من الصفر إلى الاحتراف:

جدول عرض البيانات DataGridView

ومكرر البيانات DataRepeater

لمبرمجي سي شارب

بقلم:

م. محمد حمدي غانم

هذا الكتاب صدقة جارية على روح والدي:

أ. حمدي كامل الحديدي غانم

رحمه الله وغفر له وجعل مثواه الجنة

لهذا أرجو من كل من يستفيد به أن يتذكر أن أبي هو الـــذي ربـــاني وعلمنـــي ولولاه بعد توفيق الله ما خرج إلى الوجود هذا الكتاب وغيره من الكتب.

فادعوا له بالرحمة والمغفرة

ومن كان منكم في الحرمين الشريفين وكان قادرا على عمل عمرة له، فجزاه الله خيرا.

أدعو الله أن يكون هذا الكتاب وباقي كتبي من العلم الذي ينتفع به، وأن يجعل الله لأبي نصيبا من ثوابه، فيكون من عمله الذي لا ينقطع بموته.

اللهم ارحم أبي واغفر له وكفر عنه سيئاته وقه من عذاب القبر وقه من عذاب النار،
وأدخله الجنة وأعلِ منزلته فيها
واحفظ والدتي وبارك في عمرها
اللهم ارحم والديّ كما ربياني صغيرا
آمين يا رب العالمين

ملحوظة:

هذا الكتاب جزء من كتاب:

من الصفر إلى الاحتراف برمجة قواعد البيانات في سي شارب ٢٠١٠

وقد نشرت جزءا آخر من الكتاب هو:

إنشاء قواعد البيانات وكتابة استعلامات SQL

يمكنم تحميله من هنا:

https://drive.google.com/file/d/1H3FqC-jEXihVI5fx-7ZBFmVGJm2D7Oi3/edit?fbclid=IwAR31PkLyHT1QTN1xNoozTWsvkwQ5-uowwQzNarL4EhPULQsy4-CGBOl1Cv0

وسأنشر الجزء الأخير الخاص بتقنية ADO.NET قريبا بـــإذن الله، فتـــابعوني علـــى صفحتي البرمجة:

https://www.facebook.com/vbandcsharp

يمكن تحميل الأمثلة المشار إليها يف هذا الكتاب من هنا:

https://drive.google.com/file/d/17jpo-/76DzVBdxXgLGQQV0Y5pQHDBt4qi

للتواصل مع الكاتب:

- بريدي الالكتروني:

msvbnet@hotmail.com

- مدونتي:

http://mhmdhmdy.blogspot.com

- قناتي على يوتيوب (تحتوي على إلقاء أكثر من ٦٠ قصيدة بصوتي):

http://www.youtube.com/user/mhmdhmdy

- صفحتي الأدبية على فيسبوك:

https://www.facebook.com/Poet.Mhmd.Hmdy

- كتبي في مجال البرمجة بلغتي فيجوال بيزيك وسي شارب:

https://drive.google.com/drive/folders/1J21xi8Aw15BFSv-GUgVOElLuYM6zoNct

- صفحة فيجوال بيزيك وسي شارب على فيسبوك:

https://www.facebook.com/vbandcsharp

كتب مجانية للكاتب للتنزيل:

١ - كتاب: "خرافة داروين، حينما تتحول الصدفة إلى علم":

http://mhmdhmdy.blogspot.com/2013/11/blog-post_29.html

٢ - كتب أدبية (أشعار وقصص وروايات):

https://mhmdhmdy.blogspot.com/2018/10/blog-post_23.html

٣- المبرمج الصغير:

http://mhmdhmdy.blogspot.com.eg/2016/10/blog-post_9.html

٤- الرسم والتلوين والصور والمجسمات لمبرمجي فيجوال بيزيك دوت نت:

http://mhmdhmdy.blogspot.com.eg/2014/05/blog-post_26.html

٥- الرسم والتلوين والصور والمجسمات لمبرمجي سي شارب:

http://mhmdhmdy.blogspot.com.eg/2014/08/c.html

٦- من الصفر إلى الاحتراف: برمجة قواعد البيانات في فيجوال بيزيك دوت نت:

http://mhmdhmdy.blogspot.com.eg/2016/02/blog-post_28.html

كتب مطبوعة للكاتب:

١. **فيجيوال بيزيك وسي شارب:** طريقك المختصر للانتقال من إحدى اللغتين إلى الأخرى.
٢. من الصفر إلى الاحتراف: **فيجيوال بيزيك دوت نت ٢٠١٧.**
٣. من الصفر إلى الاحتراف: **سي شارب ٢٠١٧.**
٤. من الصفر إلى الاحتراف: **برمجة إطار العمل لمبرمجي فيجيوال بيزيك دوت نت وسي شارب.**
٥. من الصفر إلى الاحتراف: **برمجة نماذج الويندوز لمبرمجي فيجيوال بيزيك دوت نت وسي شارب.**
٦. المدخل العملي السريع إلى **فيجيوال بيزيك دوت نت ٢٠١٧.**
٧. المدخل العملي السريع إلى **سي شارب ٢٠١٧.**
٨. أساسيات WPF لمبرمجي **فيجيوال بيزيك دوت نت.**
٩. أساسيات WPF لمبرمجي **سي شارب.**

لقراءة مقدمة وفهرس كل كتاب:

https://drive.google.com/drive/folders/1J21xi8Aw15BFSv-GUgVOElLuYM6zoNct

لشراء هذه الكتب، يتم تحويل الثمن بحوالة بريدية داخل مصر، أو بويسترن يونيون من خارج مصر، ويتم إرسال الكتب بطرد بالبريد السريع.. لمزيد من التفاصيل أرسل رسالة بالكتب المطلوبة إلى:

msvbnet@hotmail.com

كتب أجهز لكتابتها في المرحلة القادمة بإذن الله:

- برمجة قواعد البيانات بـــ Entity Framework.
- إنشاء تقارير Report Viewer و Crystal Reports وعرضها وطباعتها.
- برمجة مواقع الويب بـــ ASP.NET MVC Core.
- المواضيع المتقدمة في برمجة إطار العمل.
- الوسائط المتعددة في WPF.
- برمجة مشاريع Windows Universal Applications.
- برمجة الاندرويد بـــ Xamarin.
- برمجة الشبكات بدوت نت.

