كلّ العَلاقات الموجودة بين جداول مجموعة البيانات ـ إن وجدت:

**Dim rel As System.Data.DataRelation**

**For Each rel In DsBooks1.Relations**

**Console.WriteLine(rel.RelationName)**

**Next**

وتمتلك مجموعة العلاقات نفس الخصائص والوسائل التي تمتلكها [مجموعة الجداول](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\1-%20التعامل%20مع%20الجداول\1-%20مجموعة%20الجداول.htm).

**كائن العلاقة DataRelation:**

يمثّل العلاقة المنشأة بين جدولين.. ويمكن تعريف العلاقة كالتالي:

**Dim r As New DataRelation(اسم العلاقة, العمود الرئيسيّ, العمود الثانويّ)**

ولو كانت العلاقة بين أكثر من عمود في الجدول الرئيسيّ وأكثر من عمود في الجدول الفرعيّ، فهناك صيغة أخرى تقبل مصفوفتين في المعاملين الثاني والثالث.

ويمكن أن تضيف معاملا رابعا، لو وضعت به القيمة True، سيتمّ إنشاء قيد Constraint على العلاقة.

والمثال التالي يريك كيف تنشئ علاقة بين حقل ID في جدول المؤلّفين، وحقل AuthorID في جدول الكتب:

**Dim ID As DataColumn = DsBooks1.Tables("Authors").Columns("ID")**

**Dim AuthorID As DataColumn = \_**

**DsBooks1.Tables("Books").Columns("AuthorID")**

**Dim R As New DataRelation("AuthorsBooks", ID, AuthorID)**

لاحظ أنّ هذه العلاقة لم تُضَف إلى مجموعة البيانات إلى الآن، لهذا علينا إضافتها:

**DsBooks1.Relations.Add(R)**

وطبعا لو حاولت أن تنفّذ هذا الكود، فسيحدث خطأ، لأنّ هذه العلاقة موجودة من قبل في قاعدة البيانات، ولن يمكنك إعادة إنشائها (حتّى ولو غيّرت اسمها)، إلا إذا حذفت العلاقة القديمة أولا، بجملة كالتالية:

**DsBooks1.Relations.RemoveAt(0)**

**ويمتلك كائن العلاقة الخصائص التالية:**

|  |  |
| --- | --- |
| تقرأ أو تغيّر اسم العَلاقة. | RelationName |
| تعيد اسم الجدول الرئيسيّ في هذه العلاقة. | ParentTable |
| تعيد اسم الجدول الثانويّ في هذه العلاقة. | [ChildTable](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadatarelationclasschildtabletopic.htm) |
| تعيد مصفوفة تحتوي على الأعمدة الرئيسيّة في هذه العلاقة. | ParentColumns |
| تعيد مصفوفة تحتوي على الأعمدة الثانويّة في هذه العلاقة. | ChildColumns |
| تعيد كائنا، يمثّل القيود المفروضة على المفتاح الرئيسيّ UniqueConstraint في هذه العلاقة. | ParentKeyConstraint |
| تعيد كائنا، يمثّل القيود المفروضة على المفتاح الثانويّ ForeignKeyConstraint في هذه العلاقة. | ChildKeyConstraint |
| تعيد كائن مجموعة البيانات DataSet Object، التي تنتمي إليها هذه العلاقة. | DataSet |
| مماثلة لتلك الخاصّة [بكائن الجدول](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\1-%20التعامل%20مع%20الجداول\2-%20كائن%20الجدول.htm). | ExtendedProperties |

**كائن قيد التفرّد UniqueConstraint:**

يمثّل هذا الكائن القيد الوحيد المفروض على الحقل الذي تريد التأكّد من تفرّد قيمه وعدم تكرارها.

ولتعريف هذا القيد على الحقل ID لجدول المؤلّفين، استخدم الكود التالي:

**Dim ID As DataColumn = DsBooks1.Tables("Authors").Columns("ID")**

**Dim U As New UniqueConstraint("IDUnique", ID, True)**

حيث يمثّل المعامل الأوّل اسم القيد، والمعامل الثاني الحقل الذي سينطبق عليه، بينما وضع القيمة True في المعامل الثالث يدلّ على أنّ هذا القيد ينطبق على الحقل الأساسيّ للجدول.

وطبعا لم تتمَّ إضافة هذا القيد لجدول المؤلّفين بعد، حيث يجب أن تضيفه بالجملة التالية:

**DsBooks1.Tables("Authors").Constraints.Add(U)**

ليس هذا فحسب، بل إنّ هذا القيد لن يعمل، إلا إذا جعلت لخاصيّة "فرض القيود" EnforceConstraints الخاصّة بمجموعة البيانات القيمة True:

**DsBooks1.EnforceConstraints = True**

ولا داعي لتذكيرك أنّ هذا القيد موجود مسبّقا في قاعدة البيانات، وأنّ محاولة إنشائك له مرّة أخرى ستتسبب في حدوث خطإ.. إنّ الكود المذكور هنا للتمثيل ليس إلا.

**هذا، ويمتلك هذا القيد الخصائص التالية:**

|  |  |
| --- | --- |
| تعيد مصفوفة تحتوي على الأعمدة التي يؤثّر عليها هذا القيد. | Columns |
| تقرأ أو تغيّر اسم القيد. | ConstraintName |
| مشابهة لتلك الخاصّة بكائن الجدول [DataTable](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\1-%20التعامل%20مع%20الجداول\2-%20كائن%20الجدول.htm). | ExtendedProperties |
| تعيد True إذا كان هذا القيد مفروضا على المفتاح الأساسيّ للجدول. | IsPrimaryKey |
| تعيد الجدول الذي تنتمي إليه هذا العلاقة. | Table |

**كائن القيود الثانويّة ForeignKeyConstraint:**

يمثّل هذا الكائن القيود المفروضة على السجلات في الجدول الثانويّ، والذي يجب مراعاته عند حذف أو تعديل حقل المفتاح الأساسيّ في الجدول الرئيسيّ.

تعال نعرّف قيدا من هذا النوع:

**Dim ID As DataColumn = DsBooks1.Tables("Authors").Columns("ID")**

**Dim AuthorID As DataColumn = \_**

**DsBooks1.Tables("Books").Columns("AuthorID")**

**Dim F As New ForeignKeyConstraint("AuthorIDConst", ID, AuthorID)**

حيث يمثّل المعامل الأوّل اسم القيد، والمعامل الثاني الحقل الأساسيّ، والمعامل الثالث الحقل الثانويّ.

وهناك صيغة تسمح لك بتعريف قواعد الحذف والتحديث وقبول أو رفض التغييرات.. وسنتعرّف على هذه القواعد ضمن خصائص هذا القيد.

وطبعا يجب أن تضيف هذا القيد لمجموعة قيود جدول المؤلّفين، كالتالي:

**DsBooks1.Tables("Authors").Constraints.Add(F)**

ولا تنسَ أن القيود لا تعمل إلا إذا كانت لخاصيّة "فرض القيود" EnforceConstraints الخاصّة بمجموعة البيانات القيمة True.

وبالإضافة للخصائص **Columns** و **ConstraintName** و **ExtendedProperties** و **Table** التي تعرّفنا عليها في كائن قيد التفرّد ، فإنّ كائن القيد الثانويّ يمتلك الخصائص التالية:

|  |  |
| --- | --- |
| الجدول الرئيسيّ في هذه العلاقة. | RelatedTable |
| تعيد لك هذه الخاصيّة مصفوفة تحتوي على الأعمدة الرئيسيّة في هذه العلاقة. | RelatedColumns |
| توضّح الفعل الذي سيتمّ اتّخاذه عند قبول أو رفض التغييرات المدخلة على السجلات (عن طريق استدعاء الوسيلة AcceptChanges أو الوسيلة RejectChanges على التناظر).. وتأخذ هذه الخاصيّة إحدى قيمتي المرقّم AcceptRejectRule، وهما:  - Cascade: قبول التغييرات أو رفضها على التوالي.. بمعنى أنّ قبول التغييرات في السجلّ الرئيسيّ يؤدّي لقبول التغييرات في السجلات الثانويّة المرتبطة معه بالعلاقة.  - None: لا يتمّ اتخاذ أيّ فعل عبر العلاقة، في حالتي قبول التغييرات ورفضها. | AcceptRejectRule |
| توضّح الفعل الذي سيتمّ اتّخاذه عند حذف أحد السجلات من الجدول الرئيسيّ.. وتأخذ هذه الخاصيّة إحدى قيمتي المرقّم Rule، وهي:  - Cascade: سيؤدّي حذف السجلّ إلى حذف كلّ السجلات الثانويّة المرتبطة به.. فمثلا، سيؤدّي حذف أحد المؤلّفين من جدول المؤلّفين، إلى حذف كلّ كتبه من جدول الكتب.  - SetDefault:. سيؤدّي حذف السجلّ إلى وضع القيم الافتراضيّة في الحقول الثانويّة المرتبطة به.  - SetNull: سيؤدّي حذف السجلّ إلى إفراغ الحقول الثانويّة المرتبطة به من محتوياتها.  - None: لن يتمّ اتخاذ أيّ فعل، ولكن سيتمّ إطلاق استثناء لإخبارك بأنّ الجداول صارت تحتوي على بيانات غير صحيحة (لأنّ بعض الحقول تنتمي بعلاقة لسجلّ محذوف). | DeleteRule |
| توضّح الفعل الذي سيتمّ اتّخاذه عند تحديث أحد السجلات من الجدول الرئيسيّ.. وتأخذ هذه الخاصيّة إحدى قيمتي المرقّم Rule، التي شرحناها في الخاصيّة السابقة (طبعا مع الأخذ في الاعتبار أنّ الفعل هو تحديث السجلات وليس حذفها). | UpdateRule |

**التعامل مع العُروض Views**

هناك كائنان هامّان، يمدّانك بالوسائل والخصائص اللازمة للتعامل مع العُروض، وهما:

* [**كائن العرض DataView:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\3-%20التعامل%20مع%20العُروض\1-%20كائن%20العرض.htm)
* [**كائن سجلّ العرض DataRowView:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\3-%20التعامل%20مع%20العُروض\2-%20كائن%20سجلّ%20العرض.htm)
* [**كائن إدارة العرض DataViewManager:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\3-%20التعامل%20مع%20العُروض\3-%20كائن%20إدارة%20العرض.htm)
* [**كائن إعدادات العرض DataViewSetting:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\3-%20التعامل%20مع%20العُروض\4-%20كائن%20إعدادات%20العرض.htm)

**كائن العرض DataView:**

يمثّل هذا الكائن إحدى الطرق التي يمكن عرض البيانات بها للمستخدم (راجع ذلك في [أوّل فصول قواعد البيانات](file:///C:\Users\Administrator\Desktop\vb.net\15-%20تركيب%20قواعد%20البيانات\4-%20إنشاء%20قواعد%20بيانات\3-%20أهمّ%20عناصر%20الرابطة\2-%20العروض.htm)).

