**الفصل السابع عشر**

**بناء تطبيقات البيانات باستخدام تقنية ADO.NET**

* [**الخادم Server والعميل Client:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\1-%20الخادم%20والعميل.htm)
* [**تقنية ADO.NET:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\2-%20تقنية.htm)
* [**وماذا عن لغة XML؟**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\3-%20وماذا%20عن%20لغة.htm)
* [**موصل البيانات DataAdapter:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\32-%20موصل%20البيانات.htm)
* [**المعالج السحري لضبط موصل البيانات:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\33-%20المعالج%20السحري%20لإعداد%20موصّل%20البيانات.htm)
* [**إنشاء مجموعة البيانات Creating a DataSet:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\4-%20إنشاء%20مجموعة%20البيانات.htm)
* [**الأداة "جدول البيانات" DataGrid Control:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\5-%20جدول%20البيانات.htm)
* [**ربط البيانات Data Binding:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\6-%20ربط%20البيانات.htm)
* [**المعالج السحري لنموذج البيانات DataForm Wizard:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\7-%20المعالج%20السحري%20لنموذج%20البيانات.htm)
* [**برمجة موصّل البيانات DataAdapter:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\8-%20برمجة%20موصّل%20البيانات.htm)
* [**الاتصال بقاعدة البيانات مباشرة:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\9-%20الاتصال%20بقاعدة%20البيانات%20مباشرة.htm)

**الخادم Server والعميل Client:**

في هذا الفصل سنتعلّم أساسيّات بناء تطبيقات قواعد البيانات، سواء تلك التي تتعامل مع قاعدة بيانات على نفس الكمبيوتر، أو التي تتعامل مع قاعدة بيانات على كمبيوتر آخر على الشبكة Network.. ولا تقلق، ففي كلتا الحالتين لن تحتاج لجهاز غير جهازك لتجربة الأمثلة، فإعداد SQL Server على جهازك سيمكّنك من استخدام جهازك كخادم Server، والاتصال بقواعد بيانات SQL Server المنشئة عليه، كما رأينا في الفصلين السابقين.

إنّ الخادم Server ما هو إلا جهاز يعمل عليه تطبيق SQL Server، وكلّ دوره ينحصر في قراءة البيانات المطلوبة أو حفظ البيانات الواردة.. بينما العميل Client هو أيّ جهاز كمبيوتر آخر تمّ إعداد تطبيقك عليه.. وبهذا يمكن أن يكون هناك آلاف العملاء، كلّ منهم يحاول الاتصال بقاعدة البيانات على الخادم في نفس الوقت، والتعامل مع بياناتها، ثمّ إرسال أيّ تعديلات تمّ إجراؤها عليها إلى الخادم مرّة أخرى، ليتمّ حفظها في قاعدة البيانات.. هنا تقع على تطبيقك مسئوليّة دقّة وكفاءة وصحّة الوظائف المطلوبة من قاعدة البيانات.

إنّ هذه الطريقة في تقسيم العمل تحتوي على ميزاتٍ هامّة:

* وجود قاعدة البيانات على الخادم يوفّر لمستخدمها مساحة تخزين كبيرة (بدلا من وضعها على جهازه)، خاصّة حينما تكون قاعدة البيانات ضخمة.
* وجود قاعدة البيانات على الخادم يترك للجهة المسئولة عنها مهمّة تحديثها باستمرار، وهو أفضل من اضطرار كلّ مستخدم لشراء نسخة حديثة من قاعدة البيانات كلّ فترة.
* إجراء العمليّات على جهاز العميل أسرع بكثير من إرسال المهمّة للخادم لتنفيذها ثمّ إرسال الناتج، وذلك لأنّ الطبيعيّ أن يكون هناك عدد ضخم من المستخدمين الذين يجرون عمليّات متعدّدة في نفس اللحظة.

وإن كان هذا النظام يضع على عاتق المبرمج بعض التحدّيات، مثل التعارض الذي يمكن أن ينتج عندما يحذف أحد المستخدمين بعض السجلات، بينما مستخدم آخر يحدّث قيمها!.. أو عندما يحاول أكثر من مستخدم تحديث نفس السجلات بطرق مختلفة في نفس الوقت!

عامّة توجد هناك حلول كثيرة لمثل هذه الأمور، عليك أنت اختيار ما يناسبك منها وما يناسب أداء تطبيقك.

وفي تعاملنا مع قواعد البيانات سنستخدم خليّة Class تسمّى ADO.Net، حيث تختصر هذه الحروف عبارة "أداة كائن البيانات" ActiveX Data Object.. هذه الأداة ستمنحها إمكانيّات هائلة للتعامل مع قواعد البيانات وحلّ المشاكل التي قد تواجهنا.

**تقنية ADO.NET:**

صمّمت تقنية ADO.NET كتطوّر لتقنية ADO، ولكنّ الحقيقة أنّهما غير متشابهتين.. فتقنية ADO تفترض أنّ العميل يبقى على اتصال بقاعدة البيانات طوال مدّة تعامله معها عبر الشبكة، حيث يحصل على البيانات من أيّ جدول يريده ويحدّثها عبر نفس الاتصال.

هذه الطريقة تبدو عقيمة، ففتح الاتصال طوال الوقت مع قاعدة البيانات عبر الشبكة غير عمليّ، كما أنّ إجراء العمليّات على الشبكة أبطأ.. هذا ما جعل المبرمجين يستخدمون تقنية ADO بطريقة غريبة، فهم يتصلون بقاعدة البيانات، وينسخون البيانات المطلوبة إلى أجهزتهم، ثمّ يغلقون الاتصال ويجرون العمليّات المطلوبة على البيانات.. فإذا كانت هناك تغييرات يجب حفظها في قاعدة البيانات، فإنّهم يتصلونَ بقاعدة البيانات مرّة أخرى ويحفظون البيانات ثمّ يغلقون الاتصال.

لذا فقد أضافت ميكروسوفت لتقنية ADO.NET إمكانيّة التعامل مع البيانات بعد إغلاق الاتصال Disconnected Mode، حيث يتمّ تحميل البيانات في ذاكرة الجهاز، ليتعامل معها المستخدم على جهازه، وعندما يريد حفظ التغييرات، يتمّ فتح الاتصال مرّة أخرى لنقل البيانات إلى قاعدة البيانات.

ولكن كيف يتعامل العميل مع البيانات على جهازه؟.. إنّه يحتاج لوضعها في تركيب يشبه تركيب قاعدة البيانات الأصليّ.. هذا التركيب يسمّى "مجموعة البيانات" DataSet، وهي تمكّنك من التعامل مع البيانات بالحذف والإضافة والتعديل، كما تمكّنك من دمج مجموعات من السجلات من أكثر من قاعدة بيانات.. كلّ ذلك دونَ أن تؤثّر على قاعدة البيانات الأصليّة في شيء، إلا إذا أردتَ حفظ هذه التغييرات لقاعدة البيانات عن طريق إجراء اتصال جديد بها وتحديثها.

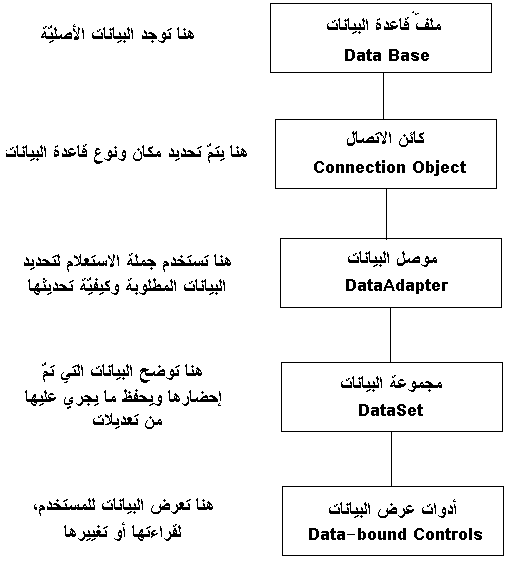
إنّ بإمكانك تحميل كلّ قاعدة البيانات في "مجموعة البيانات" DataSet، وإن كان الأكثر عمليّةً هو أن تحمّل الجزء الذي تريد التعامل معه من البيانات، وذلك عن طريق جملة SQL مناسبة (راجع الفصل السابق).

ويمكنك القول إنّ "مجموعة البيانات" DataSet هي أشبه ما تكون بمجموعة السجلات RecordSet في تقنيات DAO و RDO و ADO، حيث كانت الجداول ونتائج الاستعلامات تُحفظ فيها، وإن كانت مجموعة البيانات أرقى من مجموعة السجلات، فهي بمثابة صورة مصغّرة من قاعدة البيانات، تحتوي على الجداول، والعلاقات فيما بينها Relations، والقيودConstraints المفروضة على قيم الحقول، بما في ذلك فرض التكامل المرجعي Referential Integrity بين الجداول.

ولتحميل البيانات في مجموعة البيانات DataSet، يجب عليك أولا أن تتصل بقاعدة البيانات.. إنّ هذا يتمّ باستخدام كائن الاتصال Connection object، حيث يسمح لك بتحديد قاعدة البيانات التي ستتعامل معها.

وهناك كائن آخر يقف بين مجموعة البيانات وقاعدة البيانات.. ذلك هو "موصّل البيانات" DataAdapter Object، الذي يحتوي على المعلومات اللازمة لنقل البيانات من وإلى قاعدة البيانات.. هذه المعلومات عبارة عن استعلامات SQL اللازمة لقراءة سجلات قاعدة البيانات أو تحديثها أو حذفها أو إضافة بعض الصفوف الجديدة.

والشكل التالي يشرح هذه التقنية:



إنّ هذا التخطيط يقسّم العمل إلى طبقات مستقلّة في وظيفتها، ممّا يجعل من مجموعة البيانات أداةً متخصّصةً في عرض البيانات مهما كان نوع قاعدة البيانات التي تنتمي إليها، أو كيفيّة قراءتها وحفظها، فهذه الوظائف من اختصاص موصّل البيانات DataAdapter، الذي لا يهتمّ بدوره بموضع قاعدة البيانات أو كيفيّة الاتصال بها، فهذه مهمّة أداة الاتصال Connection.. وبهذا يمكنك أن تكتب أجزاءً عامّة من الكود الذي يتعامل مع مجموعة البيانات، دون أن تحتاج لتغييره تبعا لنوع قاعدة البيانات أو لنوع الاستعلام.. كما أنّ بإمكانك حفظ بيانات مجموعة البيانات في ملفّ XML وإعادة قراءتها منه مرّة أخرى.