سجلوا إعجابكم بصفحتي البرمجية لمتابعة صـــدور هـــذه الكتـــب بـــإذن الله، والاســـتفادة بالملاحظات البرمجية العملية التي أنشرها على الصفحة:

https://www.facebook.com/vbandcsharp

للتواصل مع باقي المبرمجين وتبادل الأسئلة والخبرات، يمكنكم الانضمام إلى هذه المجموعة:

https://www.facebook.com/groups/123809374886424/

محتويات الكتاب

-١٥-

جدول عرض البيانات DataGridView

ملحق ١

الفئات التي يستخدمها جدول عرض البيانات

- فئة عمود مربعات الاختيار DataGridViewCheckBoxColumn Class
- فئة عمود الصور DataGridViewImageColumn Class
- فئة عمود الوصلات DataGridViewLinkColumn Class
- فئة عمود القوائم المركبة DataGridViewComboBoxColumn
- فئة مجموعة صفوف جدول العرض DataGridViewRowCollection
- فئة صف جدول العرض DataGridViewRow Class
- فئة خانة جدول العرض DataGridViewCell Class
- فئة خانة مربع النص DataGridViewTextBoxCell Class
- فئة خانة الزر DataGridViewButtonCell Class
- واجهة خانة التحرير IDataGridViewEditingCell Interface
- فئة خانة مربع الاختيار DataGridViewCheckBoxCell Class
- فئة خانة الصور DataGridViewImageCell Class
- فئة خانة الوصلة DataGridViewLinkCell Class
- فئة خانة القائمة المركبة DataGridViewComboBoxCell Class
- واجهة أداة التحرير IDataGridViewEditingControl Interface
- فئة أداة تحرير مربع النص DataGridViewTextBoxEditingControl
- فئة أداة تحرير القائمة DataGridViewComboBoxEditingControl
- فئة الخانة الرئيسية DataGridViewHeaderCell Class
- فئة خانة رأس العمود DataGridViewColumnHeaderCell
- فئة الخانة العلوية اليسرى DataGridViewTopLeftHeaderCell
- فئة خانة رأس الصف DataGridViewRowHeaderCell
- فئة طراز خانة جدول العرض DataGridViewCellStyle Class
- فئة طراز الحافة المتطور DataGridViewAdvancedBorderStyle

-١٦-

شبكة البيانات DataGrid

-١٧-

مُكرِّر البيانات Data Repeater

مقدمة

يشرح الكتاب بالتفصيل جدول عرض البيانات DataGridView وشبكة البيانات DataGrid ومكرر البيانات DataRepater.

ويحتوي الكتاب على أمثلة عملية تعلمك كيف تتعامل مع هذه الأدوات، وكيف تنشئ أنواعا جديدة من أعمدة جدول العرض، تعرض خاناتها أداة اختيار التاريخ أو شجرة منسدلة أو أي نوع آخر تريده من الأدوات، كيف تجعل جدول العرض يعمل في الوضع الافتراضي Virtual Mode وكيف تضيف إليه تقنية تقسيم السجلات على صفحات Paging، وكيف ترسم مستطيلا حول الصف الحالي في جدول العرض، وكيف تنشئ قالبا لعرض كل سجل، وكيف تكرر عرضه باستخدام مكرر البيانات DataRepeter، وكيف تستخدم مكرر البيانات في الوضع الافتراضي Virtual Mode.

وغير هذا الكثير.

والله ولي التوفيق

الرموز المستخدمة في هذا الكتاب:

	سجل Structure.
	فئة Class.
	واجهة Interface.
	ثابت Constant.
	خاصية Property يمكنك قراءة أو تغيير قيمتها.
	خاصية للقراءة فقط Read Only Property.
	وسيلة Method.
	معامل Operator.
	حدث Event.
S	هذا العنصر ثابت Static، يمكن استخدامه عبر اسم الفئة مباشرة.

-١٥-

جدول عرض البيانات
DataGridView

توجد الأداة DataGridView تحت الشريط Data في صندوق الأدوات Toolbox، وهي تصلح لعرض جداول قواعد البيانات، وتتيح لك تنسيق البيانات المعروضة بالشكل الذي يناسبك، كما تتيح للمستخدم إضافة السجلات وحذفها، وتغيير قيم خاناتها، حيث تحفظ هذه التعديلات مباشرة في الجدول الأصلي في مجموعة البيانات من خلال آلية الربط، ويمكنك بعد هذا إجراء عملية تحديث Update لإرسال هذه التغييرات إلى قاعدة البيانات.

ملحوظة:
الأداة DataGridView هي تطوير لأداة قديمة اسمها DataGrid وهي ما زالت متاحة للاستخدام لكنها لا تظهر في شريط الأدوات إلا إذا قمت أنت بإضافتها إليه بالطريقة المألوفة.. وننصح باستخدام الأداة DataGridView لأنها تملك قدرات أكثر بكثير، وإن كانت الأداة DataGrid تتفرد بالقليل من الميزات، لهذا سنتعرف عليها في الفصل التالي.

ولربط جدول العرض بأحد جداول قاعدة البيانات، يمكنك استخدام الخاصيتين DataSource و DataMember بإحدى الطريقتين التاليتين:

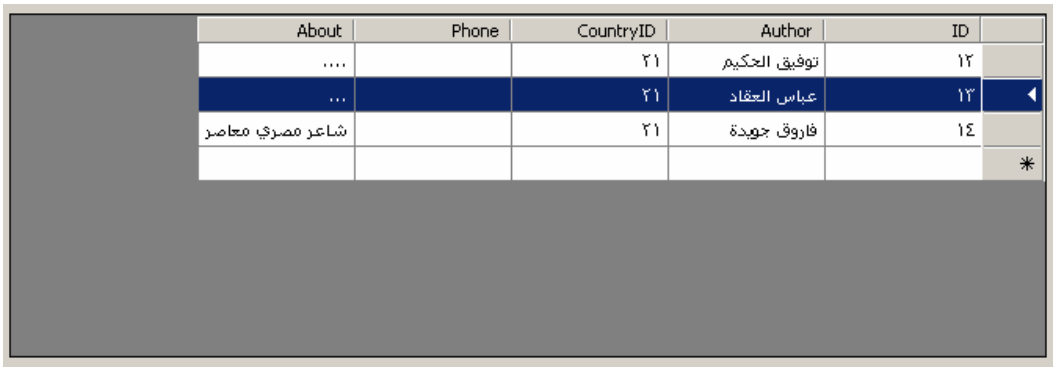
١- أن تضع مجموعة البيانات في الخاصية DataSource، وتضع اسم الجدول في الخاصية DataMember كالتالي:

DataGridView1.DataSource = DsBooks;
DataGridView1.DataMember = "Authors";

٢- أن تضع الجدول مباشرة في الخاصية DataSource كالتالي:

DataGridView1.DataSource = DsBooks.Tables["Authors"];

في كلتا الحالتين سيظهر جدول العرض كما في الصورة:



	ID	Author	CountryID	Phone	About
	١٢	توفيق الحكيم	٢١		....
◀	١٣	عباس العقاد	٢١		...
	١٤	فاروق جويدة	٢١		شاعر مصري معاصر
*					

كما تلاحظ في الصورة، يتكون جدول عرض البيانات مما يلي:

- أعمدة Columns، ولكل عمود منها رأس Header، وهو خانة ثابتة من خانات الجدول تظهر أعلى العمود وتعرض عنوان العمود، لهذا سنسمي رأس العمود أحيانا بخانة العنوان.
- صفوف Rows، ولكل منها هامش عند الضغط عليه يتم تحديد الصف.. هذا الهامش يسمى رأس الصف، أو خانة عنوان الصف، وهو يعرض أيقونة التحرير عند الكتابة في أي خانة في الصف، ويعرض أيقونة الخطأ عند وجود قيم خاطئة في الصف، كما يمكنك أن تكتب فيه نصا كعنوان للصف.. لاحظ أن الصف الأخير (الذي تسبقه النجمة *) هو صف جديد New Row، عندما يكتب فيه المستخدم تتم إضافته إلى الجدول، ويظهر صف جديد بدلا منه.
- خلايا Cells: وهي خانات الجدول، ويمكنك ضغطها بالفأرة لبدء تحريرها، كما يبتم بدء التحرير بمجرد الكتابة من لوحة المفاتيح أثناء تحديد الخانة.. وعندما تكون الخانة في وضع التحرير، يظهر فيها مربع نص للكتابة فيه.. ويمكن إنهاء التحرير بضغط Enter أو إلغائه بضغط Esc لتعود القيمة الأصلية للخانة.. ولا يتم قبول التغييرات

التي حدثت في خانات أحد الصفوف إلا إذا ضغط المستخدم CTRL+Enter، أو انتقل إلى صف آخر.