ولحسن الحظّ، هناك أداة تمثّل هذا الكائن، ستجدها تحت شريط Data في صندوق الأدوات باسم DataView.. ويمكنك أن تضيف نسخا من هذه الأداة للنموذج، حيث ستضاف لصينيّة المكوّنات Component Tray أسفل مصمّم النماذج.. حينئذٍ يمكنك تعديل خصائصها باستخدام نافذة الخصائص.

ويمكنك ربط مربّعات النصوص واللافتات والقوائم وجدول البيانات DataGrid بهذه الأداة، بنفس الطريقة التي تربطها بها بجداول مجموعة البيانات، وبذلك يمكنك عرض بيانات هذه الأداة للمستخدم.

وقد تعاملنا مع كائن العرض من قبل، فخاصيّة DefaultView الخاصّة [بكائن الجدول](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\1-%20التعامل%20مع%20الجداول\2-%20كائن%20الجدول.htm) من نوع هذا الكائن.

**ويمتلك كائن العرض الخصائص التالية:**

|  |  |
| --- | --- |
| الجدول الذي يتمّ عرض سجلاته. | Table |
| الشرط الذي على أساسه يتمّ انتقاء السجلات من الجدول الأصليّ لعرضها.. (راجع خاصيّة DefaultView الخاصّة [بكائن الجدول](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\1-%20التعامل%20مع%20الجداول\2-%20كائن%20الجدول.htm)). | RowFilter |
| [حالة السجلات](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\7-%20تحديث%20مجموعات%20البيانات\6-%20حالات%20السجلّ.htm) التي تريد عرضها (السجلات الأصليّة، المعدّلة، المحذوفة، المضافة... إلخ). | RowStateFilter |
| استخدم هذه الخاصيّة للتعامل مع السجلات المعروضة عن طريق أرقامها.. لاحظ أنّ هذه الخاصيّة هي الخاصيّة الافتراضيّة، ويمكن ألا تكتب اسمها. | Item |
| عدد السجلات المعروضة. | Count |
| اجعل هذه الخاصيّة True لتسمح بحذف السجلات المعروضة. | AllowDelete |
| اجعل هذه الخاصيّة True لتسمح بتحرير السجلات المعروضة. | AllowEdit |
| اجعل هذه الخاصيّة True لتسمح بإضافة بعض السجلات لتلك المعروضة. | AllowNew |
| نصّ يمثّل عمودا (أو أكثر) ليتمّ الترتيب تبعا له.. ويمكن أن يحتوي هذا النصّ على اتجاه الترتيب (تصاعديّ ASC أو تنازليّ DESC).. مثال:  **DataView1.Sort = "Author DESC"** | Sort |
| اجعل هذه الخاصيّة True لترتيب السجلات افتراضيّا. | ApplyDefaultSort |

وبالإضافة للوسائل **BeginInit** ـ **EndInit** ـ **CopyTo** التي تعرّفنا عليها سابقا في كائن الجدول، فإنّ كائن العرض يمتلك الوسائل التالية:

|  |  |
| --- | --- |
| تضيف سجلا جديدا للعرض.. (لاحظ أنّ هذا السجلّ سيضاف للجدول الأصليّ).. وتعيد هذه الوسيلة كائنا من النوع "سجلّ العرض" DataRowView. | AddNew |
| تحذف السجلّ الذي تُمدُّها برقمه. | Delete |
| استخدم هذه الوسيلة للبحث عن قيمة ما في السجلات المعروضة، حيث تعيد لك رقم أول سجلّ يحتوي على القيمة المطلوبة.. ويمكن أن ترسل لها مصفوفة من القيم لتعثر على السجلّ الذي تحتوي خاناته المناظرة على هذه القيم.. وفي حالة عدم العثور على أيّ سجلّ، تعيد هذه الوسيلة -1. | Find |
| مماثلة للوسيلة السابقة، ولكنّها تعيد مصفوفة (خاناتها من النوع DataRowView) تحتوي على كلّ السجلات التي عثرت بها على القيم المطلوبة. | FindRows |

كما يمتلك هذا الكائن الحدث ListChanged، الذي يحدث عندما يتمّ إضافة سجلّ إلى أحد الجداول المعروضة أو عندما يتمّ حذف سجلّ منها.

**كائن سجلّ العرض DataRowView:**

يمثّل هذا الكائن أحد السجلات المعروضة في كائن العرض DataView.. ويمتلك هذا الكائن الخصائص التالية:

|  |  |
| --- | --- |
| تعيد كائن العرض الذي ينتمي إليه هذا السجلّ. | DataView |
| تعيد True إذا كان السجلّ الحاليّ يتمّ تحريره. | IsEdit |
| تعيد True إذا كان السجلّ الحاليّ جديدا. | IsNew |
| هذه الخاصيّة افتراضيّة، وهي تمكّنك من قراءة قيمة إحدى خانات هذا السجلّ، تبعا لاسم العمود الذي ترسله لها كمعامل. | Item |
| تعيد كائن السجل DataRow الذي يمثّل السجلّ الحاليّ. | Row |
| تعيد قيمة تدلّ على ما إذا كان السجلّ أصليّا، أم معدّلا، أم قيد التعديل، ... إلخ.  [وسنتعرّف على هذا لاحقا](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\7-%20تحديث%20مجموعات%20البيانات\5-%20إصدارات%20السجلّ.htm). | RowVersion |

وبالإضافة للوسائل **BeginEdit** ـ **EndEdit** ـ **CancelEdit** ـ **Delete** (التي تعرّفنا عليها في كائن السجلّ [DataRow](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\1-%20التعامل%20مع%20الجداول\3-%20كائن%20الصفّ.htm))، يمتلك السجلّ المعروض وسيلةً هامّة، وهي CreateChildView.. هذه الوسيلة تُستخدم إذا كان السجلّ المعروض موجودا في جدول أساسيّ يرتبط بعلاقة بجدول ثانويّ.. في هذه الحالة تستقبل هذه الوسيلة اسم العلاقة (أو كائنا يمثّلها)، وتعيد كائن عرضٍ DataView يحتوي على كلّ السجلات الموجودة في الجدول الثانويّ، المرتبطة بعلاقة بهذا السجلّ.. معنى هذا أنّك تستطيع استخدام هذه الوسيلة للحصول على كائن عرض يحتوي على كلّ كتب المؤلّف الذي يمثله السجلّ الحاليّ.. انظر للكود التالي:

**Dim DRV As DataRowView**

**DRV =DsBooks1.Tables("Authors").DefaultView(0)**

**Dim DV As DataView = DRV.CreateChildView("AuthorsBooks")**

**كائن إدارة العرض DataViewManager:**

يمثّل هذا الكائن كلّ الجداول المعروضة من مجموعة البيانات، حيث يحتوي على مجموعة إعدادات العرض DataViewSettingCollection لكلّ جداول من جداول مجموعة البيانات.

ويمكن الحصول على هذا الكائن من مجموعة البيانات، باستخدام خاصيّة DefaultViewManager، كالتالي:

**Dim DVM As**  [**DataViewManager**](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadataviewsettingcollectionclasstopic.htm)

**DVM = DsBooks1.DefaultViewManager**

أو يمكن فعل ذلك عن طريق إرسال مجموعة البيانات كمعامل لمنشئ هذا الكائن عند تعريفه كالتالي:

**Dim DVM As**  [**New DataViewManager**](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdatadataviewsettingcollectionclasstopic.htm) **(DsBooks1)**

جدير بالذكر أنّ هناك أداة تمثّل هذا الكائن، ولكنّك لن تجدها في صندوق الأدوات.. عليك أولا أن تضيفها له.. اضغط بزرّ الفأرة الأيمن على صندوق الأدوات، ومن القائمة الموضعيّة اضغط الأمر Customize ToolBox.. اضغط شريط .Net Components، وابحث عن الأداة DataViewManager في قائمة الأدوات.. ضع علامة الاختيار بجوار هذه الأداة، وأغلق مربّع الحوار.. يمكنك الآن أن تضيف نسخةً من هذه الأداة إلى النموذج.

وفيما يلي أهمّ خصائص هذا الكائن:

|  |  |
| --- | --- |
| تقرأ أو تغيّر مجموعة البيانات التي يتعامل معها هذا الكائن. | DataSet |
| تعيد مجموعة إعدادات العرض DataViewSettingCollection، التي تحتوي على إعدادات العرض لكلّ جدول في مجموعة البيانات. | DataViewSettings |

ويمتلك هذا الكائن الحدث ListChanged، الذي يحدث عندما يتمّ إضافة سجلّ إلى أحد الجداول المعروضة أو عندما يتمّ حذف سجلّ منها.

**كائن إعدادات العرض DataViewSetting:**

يحتوي هذا الكائن على إعدادات العرض الافتراضيّة التي يتمّ استخدامها مع الجداول المعروضة.

ويمتلك هذا الكائن الخصائص:

**ApplyDefaultSort** ـ **DataViewManager** ـ **RowFilter** ـ **RowStateFilter** ـ **Sort** ـ **Table**.

وكلّها كما ترى مألوفة لنا.