**ملحوظة لمبرمجي VB6:**

انتهى بلا رجعة عصر أدوات DAO وما يرتبط بها من أدوات ويندوز.. ولكن إذا كنت ما زلت مغرما بالطريقة القديمة للتعامل مع قواعد البيانات، فيمكنك أن تستخدم الأداة ADO وما يرتبط بها من أدوات ويندوز، وذلك بإضافتها إلى صندوق الأدوات ToolBox، عن طريق ضغطه بزرّ الفأرة الأيمن، واختيار Customize ToolBox من القائمة الموضعيّة، وضغط شريط .NET Framework Components.. ستجد أوّل عنصر في القائمة أداة البيانات ADODC.. اضغط مربّع الاختيار المجاور لها.

يمكنك كذلك ضغط شريط COM Components واختيار أيّ من الأدوات التالية من القائمة:

**Microsoft ADO Data Control, Version 6.0 (OLEDB)**

**Microsoft DataCombo Control, Version 6.0 (OLEDB)**

**Microsoft DataGrid Control, Version 6.0 (OLEDB)**

**Microsoft DataList Control, Version 6.0 (OLEDB)**

**Microsoft DataRepeater Control, Version 6.0**

**Microsoft DBCombo Control, Version 6.0**

**Microsoft DBList Control, Version 6.0**

كما يمكنك أن تستخدم أدوات RDO، وذلك بإضافة الأداة:

**Microsoft RemoteData Control, Version 6.0**

وإن كنّا ننصحك دائما بالانتقال إلى الأحدث، إلا إذا كانت لديك برامج قديمة تريد تحديثها من دون إعادة بنائها بالكامل من جديد.. وفي هذا الصدد يجب أنّ ننوّه إلى أنّ أيّ برنامج VB6 يستخدم أداة البيانات Data Control، لن يمكنك تحديثه إلى VB.Net، إلا إذا استبدلت بهذه الأداة في VB6 الأداة ADODC، ثمّ قمت بتحديث المشروع، حيث إنّ الأداة الأولى تتعامل بنظام DAO المنقرض، بينما الأخيرة تتعامل بنظام ADO الذي ما زال متاحا.

**وماذا عن لغة XML؟**

تسير تقنية ADO.NET و لغة XML متعانقتَيِ اليدين.. إنّ XML هي طريقة لتمثيل أيّ بيانات منظّمة، وذلك بتحويلها لنصّ يعبّر عنها.. وبهذا تصلح لغة XML للتعبير عن أيّ نوع من أنواع البيانات، مثل الجداول والصور وغيرهما.

ورغم أنّ الملفّات النصّيّة أكبر حجما من الملفّات الثنائيّة Binary Files، إلا إنّ الأولى صالحة للتعامل مع أيّ تطبيق بل مع أيّ نظام تشغيل.. لهذا فقد صارت لغة XML في السنوات الأخيرة هي أنسب وسيلة لنقل البيانات عبر الإنترنت، وذلك حتّى تتجاوز مشاكل عدم التوافق بين التطبيقات وأنظمة التشغيل المختلفة.. شيء طبيعيٌّ إذن أن تستخدم تقنية ADO.NET لغة XML في نقل البيانات بين الخادم والعميل.

ولحسن الحظّ، أنت لا تحتاج لكتابة أو قراءة أيّ كود XML بنفسك ـ وإن كان ذلك متاحا لك لو احتجته ـ وذلك لأنّ تقنية ADO.NET تستخدم هذه اللغة كطبقة داخليّة، فهي تنتجها وتقرأ البيانات منها بطريقة آليّة.

ومن الإمكانيّات التي تتيحها لك لغة XML، قدرتك على استخدامها لإنشاء مجموعة بيانات DataSet بدون الاعتماد على أيّ قاعدة بيانات.. في هذه الحالة ستوضع البيانات في الذاكرة، ولو شئت الاحتفاظ بها، فيمكنك حفظها في ملفّ، ثمّ إعادة تحميلها مرّة أخرى حينما تريد.

**موصل البيانات DataAdapter:**

ابدأ مشروعا جديدا بالطريقة المعتادة، وافتح متصفّح خوادم الانترنت Server Explorer.. اختر قاعدة بيانات الكتب التي أنشأناها بـ SQL Server.. اختر واحدا من جداولها (وليكن جدول المؤلّفين).. اسحبه بالفأرة وأسقطه على النموذج.. سيقوم VB بوضع أداتين جديدتين على صينية المكوّنات Components Tray هما SqlConnection1 و SqlDataAdapter1.

ولو نظرت في خصائص الأداة SqlConnection1 لوجدت فيها خاصيّة "نصّ الاتصال" ConnectionString، وبها نصّ يحتوى على المعلومات التالية:

**data source = MHMD; ' أو اسم جهازك**

**initial catalog = Books;**

**integrated security = SSPI;**

**workstation id = MHMD; ' أو اسم جهازك**

**packet size=4096**

وهي كلّ المعلومات المطلوبة للاتصال بقاعدة بياناتك.

وإذا نظرت في خصائص الأداة SqlDataAdapter1، فستجد أنّ من ضمنها الخصائص المهمّة التالية:

**أمر التحديد SelectCommand:**

هذا هو الأمر المسئول عن استرجاع السجلات من الجدول لتحميلها في مجموعة البيانات.. ستجد أنّ قيمة هذه الخاصيّة هي SqlSelectCommand1.. هذا هو اسم الأمر ـ الذي يمكن أن تستخدمه في الكود ـ وليس صيغته.. اضغط علامة "+" المجاورة لاسم الخاصيّة.. ستنسدل مجموعة من الخصائص الفرعيّة، ستجد من بينها الخاصيّة "نصّ الأمر" CommandText.. في هذه الخاصيّة ستجد جملة SQL التالية:

**SELECT ID, Author FROM Authors**

واضح تماما أنّها تحدّد جدول المؤلّفين بكامل حقوله وسجلاته.. ولا مانع أن تعدّل هذه الجملة كما تشاء.. وحتّى يكون هذا التعديل أسهل، اضغط زرّ الانتقال المجاور لجملة SQL.. آه.. ها هي ذي نافذة باني الاستعلام Query Builder تظهر لنا.. مرحى: كلّ شيء سيتمّ بطريقة مرئيّة سهلة ومريحة!

وهناك طريقة أخرى لتحرير أمر التحديد.. اضغط الأداة SqlDataAdapter1 في صينيّة النظام بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضعط الأمر Configure Data Adapter.. ستجد هذا الأمر كذلك موجودا أسفل نافذة الخصائص.. وعند ضغطه سيتمّ عرض معالج سحريّ Wizard ليساعدك في تصميم جملة SQL.. وسنتعرّف على خطوات هذا المعالج فيما بعد.

**أمر الإدراج InsertCommand:**

هذا هو الأمر المسئول عن إدراج السجلات في جداول قاعدة البيانات.. ولو فتحت الخاصية الفرعيّة "نصّ الأمر" CommandText لوجدت فيها ما يلي:

**INSERT INTO Authors(Author) VALUES (@Author);**

**SELECT ID, Author FROM Authors**

**WHERE (ID = @@IDENTITY)**

تعرف طبعا أنّ أيّ نصّ يبدأ بالرمز @ هو عبارة عن متغيّر، وذلك حتّى يكون هذا الأمر عامّا، تستخدمه أكثر من مرّة، بقيمة مختلفة (مؤّلف مختلف) للمتغيّر في كلّ مرّة.

ولكن ماذا عن جملة SELECT التي تلي جملة الإضافة؟

إنّ دورها ينحصر في تحديد السجلات التي تمّت إضافتها لقاعد البيانات من قبل مستخدمين آخرين في فتر قطع الاتصال، ليتمّ إضافتها للسجلات الموجودة في مجموعة البيانات.

**أمر الحذف DeleteCommand:**

هذا هو الأمر المسئول عن حذف بعض السجلات من جداول قاعدة البيانات.

**أمر التحديث UpdateCommand:**

هذا هو الأمر المسئول عن تحديث السجلات في جداول قاعدة البيانات، وذلك بنقل السجلات من مجموعة البيانات إلى قاعدة البيانات.

ولكن ماذا بخصوص قاعدة بيانات الكتب التي أنشأناها بتطبيق Access.

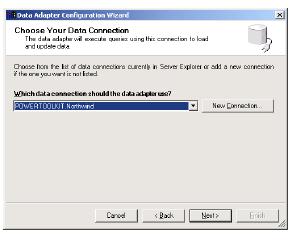
عادي.. استخدم نفس الخطوات السابقة معها.. كلّ ما سيختلف هو نوع أداتي الاتصال والتوصيل، فسيكونان من النوعين: OleDbConnection و OleDbDataAdapter، وما ذلك إلا لأنّ جمل SQL التي تتعامل مع Access بها اختلافات طفيفة عن تلك التي تتعامل مع SQL Server.. لكنْ بخلاف هذا ستجد لهذين النوعين من الأدوات نفس الخصائص والوسائل.

**المعالج السحري لإعداد موصّل البيانات Data Adapter Configuration Wizard:**

استخدم هذا المعالج لتسهيل ضبط وظيفة وخصائص موصل البيانات.

اضغط موصّل البيانات بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر Configure Data Adapter.. عندئذٍ ستظهر لك نافذة الترحيب الخاصّة بالمعالج السحري.. اضغط زرّ Next.

في النافذة التالية سيُطلبُ منك إدخال اسم كائن الاتصال:



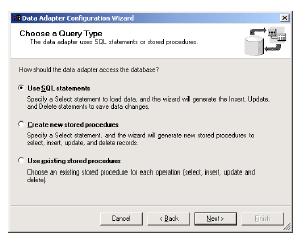
ستجد أنّ الاختيار الافتراضي هو اسم قاعدة بيانات Books المنشأة بـ SQL Server

وفي القائمة المركّبة ستجد اتصالا آخر بقاعدة البيانات المنشأة بـ Access.

ولو أردت إنشاء اتصال جديد تماما بقاعدة بيانات أخرى، فاضغط زرّ New Connection لتظهر لك نافذة خصائص ارتباط البيانات Data Link Properties.

عامّة اترك الاختيار الافتراضيّ كما هو، واضغط Next.

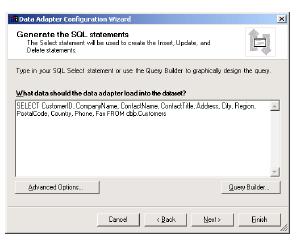
في النافذة التالية يمكنك اختيار نوع الاستعلام الذي ستستخدمه لإحضار البيانات من قاعدة البيانات:



ويمكنك الاختيار بينَ استخدام جمل SQL أو استخدام الإجراءات المخزّنة Stored Procedures التي أنشأتها من قبل، أو إنشاء إجراءات مخزّنة جديدة.. وفي حالة استخدام Access الذي لا يتعامل مع الإجراءات المخزّنة، ستجد أنّ الخيارين الأخيرين غير فعّالين Disabled..