- خلفية جدول العرض، وهي المنطقة الخالية التي لا تظهر فيها الخانات.

ويمكنك ربط جدول العرض بمصادر بيانات أخرى غير جداول البيانات.. والمشروع BindGridToArray يريك كيف يمكن ربط جدول العرض بمصفوفة تحتوي على كائنات من نوع الفئة Student، وبسطر واحد من الكود:

Grd.DataSource = Std;

كل ما فعلناه هو استخدام المصفوفة كمصدر للبيانات، ليقوم جدول العرض تلقائيا بإنشاء أعمدة بأسماء خصائص الفئة Student، ووضع قيم المصفوفة فيها.. منتهى البساطة والروعة!

لكن لا تدع هذه البساطة تخدعك، فجدول العرض أداة ضخمة، وهناك عدد كبير من الفئات Class لتمثيل مكوناتها، والتي ستجدها مشروحة بالتفصيل في الملحق رقم ١: فئات جدول العرض، فارجع إليه كلما مرت عليه إحدى هذه الفئات هنا.

فئة جدول عرض البيانات DataGridView Class

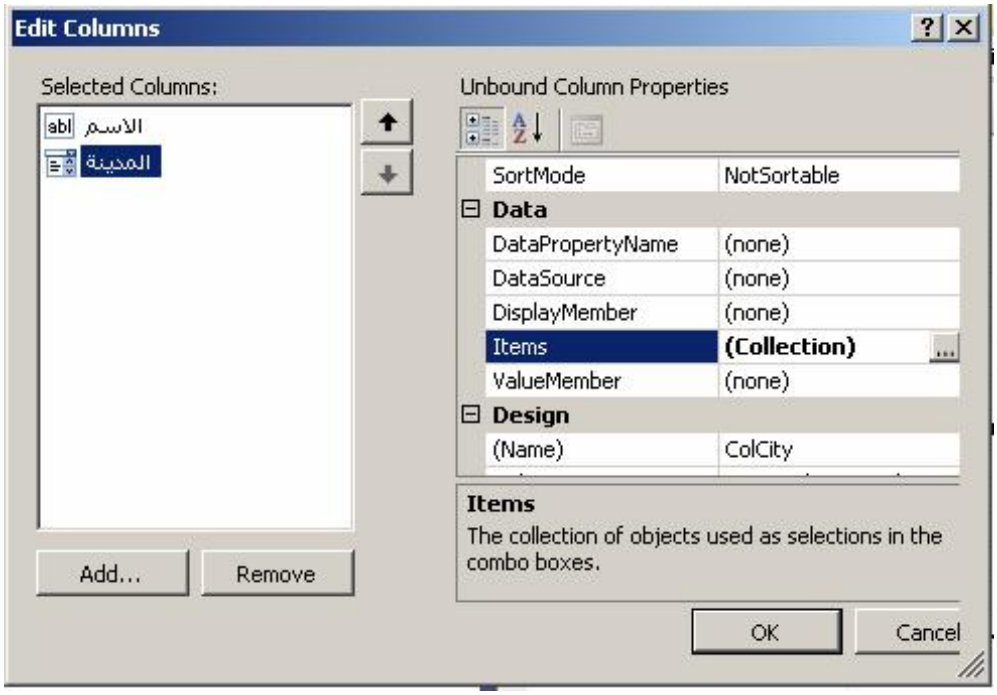
هذه الفئة ترث فئة الأداة الأم Control Class، وهي تعرض وتتحكم في الأعمدة والصفوف والخانات التي تعرفنا عليها.
ونظرا لأن هذه الأداة تمتلك عددا هائلا من الخصائص والوسائل، فسنقسمها إلى مجموعات حسب الوظيفة ليسهل علينا فهمها.

التعامل مع أعمدة جدول العرض:

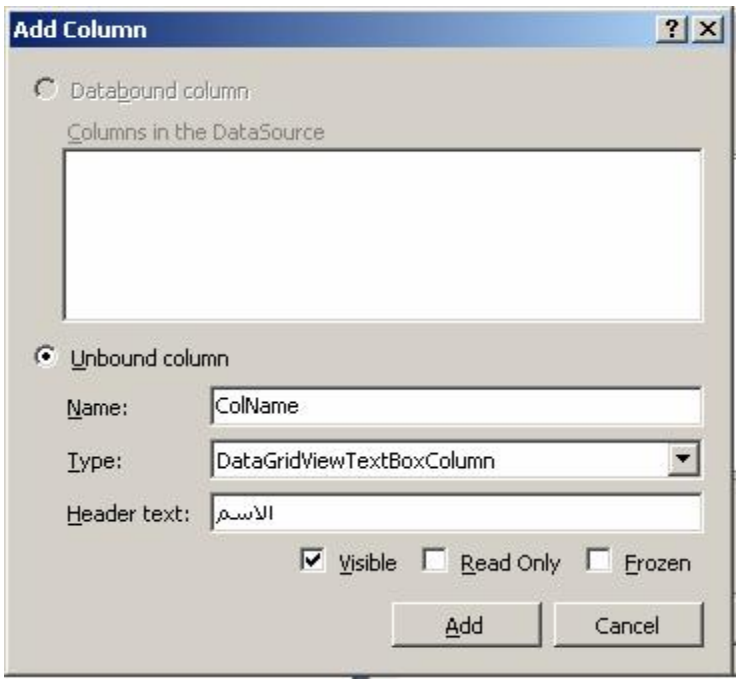
يقدم لك جدول العرض الخصائص التالية، للتعامل مع الأعمدة:

الأعمدة Columns:

تعيد مجموعة أعمدة جدول العرض DataGridViewColumnCollection، التي تحتوى على كائنات الأعمدة DataGridViewColumn Objects الموجودة في جدول العرض.
ويمكنك إضافة الأعمدة إلى هذه المجموعة في وقت التصميم باستخدام نافذة الخصائص، وذلك بضغط زر الانتقال الموجود في خانة قيمة هذه الخاصية، حيث ستظهر لك النافذة الموضحة في الصورة:
في هذه النافذة تظهر أسماء الأعمدة في القائمة اليسرى، بينما تظهر خصائص العمود المحدد في الجزء الأيمن.. على سبيل المثال، تريك الصورة خصائص العمود المسمى "المدينة"، وهو عمود يعرض قائمة منسدلة.. ويمكنك إضافة العناصر إلى هذه القائمة بضغط زر الانتقال الموجود في خانة الخاصية Items في قسم الخصائص، حيث ستظهر لك نافذة تتيح لك إضافة عناصر إلى المجموعة.