**التعامل مع الارتباطات Bindings**

رأينا كيف نربط الأدوات بجداول مجموعة البيانات، لتعرض محتويات حقول معيّنة.. هنا نرى كيف نتحكّم في هذه الارتباطات:

* [**كائن الارتباط Binding:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\4-%20التعامل%20مع%20الارتباطات\1-%20كائن%20الارتباط.htm)
* [**كائن ربط المحتوي BindingContext:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\4-%20التعامل%20مع%20الارتباطات\2-%20كائن%20ربط%20المحتوي.htm)
* [**كائن أساس الارتباط BindingManagerBase:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\4-%20التعامل%20مع%20الارتباطات\3-%20كائن%20أساس%20الارتباط.htm)

**كائن الارتباط Binding:**

يقوم هذا الكائن بربط قيمة إحدى خصائص كائن ما بإحدى خصائص أداة معيّنة، بحيث يتمّ وضع قيمة خاصيّة الكائن في خاصيّة الأداة تلقائيّا كلّما تغيّرت، والعكس.. باختصار: تصير الخاصيّتان كأنّهما خاصيّة واحدة.

وما يعنينا هنا، هو استخدام هذا الكائن في ربط محتويات أحد خانات الجدول، الواقعة في السجلّ الحاليّ في عمود محدّد، بإحدى خصائص الأداة (مثل خاصيّة النصّ لمربّع النصّ).

ولقد رأينا كيفيّة فعل ذلك باستخدام نافذة الخصائص في الفصل السابق.. الآن نرى كيف نفعل هذا برمجيّا من خلال الكود.

سنعرّف الآن كائنا يربط خاصيّة Text الخاصّة بمربّع النصّ باسم الكتاب في جدول الكتب:

**Dim B As New Binding("Text", DsBooks1, "Books.Book"))**

أعتقد أنّ المعاملات واضحة: الخاصيّة، مجموعة البيانات، اسم العمود في أحد جداول مجموعة البيانات.

وطبعا تعرف أنّ هذا الارتباط لن يأخذ حيّزا من التنفيذ إلا إذا أضفناه لمجموعة الارتباطات ControlBindingsCollection الخاصّة بمربّع النصّ، والتي تمثّلها خاصيّة "ارتباطات البيانات" DataBindings، كالتالي:

**TextBox1.DataBindings.Add (B)**

ولحسن الحظّ لا تمتلك مجموعة ارتباطات الأداة ControlBindingsCollection خصائصَ أو وسائلَ جديدةً علينا حتّى تستحقّ أن نشرحها منفردة.

**نفرغ إذن للتعرّف على خصائص كائن الارتباط Binding:**

|  |  |
| --- | --- |
| تعيد الأداة التي ينتمي إليها الارتباط الحاليّ. | Control |
| اسم الخاصيّة التي تشارك بها الأداة في عمليّة الارتباط. | PropertyName |
| تعيد مصدر البيانات (مجموعة البيانات) المشتركة في هذا الارتباط. | DataSource |
| تعيد كائنا يحتوي على معلومات حول عنصر البيانات المرتبط بالأداة. | BindingMemberInfo |
| تعيد كائن أساس الارتباط BindingManagerBase، الذي يتيح لك التحكّم في الارتباط الحاليّ.. وقد دارت بيننا مناقشة ساخنة بخصوصه في [الفصل السابق](file:///C:\Users\Administrator\Desktop\vb.net\17-%20تطبيقات%20قواعد%20البيانات\6-%20ربط%20البيانات\2-%20كائن%20ربط%20المحتوى.htm). | BindingManagerBase |
| تعيد True إذا كان الارتباط فعالا (سنرى بعد قليل [كيف نوقف الارتباط عن العمل مؤقّتا](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\3-%20كائن%20أساس%20الارتباط.htm)). | IsBinding |

**هذا، ويمنحك كائن الارتباط الحدثين التاليين، اللذين يعتبران في غاية الأهميّة:**

|  |  |
| --- | --- |
| يحدث قبل كتابة قيمة جديدة في خاصيّة الأداة، وذلك حتّى يسمح لك بتنسيق البيانات ليتمّ كتابتها بالشكل الذي تريده.  ويمتلك المعامل e لهذا الحدث الخاصيتين:  - DesiredType: تعيد نوع البيانات التي سيتمّ عرضها في الأداة.  - Value: تسمح لك بقراءة وتغيير البيانات التي سيتمّ عرضها في الأداة.. انظر للمثال التالي:  e.Value = CType(e.Value, decimal).ToString("c") | Format |
| يحدث عندما تتغيّر قيمة خاصيّة الأداة، وذلك حتّى بسمح لك باستعادة تنسيق البيانات الأصليّ، ليتمّ نقلها لمجموعة البيانات.  والمعامل e لهذا الحدث مماثل لذلك الخاصّ بالحدث السابق.. انظر كيف نستعيد الرقم الأصليّ من النصّ المنسّق:  **e.Value = Decimal.Parse(e.Value.ToString, \_**  **NumberStyles.Currency, Nothing)** | Parse |

**كائن ربط المحتوي BindingContext:**

تعرّفنا في [الفصل السابق](file:///C:\Users\Administrator\Desktop\vb.net\17-%20تطبيقات%20قواعد%20البيانات\6-%20ربط%20البيانات\2-%20كائن%20ربط%20المحتوى.htm) على كائن ربط المحتوى، الذي تمنحه لنا خاصيّة BindingContext الخاصّة بالنموذج، ورأينا كيف يمكن استخدامه لإجراء الحوار بين مجموعة البيانات والبيانات المعروضة في أدوات النموذج:

**Dim BC As BindingContext = Me.BindingContext**

ويجب أن تلاحظ أن عليك استخدام كائن ربط المحتوي الخاصّ بالأدوات الحاوية Container Controls (مثل اللوحة Panel أو مربّع التجميع GroupBox)، إذا كانت تحتوي الأدوات المرتبطة بالجداول.

ولا يمتلك كائن ربط المحتوي سوى خاصيّة وحيدة، هي خاصيّة العنصر Item (وهي الخاصيّة الافتراضيّة)، التي تستطيع من خلالها الحصول على كائنات أساس الارتباط BindingManagerBase، التي تتيح لك التحكّم في الارتباط بين أدوات النموذج وجداول مجموعة البيانات.

**كائن أساس الارتباط BindingManagerBase:**

يتيح لك هذا الكائن التحكّم في الارتباط المنعقد بين أحد الجداول وبين مجموعة من الأدوات.. ولتعريف نسخة من هذا الكائن، استخدم الصيغة التالية:

**Dim BM As BindingManagerBase**

وللتحكّم في الرابطة بين الأدوات وجدول الكتب، استخدم خاصيّة Item الخاصّة بكائن ربط المحتوي.. لاحظ أن Item هي الخاصيّة الافتراضيّة، لذا فلن نكتب اسمها:

**BM = Me.BindingContext(DsBooks1, "Book")**

الآن يمكنك عرض السجلّ التالي لسجلّ الكتب المعروض حاليّا في الأدوات بالجملة التالية:

**BM.Position += 1**

**والجدول التالي يوضّح خصائص كائن أساس الارتباط:**

|  |  |
| --- | --- |
| تعيد مجموعة ارتباطات BindingsCollection تحتوي على الارتباطات الموجودة بين أدوات النموذج والجدول المحدّد في كائن ربط المحتوى. | Bindings |
| عدد السجلات المرتبطة. | Count |
| تقرأ أو تغيّر موضع السجلّ الذي تعرضه أدوات النموذج حاليّا. | Position |

**أمّا الجدول التالي فيوضّح أهمّ وسائل كائن أساس الارتباط:**

|  |  |
| --- | --- |
| تضيف سجلا جديدا للجدول الذي تعرض الأدوات سجلاته. | AddNew |
| تُلغي عمليّة التحرير الحاليّة، بحيث يستعيد السجلّ القيمة السابقة لعمليّة التحرير. | CancelCurrentEdit |
| تُنهي عمليّة التحرير، حيث سيقبل السجلّ القيم التي أدخلها المستخدم.. استدعِ هذه الوسيلة قبل حفظ البيانات لقاعدة البيانات. | EndCurrentEdit |
| تقرأ أو تغيّر المجموعة Collection التي تحتوي على خصائص مصدر البيانات DataSource. | GetItemProperties |
| تحذف السجل الموجود في الموضع المرسل كمعامل. | RemoveAt |
| توقف الارتباط بين الأدوات ومجموعة البيانات مؤقّتا. | SuspendBinding |
| تعيد الارتباط بين الأدوات ومجموعة البيانات. | ResumeBinding |

جدير بالذكر أنّ كائن أساس الارتباط يمتلك الحدث PositionChanged، الذي ينطلق كلّما تغيّر موضع السجلّ الحاليّ.. ولا أعتقد أنّك تحتاج لتوجيهٍ بخصوص تعريف متغيّر من هذا الكائن على مستوى النموذج مع استخدام كلمة WithEvents حتّى تتمكّن من استخدام هذا الحدث الخاصّ به.. أو يمكنك استخدام جملة AddHandler لو شئت ([راجع الفصل الخامس، الفقرة الخاصّة بالنماذج المرنة](file:///C:\Users\Administrator\Desktop\01-%20أساسيّات%20اللغة\05-%20Forms%20النماذج\4-%20بناء%20نماذج%20مرنة\4-%20إنشاء%20مستجيبات%20للأحداث%20في%20وقت%20التنفيذ.htm)).

**مجموعات البيانات محدّدة النوع Typed DataSets:**

عرفنا من قبل أنّ الجملة التالية تقرأ اسم المؤلّف الموجود في الصفّ الثالث في جدول المؤلّفين:

**DsBooks1.Tables("Authors").Rows(2).Item("Author")**

واضح طبعا أنّها جملة طويلة تدفع للاستياء.. إذن ما رأيك في الجملة التالية:

**DsBooks1.Authors(2).CustomerID**

مختصرةٌ، أليس كذلك؟

إنّ الجملتين كلتيهما ـ وللعجب ـ متكافئتان، وإن كانت الأولى عامّة، والثانية خاصّة لا تصلح إلا عند التعامل مع مجموعة البيانات محدّدة النوع Typed DataSet.