اضغط Next.

النافذة التالية خاصّة بإنشاء جملة SQL:

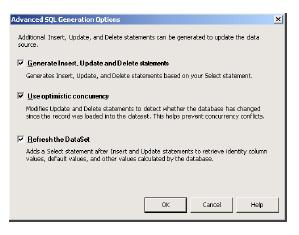


وفي هذا الصدد، لديك مربّع نصّ به جملة SELECT مبدئيّة، تقوم باسترجاع الجدول كلّه.. عدّل نصّ الجملة يدويّا لو أردت، أو اضغط زرّ Query Builder لاستخدام باني الاستعلام في إنشاء جملة SQL.

عامّة لن نحتاج لإجراء تعديلات في عرض جدول المؤلّفين في مثالنا هذا.. إذن لماذا أشرح لك هذه الخطوات؟.. اعتبره مقلبا!!

ولكن ماذا عن زرّ "خيارات المتقدّمة" Advanced Options؟

لو ضغطت هذا الزرّ فستظهر لك نافذة بها الخيارات التالية:



1. Generate Insert, Update And Delete Statements:

استخدم هذا الخيار إذا كنت تنتوي إجراء أيّ تعديل في قاعدة البيانات.. في هذه الحالة سيتمّ إنتاج جمل التحديث والإدراج والحذف آليا، بحيث تكون مناظرة لجملة Select التي أنشأتها.

1. استخدم المطابقة المتفائلة Use Optimistic Concurrency:

لو اخترت هذا الخيار، فأنت تفترض ـ متفائلا ـ أنّ قاعدة البيانات لن يتمّ تعديلها من قِبَلِ مستخدمين آخرين في الوقت الذي تتعامل فيه مع السجلات على جهازك.. حذار، فلو حدث العكس وكانت السجلات قد تغيّرت من قبل مستخدمين آخرين، فلن يتمّ تحديث التعديلات التي أجريتها أنت على هذه السجلات!

وطبعا لو كنت "متشائما" بخصوص هذا الأمر، فلا تختر هذا الخيار.. في هذه الحالة سيتمّ حفظ تعديلاتك بدلا من تعديلات المستخدمين الآخرين، وليشربوا من البحر!!

1. إنعاش مجموعة البيانات Refresh The DataSet:

باختيار هذا الخيار، سيتمّ وضع جملة SELECT بعد جملتي الإدراج والتحديث، وذلك حتّى يتمّ إنعاش مجموعة البيانات على جهازك لتتطابق مع قاعدة البيانات بعد تعديلها.. هذا الخيار غير متاح في قواعد بيانات Access.

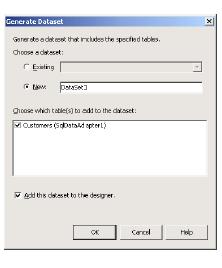
اختر ما يناسبك من هذه ثم اضغط OK.. اضغط Next للانتقال للنافذة التالية.

ستظهر لك آخر نافذة في المعالج السحريّ، وهي تعرض ملخّصا لخياراتك.. اضغط زر Finish لإنهاء العمليّة.

**إنشاء مجموعة البيانات Creating a DataSet:**

اضغط أداة توصيل البيانات DataAdapter في صينيّة المكوّنات بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر "إنتاج مجموعة البيانات" Generate Dataset.. ويمكنك أن تجد نفس الأمر أسفل نافذة الخصائص وفي القائمة الرئيسيّة Data.

سيظهر لك مرّبع حوار "إنتاج مجموعة البيانات":



في هذه النافذة ستجد الاختيارات معدّة لإنشاء مجموعة بيانات جديدة تسمّى DataSet1 تحتوي على جدول المؤلّفين.. لا تغيّر هذه الخيارات واضغط OK.

ستتمّ إضافة الأداة DataSet11 لصينيّة المكوّنات.. ليس هذا فحسب، بل ستتمّ إضافة الملفّ DataSet1.xsd إلى ملفّات المشروع في متصفّح المشاريع، وهو عبارة عن خليّة Class تصف تركيب البيانات التي ستستقبلها من قاعدة البيانات.. وتعتبر مجموعة البيانات نسخة Instance من هذه الخليّة.

دعنا نرى ماذا فعلنا حتّى هذه اللحظة.. افتح القائمة الرئيسيّة Data Menu، واضغط الأمر Preview Data.. ويمكنك أن تجد هذا الأمر أيضا في القائمة الموضعيّة لأداة توصيل البيانات DataAdapter في صينيّة المكوّنات، أو في أسفل نافذة الخصائص.. هذا الأمر سيفتح نافذة استعراض البيانات، التي ستكون فارغة مبدئيّا.. اضغط زر ملء مجموعة البيانات Fill Dataset.. سيتمّ ملء جدول المؤلّفين بالبيانات.

وإذا كانت مجموعة البيانات تحتوي على أكثر من جدول، فيمكن اختيار اسم الجدول الذي تريد عرضه من القائمة الموجودة على يسار النافذة.

**جدول البيانات DataGrid Control:**

لا ريب أنّك تتساءل عن كيفيّة عرض البيانات للمستخدم، فبعد كلّ هذا الشرح لم ننشئ تطبيقا واحدا ولم نحصل على نتائج!

إذن فخذ خُطوةً نحو الذي تتمنّاه!

إنّ أنسب أداة لعرض البيانات للمستخدم هي الجدول DataGrid، وهو يمنحك قدرات هائلة تشبه قدرات جداول Access، حتّى إنّها تمكّنك من عرض البيانات المترابطة، مثل أن تعرض جدولا فرعيّا لكلّ مؤلّف يحتوى على أسماء الكتب التي ألّفها، ليس تماما كما رأينا في Access، ولكنّ بطريقة مقاربة.

* [**كيفيّة استخدام جدول البيانات:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\5-%20جدول%20البيانات\2-%20كيفيّة%20استخدام%20جدول%20البيانات.htm)
* [**تنسيق جدول البيانات:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\5-%20جدول%20البيانات\3-%20تنسيق%20جدول%20البيانات.htm)
* [**عرض الجداول المترابطة:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\5-%20جدول%20البيانات\4-%20عرض%20الجداول%20المترابطة.htm)

**كيفيّة استخدام جدول البيانات:**

ضع نسخة من الأداة DataGrid على النموذج.. في نافذة الخصائص حدّد خاصيّة "مصدر البيانات" DataSource.. في خانة القيمة اضغط زرّ إسدال القائمة.. ستجد القيمتين التاليتين: DataSet11، DataSet11.Authors.

الآن أنت بالخيار: فإمّا أن تجعل مصدر البيانات هو مجموعة البيانات DataSet11 نفسها، وفي هذه الحالة ستحتاج لاستخدام خاصيّة "عنصر البيانات" DataMember، وجعل قيمتها Authors حتّى يتمّ عرض جدول المؤلّفين.. وإمّا أن تجعل مصدر البيانات هو DataSet11.Authors مباشرةً.

وفي كلتا الحالتين ستظهر أسماء الحقول في الجدول على النموذج في وقت التصميم، ولكن بدون عرض البيانات.

شغّل البرنامج الآن، وانظر ماذا ترى.. أيضا ما زال الجدول فارغا!

إنّ هذا يعود لأنّ مجموعة البيانات نفسها ما زالت فارغة، فهي لا تمتلئ إلا باستدعاء الوسيلة Fill الخاصة بموصّل البيانات DataAdapter، وهي تأخذ معاملين: اسم مجموعة البيانات التي سيتم ملؤها، واسم الجدول الذي ستجلب منه البيانات.. ويمكن عدم إرسال المعامل الأخير إذا كانت مجموعة البيانات تتعامل مع جدول واحد فحسب.

أضف زرّا للنموذج، واكتب فيه ما يلي:

**DataSet11.Clear() 'إزالة البيانات الموجودة حاليا**

**' إضافة البيانات الجديدة**

**SqlDataAdapter1.Fill(DataSet11, "Authors")**

الآن لو شغّلت البرنامج وضغطت الزرّ، فسيتمّ ملء الجدول بأسماء المؤلّفين.



إنّ بإمكانك أن تعدّل بيانات الجدول، ولكنّ هذه التعديلات ستكون خاصّة بمجموعة البيانات ولن تؤثّر على قاعدة البيانات الأصليّة.. فإذا أردت أن ترسل التعديلات لقاعدة البيانات، فاستخدم الوسيلة Update الخاصّة بموصّل البيانات، ولها نفس معاملي الوسيلة Fill.

ضع زرّا آخر على النموذج، وأضف إليه الجملة التالية:

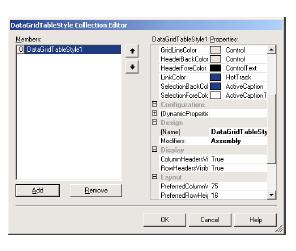
**SqlDataAdapter1.Update(DataSet11, "Authors")**

ويتمّ تحديث سجلات قاعدة البيانات واحدا فواحدا، فإذا حدث خطأ عند محاولة تحديث واحد منها، فإنّ عمليّة التحديث تتوقّف ولا تستمرّ.. فإذا أردت تغيير هذا التصرّف، فضع القيمة True في خاصيّة "استمرار التحديث عند حدوث خطإ" ContinueUpdateOnError، في أداة توصيل البيانات DataAdapter.

**تنسيق جدول البيانات DataGrid:**

يمكنك أن تحصل على تنسيقات جاهزة للجدول، وذلك بضغط الجدول بزرّ الفأرة الأيمن، واختيار الأمر AutoFormat من القائمة الموضعيّة، حيث ستظهر لك نافذة تحتوي على قائمة بأسماء التنسيقات المتاحة، مع عرض لتأثير كلّ منها على جدول المعاينة.

كما يمكنك أن تنشئ تنسيقات جديدة، وذلك باستخدام خاصيّة "أنماط الجداول" TableStyles.. حدّد هذه الخاصيّة في نافذة الخصائص، واضغط زرّ الانتقال المجاور لها.. ستظهر لك نافذة تحرير مجموعة أنماط الجداول.



إنّ بإمكان جدول البيانات DataGrid أن يعرض جدولا Table واحدا من قاعدة البيانات أو أكثر، لهذا فإنّ بإمكانك اختيار تنسيق مختلف لكلّ جدول.. ولكن في حالتنا هذه، لا تحتوي مجموعة البيانات إلا على جدول واحد، لهذا فلن نضيف إلا كائن تنسيق جدول DataGridTableStyle object واحدًا فقط المجموعة Collection.