ويمكنك حذف أي عمود من القائمة بتحديده وضغط الزر Remove.. ويمكنك إضافة عمود جديد بضغط الزر Add، حيث ستظهر لك نافذة فرعية تتيح لك إدخال بيانـــات العمود، كما هو موضح في الصورة:



هذه النافذة تتيح لك اختيارين:

١- إنشاء عمود مرتبط بمصدر بيانات Data-bound Column:

هذا الاختيار يكون متاحا فقط إذا كان جدول العرض مرتبطا بمصدر البيانات من خلال الخاصية DataSource.. في هذه الحالة ستعرض القائمة العلوية أسماء الأعمدة المتاحة في مصدر البيانات، وعليك تحديد واحد منها لربط العمود الجديد به.. لاحظ أنك لا تستطيع عرض أعمدة مرتبطة بأكثر من جدول بيانات في نفس جدول العرض.

٢- إنشاء عمود غير مرتبط بمصدر بيانات Unbound Column:

هذا الاختيار متاح دائما، ولو فعّلته فيجب عليك كتابة تفاصيل العمود كما يلي:

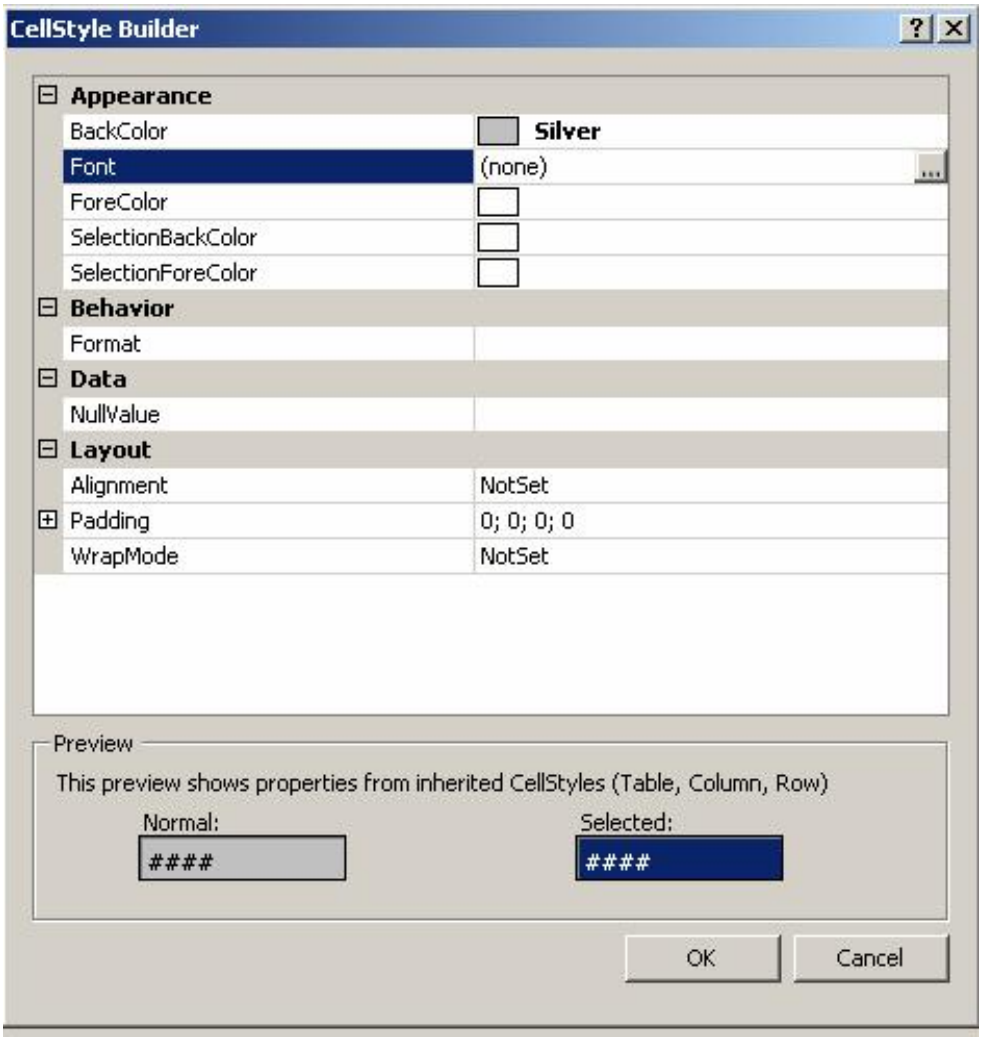
- كتابة الاسم البرمجي للعمود في الخانة Name.. لاحظ أن كل عمود تنشئه في هذه النافذة، يتم تعريف متغير بنفس اسمه على مستوى النموذج، لتستطيع استخدامه مباشرة في التعامل مع العمود، مما يجعل الكود مختصرا.. لهذا عليك اختيار اسم مناسب للعمود يدل على وظيفته، مع وضع بادئة مميزة له (ولتكن Col) كي لا يتعارض مع أي متغيرات أخرى معرفة في البرنامج.. يمكنك مثلا أن تسمي عمود المؤلفين ColAuthors، وعمود الكتب ColBooks.. وهكذا.
- اختيار نوع العمود من القائمة المنسدلة Type.. والعمود النصي DataGridViewTextBoxColumn هو النوع الافتراضي، ويمكنك اختيار أي نوع آخر كعمود القوائم المركبة DataGridViewCombBoxColumn.
- كتابة عنوان العمود (الذي سيعرضه الجدول)، في الخانة Header Text.
- إذا كان العمود خفيا لن يظهر للمستخدم، فأزل علامة الاختيار من مربع الاختيار Visible.

- إذا كان العمود للقراءة فقط ولا يمكن للمستخدم تحرير خاناتـــه، فضـــع علامة الاختيار في مربع الاختيار Read Only.
- إذا كان العمود ثابتا، ولا يمكن للمستخدم تغيير عرضه بالفـــأرة، فضـــع علامة الاختيار في مربع الاختيار Frozen.

وبعد أن تنتهي من إدخال تفاصيل العمود، اضغط الزر Add لإضافته إلى مجموعـــة الأعمدة.. لاحظ أن هذا لن يغلق هذه النافذة، بل سيعيد خاناتها إلى قيمها الافتراضـــية ليتيح لك إنشاء عمود جديد مباشرة.

لاحظ أنك تستطيع إنشاء أعمدة مرتبطة واعدة غير مرتبطة في نفس الجـــدول.. لكـــن هذا سيعقد الأمور عليك، لأن جدول العرض يمحو قيم الأعمدة غير المرتبطة عنـــدما يقوم بتحديث قيم الأعمدة المرتبطة، مثلما يحدث عند ضـــغط رأس العمـــود لترتيـــب الصفوف!.. لهذا بذلنا بعض الجهد في المشروع CustomDataSet للمحافظة علـــى قيم العمود غير المرتبط الذي يعرض أسماء المواد الدراسة في نافذة درجات الطالب.