ماذا يعني هذا المصطلح؟

يعني أنّ مجموعة البيانات قد تمّ إنشاؤها في وقت التصميم، ممّا سمح لـ VB بالقيام بإنشاء خليّة Class تحتوي على تركيبها بما تحتويه من جداول وعلاقات.. وهكذا يمكن استخدام الجملة المختصرة السابقة.

ولكن لو قمت بإنشاء مجموعة البيانات في وقت التنفيذ باستخدام الكود، فلا بدّ من استخدام الجملة الطويلة الأولى.. لاحظ أنّ ذلك يختلف عن إنشاء مجموعة بيانات في وقت التصميم، ثمّ إنشاء نسخ منها في الكود في وقت التنفيذ.

ونظرا لأنّ كلّ ما أنشأناه من مجموعات البيانات حتّى الآن محدّدة النوع Typed، فلا ريبَ أنّك تتساءل: أين إذن تلك الخلايا التي تمّ إنشاؤها لمجموعات البيانات تلك؟

حسنا: افتح أيّ مشروع أنشأناه في الفصل السابق، واضغط زر "عرض كل الملفّات " Show All Files الموجود أعلى نافذة متصفّح المشاريع Solution Explorer.. ستجد أنّ ملفّا امتداده .vb قد ظهر تحت الملفّ ذا الامتداد .XDS الذي يمثّل مجموعة البيانات.. هذا الملفّ الجديد هو الخليّة التي تحتوي على تعريف جداول وحقول مجموعة البيانات.. افتح هذه الخليّة وتصفّحها.. ولكن لا تحاول تغيير أيّ شيء بها.. ستلاحظ أنّ هذه الخليّة تحتوي على تعريفات خلايا أخرى، هي خلايا الجداول وصفوف الجداول التي تحتوي عليها مجموعة البيانات.. وطبعا لكلّ خليّة من هذه الخلايا وسائلها وخصائصها وأحداثها.

ولقد رأينا في المواضيع السابقة كيف نتعامل مع الجداول والسجلات من خلال كائنات مجموعة البيانات.. وكنّا نكتب الكود التالي لإضافة سجلّ جديد لجدول الكتب:

**Dim R As DataRow**

**R = DsBooks1.Tables("Books").NewRow**

**DsBooks1.Tables("Books").Rows.Add(R)**

ونظرا لأنّ مجموعة البيانات DsBooks1 منشأة من المخطّط Books في وقت التصميم، فإنّ لدينا خليّة تحمل نفس اسم المخطّط Books، يمكننا استخدامها في إعادة كتابة الكود السابق كالتالي:

**Dim R As DataSet1.BooksRow**

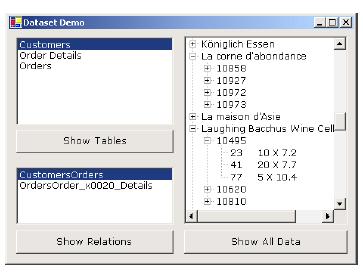
**R = DsBooks1.Books.NewBooksRow**

**DsBooks1.Books.AddBooksRow(R)**

وأنت بالخيار في الكود الذي تفضّل استخدامه.

**مشروع Datasets:**

صمّم نموذجا كما في الشكل التالي:



وأنشئ مجموعة بيانات تحتوي على جدولي المؤلّفين والكتب، بالطريقة التقليديّة.. سمّ هذه المجموعة Authors.. ولا تنسَ إنشاء علاقةٍ بينَ جدولي المؤلّفينَ والكتب، في مخطّط XML بنفس الطريقة التي تعلّمناها في الفصل السابق.

عند ضغط زرّ Show Tables، سنعرض أسماء الجداول الموجودة في مجموعة البيانات، لتظهر في القائمة العلويّة اليسرى:

**ListBox1.Items.Clear()**

**Dim tbl As System.Data.DataTable**

**For Each tbl In Authors1.Tables**

**ListBox1.Items.Add(tbl.TableName)**

**Next**

عند ضغط زرّ Show Relations، سنعرض أسماء العلاقات الموجودة بين الجداول، لتظهر في القائمة الثانية.

**ListBox2.Items.Clear()**

**Dim rel As System.Data.DataRelation**

**For Each rel In Authors1.Relations**

**ListBox2.Items.Add(rel.RelationName)**

**Next**

ولو ضغطت اسم أيّ علاقةٍ في القائمة، فستظهر رسالةٌ تعرضُ خصائص هذه العلاقة:

**Dim rel As System.Data.DataRelation**

**rel = Authors1.Relations.Item(ListBox2.Text)**

**Dim msg As String**

**msg = "الجدول الرئيسيّ: " & rel.ParentTable.TableName & vbCrLf**

**msg & = "الجدول الفرعيّ: " & rel.ChildTable.TableName & vbCrLf**

**msg &= rel.ParentColumns(0).ColumnName**

**msg &= " = " & rel.ChildColumns(0).ColumnName**

**MsgBox(msg)**

وعند ضغط زرّ Show All Data، سيتمّ عرض جميع أسماء المؤلّفين في الشجرة TreeView، وتحت كلّ مؤلّف، سنعرض أسماء كتبه كعناصر فرعيّة.. في البداية يجب أن تحمّل البيانات في مجموعة البيانات.. يمكن أن تفعل هذا في ضغط هذا الزرّ، وإن كان سيؤدّي إلى مرور بعض الوقت قبلَ ظهور العناصر في الشجرة.. أو يمكنك أن تفعل ذلك في حدث تحميل النموذج.. وسأترك لك كتابة هذا الكود اليسير.

تعال إذن نرى كود ملء الشجرة بأسماء المؤلّفينَ وكتبهم.. ويمكنك أن تجد كلّ المعلومات اللازمة للتعامل مع الشجرة، في [فصل الأدوات المتقدّمة](file:///C:\Users\Administrator\Desktop\vb.net\03-%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة.htm):

**TreeView1.Nodes.Clear()**

**' وضع اسم العلاقة المنشأة على جدول المؤلّفين في متغيّر، لاستخدامها فيما بعد**

**Dim AuthorRelation As String**

**AuthorRelation = Authors1.Authors.ChildRelations(0).RelationName()**

**' تعريف متغيّر يمكن أن يحتوي على صف من صفوف جدول المؤلّفين**

**Dim row As Authors.AuthorsRow**

**' تعريف مصفوفة ومتغيّر يمكن أن يحتويا على صفوف من جدول الكتب**

**Dim BooksRow(), BRow As Authors.BooksRow**

**' تعريف متغيّر يمكن أن يحتوي على فرع من فروع الشجرة**

**Dim ANode As TreeNode**

**' جملة تكراريّة لقراءة كلّ صفوف المؤلّفين واحدًا بعد آخر**

**For Each row In Authors1.Authors**

**' إضافة اسم المؤلّف للشجرة**

**ANode = TreeView1.Nodes.Add(Trim(row.Author))**

**' وضع كتب المؤلّف في مصفوفة، باستخدام العلاقة بين جدولي المؤلّفين والكتب**

**BooksRow = row.GetChildRows(AuthorRelation)**

**' جملة تكراريّة لقراءة كلّ خانات مصفوفة الكتب واحدة بعد أخرى**

**For Each BRow In BooksRow**

**' إضافة الكتاب إلى فرع المؤلّف الحالي**

**ANode.Nodes.Add(Trim(BRow.Book))**

**Next**

**Next**

**تحديث مجموعات البيانات Updating DataSets**

* [**التحدّيات:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات\1-%20التحدّيات.htm)
* [**حلّ مشكلة معرّفات الحقول IDS:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات\2-%20حلّ%20مشكلة%20معرّفات%20الحقول.htm)
* [**الانتقالات Transactions:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات\3-%20الانتقالات.htm)
* [**إجراء عمليّات التحديث:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات\4-%20إجراء%20عمليّات%20التحديث.htm)
* [**إصدارات السجلّ Row Versions:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات\5-%20إصدارات%20السجلّ.htm)
* [**حالات السجلّ Row States:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات\6-%20حالات%20السجلّ.htm)
* [**مزج مجموعات البيانات:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات\8-%20مزج%20مجموعات%20البيانات.htm)
* [**التعامل مع السجلات التي بها أخطاء:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\18-%20ADO.NET%20برمجة%20كائنات\7-%20تحديث%20مجموعات%20البيانات\7-%20التعامل%20مع%20السجلات%20التي%20بها%20أخطاء.htm)

**التحدّيات:**

رأينا في الفصل السابق كيف نستخدم موصّل البيانات DataAdapter لحفظ التغييرات التي تحدث لسجلات مجموعة البيانات في قاعدة البيانات..

للأسف: ليست هذه طريقة يعتمد عليها.. افترض أنّ هناك تغييرا في بيانات 30 سجلا، تحاول حفظها في قاعدة البيانات.. ستبرز المشكلة إذا كانت هناك أخطاء في بعض هذه السجلات، ففي هذه الحالة لن يتمّ حفظ السجلات التي بها أخطاء.. معنى هذا أنّك تحتاج للبحث عن هذه السجلات، حيث ستعرضها للمستخدم وتطلب منه تعديل ما بها من أخطاء، قبل أن تحاول حفظها من جديد.

ولكنّ هناك حالات لا يمكن أن تطلب فيها من المستخدم أن يعدّل الخطأ.. افترض أنّك تريد حفظ التغييرات في بعض السجلات، فاكتشفت أنّها قد حذفت من قاعدة البيانات بواسطة مستخدم آخر!.. في هذه الحالة سيحدث خطأ، ولن يستطيع موصّل البيانات إكمال حفظ باقي السجلات.

مشكلة أخرى هي مشكلة القيود Constrains المفروضة على قيم الحقول، مثل أن ترفض القيم السالبة لأحد الحقول.. إنّ هذه القيود لا تنتقل تلقائيّا من قاعدة البيانات إلى مجموعة البيانات، ممّا يعنى أنّ حقول قاعدة البيانات قد تستقبل الكثير من القيم المرفوضة.. في هذه الحالة لن تقبل قاعدة البيانات هذه القيم.