اضغط زر الإضافة Add.. سيُضاف الكائن DataGridTableStyle1 للمجموعة.. وفي جدول الخصائص الخاصّة به، ستجد أنّ بإمكانك التحكّم في كثير من سمات الجدول، مثل خطّ العنوان HeaderFont، وألوان خطوط الجدول وخلفيّته، ولون الأعمدة الفرديّة، والزوجيّة... إلخ.

ولكي ينطبق هذا التنسيق على جدول Table بعينه، اكتب اسم هذا الجدول (اسمه في مجموعة البيانات.. مثل Authors) في خاصيّة "اسم التخطيط" MappingName.

وليس هذا كلّ شيء، فلو نظرت للخاصيّة التي تعلو خاصيّة MappingName مباشرة، لوجد أنّها خاصيّة "أنماط أعمدة الجدول" GridColumnStyles، وهي مجموعة أخرى Collection، تمنحك قدرات كبيرة في تنسيق أعمدة الجدول.. اضغط زرّ الانتقال المجاور لهذه الخاصيّة.. ستظهر لك نافذة جديدة مشابهة للنافذة الأولى.. يمكنك أن تضيف كائنا لكلّ عمود، وبهذا تستطيع تغيير خصائص كلّ عمود على حدة، مثل عرض العمود وعنوانه ومحاذاته.. وطبعا ستجد لكلّ كائن تضيفه خاصيّة MappingName، وذلك لكي تحدد عن طريقها اسم العمود الذي ستنطبق عليه هذه التنسيقات.

**عرض الجداول المترابطة:**

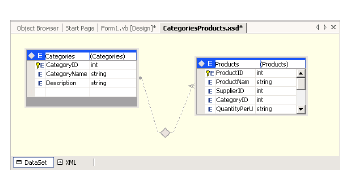
ابدأ مشروعا جديدا، ومن متصفّح خوادم الإنترنت اسحب جدولي المؤلّفين والكتب (دعنا نتعامل مع قاعدة بيانات Access هذه المرّة) وألقِهما على النموذج (يمكن أن تسحب العنصر Tables ليتمّ وضع كلّ الجداول التي تندرج تحته مباشرة).. سيتمّ وضع أداة الاتصال واثنين من أدوات التوصيل على النموذج، ولكنهما في هذه المرّة سيكونان من النوع OleDB بدلا من النوع SQL.. لن تجد فارقا في التعامل بين هذين النوعين، فلقد قلنا إن الاختلافات تنحصر في بعض صيغ لغة SQL، وهي اختلافات ضئيلة.. فمثلا، لو فتحت خاصيّة "أمر الإدراج" InsertCommand لوجدت جملة SQL في Access تستخدم علامة الاستفهام ? بدلا من أسماء المتغيّرات التي تبدأ بالرمز @ في SQL Server، مع عدم وجود جملة SELECT تالية لها.

**INSERT INTO Authors(Author) VALUES (?)**

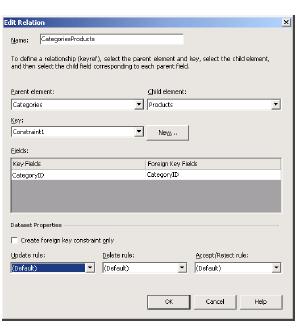
أعد تسمية موصلي البيانات بـ DAAuthors و DABooks، ليكون اسماهما أكثر منطقيّة ودلالة عند استخدامهما في الكود.

قم بإنشاء مجموعة البيانات بالطريقة التقليديّة، ولكن في نافذة إنشائها ضع علامة الاختيار أمام اسمي الجدولين هذه المرّة.. اضغط OK.. ومن نافذة الخصائص غيّر اسم مجموعة البيانات إلى AuthorsBooks1.

اضغط الملفّ ذا الامتداد XSD في متصفّح المشاريع.. ستظهر لك نافذة تعرض تركيب الجدولين ولكن بدون أيّ علاقات بينهما.. تعال نحدّد نحن هذه العلاقات.



افتح صندوق الأدوات.. ستجده مختلفا هذه المرّة، حيث سيعرض الأدوات الملائمة لتحرير مخطّطات XML.. من بين هذه الأدوات ستجد أيقونة العلاقات Relation.. اسحبها وأسقطها على جدول المؤلّفين.. ستظهر لك نافذة تحرير العلاقة كما هو موضّح بالشكل.



لن تجد صعوبة في تحديد مواصفات العلاقة، فهذه النافذة شديدة الشبه بنافذة إنشاء العلاقة التي تعرّفنا عليها في الفصلين السابقين.

اجعل العنصر الأساسي Parent Element هو جدول المؤلّفين وفي خانة الحقل المفتاحي Key Field اختر الحقل ID (لن تستطيع اختيار غيره، لأنّه هو المفتاح الأساسيّ)، والعنصر الفرعيّ Child Element هو جدول الكتب، وفي خانة الحقل المفتاحيّ اختر الحقل AuthorID.. اضغط OK.

سيتمّ رسم خطّ العلاقة في مخطّط XML بنفس الطريقة المألوفة.

وإذا أردت تعديل شيء في خصائص العلاقة، يمكنك أن تضغط بزرّ الفأرة الأيمن رمز الجوهرة في منتصف خطّ العلاقة، ومن القائمة الموضعيّة اضغط الأمر "تحرير العلاقة" Edit Relation، لتظهر لك مرّة أخرى نافذة تحرير العلاقة.

ضع جدول بيانات DataGrid على النموذج، واربطه بمجموعة البيانات، عن طريق وضع القيمة AuthorsBooks1 في خاصيّة DataSource.

الآن لم يبقَ أمامنا إلا تحميل هذين الجدولين المترابطين في مجموعتي البيانات.. هذا ما سنفعله في حدث تحميل النموذج:

**Private Sub Form1\_Load(ByVal sender As System.Object, \_**

**ByVal e As System.EventArgs) Handles MyBase.Load**

**DAAuthors.Fill(AuthorsBooks1, "Authors")**

**DABooks.Fill(AuthorsBooks1, "Books")**

**End Sub**

لو شغّلت البرنامج الآن فلن تظهر في الجدول أيّ بيانات.. فقط علامة "+"، لو ضغطتها فستعرض لك اسمي جدولي المؤلّفين والكتب.



اضغط جدول المؤلّفين.. سيمتلئ الجدول بسجلات جدول المؤلّفين.. وأمام كلّ سجلّ سترى علامة "+".. لو ضغطت هذه العلامة، فسيظهر لك اسم العلاقة بين الجدولين، كما في الصورة التالية:



اضغط اسم العلاقة بالفأرة.. سيتمّ عرض أسماء الكتب الخاصّة بهذا المؤلّف، الذي سيظهر اسمه أعلى الجدول.



ولو أردت العودة لجدول المؤلّفين، فاضغط السهم المرسوم في الركن العلويّ الأيمن من جدول البيانات.. ولو أردت إخفاء اسم المؤلّف المكتوب أعلى الجدول فاضغط الرمز المجاور للسهم.. ولو أردت إعادة إظهاره، فاضغط نفس الرمز مرّة أخرى.

لاحظ أنّ بإمكانك ترتيب سجلات الجدول تصاعديّا، تبعا لقيم أيّ عمود فيه، وذلك بمجرّد ضغط رأس هذا العمود Header.. ولو أعدت ضغط نفس العمود، فسيتمّ عكس الترتيب.. ولو أردت منع هذا الأمر، فاجعل خاصيّة "السماح بالترتيب" AllowSorting لجدول البيانات بالقيمة False.

**ربط البيانات Data Binding:**

ربما تُرضيك طريقة عرض السجلات في جدول البيانات، ولكنّها لن تُغنيك عن رغبتك أحيانا في عرض البيانات في أدوات أخرى كاللافتات ومربّعات النصّ والقوائم، وذلك حتّى تصمّم واجهات استخدام أكثر حرّيّةً ومناسبةً لأغراضك.

فلتكن مطمئنّا، فلن تحتاج لكتابة كود لذلك.. كلّ ما عليك هو أن تربط الأداة التي تناسبك بحقل معيّن من جدول ما، لتعرض هذه الأداة قيم هذا الحقل.. وكلّما تحرّكت في سجلات مجموعة البيانات، يتمّ تغيير القيمة المعروضة في الأداة آليّا.. ولو غيّرت القيمة الموجودة في الأداة، فسيتمّ حفظ هذا التغيير لمجموعة الأدوات آليّا بمجرّد مغادرتك للسجلّ الحالي.. وطبعا لن يتمّ حفظ التغيير لقاعدة البيانات نفسها، إلا إذا قمت أنت بتحديثها.

* [**مشروع ViewEditBooks:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\6-%20ربط%20البيانات\1-%20مشروع%20عرض%20وتحرير%20الكتب.htm)
* [**كائن ربط المحتوى BindingContext Object:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\6-%20ربط%20البيانات\2-%20كائن%20ربط%20المحتوى.htm)
* [**ربط الأدوات الأكثر تعقيدا:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\6-%20ربط%20البيانات\3-%20ربط%20الأدوات%20الأكثر%20تعقيدا.htm)

**مشروع ViewEditBooks:**

ابدأ مشروعا جديدا وأسمه ViewEditBooks، وصمّم واجهة استخدامه كما في الصورة التالية:



اسحب جدول المؤلّفين إلى النموذج، وحدّد أداة التوصيل التي تمّ إنشاؤها، ومن نافذة الخصائص، غيّر اسمها إلى DABooks.. حدّد خاصيّة SelectCommand وأسدل عناصرها الفرعيّة.. عدل جملة SQL الموجودة في خاصيّة CommandText لتصبح كالتالي:

**SELECT Books.ID, Books.Book, Authors.Author**

**FROM Authors**

**INNER JOIN Books ON Authors.ID = Books.AuthorID**

استخدم باني الاستعلام لو أردت (وذلك بضغط زر الانتقال في خانة القيمة لهذه الخاصيّة)، أو اكتب هذه الجملة يدويّا (في سطر واحد طبعا، لأنّ المكان المخصّص لكتابة قيمة الخاصيّة لن يسمح بأكثر من سطر).

لاحظ أنّنا في مثالنا هذا لن نعرض جدولا موجودا في قاعدة البيانات، بل سنعرض جدولا مستنتجا من جدولي المؤلّفين والكتب.