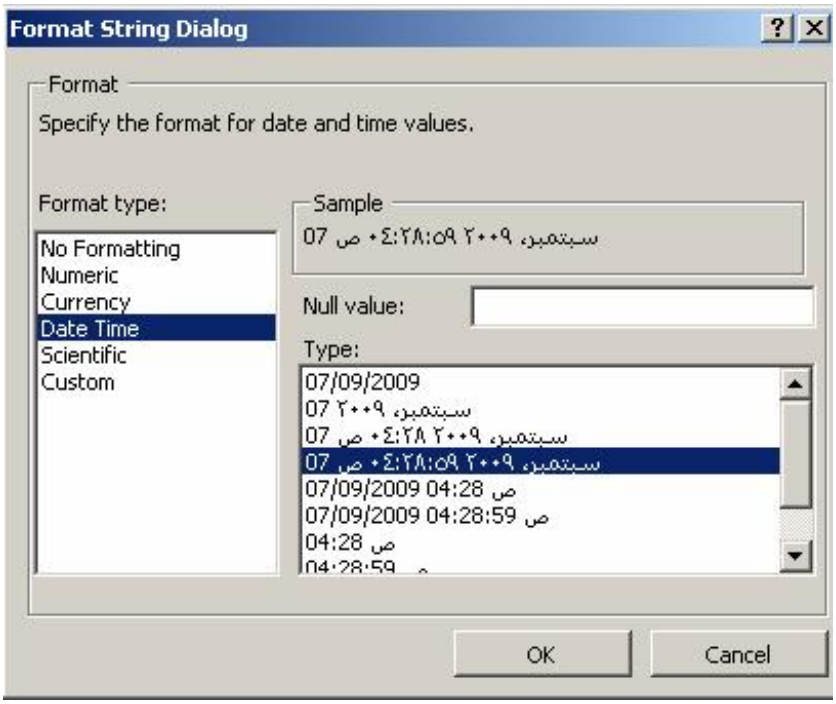
بعد إن تنتهي من إنشاء كل الأعمدة التي تريدها، اضغط Cancel لإغلاق هذه النافذة والعودة إلى النافذة السابقة، حيث ستجد الأعمدة التي أنشـــأتها موجـــودة فـــي قائمـــة الأعمدة، ومن ثم يمكنك تحديد كل منها وتغيير خصائصه كما تريد.. مـــثلا، إذا أردت توسيط النص في خانات العمود، فحدد هذا العمود في القسم الأيسر، ومن القسم الأيمن اختر الخاصية DefaultCellStyle، واضغط الزر الموجود في خانة قيمتها، لعرض نافذة باني طراز الخانة CellStyle Builder، التي تتيح لك تغيير خصـــائص شـــكل الخانة بصورة مرئية.. كما في الصورة:



هذه النافذة تتيح لك وضع قيم كائن طراز الخانة CellStyle بشـــكل مرئـــي وســـهل، وهي متاحة للاستخدام أيضا في نافذة الخصائص مع كل الخصائص التي تتعامل مـــع طراز الخانة، مثل الخاصية DefaultCellStyle.. ويمكنك استخدام هذه النافذة كمـــا يلي:

- اضغط زر الإسدال في خانات خصائص الألوان، لعرض مربع اختيار اللون.
- اضغط زر الإسدال في خانتي المحاذاة Alignment و WrapMode لاختيار القيمة المناسبة من القائمة المنسدلة.
- اضغط زر الانتقال في خانة الخط Font، لعرض مربع اختيار الخط.

- اضغط زر الانتقال في خانة التنسيق Format، لعرض مربع إنشاء نص التنسيق، وهو كما في الصورة:



في هذه النافذة يمكنك اختيار نوع التنسيق من القائمة اليسرى، حيث ستظهر في الجانب الأيمن بعض الاختيارات التي تتيح لك إنشاء صيغة هذا التنسيق، وهي:

- مربع نص القيمة المنعدمة Null Value لتكتب فيه القيمة التي ستستخدم عندما يترك المستخدم الخانة فارغة.
- قائمة تعرض لك صيغ التاريخ المختلفة لتختار منها.
- مربع رقمي NumericUpDown يتيح لك تحديد عدد الخانات العشرية في صيغ الأرقام والعملة والنسب المئوية.

ما يعنيا هنا هو الخاصية Alignment التي تتيح لك اختيار محاذاة النص في خانات العمود.. ولتوسيط النص، اختر القيمة MiddleCentre من القائمة المنسدلة، واضغط Ok للعودة إلى نافذة الأعمدة.

وبعد أن تنتهي من إنشاء كل الأعمدة وضبط خصائصها، اضغط OK لإغلاق النافذة.. ستجد الأعمدة التي أنشأتها قد ظهرت في جدول العرض في وقت التصميم.

عدد الأعمدة ColumnCount:

تقرأ أو تغير عدد الأعمدة الموجودة في جدول العرض.. لاحظ أن وضع الرقم ٠ في هذه الخاصية سيحذف كل أعمدة الجدول، وإذا جعلت لها قيمة أقل من عدد الأعمدة الحالية، فسيتم حذف بعض الأعمدة من نهاية مجموعة الأعمدة Columns، أما إذا وضعت فيها عددا أكبر من عدد الأعمدة الحالي، فسيتم إنشاء أعمدة نصية وإضافتها إلى نهاية مجموعة الأعمدة، وستكون هذه الأعمدة بدون عناوين.. وتسبب هذه الخاصية خطأ إذا حاولت استخدامها مع جدول عرض مرتبط بمصدر بيانات.

السماح للمستخدم بترتيب الأعمدة AllowUserToOrderColumns:

إذا جعلت قيمة هذه الخاصية True، فسيستطيع المستخدم سحب الأعمدة من مواضعها لإعادة ترتيبها.. والقيمة الافتراضية لهذه الخاصية False.

السماح للمستخدم بتغيير حجم الأعمدة AllowUserToResizeColumns:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيستطيع المستخدم سحب الحافة اليمنى للعمود بالفأرة لتكبير عرضه أو تصغيره.

إنتاج الأعمدة تلقائيا AutoGenerateColumns:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم إنشاء الأعمدة تلقائيا عند ربط جدول العرض بمصدر البيانات.

طريقة التحجيم التلقائي للأعمدة AutoSizeColumnsMode:

توضح كيف يتم تغيير عرض الأعمدة تلقائيا تبعا لمحتوياتها، وهي تأخذ إحدى قيم المرقم DataGridViewAutoSizeColumnsMode، وهو يملك نفس قيم المرقم

DataGridViewAutoSizeColumnMode الذي تعرفنا عليه سابقا، ما عدا القيمة NotSet.

رءوس الأعمدة مرئية ColumnHeadersVisible:

إذا جعلت قيمة هذه الخاصية False، فلن يتم عرض الصف الذي يحتوي على رؤوس الأعمدة.. والقيمة الافتراضية لهذه الخاصية هي True.

ارتفاع رءوس الأعمدة ColumnHeadersHeight:

تقرأ أو تغير ارتفاع الصف الذي يحوي رءوس الأعمدة.