إذن ينبغي عليك أن تكتب الكود المناسب للتحقّق من الصحّة، قبل أن تقبل البيانات من المستخدم وتضعها في مجموعة البيانات.

نوع آخر من الأخطاء المحتملة، يمكن أن يحدث عند إضافة مؤلّف لجدول المؤلّفين.. في هذه الحالة سيتمّ إنتاج قيمة الحقل ID آليّا، ولتكن 20.. ولكن ماذا لو كان هناك مستخدم آخر في هذه اللحظة قد أضاف مؤلّفا إلى مجموعة البيانات على جهازه، فأصبحت قيمة الحقل ID له 20، وسبقك لحفظ هذا السجلّ إلى قاعدة البيانات؟.. لو حاولت الآن حفظ مؤلّفك الجديد فلن تستطيع، لأنّ الحقل ID لا يقبل القيم المكرّرة!.. إنّ هذه مشكلة تحتاج للكثير من المجهود لحلّها.

أحد الحلول الممكنة عند تصميم قاعدة بيانات جديدة، هو استخدام نوع البيانات المسمّى GUID ( المعرّف المتفرّد العامّ Globally Unique Identifier) للحقل ID.. إنّ هذا يجعل الكمبيوتر ينتج رقما كبيرا لكلّ حقل، يصعب تكراره إلا في أندر الحالات.. وطبعا هذه الحالات النادرة ما زالت تحتاج لحلّ!

ويمنحك إطار العمل الخليّة Guid، وهي تمتلك وسيلة وحيدة، هي Guid.NewGuid، حيث تعيد لك معرفا متفردا عاما.

ولكنّ ماذا نفعل عندما نتعامل مع قواعد بيانات مصمّمة سابقا، والحقل ID فيها يتبع نظام الترقيم التلقائي؟.. لا مهرب من مواجهة المشكلة!

كلّ هذه المشكلات وغيرها تحتاج منّك للتدخّل لتحديث السجلات يدويّا، حتّى تستطيع التعامل مع كلّ مشكلة بما يناسبها.

**حلّ مشكلة معرّفات الحقول IDs:**

رأينا أنّ مشكلة الترقيم التلقائيّ تنتج أساسا لأنّنا نضيف سجلا جديدا لمجموعة البيانات، في نفس وقت إضافة مستخدم آخر لسجل جديد لمجموعة بياناته، ممّا يجعل للسجلين نفس ترقيم الحقل ID.. وطبعا سجلّ واحد فقط هو الذي ستقبله قاعدة البيانات عند تحديثها، بينما سترفض الآخر، لأنّ قيمة الحقل ID غير قابلة للتكرار.

حسنا.. هناك حلّ في غاية البساطة.. لا تُضِفِ السجلَّ الجديدَ لمجموعة البيانات، بل أضفْه لقاعدة البيانات نفسها مباشرةً، باستخدام كائن الأمر Command Object الذي ناقشناه [في الفصل السابق](file:///C:\Users\Administrator\Desktop\vb.net\17-%20تطبيقات%20قواعد%20البيانات\9-%20الاتصال%20بقاعدة%20البيانات%20مباشرة.htm).. هذا سيضمن أنّ قاعدة البيانات ستمنح السجلّ الجديدة ترقيما متفرّدا للحقل ID.

في البداية، سننشئ إجراءا مخزّنا Stored Procedure يضيف حقلا للجدول (وتستطيع استخدام جمل SQL مباشرةً، عبر أمر الإدراج InsertCommand في كائن الأوامر Command Object).

ها هو ذا الإجراء المخزّن الذي يضيف حقلا لجدول الكتب (راجع الإجراءات المخزّنة [في الفصل قبل السابق](file:///C:\Users\Administrator\Desktop\vb.net\16-%20SQL%20لغة%20الاستعلام%20المركّبة\5-%20الإجراءات%20المخزّنة.htm)):

**ALTER PROCEDURE NewBook**

**(**

**@AuthorID nchar(30),**

**@Book nchar(30)**

**)**

**AS**

**INSERT INTO Books (AuthorID, Book) VALUES(@AuthorID, @Book)**

**RETURN (@@IDENTITY)**

لاحظ أنّ الإجراء يعيد لك قيمة الحقل ID، وذلك عن طريق التعبير:

**RETURN (@@IDENTITY)**

وسنرى بعد قليل كيف نستقبل هذه القيمة المعادة في تطبيقنا.

ولكي نجعل كلامنا (خفيفا عليك)، سنطبّقه عمليّا.

انسخ تطبيق EditBooks من مجلّد برامج الفصل السابق إلى أيّ مجلّد آخر وافتحه.. سنقوم بإضافة بعض التعديلات لهذه التطبيق.

اسحب الإجراء المخزّن NewBook من متصفّح خوادم الإنترنت Server Explorer وألقه على النموذج.. سيقوم VB تلقائيّا بإنشاء كائن اتصال وكائن أوامر يستخدم هذا الإجراء المخزّن، ويحتوي على المعاملات Parameters اللازمة لإرسال معاملات الإجراء المخزّن واستقبال القيمة العائدة منه.. باختصار: ليس مطلوبا منك فعل أيّ شيء!

الآن تعال نكتب الكود الذي يضيف كتابا للمؤلّف الحاليّ.. علينا أن نتّبع هذه الخطوات:

1- سنطلب من المستخدم إدخال اسم الكتاب في مربّع إدخال InputBox.

**Dim BookName As String**

**BookName = InputBox("أدخل اسم الكتاب")**

**If BookName = "" Then Exit Sub**

2- هذا الاسم سنمرّره للمعامل Book@ الخاصّ بالإجراء المخزّن:

**SqlCommand1.Parameters("@Book").Value = BookName**

3- علينا أن نعرف رقم المؤلّف الحالي ID.. سنستخدم كائن ربط المحتوي لنعرف رقم السجلّ الحاليّ.. ثمّ سنقرأ قيمة الحقل ID المناظرة لهذا السجل:

**Dim Pos As Integer**

**Pos = Me.BindingContext(objBooks, "Authors").Position**

**Dim AID As Integer = objBooks.Authors(Pos).ID()**

4- هذا الرقم سنمرّره للمعامل @AuthorID الخاصّ بالإجراء المخزّن:

**SqlCommand1.Parameters("@AuthorID").Value = AID**

5- سنفتح الاتصال مع قاعدة البيانات، ونستدعي الوسيلة ExecuteScalar، حتّى تعيد لنا القيمة المعادة من الإجراء المخزّن، ثمّ نغلق الاتصال:

**SqlConnection1.Open()**

**SqlCommand1.ExecuteScalar()**

**SqlConnection1.Close()**

6- وأخيرا سنقرأ القيمة المعادة (التي تمثّل رقم الكتاب المضاف ID)، عن طريق المعامل @RETURN\_VALUE.

**Dim BookID As Integer**

**BookID = SqlCommand1.Parameters("@RETURN\_VALUE").Value**

لاحظ أنّ مجموعة البيانات لن تشعر بالحقل الذي تمّت إضافته لقاعدة البيانات.. ولمعالجة هذه المشكلة، يمكنك إعادة تحميل قاعدة البيانات في مجموعة البيانات.. طبعا هذا حلّ غير عمليّ، خاصّةً لو كان حجم قاعدة البيانات ضخما.

حلّ آخر قد يرد إلى ذهنك، هو أن تضيف سجلا جديدا لمجموعة البيانات، له نفس بيانات السجلّ الذي أضفناه لقاعدة البيانات، فكأنّه نسخة منه.. طبعا هذا ليس حلا، فهذه النسخة قد تأخذ رقم ID مختلفا تماما عن ذلك الموجود في قاعدة البيانات، وبهذا نكون قد عدنا للمشكلة التي هربنا منها!

هنا يأتي دور الرقم ID الذي استعدناه.. سنستخدم هذا الرقم لتحديد الحقل الذي تمّت إضافته لقاعدة البيانات، ونحمّله إلى مجموعة البيانات (دون أن نمحو المحتويات السابقة).. تبدو فكرةً طيّبة.. ولتنفيذها يجب أن يكون لدينا موصل بيانات DataAdapter، يحتوي على أمر تحديد SelectCommand مناسب، لنستخدمه في نقل السجل المنشود إلى مجموعة البيانات.. طبعا من اليسير عليك إضافة موصّل بيانات للنموذج وضبط خصائصه.. ولكنّنا هنا سنفعل شيئا مختلفا.. لن نضع أيّ أداة على النموذج.. سنفعل كلّ شيءٍ من الكود!.. اتبع معي هذه الخطوات:

7- سنعرّف كائن أمر Command Object ونصله بقاعدة البيانات، كالتالي:

**Dim Slct As New SqlClient.SqlCommand()**

**Slct.Connection = SqlConnection1**

8- سنعرّف معاملا Parameter، حتّى نستخدمه بعد ذلك في جملة SQL، كمعامل إدخال Input، نرسل به الرقم ID الذي يميّز الحقل الذي نريده.. تذكّر أنّنا نحتفظ بقيمة هذا المعامل في المتغيّر BookID:

**Dim p As New SqlClient.SqlParameter()**

**p.ParameterName = "@ID"**

**p.Direction = ParameterDirection.Input**

**p.SqlDbType = SqlDbType.Int**

**p.Value = BookID**

9- الآن سنضيف المعامل الذي عرّفناه لمجموعة معاملات كائن الأوامر:

**Slct.Parameters.Add(p)**

10- والآن سنكتب جملة SQL التي تعيد لنا الحقل المطلوب.. في هذه الحالة سيكون نوع الأمر CommandType نصّيّا:

**Slct.CommandType = CommandType.Text**

**Slct.CommandText = "SELECT \* FROM Books WHERE ID = @ID"**

11- ثمّ سنعرّف موصّل بيانات، ونضع في أمر التحديد الخاصّ به كائن الأمر الذي أنشأناه:

**Dim DA As New SqlClient.SqlDataAdapter()**

**DA.SelectCommand = Slct**

12- وأخيرا، سنستخدم موصّل البيانات في (ملء) جدول الكتب في مجموعة البيانات بالسجل الذي حصلنا عليه.. ونظرا لأنّنا لم نَمْحُ البيانات السابقة من مجموعة البيانات، فإنّ السجلّ سيضاف إليها.. وهو المطلوب:

**DA.Fill(objBooks, "Books")**

ها هو ذا الكود كاملا.. لاحظ أنّ علينا أن نكتب الكود الذي يعالج الاستثناءات، حتّى لا يحدث أيّ خطإ غير متوقّع:

**Try**

**Dim BookName As String**

**BookName = InputBox("أدخل اسم الكتاب")**

**If BookName = "" Then Exit Sub**

**SqlCommand1.Parameters("@Book").Value = BookName**

**Dim Pos As Integer**

**Pos = Me.BindingContext(objBooks, "Authors").Position**

**SqlCommand1.Parameters("@AuthorID").Value = \_**

**objBooks.Authors(Pos).ID()**

**SqlConnection1.Open()**

**SqlCommand1.ExecuteScalar()**

**SqlConnection1.Close()**

**Dim BookID As Integer = \_**

**SqlCommand1.Parameters("@RETURN\_VALUE").Value**

**Dim Slct As New SqlClient.SqlCommand()**

**Slct.Connection = SqlConnection1**

**Dim p As New SqlClient.SqlParameter()**

**p.ParameterName = "@ID"**

**p.Direction = ParameterDirection.Input**

**p.SqlDbType = SqlDbType.Int**

**p.Value = BookID**

**Slct.Parameters.Add(p)**

**Slct.CommandType = CommandType.Text**

**Slct.CommandText = "SELECT \* FROM Books" \_**

**&"WHERE ID = @ID"**

**Dim DA As New SqlClient.SqlDataAdapter()**

**DA.SelectCommand = Slct**

**DA.Fill(objBooks, "Books")**

**Catch exc As Exception**

**MsgBox(exc.Message)**

**End Try**

وهناك طريقة ثالثة يمكن بواسطتها إضافة الحقل الجديد لمجموعة البيانات، عن طريق استخدام الوسيلة Merge.. سنتعرّف عليها لاحقا.

**الانتقالات Transactions:**

كثيرا ما تحتاج لإجراء مجموعة من عمليّات التحديث والتعديل المتتالية لقاعدة البيانات.. وكثيرا ما تحتاج للتراجع Roll Back عن كلّ هذه العمليّات مرّة واحدة لأنّ واحدة منها لم تكن مقبولة.

لذلك، تقدّم لك تقنية ADO.NET الانتقالات Transactions، التي ستبدو لك في غاية البساطة عند استخدامها، لأنّ ADO.NET سيقوم بالعمل كلّه.. والفكرة ببساطة تتلخّص في الخوارزميّة التالية:

- ابدأ مجموعة من العمليّات على قاعدة البيانات Begin Transactions

- إذا فشلت إحدى هذه العمليّات، أو ألغى المستخدم الأمر كلّه:

تراجع عن كلّ العمليّات Roll Back.

- وإذا مضى كلّ شيءٍ على ما يرام:

احفظ كلّ العمليّات في قاعدة البيانات نهائيّا Commit.

تعال نرى ذلك بطريقة عمليّة، وذلك بإنشاء تطبيق Transaction، الذي ستجده في مجلّد برامج هذه الفصل.

واجهة هذا التطبيق بسيطة، تحتوي على زرّ واحد فقط، عند ضغطه يعرض للمستخدم رسالة تطلب منه إدخال اسم كتاب لإضافته لجدول الكتب (سنضيف هذا الكتاب لأيّ مؤلّف للتسهيل، وليكن المؤلّف رقم 4).. لو نجحت إضافة الكتاب، فستظهر رسالة ثانية، تطلب إدخال اسم كتاب آخر.. لو ضغط المستخدم زرّ الإلغاء، فسيتمّ إلغاء العمليّة بأكملها، ولن يتمّ حفظ الكتاب الأوّل أيضا.. ولو أدخل المستخدم اسم الكتاب الثاني، فسيتمّ إضافته لجدول الكتب، وحفظ هذه التغييرات نهائيّا بقاعدة البيانات.. تعال نرى كيف نفعل ذلك:

اسحب الإجراء المخزّن NewBook من متصفّح الخوادم إلى النموذج، لتحصل تلقائيّا على أداة اتصال Connection Object و كائن أوامر Command Object جاهزين للعمل.

نتفرّغ إذن لكود الزرّ:

هذه البداية مفهومة طبعا:

**Dim BookName As String = InputBox("أدخل اسم الكتاب الاوّل")**

**If BookName = "" Then Exit Sub**

أمّا السطران التاليان، فيهيّآن معاملات الإجراء المخزّن، كما شرحنا سابقا:

**SqlCommand1.Parameters("@Book").Value = BookName**

**SqlCommand1.Parameters("@AuthorID").Value = 4**

وطبعا لا بدّ أن نفتح الاتصال بيننا وبين قاعدة البيانات، قبل أن نبدأ التعامل معها:

**SqlConnection1.Open()**

لتحديد بداية عمليّات الانتقال، استخدم الوسيلة BeginTransaction الخاصّة بكائن الاتصال.. هذه الوسيلة تعيد كائنا من النوع SqlTransaction.. ولكي يمكنك استخدام هذا الكائن لمراقبة العمليّات التي سنبدأها على قاعدة البيانات، ضعه في خاصيّة Transaction الخاصّة بكائن الأوامر:

**SqlCommand1.Transaction = SqlConnection1.BeginTransaction**

الآن سننفّذ الإجراء المخزّن، لحفظ الكتاب الأوّل:

**SqlCommand1.ExecuteScalar()**

ثمّ سنطلب من المستخدم إدخال الكتاب الثاني.. فإذا ضغط زرّ الإلغاء، أو ترك اسم الكتاب فارغا، فسنستخدم الوسيلة Rollback الخاصّة بالكائن SqlTransaction لإلغاء العمليّة بكاملها.. معنى هذا أنّ الكتاب الأوّل لن يتمّ حفظه:

**BookName = InputBox("أدخل اسم الكتاب الثاني")**

**If BookName = "" Then**

**SqlCommand1.Transaction.Rollback()**

**MsgBox("لقد اخترت أن تُلغي العمليّة بأكملها")**

**Exit Sub**

**End If**

أمّا إذا أدخل اسم الكتاب الثاني، فسنقوم بتهيئة معاملات الإجراء المخزّن واستدعائه لحفظ هذا الكتاب:

**SqlCommand1.Parameters("@Book").Value = BookName**

**SqlCommand1.ExecuteScalar()**

وأخيرا، سنحفظ هذه التغييرات في قاعدة البيانات، باستخدام الوسيلة "إيداع" Commit الخاصّة بالكائن SqlTransaction.. طبعا مع إغلاق الاتصال بقاعدة البيانات:

**SqlCommand1.Transaction.Commit()**

**MsgBox("تمّت العمليّة بنجاح")**

**SqlConnection1.Close()**

وحتّى لا تشعر بالاستياء من استخدام جمل طويلة على شاكلة:

**SqlCommand1.Transaction.Commit()**

يمكنك تعريف متغيّر من النوع SqlTransaction، واستخدامه كالتالي:

**Dim Trans As SqlClient.SqlTransaction**

**Trans = SqlConnection1.BeginTransaction**

**SqlCommand1.Transaction = Trans**

**……………..**

**Trans.Commit**

**…………..**

**Trans.RollBack**

بقي شيء هامّ.. استخدام معالجات الاستثناء، توقّيا لأيّ خطإ غير محتمل أثناء التعامل مع قاعدة البيانات، حيث سنقوم في هذه الحالة بإلغاء العمليّات وعرض رسالة للمستخدم.. ها هو ذا الكود كاملا:

**Dim BookName As String = InputBox("أدخل اسم الكتاب الاوّل")**

**If BookName = "" Then Exit Sub**

**SqlCommand1.Parameters("@Book").Value = BookName**

**SqlCommand1.Parameters("@AuthorID").Value = 4**

**Try**

**SqlConnection1.Open()**

**SqlCommand1.Transaction = SqlConnection1.BeginTransaction**

**SqlCommand1.ExecuteScalar()**

**BookName = InputBox("أدخل اسم الكتاب الثاني")**

**If BookName = "" Then**

**SqlCommand1.Transaction.Rollback()**

**MsgBox("لقد اخترت أن تلغي العمليّة بأكملها")**

**Exit Sub**

**End If**

**SqlCommand1.Parameters("@Book").Value = BookName**

**SqlCommand1.ExecuteScalar()**

**SqlCommand1.Transaction.Commit()**

**MsgBox("تمّت العمليّة بنجاح")**

**Catch Exc As Exception**

**SqlCommand1.Transaction.Rollback()**

**MsgBox("تمّ إلغاء العمليّة بسبب الخطإ التالي:" & vbCrLf & Exc.Message)**

**Finally**

**SqlConnection1.Close()**

**End Try**

**إجراء عمليّات التحديث Updating:**

سنتعامل الآن مع المشكلة العتيدة: تلك هي المأساة التي ستحدث عندما تحاول أن تحفظ البيانات التي قمت بتعديلها من مجموعة البيانات DataSet إلى قاعدة البيانات، فتفاجأ بانّ مستخدمين آخرين قد غيّروا معالم قاعدة البيانات بالحذف والتعديل والإضافة، ممّا سيعوق حفظ البيانات الخاصّة بك.

وتسمّى المشكلة الناتجة عن محاولة مستخدمين أو أكثر تحديث نفس البيانات فيي قاعدة البيانات في نفس اللحظة باسم "التطابق" Concurrency.. ويمكن علاجها بأحد حلّين: التطابق المتفائل Optimistic Concurrency، والتطابق المتشائم Pessimistic Concurrency.