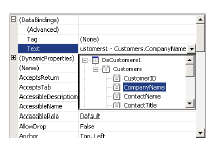
أحببت أن ألفت نظرك لهذا، حتّى لا تعتقد أنّك مضطرّ لإنشاء مجموعة البيانات لتعرض الجداول "الحقيقيّة" فحسب.. لا.. بل يمكنها كذلك أن تعرض أيّ جدول تعيده أيّ جملة SQL مهما كان تعقيدها.

قم الآن بإنشاء مجموعة البيانات بالطريقة المعهودة، وأسمها Books.

نريد ربط مربّعات النصوص بهذه المجموعة.. ستجد الأمر في غاية البساطة.

حدد أحد مربّعات النصّ، وليكن ذلك الذي سنعرض فيه رقم الكتاب.. ومن نافذة الخصائص حدّد المقطع المسمّى DataBinding (ستجده في بداية الخصائص، خروجا عن الترتيب الأبجديّ للخصائص).. تحت هذا المقطع، ستظهر لك أسماء الخصائص التي يمكن ربطها بمجموعة البيانات لهذه الأداة.. وبالنسبة لمربّع النصّ، ستجد أنّ بإمكانك ربط خاصيتي Tag و Text.. في الغالب يتمّ ربط خاصيّة Tag برقم السجلّ ID، وذلك لتسهيل التعامل مع الحقل عند الحاجة.. ولكن في مثالنا هذا، سنعرض هذا الرقم في خاصيّة Text لتظهر للمستخدم.

حدّد خاصيّة Text، واضغط زرّ الإسدال الموجود في خانة القيمة.. ستظهر لك قائمة بكلّ أسماء مجموعات البيانات الموجودة على النموذج.



وفي مثالنا هذا لن تظهر إلا مجموعة البيانات Books1.. اضغط علامة "+" المجاورة لها.. ستنسدل أسماء الجداول التي تحتويها.. وفي حالتنا هذه سيظهر لك اسم جدول المؤلّفين فقط.. اضغط علامة "+" المجاورة لها.. ستنسدل أسماء الحقول التي يحتويها.. اضغط الحقل ID.. لقد انتهى الأمر!

الآن تمّ ربط مربّع النصّ بالحقل ID، بحيث لو ملأنا مجموعة البيانات بالسجلات، فسيتمّ عرض قيمة هذا الحقل ـ للسجلّ الحاليّ ـ في مربّع النصّ هذا.. وكلّما تحرّكنا من سجلّ لآخر، يتمّ عرض قيمة الحقل ID للسجلّ الجديد آليّا.

لاحظ أنّنا جعلنا لخاصيّة ReadOnly لمربّع النصّ القيمة True، وذلك لمنع المستخدم من محاولة تغيير هذا الحقل، فقيمته غير قابلة للتغيير في قاعدة البيانات.

اتبع نفس هذه الخطوات، لربط مربعي النصّ الآخرين بحقلي Book و Author.. وسأترك لك كذلك كتابة كود الزرّ الذي يملأ مجموعة البيانات بالسجلات، وكود الزرّ الذي يحفظ هذه البيانات في قاعدة البيانات بعد تعديلها، فليس فيهما جديد عمّا رأيناه من قبل.

عند هذه النقطة، لو شغّلت المشروع وضغطت زرّ تحميل البيانات، فستظهر قيم السجلّ الأوّل في مربّعات النصّ، كلّ حقل في مربّعه الذي حدّدناه.

نريد الآن أن نكتب كود الأزرار التي تسمح للمستخدم بالتنقّل بين سجلات مجموعة البيانات.

**كائن ربط المحتوى BindingContext Object:**

هذا الكائن هو أحد خصائص النموذج Form، وهو يمكّنك من تغيير السجلّ الحاليّ في أيّ مجموعة بيانات موجودة على النموذج.

ويمتلك رابط المحتوى عدّة خصائص، أهمّها الخاصيّة الافتراضيّة Item التي تعيد كائنا نستطيع من خلاله استخدام خاصيّة الموضع Position، التي تشير إلى السجلّ الحاليّ في مجموعة البيانات، وعن طريقها يمكن الانتقال بين السجلات.

إنّ أيّا من التعبيرين التاليين المتماثلين، يمثّل السجلّ الحالي:

**Me.BindingContext.Item(Books1, "Authors").Position**

**Me.BindingContext(Books1, "Authors").Position**

تلاحظ أنّنا نمدّ خاصيّة Item الخاصّة برابط المحتوى باسم مجموعة البيانات، واسم الجدول الذي نريد التعامل معه داخلها.

الآن يمكنك أن تضع في خاصيّة الموضع رقم السجلّ الذي تريد الانتقال إليه.. وطبعا تعرف أنّ أول سجلّ يحمل رقم 0، ولكن يا ترى ما هو رقم آخر سجل؟

يمكنك استخدام خاصيّة Count، حيث سيكون رقم السجلّ الأخير هو:

**Me.BindingContext(Books1, "Authors").Count -1**

أعرف أنّك ستتساءل في استياء:

- هل سأُضطرّ في كلّ سطر لكتابة كلّ هذه الجمل الطويلة؟

- سأستاء أنا منك لو لم تجد أنت الحلّ؟

- إم.. دعني أفكّر قليلا..... آه.. نستخدم مقطع With…. End With.. يا لي من عبقريّ!

- كنتُ سأصابُ بالإحباط لو لم تكن بمثل هذه العبقريّة الفذّة!.. انظر مثلا للكود الذي ينقلنا للسجلّ التالي:

**With Me.BindingContext(Books1, "Authors")**

**If .Position < .Count - 1 Then .Position += 1**

**End With**

- آه.. هكذا فعلا صار الكود أكثر اختصارا.

- ما زال هناك حلّ آخر.. هل تستطيع أن تخمّنه؟

- حلٌّ آخر!.. حلٌّ آخر؟.. لا أعتقدُ أنّ مخيّلتي تُسعفني.

- اعتصر مخّك قليلا، ولكن حذار أن تبلل الكمبيوتر!!

- ربما... ربما لو كان هناك نوعٌ من المتغيرات يستطيع استقبال القيمة المعادة من خاصيّة Item، كنّا استطعنا استخدامه.

- أبشر إذن.. هناك مثل هذا الكائن.

- إليّ به.

- هذا الكائن يسمّى "أساس إدارة الربط" BindingManagerBase.

- يع.. ما هذه الكلكعة؟

- إن هي إلا أسماء.. ما عليك من الأسماء!

- ما عليّ من الأسماء؟!.. أتهرّج؟.. إذن كيف أحفظها لأكتبها في الكود؟

- افعل نفس ما أفعله أنا!

- وما هذا الذي تفعله أنت إن شاء الله؟

- أحلّق بالفأرة فوق كلمة BindingContext في محرّر الكود.. ما هي إلا ثوانٍ ويظهر تلميح يعرض لي صيغة تعريف خاصيّة Item.. هذه الصيغة تنتهي بكلمة As يليها نوع الكائن.. وفي حالتنا هذه ستجد أنّه من النوع:

**System.Windows.Forms.BindingManagerBase**

هذه هي الصيغة الكاملة، ولكن نظرًا لأنّ الفضاء System.Windows.Forms موجود في مراجع البرنامج References، لهذا يمكنك اختصار الاسم إلى BindingManagerBase فقط.. هذا هو نوع المتغيّر الذي نحتاجه.. الأمر بسيط.. أليس كذلك؟

- يا ابن الإيه!

- تعال نرى إذن كيف نكتب الكود السابق باستخدام هذا الـ BindingManagerBase:

**Dim T As BindingManagerBase**

**T = Me.BindingContext(Books1, "Authors")**

**If T.Position < T.Count - 1 Then T.Position += 1**

ما رأيك في هذه الوسيلة؟

- شغّال.. يمشي!

- إذن فسأترك لك كتابة الكود الذي يعرض السجلّ السابق، وذلك الذي يعرض السجلّ الأوّل، وذلك الذي يعرض السجلّ الأخير.

- آخ!

- شيءٌ وحيد فقط أريد إيضاحه لك قبل أن أتركك تحاول كتابة الكود بنفسك.. ذلك هو إضافة بعض الكود، لمعالجة أيّ خطإٍ قد يحدث نتيجة لوضع المستخدم أيّ قيمة غير مسموح بها في خانة اسم المؤلّف أو خانة اسم الكتاب.. هذا هو الكود المعدّل:

**Dim T As BindingManagerBase**

**T = Me.BindingContext(Books1, "Authors")**

**If T.Position < T.Count - 1 Then**

**Try**

**T.Position += 1**

**Catch dataException As Exception**

**MsgBox(dataException.Message)**

**T.CancelCurrentEdit()**

**End Try**

**End If**

حيث استخدمنا الوسيلة "إلغاء عمليّة التحرير الحاليّة" CancelCurrentEdit لإلغاء القيمة التي تسبّبت في حدوث الخطإ، وإعادة قيم حقول السجلّ إلى ما كانت عليه.

**ملحوظة لمبرمجي VB6:**

لعلّك تتساءل: أين ذهبت الوسائل MoveNext و MovePrev و MoveFirst و MoveLast التي تعوّدتَ على استخدامها للانتقال بين سجلات الجدول؟

لم تعُد بك حاجة لهذه الوسائل، وذلك لأنّ البيانات الآن محمّلة في الذاكرة، وبإمكانك التنقل بحرّيّة من سجلّ لآخر.. أمّا حينما كنت تتعامل مع السجلات في ملفّ قاعدة البيانات، فقد كان لزاما عليك أن تنتقل من سجلّ لآخر بالترتيب، حتّى تصل لموضع السجلّ المطلوب!

**ربط الأدوات الأكثر تعقيدا**

نظرا لأنّ جدول الكتب صغير الحجم، فإنّ التطبيق السابق يبدو مناسبا.. ولكن في قواعد البيانات الضخمة، فإنّ التحرّك بين آلاف السجلات واحدا بعد آخر يبدو نوعا من العبث.. لهذا لا بدّ من إنشاء واجهة أكثر ملاءمة لهذا الوضع.

في مثل هذه الحالات يمكن استخدام القوائم Lists لعرض محتويات حقل بأكمله، بحيث يمكن للمستخدم استخدام المنزلق Scroll لتصفّح السجلات.. تعال نرى ذلك عمليّا:

* [**مشروع LookupAuthors:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\3-%20ربط%20الأدوات%20الأكثر%20تعقيدا\4-%20مشروع%20المؤلّفين.htm)
* [**مشروع Books:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\3-%20ربط%20الأدوات%20الأكثر%20تعقيدا\5-%20مشروع%20الكتب.htm)

**مشروع LookupAuthors:**

كلّ ما سنفعله هو تطوير المشروع السابق، باستبدال القائمة المركّبة ComboBox بأزرار الانتقال.. صمّم النموذج ليبدو كما في الصورة.