طريقة تغيير ارتفاع رءوس الأعمدة ColumnHeadersHeightSizeMode:

تحدد كيفية تغير ارتفاع الصف الذي يحتوي على رءوس الأعمدة، وهي تأخذ إحدى قيم المرقم DataGridViewColumnHeadersHeightSizeMode:

EnableResizing	السماح للمستخدم بتغيير ارتفاع رءوس الأعمدة.
DisableResizing	عدم السماح للمستخدم بتغيير ارتفاع رءوس الأعمدة.
AutoSize	تغيير ارتفاع رءوس الأعمدة تلقائيا ليناسب عناوينها.

عرض الجزء المختفي من أول عمود ظاهر FirstDisplayedScrollingColumnHiddenWidth:

تعيد عرض الجزء المختفي من أول عمود ظاهر على الشاشة حاليا.

رقم أول عمود ظاهر FirstDisplayedScrollingColumnIndex:

تعيد رقم أول عمود ظاهر على الشاشة حاليا.. ويمكنك أيضا تغيير قيمة هذه الخاصية، للانزلاق إلى العمود الذي أرسلت رقمه إليها، ليصير أول عمود ظاهر.. والمثال التالي ينزلق إلى العمود العاشر إن لم يكن ظاهرا على الشاشة:

```
Dgv.FirstDisplayedScrollingColumnIndex = 9;
```

ويؤدي استخدام رقم عمود غير موجود إلى حدوث خطأ في البرنامج.. على سبيل المثال، ستسبب الجملة السابقة خطأ إذا كان عدد الأعمدة أقل من ١٠.

الأعمدة المحددة SelectedColumns:

تعيد مجموعة أعمدة DataGridViewSelectedColumnCollection، وهي مجموعة ترث الفئة BaseCollection وتمثل واجهة القائمة IList، وتحتوي على الأعمدة المحددة حاليا في الجدول.

عمود الترتيب SortedColumn:

تعيد كائن العمود DataGridViewColumn الذي تم ترتيب صفوف جدول العرض تبعا لترتيب خاناته.. وتعيد هذه الخاصية Nothing إذا لم يكن جدول العرض مرتبا.

طراز حواف رؤوس الأعمدة ColumnHeadersBorderStyle:

تحدد شكل إطار خانات العناوين، وهي تأخذ إحدى قيم المرقم DataGridViewHeaderBorderStyle التالية:

None	لا توجد إطارات.
Single	إطار من خط مفرد.
Raised	إطار بارز.
Sunken	إطار غائر.
Custom	إطار مخصص.. هذه القيمة للقراءة فقط، ولا يمكنك وضعها بنفسك، وإنما تتغير تلقائيا عند تغيير قيمة الخاصية AdvancedColumnHeadersBorderStyle.

الطـــــــــراز الافتراضـــــــــي لخانـــــــــات رءوس الأعمـــــــــدة ColumnHeadersDefaultCellStyle:

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle المستخدم مع خانـات رؤوس الأعمدة.. ويمكنك تغيير هذا الطراز بشكل مرئي من نافذة الخصائص، وذلـك بضغط زر الانتقال الموجود في خانة قيمة هذه الخاصية لعرض نافـذة بـاني طـراز الخانة CellStyle Builder.

الطـــــــــراز المتطـــــــــور لحافـــــــــة رؤوس الأعمـــــــــدة AdvancedColumnHeadersBorderStyle:

تعيد كائن الطـراز المتطـور DataGridViewAdvancedBorderStyle الـذي يتعامل مع إطارات خانات عناوين الأعمدة.

طـــــــراز الحافـــــــة المضـــــــبوط للخانـــــــة العلويـــــــة اليســـــــرى AdjustedTopLeftHeaderBorderStyle:

تعيد كائن الطـراز المتطـور DataGridViewAdvancedBorderStyle الـذي يتعامل مع إطارات الخانة العلوية اليسرى في الجدول.

تفعيل الطرازات الشكلية للخانات الرئيسية EnableHeadersVisualStyles:

إذا جعلت قيمة هذه الخاصية True، فسيتم استخدام طراز الحواف وطـراز الخانـات مع رءوس الأعمدة ورءوس الصفوف.. لاحظ أن هذا سيمنع تأثير بعض الخصـائص الموجـــــودة فـــــي الخاصـــــية ColumnHeadersDefaultCellStyles أو RowHeadersDefaultCellStyles، فمثلا: لو غيرت لون خلفية الخانات الرئيسية في تلك الخاصيتين فلن يغير هذا شيئا إلا إذا وضعت القيمـة False فـي الخاصـية EnableHeadersVisualStyles.

كما يمدك جدول العرض بالوسائل التالية للتعامل مع الأعمدة:

عدد الأعمدة المعروضة DisplayedColumnCount:

تعيد عدد الأعمدة التي يراها المستخدم على الشاشة في هذه اللحظة، ولها معامل منطقي، إذا جعلت قيمته True فسيدخل ضمن العدد الأعمدة التي يظهر جزء منها فقط.

تعديل طراز حافة عنوان العمود AdjustColumnHeaderBorderStyle:

تعدل شكل إطار رأس العمود.. وتستقبل هذه الوسيلة المعاملات التالية:

- كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الخاص بالعمود الذي سيتم تعديله.
- كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الذي سيستخدم لحفظ التغييرات البينية التي تحدث لرأس العمود.
- معامل منطقي، أرسل إليه True إذا كان العمود هو أول عمود معروض في الجدول.
- معامل منطقي، أرسل إليه True إذا كان العمود هو آخر عمود مرئي في الجدول.

وتعيد هذه الوسيلة كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الذي يمثل طراز الحافة المعدل.
لاحظ أنك لست مضطرا إلى استخدام هذه الوسيلة يدويا، فجدول العرض يستدعيها تلقائيا لضبط شكل حواف الخانات الرئيسية للأعمدة عند رسمها.

تحجيم العمود تلقائيا AutoResizeColumn:

تغير عرض العمود المطلوب، ليناسب محتويات خاناته.. ولها صيغتان:

١. الصيغة الأولى تستقبل رقم العمود المراد تحجيمه.

٢. الصيغة الثانية تزيد على الصيغة السابقة بمعامل ثان، يستقبل إحدى قيم المرقم DataGridViewAutoSizeColumnMode لتوضح طريقة تحجيم العمود، وقد سبق لنا التعرف عليه.

تحجيم تلقائي للأعمدة AutoResizeColumns:

تغير عرض جميع أعمدة الجدول لتلائم محتويات خاناتها.. ولها صيغتان:

١. الصيغة الأولى بدون معاملات.

٢. الصيغة الثانية تستقبل إحدى قيم المرقم DataGridViewAutoSizeColumnMode لتوضح طريقة تحجيم الأعمدة.

تغيير ارتفاع رؤوس الأعمدة تلقائيا AutoResizeColumnHeadersHeight:

تغير ارتفاع رءوس الأعمدة تلقائيا لتناسب محتوياتها.. ولها صيغتان:

١- الصيغة الأولى بدون معاملات، وهي تغير ارتفاع صف العناوين ليراعي محتويات جميع خاناته.

٢- والصيغة الثانية تستقبل رقم العمود الذي يجب مراعاة محتويات خانة عنوانه عند تغيير ارتفاع الصف.