وتقوم تقنية ADO.NET أساسا على فكرة التطابق المتفائل Optimistic Concurrency، التي تفترض أن قاعدة البيانات لن يتمّ تعديلها من قبَلِ أيّ مستخدم، أثناء تعاملك مع بياناتها في مجموعة البيانات.. فإذا حدث العكس، ووجد ADO.NET أيّ تغيير في أحد الحقول في قاعدة البيانات، فإنّ التغييرات لا يتمّ حفظها.

ولكنّ هذا لا يعني أنّك لا تستطيع استخدام التطابق المتشائم Pessimistic Concurrency ([راجع الفصل السابق](file:///C:\Users\Administrator\Desktop\vb.net\17-%20تطبيقات%20قواعد%20البيانات\33-%20المعالج%20السحري%20لإعداد%20موصّل%20البيانات.htm))، الذي يقوم أساسا على مبدإ "آخر تحديث يكسب!".. حيث إنّ السجلات يتمّ حفظها إلى قاعدة البيانات، حتّى ولو كانت هناك تعديلات قد أجراها مستخدم آخر عليها.. إنّك تفرض سجلاتك على قاعدة البيانات رغم أنف الجميع!.. إنّ مثل هذه الحالة تناسب أنظمة حجز رحلات الطيران، لأنّ آخر تعديل في مواعيد الحجز هو الأولى بالاعتبار.

**إصدارات السجلّ DataRow’s Versions:**

ولكن كيف تعرف مجموعة البيانات DataSet أنّ أحد سجلات قاعدة البيانات قد تمّ تعديله منذ آخر اتصال؟

إنّ هذا يتمّ عن طريق حفظ نسخ مختلفة من كلّ سجلّ في مجموعة البيانات:

|  |  |
| --- | --- |
| Original | السجلّ الأصليّ الذي تمّ إحضاره من قاعدة البيانات.. وعند محاولة حفظ السجلّ، تتأكّد مجموعة البيانات أولا أنّ السجلّ الموجود في قاعدة البيانات مطابق لهذه النسخة. |
| Current | السجلّ الحاليّ الذي يحتوي على التعديلات التي أدخلها المستخدم، والتي يتمّ حفظها لقاعدة البيانات في عمليّة التحديث. |
| Proposed | السجلّ المقترح، الذي يحتوي على القيم التي ما زال المستخدم يدخلها حاليّا.. وأنت تعرف بالطبع أنّ هذه القيم لا تحفظ في مجموعة البيانات إلا إذا انتقل المستخدم إلى سجلّ آخر، أو إذا طلب حفظها (بضغط زرّ ما مثلا لتطبيق وسائل الحفظ).. وعندما يحفظ هذا السجلّ، يصير هو السجلّ الحاليّ. |
| Default | السجلّ الافتراضيّ، الذي يحتوي على القيم الافتراضيّة لحقول السجلّ. |

ولكي تحدّد القيمة التي تريد قراءتها من هذه القيم، استخدم المعامل الثاني في خاصيّة DataRow.Item.. والجملة التالية تقرأ القيمة الأصليّة لاسم الكتاب في السجلّ الأوّل من جدول الكتب:

**DsBooks11.Books.Rows(0).Item("Book", DataRowVersion.Original)**

بالمثل، يمكنك قراءة القيمة الحاليّة أو الافتراضيّة أو المقترحة لأيّ سجلّ.

ولكي تختبر ذلك، أضف زرّا لمشروع EditBooks، واجعله يعرض رسالة تسرد النسخ المختلفة للحقل Book في السجلّ الأوّل من جدول الكتب.. حاول أن تغيّر اسم المؤلف في هذا السجلّ ليتّضح لك الأمر.

وحتّى لا يحدث أيّ خطإ، نتيجة لعدم وجود نسخة معيّنة من الحقل (مثلا، لن توجد النسخة المقترحة Proposed إلا عندما يكون السجلّ تحت التعديل حاليّا، وهذا لا يتوافر باستمرار)، استخدم الوسيلة HasVersion للتأكّد من وجود النسخة قبل استخدامها، وذلك كالتالي:

**Dim row As DataRow = DsBooks11.Books.Rows(0)**

**If row.HasVersion(DataRowVersion.Proposed) Then**

**MsgBox("Books" & row.Item("Book", \_**

**DataRowVersion.Proposed))**

**End If**

حالات السجلّ DataRow States:

بخلاف إصدارات السجلّ المختلفة، توجد للسجلّ حالات مختلفة:

|  |  |
| --- | --- |
| Added | مُضاف.. هذا السجلّ ليس موجودا في قاعدة البيانات، وإنّما تمّت إضافته لمجموعة البيانات. |
| Deleted | محذوف.. تمّ حذف هذا السجلّ من مجموعة البيانات، ولكنّه ما زال موجودا في قاعدة البيانات. |
| Detached | مستقلّ.. تمّ إنشاء هذا السجلّ ولكنّه لم يحفظ في مجموعة البيانات بعد. |
| Modified | معدّل.. تمّ تعديل هذا السجلّ، ولكن لم يتمّ حفظ التعديلات لقاعدة البيانات بعد. |
| Unchanged | لم يتمّ تغيير محتويات السجلّ، منذ أن تمّ تحميله من قاعدة البيانات (أو منذ آخر حفظ). |

ويمكنك أن تستخدم هذه الحالات في تحديث قاعدة البيانات.. فبدلا من إرسال كلّ السجلات إلى قاعدة البيانات، يمكنك أن تُرسل فقط تلك السجلات المعدّلة والمضافة والمحذوفة (ليتمّ حذفُها فعلا من قاعدة البيانات)، وبهذا توفّر الكثير من الوقت.. انظر كيف نستعيد السجلات المعدّلة فقط.. تذكّر أنّ DsBooks هو اسم خليّة مجموعة البيانات، و DsBooks1 هو اسم نسخة مجموعة البيانات المعرّفة منها:

**Dim UpdatedRows As DsBooks**

**UpdatedRows = DsBooks1.GetChanges(DataRowState.Modified)**

أعتقد أنّك قد استوعبت اللعبة: أنشأنا نسخة جديدة من مجموعة البيانات، وملأناها بالسجلات المعدّلة.. الآن يمكنك أن تحدّث قاعدة البيانات بمجموعة البيانات الجديدة هذه، كالتالي:

**MsgBox(DataAdapter1.Update(UpdatedRows))**

لاحظ أنّ الرسالة ستعرض عدد السجلات التي تأثّّرت في قاعدة البيانات، لأنّ هذه هي القيمة المعادة من الوسيلة Update.

ويمكنك اتّباع نفس الطريقة مع السجلات المضافة والمحذوفة.. ولا تنس كتابة معالج الاستثناء، توقّيا لأيّ أخطاء تحدث أثناء محاولة الحفظ، وما أكثرَها!.. ويفضّل اتّباع الترتيب التالي عند التحديث: السجلات المعدّلة، السجلات المضافة، السجلات المحذوفة.

ولو أردت أن تحصل على كلّ السجلات التي حدث بها تغيير مرّة واحدة، فيمكنك استدعاء الوسيلة GetChanges بدون معاملات، كالتالي:

**UpdatedRows = DsBooks1.GetChanges()**

ويمكنك أن تتأكّد أولا إذا كانت هناك سجلات قد حدثت بها اختلافات، عن طريق استدعاء الوسيلة HasChanges، التي تعيد True في هذه الحالة.

ها هو الإجراء كاملا:

**Private Sub UpdateTable(ByVal sender As System.Object, \_**

**ByVal e As System.EventArgs) Handles bttnUpdate.Click**

**Dim UpdatedRows As Books**

**UpdatedRows = DsBooks1.GetChanges(DataRowState.Modified)**

**If Not UpdatedRows Is Nothing Then**

**DataAdapter1.Update(UpdatedRows)**

**End If**

**UpdatedRows = DsBooks1.GetChanges(DataRowState.Added)**

**If Not UpdatedRows Is Nothing Then**

**DataAdapter1.Update(UpdatedRows)**

**End If**

**UpdatedRows = DsBooks1.GetChanges(DataRowState.Deleted)**

**If Not UpdatedRows Is Nothing Then**

**DataAdapter1.Update(UpdatedRows)**

**End If**

**End Sub**

ولكن.. هل سنترك حالة السجلات بعد حفظها في قاعدة البيانات، كما هي في مجموعة البيانات؟

بالطبع لا.. عليك أن تغيّر هذه الحالة.. إمّا بأن تعيد ملأ مجموعة البيانات من قاعدة البيانات من جديد (وهو حلّ غير عمليّ في حالة قواعد البيانات الضخمة).. وإمّا باستدعاء الوسيلة "قبول التغييرات" AcceptChanges، ليتمّ معاملة السجلات الحاليّة Current باعتبارها هي السجلات الأصليّة Original، بالضبط كأنّك حمّلتها للتوّ من قاعدة البيانات:

**DsBooks1.AcceptChanges()**

ولكن.. ما أدرانا أنّ البيانات الموجودة في قاعدة البيانات ما زالت هي نفسها الموجودة في مجموعة البيانات؟.. لا تنسَ أنّ هناك مستخدمين غيرنا.

طيب.. هناك حلّ.

إنّ مجموعة البيانات التي وضعنا بها السجلات التي حدثت بها تغييرات، وقمنا بحفظها في قاعدة البيانات ستتلقّى من قاعدة البيانات بيانات محدّثة (سواء بإعادة تحميلها من جديد، أو بإرسال أيّ بيانات جديدة أضافتها قاعدة البيانات ـ مثل الرقم التلقائيّ الذي تمّ توليده لبعض الحقول).. إذن يجبَ علينا أن نضيف هذه التحديثات لمجموعة البيانات الأصليّة.. بسيطة: سنستخدم الوسيلة Merge لدمج مجموعتي البيانات معا، وذلك قبل استدعاء الوسيلة AcceptChanges.. هذا هو الموضوع بعد التالي.

**مزج مجموعات البيانات:**

تمتلك مجموعة البيانات الوسيلة Merge التي تتيح لك مزج سجلات أخرى بسجلاتها.. إنّ هذا يعني أنّ السجلات الجديدة سيتمّ إضافتها، أمّا السجلات التي تشترك في الرقم ID فسيتمّ تعديل بياناتها وحالتها تبعا لمجموعة البيانات الممزوجة.