لن تصلح مجموعة البيانات السابقة.. عدّل جملة التحديد SelectCommand الخاصة بموصل البيانات لتعيد على جدول المؤلّفين الأصليّ كما هو في قاعدة البيانات.. سمّ مجموعة البيانات الجديدة Authors1.

حدّد القائمة المركّبة، ومن نافذة الخصائص اختر خاصيّة "مصدر البيانات" DataSource، وضع بها القيمة Authors1.. ثمّ اختر خاصيّة "العنصر المعروض" DisplayMember، واجعل قيمتها Authors.Author.

إذا شغّلت التطبيق الآن، وضغطت زرّ ملء مجموعة البيانات، فستجد أنّ القائمة المركّبة قد امتلأت بأسماء المؤلّفين.. المدهش أنّك لو ضغطت اسم أيّ مؤلّف، فسيتمّ يتمّ عرض رقمه في مربّع النصّ، دون أن نكتب أيّ كود لذلك!.. إنّ ضغط اسم المؤلّف يغيّر السجلّ الحاليّ، لهذا يتمّ عرض الرقم ID لهذا السجلّ الجديد.

**ملحوظة عجيبة:**

لكي تعمل القائمة المركّبة ComboBox على تغيير السجلّ الحاليّ Current Record في مجموعة البيانات DataSet تلقائيّا عند ضغط أيّ عنصر، فيجب أن تكون لخاصيّة DataSource القيمة Authors1، ولخاصيّة DisplayMember القيمة Authors.Author.

العجيب أنّ القائمة المركّبة لن تقوم بذلك إذا جعلت لخاصيّة DataSource القيمة Authors1.Author، ولخاصيّة DisplayMember القيمة Author.. ولا تسلني لماذا!!.. حينما أعرفُ فسوف أخبرك!!

عامّة لو أردت أن تكتب الكود الذي يغيّر السجلّ الحاليّ عند ضغط عناصر القائمة المركّبة، فيكفيك أن تلاحظ أنّ ترتيب العناصر في القائمة المركّبة هو نفس ترتيبها في مجموعة البيانات، أيّ أنّ رقم السجلّ الحاليّ، هو نفسه رقم العنصر المحدّد في القائمة المركّبة.. لهذا يكفي أن تغيّر قيمة خاصيّة الموضع BindingContext.Position لتصبح مساوية لرقم العنصر المحدّد ComboBox1.SelectedIndex، وذلك في الحدث SelectedIndexChanged.. هذه هي الطريقة العقيمة التي يتبنّاها المرجع الأصليّ!!

**مشروع Books:**

نريد الآن تطوير المشروع السابق، بحيث نضيف للنموذج قائمة مركّبة أخرى تعرض أسماء كتب المؤلّف المحدّد حاليّا في القائمة المركّبة الأولى.

صمّم النموذج كما في الصورة التالية.



والآن اتبع هذه الخطوات:

1- اسحب جدولي المؤلّفين والكتب وألقِهما على النموذج.

2- أنشئ مجموعة بيانات تحتوي على الجدولين.. أسمِها Books.

3- انقر مرّتينِ بالفأرة على الملفّ Books.xsd في متصفّح المشاريع.. ستظهر لك نافذة مخطّط XML.. أنشئ علاقة بين الجدولين كما فعلنا من قبل.

4- الآن تعال نربط الأدوات بمجموعة البيانات:

- بالنسبة للقائمة المركّبة التي ستعرض أسماء المؤلّفين، ومربّع النصّ الذي سيعرض رقم المؤلّف، لن يختلف شيءٌ عن المثال السابق.

- بالنسبة للقائمة المركّبة التي ستعرض أسماء الكتب سيختلف الأمر قليلا.. حدّد خاصيّة DataSource في نافذة الخصائص، وضع فيها القيمة Books1.. ثمّ حدّد خاصيّة DisplayMember، واضغط زرّ الإسدال الموجود في خانة القيمة.. لا تختر جدول الكتب.. اختر جدول المؤلّفين Authors.. ستجد ضمن عناصره الفرعيّة عنصرا جديدا، هو AuthorsBooks.. هذا العنصر ناتج عن العلاقة التي أنشأناها.. اختر هذا العنصر.. ستجد تحته أسماء حقول جدول الكتب.. اختر الحقل Book.. بهذا لن تعرض قائمة الكتب كلّ الكتب الموجودة في قاعدة البيانات.. ستعرض فقط الكتب التي تنتمي للمؤلّف الحاليّ.

- اتبّع إجراءً مشابها مع مربّع النصّ الذي سيعرض رقم الكتاب.. اربط خاصيّة Text بالحقل ID الموجود في العنصر AuthorsBooks تحت جدول المؤلّفين.

5- وأخيرا، اكتب الكود الذي يملأ مجموعة البيانات بسجلات الجدولين.

الآن لو جرّبت البرنامج، فلا ريبَ أنّك ستتيه دهشةً وسعادةً، فلديك واجهة استخدام رائعة، تعمل بطريقة مثاليّة دونَ سطرٍ واحد من الكود ـ سوى ذلك الخاص بملء مجموعة البيانات فقط!!

وحتّى نطرق الحديد وهو ساخن، تعال نرى إمكانيّة أخرى مدهشة.. تلك هي المعالج السحري لنموذج البيانات.

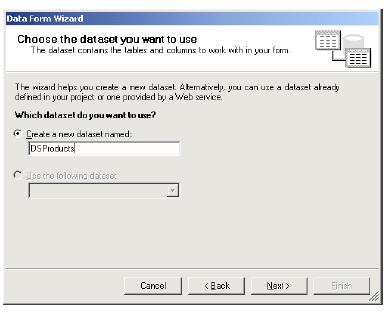
**المعالج السحري لنموذج البيانات DataForm Wizard:**

هذا المعالج سينشئ لك بطريقة سحريّة نموذجا تستطيع استخدامه لإدخال البيانات، بما في ذلك قسط كبير من الكود الذي يجعله يعمل.. تعال نرى ذلك:

- ابدأ مشروعا جديدا وأسمه EditBooks.. احذف النموذج الافتراضيّ Form1، ثمّ اضغط الأمر Add Windows Form من القائمة الرئيسيّة Project.. في مربّع الحوار الذي سيظهر اختر العنصر DataForm Wizard واضغط الزرّ Open.

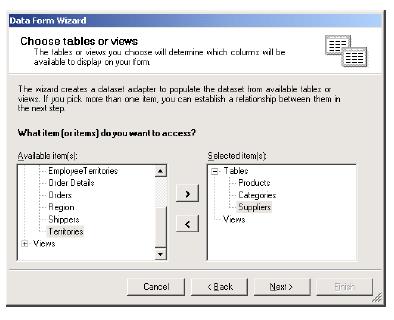
- ستظهر لك شاشة الترحيب الخاصّة بالمعالج السحريّ.. اضغط Next.

- في النافذة الثانية، عليك إدخال اسم مجموعة البيانات DataSet التي ستبني عليها النموذج.. ونظرا لأنّ تطبيقنا لا يحتوي على واحدة، اكتب اسم مجموعة بيانات جديدة (وليكن Books).. اضغط Next.



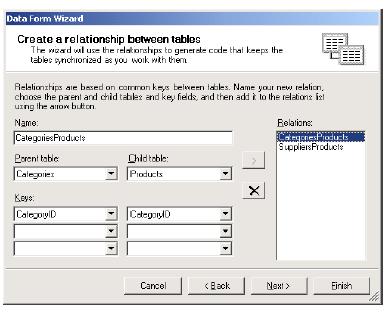
- النافذة الثالثة ستطلب منك اختيار اتصالٍ بقاعدة بيانات موجودة، أو إنشاء اتصال جديد.. اتصل بقاعدة البيانات المنشأة بـ SQL Server.. اضغط Next.

- النافذة الرابعة تعرض لك قائمة (على اليسار) بها كلّ الجداول Tables وطرق العرض Views الموجودة في قاعدة البيانات.



اختر منها واحدة أو أكثر، ليتمّ بناء النموذج عليها.. كلّ ما عليك هو تحديد العنصر، وضغط الزرّ الذي يشير للاتجاه المناسب، وذلك لنقل العناصر بين القائمتين اليمنى واليسرى.. انقل جدولي المؤلّفين والكتب للقائمة اليمنى.. اضغط Next.

- النافذة الخامسة تسمح لك بتحديد العلاقات بين الجدولين.. أنشئ علاقة بين الحقل ID في جدول المؤلّفين، والحقل AuthorID في جدول الكتب.

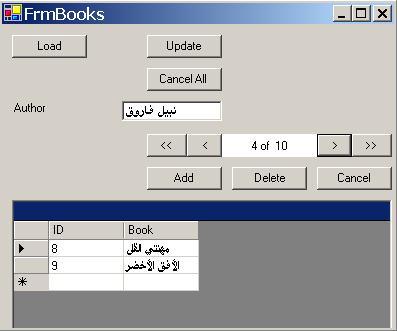


ولا تنس أن تضع اسما للعلاقة، وليكن AuthorBooks.. بعد أن تنتهي، اضغط الزرّ الذي يحمل العلامة ">" لنقل العلاقة لقائمة العلاقات.. إنّ بإمكانك إنشاء أيّ عدد تحبّ من العلاقات بينّ أيّ عدد من الجداول.. اضغط Next.

- النافذة السادسة، تسمح لك بتحديد الجداول والحقول التي سيتعامل معها النموذج.. أزل حقل ID من جدول المؤلّفين، وحقل AuthorID من جدول الكتب.. اضغط Next.

- النافذة الأخيرة تتيح لك اختيار طراز عرض البيانات على النموذج.. لديك هنا طريقتان: العرض في جدول بيانات DataGrid، أو عرض بيانات كلّ حقل في أداة مستقلّة.. أنصحك بتجربة الحالتين، وإن كنّا سنختار هنا الحالة الثانية.. ستجد أنّ بإمكانك اختيار بعض أزرار الوظائف لإضافتها للنموذج (مثل إضافة أو حذف سجلات، أو حفظ التعديلات أو إلغائها... إلخ).. اضغط زرّ Finish لإنهاء المعالج.

ما هي إلا ثوانيَ، وتجد أن نموذجا بأدواته وكوده قد أضيف لتطبيقك.. هذا النموذج سيعمل بدون إضافة حرف واحد من الكود إليه.. مبروك!



عامّة يمكنك تطوير هذا النموذج ليناسب تطلّعاتك.. مثلا، يمكنك إضافة قائمة ListBox تعرض أسماء المؤلّفين، لتسمح للمستخدم باختيار اسم المؤلّف الذي يريده (كما رأينا من قبل)، بدلا من استخدام الأزرار التي أضافها المعالج السحريّ.