معرفة مستطيل عرض العمود GetColumnDisplayRectangle:

تعيد كائن مستطيل Rectangle يحتوي على موضع وأبعاد العمود المطلوب، وهي تستقبل معاملين:

- رقم العمود المطلوب.
- معامل منطقي إذا جعلت قيمته True فستعيد هذه الوسيلة المستطيل المحيط بالجزء المعروض من العمود على الشاشة، وإذا جعلته False، فستعيد المستطيل المحيط بكامل العمود حتى لو كان جزء منه غير ظاهر على الشاشة.

إبطال رسم العمود InvalidateColumn:

أرسل إلى هذه الوسيلة رقم العمود، لتقوم بإبطال رسمه في الجدول، مما يجبره على إعادة رسم نفسه من جديد.

كما يمدك جدول العرض بالأحداث التالية للتعامل مع الأعمدة، علما بأن المعامل الثاني e في معظم هذه الأحداث من النوع DataGridViewColumnEventArgs، وهو يمتلك الخاصية Column التي تعيد كائن العمود DataGridViewColumn الذي سبب انطلاق الحدث.. لهذا لن نكرر ذكر هذا في الأحداث، وسنذكر نوع المعامل فقط إذا كان مختلفا:

إضافة عمود ColumnAdded:

ينطلق عند إضافة عمود إلى جدول العرض.

حذف عمود ColumnRemoved:

ينطلق عند حذف عمود من جدول العرض.

تغير رقم عرض العمود ColumnDisplayIndexChanged:

ينطلق عندما تتغير قيمة الخاصية DisplayIndex الخاصة بأحد أعمدة جدول العرض سواء من الكود، أو بسبب سحب المستخدم للعمود من موضعه.

النقر المزدوج على فاصل العمود ColumnDividerDoubleClick:

ينطلق عندما ينقر المستخدم مرتين بالفأرة فوق الخط الفاصل بين عمودين، لتحجيم العمود تلقائيا ليناسب محتويات خاناته.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewColumnDividerDoubleClickEventArgs، وهو يمتلك الخصائص:

	ColumnIndex	تعيد رقم العمود الذي نقره المستخدم بالفأرة.
	Button	تعيد إحدى قيم المرقم MouseButtons التي تخبرك بزر الفأرة الذي ضغطه المستخدم.

	Clicks	تعيد عدد مرات ضغط زر الفأرة.
	Delta	تعيد عدد حركات عجلة الفأرة.
	X	تعيد الموضع الأفقي لمؤشر الفأرة.
	Y	تعيد الموضع الرأسي لمؤشر الفأرة.
	Location	تعيد كائن النقطة Point، الذي يحمل موضع مؤشر الفأرة.
	Handled	إذا جعلت قيمة هذه الخاصية True، فلن تتخذ أية خطوات إضافية لمعالجة الحدث.. هذا معناه إلغاء عملية التحجيم التلقائي للعمود.

ضغط رأس العمود ColumnHeaderMouseClick:

ينطلق عند الضغط بالفأرة على رأس أحد أعمدة جدول العرض.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellMouseEventArgs، وهو يمتلك نفس خصائص الحدث السابق ما عدا الخاصية Handled، كما يمتلك الخاصية RowIndex التي تعيد رقم الصف الذي توجد به الخانة المضغوطة.. لاحظ أن هذه الخاصية ستعيد ١- في هذا الحدث، لأن صف رؤوس الأعمدة لا يدخل ضمن ترقيم صفوف الجدول.

النقر المزدوج على راس العمود ColumnHeaderMouseDoubleClick:

ينطلق عند النقر مرتين بالفأرة على أحد أعمدة الجدول.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellMouseEventHandler كما في الحدث السابق.

تغير حالة العمود ColumnStateChanged:

ينطلق عندما تتغير حالة العمود، كأن يفقد المؤشر Focus Lost.. والمعامل الثاني e من النوع DataGridViewColumnStateChangedEventArgs وهو يمتلك الخاصيتين التاليتين:

	Column	تعيد كائن العمود DataGridViewColumn الذي تغيرت حالته.
	StateChanged	تخبرك بالحالة الجديدة للعمود، وهي تعيد إحدى قيم المرقم DataGridViewElementStates الذي تعرفنا عليه من قبل عند التعرف على الخاصية DataGridViewElement.State.

تغير عرض العمود ColumnWidthChanged:

ينطلق عندما تتغير قيمة الخاصية Width الخاصة بأحد أعمدة جدول العرض، سواء برمجيا أو بواسطة المستخدم.

التعامل مع صفوف جدول عرض البيانات:

يمنحك جدول العرض الخصائص التالية للتعامل مع الصفوف:

عدد الصفوف RowCount:

تقرأ أو تغير عدد صفوف جدول العرض.. ولو وضعت في هذه الخاصية قيمة أكبر من عدد صفوف الجدول، فسيضيف هذا صفوفا جديدة إلى نهاية الجدول، بينما يؤدي وضع قيمة أصغر من عدد صفوف الجدول إلى حذف صفوف من نهاية الجدول، ولو وضعت في هذه الخاصية القيمة صفر فستمحى كل الصفوف.

الصفوف Rows:

تعيد مجموعة الصفوف DataGridViewRowCollection التي تحتوي على كائنات صفوف جدول العرض DataGridViewRow Objects.. وكل صف تضيفه إلى هذه المجموعة يظهر في جدول العرض، وكل صف تحذفه منها يختفي من جدول العرض.. وتبدأ الصفوف في هذه المجموعة بالصف رقم صفر، وهو أول صف بعد صف رؤوس الأعمدة، وتنتهي هذه المجموعة بالصف الجديد إذا كان مسموحا بعرضه في جدول العرض، أو بآخر صف حقيقي يحتوي على بيانات إن كان جدول العرض لا يعرض الصف الجديد.

لاحظ أن صف رؤوس الأعمدة هو الصف رقم -١، لكنك لا تستطيع التعامل معه من خلال هذه الخاصية، وإنما تحصل على الرقم -١ من الوسائل والأحداث التي تخبرك برقم الصف الذي حدث له تغير معين، كما سنرى فيما يلي.. وبدلا من هذا ويمكن التعامل مع أي خانة في صف الرءوس باستخدام الخاصية HeaderCell لكل عمود.. والمثال التالي سيخبرك أن خانة رأس العمود الأول توجد في الصف رقم -١:

```
MessageBox.Show(
   DGAuthors.Columns[0].HeaderCell.RowIndex.ToString( ));
```

ويريك الزر "عكس التحديد" في المشروع DataGridViewAuthorBooks مثالا على كيفية استخدام المجموعة Rows لعكس تحديد الصفوف جدول العرض،وذلك بالمرور على كل صفوف الجدول، وعكس قيمة الخاصية Selected لكل منها.