ولهذه الوسيلة الصيغ التالية:

**DsBooks1.Merge(مصفوفة سجلات)**

**DsBooks1.Merge(جدول)**

**DsBooks1.Merge(مجموعة بيانات)**

ويمكن أن تأخذ هذه الصيغ معاملا ثانيا، إذا جعلته True فسيتمّ الاحتفاظ بالتغييرات التي حدثت في مجموعة البيانات الأصليّة.

كما يمكن أن تأخذ معاملا ثالثا، يمثّل ردّ الفعل الذي سيتمّ اتخاذه لو كان تركيب مجموعتي البيانات مختلفا، وهو يأخذ إحدى قيم المرقّم MissingSchemaAction التالية:

|  |  |
| --- | --- |
| Add | لو كان هناك أعمدة جديدة، فستتمّ إضافتها لمجموعة البيانات الأصليّة. |
| AddWithKey | إضافة الأعمدة الجديدة والمعرّفات IDs الجديدة. |
| Ignore | يتمّ تجاهل العناصر الجديدة. |
| Error | لا يتمّ المزج وينطلق استثناء. |

وأثناء عمليّة المزج لا يتمّ التحقّق من صحّة القيود Constrains، فإذا ما تمّ المزج، يتمّ التحقّ منها.. فإذا كانت هناك سجلات تعارض القيود المفروضة، ينطلق استثناء.

ويمكن أن تفشل عمليّة المزج، وينطلق الحدث MergeFailed، وذلك إذا كان لسجلين في مجموعتي البيانات نفس المفتاح الأساسيّ، وكانت لخاصيّة EnforceConstraints القيمة True.

**التعامل مع السجلات التي بها أخطاء:**

ما زلنا لم نجد إجابةً شافيةً للسؤال التالي: "كيف نعرف السجلات التي لم يتمّ حفظها في قاعدة البيانات، نتيجةً لوجود أخطاء بها، وذلك حتّى نستطيع أن نطلب من المستخدم تعديل هذه الأخطاء؟"

أولا، يجب علينا أن نغيّر قيمة خاصيّة "استمرار التحديث رغم حدوث الأخطاء" ContinueUpdateOnError الخاصّة بموصّل البيانات DataAdapter إلي True.. إنّ هذا لن يرغم قاعدة البيانات على تقبّل الخطإ، ولكنّه سيرغم موصّل البيانات على استمرار عمليّة التحديث، بحيث يحفظ باقي السجلات التي لا تحتوي على أخطاء في قاعدة البيانات.

بعد هذا يجب أن نعرف السجلات التي حدثت بها أخطاء.. ويمكن هذا عن طريق اختبار خاصيّة Errors لكلّ صفّ في الجدول الذي يتمّ حفظه في مجموعة البيانات.

ونظرا لأنّ هذه طريقة غير مريحة (خاصة في حالة قاعدة البيانات الضخمة)، فإنّ من الأفضل استخدام الحدث RowUpdated الخاصّ بموصّل البيانات، والذي ينطلق بعد محاولة حفظ كلّ سجلّ.. وعن طريق خاصيّة e.Status يمكنك أن تعرف إذا كان حفظ السجلّ قد تمّ بنجاح، أم فشل نتيجةَ خطإ ما (ستكون قيمة الخاصيّة في هذه الحالة UpdateStatus.ErrorsOccurred).. هذا هو كود الإجراء:

**Private Sub DataAdapter1\_RowUpdated(sender As Object, \_**

**ByVal e As SqlClient.SqlRowUpdatedEventArgs) \_**

**Handles SqlDataAdapter1.RowUpdated**

**Select Case e.Row.RowState**

**Case DataRowState.Added**

**Console.Write("تمّت إضافة كتاب: " & \_**

**e.Row.Item("Book", DataRowVersion.Added))**

**Case DataRowState.Deleted**

**Console.Write("تمّ حذف كتاب: " & \_**

**e.Row.Item("Book", DataRowVersion.Original))**

**Case DataRowState.Modified**

**Console.Write("تمّ تعديل اسم الكتاب: " & \_**

**e.Row.Item("Book", DataRowVersion.Original))**

**End Select**

**If e.Status = UpdateStatus.ErrorsOccurred Then**

**Console.WriteLine("أخفق تحديث السجلّ")**

**Console.WriteLine(e.Errors.Message)**

**' إلغاء التغييرات التي حدثت للسجلّ**

**e.Row.RejectChanges()**

**Else**

**Console.WriteLine("تمّ تحديث السجلّ")**

**e.Row.AcceptChanges()**

**End If**

**End Sub**

**إنشاء مجموعات بيانات خاصّة:**

في هذا المقطع سننشئ مجموعات بيانات لا تناظر قواعد بيانات.. سننشئها باستخدام أدوات مخطط XML الموجودة في صندوق الأدوات، وسنربط بها جدول بيانات DataGrid بحيث نستطيع إدخال البيانات بها.. وسنرى كيف يمكن حفظ هذه البيانات في ملفّ، وكيف نعيد تحميلها منه عندما نريد.

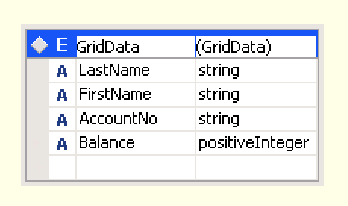
ابدأ مشروعا جديدا، أسمه XMLGrid.. في متصفّح المشاريع، اضغط اسم المشروع بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر Add New Item.. اختر العنصر XML Schema، وحدّد اسما لهذا العنصر الجديد، واضغط زرّ OK.

سيضاف مخطّط XML إلى المشروع (يحمل الامتداد .xsd).

افتح صندوق الأدوات، وانقر مرّتين بالفأرة الأداة Element لإضافة عنصر جديد للمخطّط.. هذا العنصر مماثل للجدول المعتاد في مجموعة البيانات، وسيظهر على المخطّط في صورة مستطيل فارغ، يحمل الاسم الافتراضيّ element1.. غيّر هذا الاسم على GridData (يمكنك الكتابة في خانة اسم العنصر).

من صندوق الأدوات، اسحب الأداة Attribute وأسقطها على العنصر الذي أضفناه للمخطّط.. سيعمل هذا على إضافة صفّ لعنصر المخطّط، بحيث تحمل خانة الاسم Attribute1 وخانة النوع String.. وطبعا يمكنك تغيير كلا القيمتين.. عامّةً يمكن فعل هذا مباشرة بمجرّد كتابة الأسماء التي تريدها في الصفّ الأخير في العنصر، وسيتمّ إضافة صفّ جديد تلقائيّا.. هذا الصفّ الجديد سيكون بمثابة عنصر وليس سمة.. اضغط بالفأرة حرف E على يسار الصفّ.. سيظهر لك سهم إسدال.. اضغطه، ومن القائمة المنسدلة اضغط العنصر Attribute.. سيتغيّر حرف E إلى حرف A.

صمّم المخطّط ليبدو كما في الصورة التالية:



والآن، اضغط بزرّ الفأرة الأيمن في أيّ موضع من المخطّط، ومن القائمة الموضعيّة اضغط الأمر Generate Dataset.. انتقل إلى النموذج، واستخدم صندوق الأدوات لإضافة مجموعة بيانات DataSet (تحت شريط Data).. سيظهر لك مربّع حوار إضافة مجموعة بيانات، مشيرا إلى المخطّط.. اضغط OK.

خلاص.. يمكنك استخدام مجموعة البيانات عند هذه النقطة بنفس الطريقة التي اعتدتها سابقا.

أضف للنموذج جدول بيانات DataGrid، وضع اسم مجموع البيانات xmlschema11.GridData في خاصيّة DataSource الخاصّة به.

شغّل التطبيق، واختبر أداءه.

ويمكنك إضافة أكثر من عنصر (جدول) لمخطّط XML، وإنشاء علاقات بينها (كما فعلنا من قبل في الفصل السابق).

ولكن كيف نحفظ البيانات التي أدخلناها في الجدول في ملفّ؟

لا نحتاج لأكثر من الوسيلة WriteXml الخاصّة بمجموعة البيانات، والتي تقبل كمعامل كائن بيانات الملفّ FileStream.. ابدأ باستيراد فضاء الاسم System.IO في بداية الملفّ:

**Imports System.IO**

هذا هو الكود الذي يحفظ مجموعة البيانت:

**Dim FS AS New FileStream("C:\GRID.XML", \_**

**FileMode.OpenOrCreate)**

**XmlSchema11.WriteXml(FS)**

**FS.Close()**

ولو فتحت الملفّ GRID.XML ـ بمتصفّح الإنترنت Internet Explorer مثلا ـ فستجد أنّه يحتوي على كود XML.

ولإعادة تحميل هذه البيانات من الملفّ إلى مجموعة البيانات، تُستخدم الوسيلة ReadXml، المناظرة للوسيلة السابقة:

**Dim FS As New FileStream("C:\GRID.XML", FileMode.Open)**

**XmlSchema11.Clear() ' محو أيّ بيانات سابقة بمجموعة البيانات**

**XmlSchema11.ReadXml(FS)**

**FS.Close()**

**والجدول التالي يوضّح أهمّ الوسائل التي تمنحها لك مجموعة البيانات للتعامل مع لغة XML:**

|  |  |
| --- | --- |
| تعيد نصّا عبارة عن كود XML يمثّل البيانات المخزّنة في مجموعة البيانات. | GetXml |
| تعيد نصّا عبارة عن كود XML يمثّل مخطط البيانات XSD Schema المخزّنة في مجموعة البيانات. | GetXmlSchema |
| تقرأ البيانات ومخطّطها من ملفّ XML وتحمّلها في مجموعة البيانات. | ReadXml |
| تقرأ مخطّط البيانات من ملفّ XML وتحمّله في مجموعة البيانات. | ReadXmlSchema |
| تحفظ البيانات ومخطّطها في ملفّ XML. | WriteXml |
| تحفظ مخطّط البيانات في ملفّ XML. | WriteXmlSchema |