**برمجة موصّل البيانات DataAdapter:**

يوجد عيبٌ رئيسيٌّ في كلّ التطبيقات السابقة.. ذلك أنّها تقوم بتحميل كلّ الجداول بكاملها لجهاز العميل Client.. صحيح أنّنا نتعامل مع جدولين صغيرين، لهذا لم تظهر لنا أيّ مشاكل، ولكنّ هذا لن يكون الوضع حينما نتعامل مع جداول تحتوي على عشرات الآلاف من السجلات أو أكثر.. تعال نر إذن كيف نتحكّم في كمّ السجلات التي نحمّلها للعميل:

ابدأ مشروعا جديدا أسمه AuthorsBooks، وصمّم واجهة استخدامه كما في الصورة التالية:



والفكرة في هذا المشروع، هو أن نسمح للمستخدم بكتابة الأحرف الأولى من اسم المؤلّف في مربّع النصّ، حيث سنقوم بملء القائمة بأسماء المؤلّفين التي تبدأ بهذه الحروف.. وعندما يضغط المستخدم اسم أيّ مؤلّف في القائمة، يتمّ عرض كتبه في جدول البيانات.. تعالَ نرى كيف نفعلُ ذلك.

افتح صندوق الأدوات، واضغط شريط البيانات Data.

ضع نسخةً من أداة التوصيل OleDbDataAdapter على النموذج.. سيظهر لك معالج الإعداد السحريّ.. استخدمه للاتصال بقاعدة البيانات التي أنشأناها بـ SQL Server، وتابع الخطوات حتّى تصل لنافذة إدخال الاستعلام.. اكتب فيها جملة SQL التالية:

**SELECT ID, Author**

**FROM Authors**

**WHERE (Author LIKE ?)**

لاحظ أنّ علامة الاستفهام تقوم بعمل المتغيّرات في جمل SQL المخصّصة للنوع QleDb.. إذن فهذا استعلام ذو معامل Parameterized Query، وعليك أن تمدّه بقيمة هذا المعامل قبل محاولة تنفيذه.

في نفس النافذة التي تكتب فيها جملة SQL في معالج الإعداد، اضغط الزرّ Advanced.. أزل علامة الاختيار من الخيار Generate Insert, Update, and Delete Statements، فلن نحتاج لجمل الإدراج والحذف والتحديث في تطبيقنا هذا.

بعد انتهاء خطوات المعالج، سمّ أداة التوصيل DAAuthors.. لاحظ أنّ نسخة من أداة الاتصال OleDbConnection قد أضيفت للنموذج.. لاحظ كذلك أنّنا نستخدم هذه الأداة هذه المرّة للاتصال بقاعدة بيانات SQL Server.. إنّ هذا يعني أنّ أداة الاتصال من النوع OleDbConnection لا تختلف في شيءٍ عن أداة الاتصال من النوع SqlConnection، غير أنّ هذه الأخيرة مهيأة بطريقةٍ أفضلَ للتعامل مع قواعد بيانات SQL Server.

بعد ذلك ضع نسخة أخرى من أداة توصيل البيانات على النموذج، واكتب لها جملة SQL التالية:

**SELECT ID, Book**

**FROM Books**

**INNER JOIN Authors ON AuthorID = Authors.ID**

**WHERE AuthorID = ?**

واضح طبعا أنّ هذه الجملة تسترجع من قاعدة البيانات كلّ الكتبَ التي ألّفها المؤلّف رقم ?.. هذا استعلام آخر ذو معامل، ولا بدّ من إمداده بقيمة هذا المعامل قبل تنفيذه.. لا تنس إزالة اختيار إنشاء جمل الإدراج والحذف والإضافة.. سمّ أداة التوصيل DABooks.

أنشئ الآن مجموعة بيانات للجدولين.. سمّ الأولى Authors واربطها بجدول المؤلّفين فقط (لا تضع علامة الاختيار أمام جدول الكتب)، وسمّ الثانية Books واربطها بجدول الكتب فقط.

اربط القائمة بمجموعة البيانات Authors1، بحيث تعرض أسماء المؤلّفين.

واربط جدول البيانات DataGrid بمجموعة البيانات Books1.

الآن نتفرّغ لكيفيّة تحديد قيم معاملات الاستعلام.. حدّد موصّل البيانات ADAuthors، ومن نافذة الخصائص أسدل خاصيّة SelectCommand.. تحت هذا الكائن ستجد خاصيّة "المعاملات" Parameters.. هنا يمكنك أن تمدّ الاستعلام بقيم المعاملات.

اضغط زرّ الانتقال الموجود في خانة القيمة لهذه الخاصيّة.. ستظهر لك نافذة تحرير مجموعة المعاملات.. على اليسار ستجد قائمة بأسماء المعاملات.. ونظرا لأنّنا لم نحدّد أسماء هذه المعاملات، فإنّها ستأخذ نفس أسماء الحقول التي ترتبط بها.. فمثلا ستجد لأمر التحديد في جدول المؤلّفين معاملا واحدا اسمه Author.

وعلى يمين النافذة ستجد قائمة بخصائص هذا المعامل.. ومنها:

|  |  |
| --- | --- |
| Direction | لتحديد إذا ما كان معامل إدخالٍ Input أم إخراج Output. |
| SourceColumn | لتحديد الحقل الذي يتعامل معه المعامل. |
| OleDbType | لتحديد نوع البيانات التي ستوضع في المعامل |
| Value | لوضع قيمة للمعامل، وإن كنّا سنترك ذلك للكود، لأنّ القيمة ستتغيّر مرارا على حسب اختيارات المستخدم. |

إنّه إذن وقت كتابة بعض الكود:

في البداية سنكتب الكود الذي يحصل على المؤلّفين الذين تبدأ أسماؤهم بالحروف الموجودة في مربّع النصّ.. ولكن أين نكتب مثل هذا الكود؟

سنكتبه في حدث ارتفاع الزرّ KeyUp الخاصّ بمربّع النصّ.. لا تتسرّع وتظنّ أنّنا سننفّذ جملة SQL مع كلّ حرف يكتبه أو يمحوه المستخدم.. إنّ فعل هذا سيكون جريمةً في حقّ البرمجةِ والمستخدم، وسيرزأُ البرنامج ببطء شديد!

إذن فماذا نفعل؟

سنترك للمستخدم حرّيّة اتخاذ القرار.. سنُحضر أسماء المؤلّفين عندما يضغط المستخدم زرّ الإدخال Enter من لوحة المفاتيح.. ها هو ذا الكود:

**Private Sub TextBox1\_KeyUp(ByVal sender As Object, \_**

**ByVal e As KeyEventArgs) Handles TextBox1.KeyUp**

**If e.KeyCode = Keys.Enter Then**

**DAAuthors.SelectCommand.Parameters("Author").Value \_**

**= TextBox1.Text & "%"**

**Authors1.Clear()**

**DAAuthors.Fill(Authors1)**

**' يجب استدعاء حدث ضغط القائمة، حتّى تعرض كتب أوّل مؤلّف في القائمة**

**If ListBox1.SelectedIndex <> -1 Then**

**ListBox1\_SelectedIndexChanged(Me, e)**

**End If**

**End If**

**End Sub**

أعتقد أنّ الكود واضح ولا يحتاج للشرح.

الآن علينا أن نكتب الكود الذي يملأ جدول البيانات.. يجب أولا أن نعرف رقم المؤلّف المحدّد حاليّا في القائمة.. كيف؟

إنّ ذلك ممكن عبر مجموعة البيانات، فهي عبارة عن خليّة Class تحتوي على الخصائص والوسائل اللازمة للتعامل مع الجداول التي تحتويها.. فمثلا للتعامل مع جدول المؤلّفين، يمكنك أن تكتب:

**Authors1.Authors**

وللتعامل مع سجلّ معيّن في هذا الجدول، اكتب رقمه بين قوسين كالتالي:

**Authors1.Authors**(3)

والتعامل مع حقل معيّن في هذا السجلّ، استخدم الصيغة التالية:

**Authors1.Authors**(3).ID

الآن يمكننا حلّ المشكلة السابقة، بملاحظة أنّ رقم السجلّ في الجدول هو نفسه رقم المؤلّف في القائمة.. هذا هو الكود الجديد:

**Private Sub ListBox1\_SelectedIndexChanged(ByVal sender As Object, \_**

**e As EventArgs) Handles ListBox1.SelectedIndexChanged**

**DABooks.SelectCommand.Parameters("AuthorID").Value \_**

**= Authors1.Authors(ListBox1.SelectedIndex).ID**

**Books1.Clear()**

**DABooks.Fill(Books1)**

**End Sub**

وهناك طريقة أخرى طريفة لمعرفة رقم المؤلّف في الجدول، وذلك بوضع هذا الرقم في خاصيّة ValueMember للقائمة.. حدّد هذه الخاصيّة في نافذة الخصائص، واربطها بالحقل ID في جدول المؤلّفين.. الآن لو استخدمت خاصيّة SelectedValue فلن تعيد لك اسم المؤلّف، ولكنّها ستعيد لك رقمه في الجدول.. وبهذا يمكن تعديل السطر الأوّل في الكود السابق ليصير كالآتي:

**DABooks.SelectCommand.Parameters("AuthorID").Value \_**

**= ListBox1.SelectedValue**

**الاتصال بقاعدة البيانات مباشرة**

ماذا لو لم تكن تريد أن تحمّل السجلات من قاعدة البيانات إلى جهاز العميل؟.. ماذا لو كان تطبيقك يريد التعامل مع سجلّ واحد فقط ليس أكثر؟.. ماذا لو كانت قاعدة البيانات موجودةً على نفس الجهاز؟.. أعتقد أنّك تحتاج لطريقة أخرى للتعامل مع قاعدة البيانات.. هذه الطريقة يقدّمها لك كلّ من:

- كائن الأوامر Command Object الذي يسمح لك بتنفيذ استعلام Query على قاعدة البيانات.

- وكائن قراءة البيانات DataReader، الذي يسمح لك بالمرور عبر السجلات التي حصلت عليها من الاستعلام.

* [**كائن الأوامر Command Object:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\9-%20الاتصال%20بقاعدة%20البيانات\1-%20كائن%20الأوامر.htm)
* [**كائن قراءة البيانات DataReader:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\9-%20الاتصال%20بقاعدة%20البيانات\2-%20كائن%20قراءة%20البيانات.htm)
* [**مشروع قارئ البيانات DataReader:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\9-%20الاتصال%20بقاعدة%20البيانات\3-%20مشروع%20قارئ%20البيانات.htm)
* [**مشروع StoredProcedure:**](file:///C:\Users\Administrator\Desktop\vb.net\e-%20%20Databases%20قواعد%20البيانات\17-%20تطبيقات%20قواعد%20البيانات\9-%20الاتصال%20بقاعدة%20البيانات\4-مشروع%20الإجراءات%20المخزّنة.htm)

**كائن الأوامر Command Object:**

لا جديد.. لقد تعاملنا مع هذا الكائن بالفعل من خلال موصّل البيانات.. هل تذكر ذلك؟

إنّ الخصائص SelectCommand و DeleteCommand و InsertCommand و UpdateCommand ما هي إلا كائنات من هذا النوع.

وسنرى كيف ننشئ كائن الأوامر من خلال الكود، ونضع قيما لخصائصه بأنفسنا، مثل خاصيّة الاتصال Connection، وخاصيّة "نصّ الأمر" CommandText.. بعد ذلك سنستخدم إحدى الوسائل التالية لتنفيذه:

**ExecuteReader:**

تنفّذ الأمر، وتعيد كائنا من نوع قارئ البيانات DataReader، حيث يمكنك استخدامه لقراءة النتيجة سجلا وراء سجلّ.

**ExecuteXMLReader:**

تنفّذ الأمر، وتعيد كائنا من النوع XMLDataReader، وهو مشابه لقارئ البيانات.

**ExecuteScalar:**

تنفّذ الأمر، وتعيد الخانة الموجودة في الصفّ الأوّل من العمود الأوّل في الجدول الناتج، وتتجاهل باقي الخانات.. ويمكنك استخدام هذه الوسيلة لتنفيذ دوال التجميع Aggregate Functions .

**ExecuteNonQuery:**

تنفّذ الأمر، وتعيد فقط عدد السجلات التي تأثّرت بالأمر.. ويمكنك استخدام هذه الوسيلة لتنفيذ أوامر التحديث والحذف والإدراج.

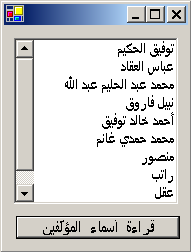
**كائن قراءة البيانات DataReader:**

إنّ قارئ البيانات DataReader أسرع بمراحل من مجموعة البيانات، ولكنّه في الوقت ذاته لا يسمح لك بتعديل السجلات، ولا حتّى بالتحرّك فيها الخلف، فأنت مجبرٌ على استخدام الوسيلة Read لقراءة السجلات واحدًا تلو الآخر، وفي أثناء ذلك يكون الاتصال بينك وبينَ قاعدة البيانات مفتوحا.. باختصار: إنّ قارئ البيانات DataReader يعيد لك نتيجةً توصفُ بأنّها للأمام فقط وللقراءة فقط Forward-only, Read-only Result.

غنيّ عن الذكر، أنّ هناك نوعين من قارئ البيانات: SqlDataReader و OleDbDataReader.

**مشروع قارئ البيانات DataReader:**

تعال نرى كيف نستخدم قارئ البيانات في قراءة أسماء المؤلّفين ووضعها في قائمة.



ضع نسخة من الأداة SqlConnection ونسخةً من الأداة SqlCommand على النموذج.

حددّ الأداة SqlConnection1 على النموذج، ومن نافذة الخصائص حدّد خاصيّة ConnectionString.. اضغط زرّ الإسدال في خانة القيمة واختر قاعدة البيانات Books من القائمة المنسدلة.

ثمّ حدّد الأداة SqlCommand1، ومن نافذة الخصائص حدّد خاصيّة Connection.. اضغط زرّ الإسدال واختر SqlConnection1 من القائمة المنسدلة.

بعد ذلك حدّد خاصيّة CommandText، واضغط زرّ الانتقال الموجود في خانة القيمة.. ستظهر لك نافذة باني الاستعلام.. استخدمه لإضافة جدول المؤلفين، واكتفِ بعرض أسماء المؤلّفين فقط.. يجب أن تكون جملة SQL كالتالي:

**SELECT Author FROM Authors**

الآنَ لم يبقَ أمامنَا إلا تنفيذ هذا الاستعلام.. اكتب ما يلي في حدث ضغط الزرّ:

**Private Sub Button1\_Click(ByVal sender As System.Object,\_**

**ByVal e As EventArgs) Handles Button1.Click**

**SqlConnection1.Open()**

**Dim SQLReader As SqlClient.SqlDataReader**

**SQLReader = SqlCommand1.ExecuteReader()**

**ListBox1.Items.Clear**

**While SQLReader.Read**

**ListBox1.Items.Add(SQLReader.Item("Author"))**

**End While**

**SqlConnection1.Close()**

**End Sub**

تلاحظ طبعا أنّنا نستخدم الوسيلة Read لقراءة السجلّ التالي، وهي تعيد False إذا لم يعد هناك المزيد من السجلات.. إنّ السجلات ليست مخزّنة في قارئ البيانات.. إنّه يطلبها من قاعدة البيانات واحدا وراء آخر، في كلّ مرّة تستدعي فيها الوسيلة Read.. فإذا أردت أن تقرأ قيمة أيّ حقل في السجلّ الحاليّ، فعليك باستخدام الخاصيّة Item، حيث تمدّها باسم الحقل (أو رقمه) كمعامل.

**مشروع StoredProcedure:**

أتذكر الإجراء المخزّن المسمّى GetAuthorBooks الذي أنشأناه في الفصل السابق؟.. أعتقد أنّ الوقت قد حان لاستخدامه.

إنّ هذا الإجراء يستقبلُ منك اسم المؤلّف، ويعيد لك كتبه.

ابدأ مشروعا جديدا، وصمّم النموذج كما تراه في الصورة التالية:



اسحب اسم الإجراء المخزّن GetAuthorBooks من متصفّح خوادم الإنترنت وأسقطه على النموذج.. سيتمّ إنشاء أداة اتصال SqlConnection1 وأداة الأوامر SqlCommand1 تلقائيّا على النموذج.

لو نظرت الآن في خصائص أداة الأوامر، فستجد أنّ خاصيّة CommandText تحمل اسم الإجراء المخزّن، في حين صارت خاصيّة "نوع الأمر" CommandType تحمل القيمة StoredProcedure.. إنّ قيمة هذه الخاصيّة’ في كلّ البرامج التي تعاملنا معها سابقا كانت Text، وذلك لنتمكّن من تنفيذ جملة SQL.. أمّا الآن فقد صار لها القيمة StoredProcedure، لنتمكّن من تنفيذ إجراء مخزّن.. وهناك قيمة ثالثة، وهي TableDirect، وذلك لو أردت أن تسترجع من قاعدة البيانات جدولا بكامله كما هو بدون أيّ تعديل.

حدّد خاصيّة المعاملات Parameters، واضغط زرّ الانتقال.. ستجدّ أنّ هناك معاملين للإجراء.. القيمة المعادة @RETURN\_VALUE، ويمكنك حذفها فنحن لم نستخدمها في الإجراء، و @Author وهو معامل إدخال، حيث نضع به اسم المؤلّف الذي نريد معرفة كتبه.

الآن نصل لآخر خطوة: تنفيذ الأمر، والحصول على النتيجة بواسطة قارئ البيانات.. اكتب الكود التالي في حدث ضغط الزرّ:

**SqlConnection1.Open()**

**SqlCommand1.Parameters("@Author").Value = TextBox1.Text**

**Dim SQLReader As sqlclient.SqlDataReader**

**SQLReader = SqlCommand1.ExecuteReader**

**ListBox1.Items.Clear**

**While SQLReader.Read**

**ListBox1.Items.Add(SQLReader.Item("Book"))**

**End While**

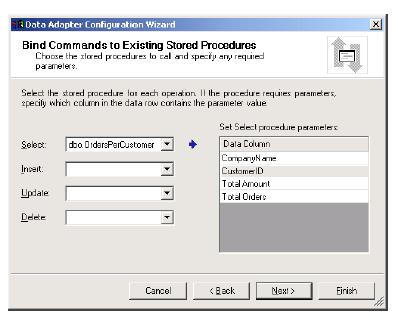
**SqlConnection1.Close()**

شغّل البرنامج، وجرّب أن تكتب اسم (توفيق الحكيم) في مربّع النص وتضغط الزرّ.

طبعا لن يكون مريحا للمستخدم أن يكتب اسم المؤلّف بنفسه، فذلك يجعله عرضه للخطإ في كتابته.. عامّةً، يمكنك أن تضيف قائمة للنموذج بدلا من مربّع النصّ، وفيها تضع أسماء المؤلّفين بأيٍّ من الطرق التي سبق لنا التعرّف عليها في هذا الفصل.

وحتّى لا تظنّ أنّك مجبر على استخدام الإجراءات المخزّنة مع قارئ البيانات فقط، تعال نرى كيف نستخدمها مع مجموعة البيانات.

أضف أداة توصيل البيانات DataAdapter إلى النموذج، واضغطها بزرّ الفأرة الأيمن.. ومن القائمة الموضعيّة اضغط الأمر Configure DataAdapter.. اتبع خطوات المعالج السحريّ كما تعرّفنا عليها سابقا (وتأكّد أنّك اخترت قاعدة البيانات المنشأة بـ SQL Server)، حتّى تصل لشاشة اختيار نوع الاستعلام، اختر الاختيار الأخير: Use Existing Stored Procedue واضغط Next.



في النافذة التالية ستجد أربع قوائم منسدلة، تمكّنك من اختيار اسم أربع إجراءات مخزّنة لتنفيذها في أوامر التحديد والإدراج والحذف والتحديث.. ولو أسدلت أيّا منها فستجدها تحتوي على اسم الإجراء GetAuthorBooks.. أسدل قائمة التحديد Selection، واختر هذا الإجراء.. أضغط Next وواصل باقي خطوات المعالج.

الآن سيظهر لك في نافذة إنشاء مجموعة البيانات، اسم الإجراء المخزّن مع أسماء الجداول التي يمكن وضعها داخل مجموعة البيانات.. اختر اسم الإجراء المخزّن.

بعد ذلك اضغط الأمر Preview Data من القائمة الرئيسيّة Data Menu.. في نافذة استعراض البيانات، اختر اسم مجموعة البيانات الجديدة من القائمة المنسدلة.. في الجدول المخصّص للمعاملات على يمين الشاشة سيظهر اسم المعامل Author.. في حقل القيمة Value اكتب أيّ اسم للتجريب، وليكن "نبيل فاروق".. اضغط الآن الزرّ Fill DataSet، لتظهر لك أسماء الكتب التي ألّفها.

أعتقد أنّك تعرف باقي الخطوات بالفعل.