الصفوف المحددة SelectedRows:

تعيد مجموعة الصفوف المحددة DataGridViewSelectedRowCollection، وهي مجموعة ترث الفئة BaseCollection وتمثل واجهة القائمة IList، وتحتوي على الأعمدة المحددة حاليا في الجدول.. لاحظ أنك لا تستطيع إضافة صفوف إلى هذه المجموعة لأنها للقراءة فقط ولا تملك الوسيلة Add.. لهذا لو أردت تحديد أحد الصفوف، فضع القيمة True في الخاصية Selected الخاصة بهذا الصف.. والكود التالي يحدد الصف الأول:

```
Dgv.Rows[0].Selected = true;
```

لكن هذا لن يزيل تحديد الصفوف المحددة سابقا، لهذا لو أردت فعل هذا، فعليك المرور عبر كل الصفوف المحددة ووضع القيمة false في الخاصية Selected الخاصة بكل منها:

```
while (DGAuthors.SelectedRows.Count != 0)
   Dgv.SelectedRows[0].Selected = false;
Dgv.Rows[0].Selected = true;
```

لاحظ أن إزالة تحديد الصف الأول يحدث مجموعة الصفوف المحددة لإزالته منها، لهذا نستمر في حذف الصف الأول من هذه المجموعة دائما إلى أن تفرغ نهائيا من محتوياتها.

وهناك طريقة أخرى أكثر كفاءة، وهي وضع مجموعة الصفوف المحددة في متغير، وإزالة تحديد كل عناصرها.. صحيح أن المتغير يشير إلى مجموعة الصفوف المحددة مرجعيا، والمفروض أن يرى التغييرات التي تحدث لها، لكن جدول العرض يلغي المجموعة كلها إذا تغير تحديد أي صف وينشئ مجموعة جديدة ويضف إليها الصفوف المحددة، لهذا يظل المرجع الذي وضعناه في المتغير يشير إلى المجموعة

القديمة.. لعل هذا يوضح لك لماذا يكون التعامل مع المجموعة SelectedRows مكلفا جدا إذا كان جدول العرض يحتوي على عدد هائل من الصفوف، فتحديث جدول العرض لهذه المجموعة عملية تتسم بعدم الكفاءة!

وستجد الكود التالي في الزر "تحديد الصف الأول" في المشروع DataGridViewAuthorBooks، وهو يزيل تحديد كل الصفوف المحددة، ثم يحدد الصف الأول في جدول العرض:

```
var SelRows = DGAuthors.SelectedRows;
foreach (DataGridViewRow R in SelRows)
   R.Selected = false;
DGAuthors.Rows[0].Selected = true;
```

قالب الصف RowTemplate:

تقرأ أو تغير كائن الصف DataGridViewRow الذي يستخدم كقالب تستمد منه الصفوف الجديدة خصائصها.. هذا مفيد إذا أردت تغيير شكل كل صفوف الجدول ووضع القيم الافتراضية للصف الجديد، فكل ما عليك هو تعريف كائن صف جديد وضبط خصائصه ثم وضعه في هذه الخاصية.. ويمكنك فعل هذا في وقت التصميم باستخدام نافذة الخصائص، فلو ضغطت العلامة + المجاورة لاسم هذه الخاصية، فستظهر بعض الخصائص الفرعية للصف الذي يعمل كقالب، ما يتيح لك التحكم في القائمة الموضعية للصف وارتفاعه وعرض الفاصل، كما يمنك استخدام الخاصية DefaultCellStyle للتحكم شكل خانات الصف.. وقد فعلنا هذا في جدول العرض الموضوع على النموذج FrmBooks في المشروع DataGridViewAuthorBooks، لجعل لون خلفية الخانات اصفر، ولون الكتابة أحمر.

السماح للمستخدم بإضافة صفوف AllowUserToAddRows:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيعرض جدول العرض صفا إضافيا فارغا في نهاية الجدول، وعند تحرير المستخدم لأي خانة من خاناته يضاف هذا الصف إلى الجدول، ويضاف بعده صف جديد فارغ.

رقم الصف الجديد NewRowIndex:

تعيد رقم الصف الجديد في جدول العرض (آخر صف في الجدول).. لاحظ أن هذه الوسيلة ستعيد -١ إذا لم يكن مسموحا للجدول بعرض صف جديد.

السماح للمستخدم بحذف الصفوف AllowUserToDeleteRows:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتمكن المستخدم من حذف الصف المحدد في جدول العرض بضغط الزر Delete من لوحة المفاتيح.

السماح للمستخدم بتحجيم الصفوف AllowUserToResizeRows:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتمكن المستخدم من تغيير ارتفاع الصف بسحب حافته بالفأرة.

الصف الحالي CurrentRow:

تعيد كائن الصف DataGridViewRow الذي يحتوي على الخانة التي بها المؤشر Focused حاليا.

طريقة التحجيم التلقائي للصف AutoSizeRowsMode:

تحدد كيف سيتم تغيير ارتفاع صفوف جدول العرض تلقائيا، وهي تأخذ إحدى قيم المرقم DataGridViewAutoSizeRowsMode التالية:

None	لا يتم تغيير ارتفاع الصفوف تلقائيا.
AllCells	تغيير ارتفاع كل صف ليناسب محتويات جميع خاناته، بما فيها الخانة الرئيسية Header.
AllCells ExceptHeader	تغيير ارتفاع كل صف ليناسب محتويات جميع خاناته، ما عدا الخانة الرئيسية.
AllHeaders	تغيير ارتفاع كل صف ليناسب محتويات الخانة الرئيسية.
DisplayedCells	تغيير ارتفاع كل صف ليناسب محتويات خاناته المعروضة على الشاشة، بما فيها الخانة الرئيسية.
DisplayedCells ExceptHeaders	تغيير ارتفاع كل صف ليناسب محتويات خاناته المعروضة على الشاشة، ما عدا الخانة الرئيسية.
DisplayedHeaders	تغيير ارتفاع كل صف ليناسب محتويات الخانات الرئيسية المعروضة على الشاشة.

هل الصف الحالي قذر IsCurrentRowDirty:

تعيد True إذا كان الصف الحالي يحتوي على تغييرات لم يتم حفظها في مصدر البيانات.. لاحظ أن التغييرات التي أجراها المستخدم يتم حفظها في مصدر البيانات في الحالات التالية:

- فور مغادرة الصف الحالي إلى صف آخر.
- إذا ضغط المستخدم Ctrl+Enter من لوحة المفاتيح وهو ما زال في الصف الحالي.
- إذا قمت باستدعاء الوسيلة Form.Validate الخاصة بالنموذج الذي يوجد عليه جدول العرض.
- إذا كان جدول العرض مرتبطا بمصدر ربط BindingSource واستدعيت الوسيلة EndEdit الخاصة به.

وقد استخدمنا هذه الخاصية في المشروع CustomDataSet في الحدث CellContentClick، وذلك لحفظ بيانات التلميذ الحالي في مجموعة البيانات قبل عرض درجاته، حتى لا يحدث خطأ عند محاولة التعامل معها.. هذا هو الكود الذي يفعل هذا:

```
if (DgStudents.IsCurrentRowDirty)
{
   DgStudents.EndEdit( ); // إنهاء التحرير
   // إجبار جدول العرض على نقل التغييرات إلى مجموعة البيانات
   this.Validate( );
}
```

رقم أول صف معروض FirstDisplayedScrollingRowIndex:

تعيد رقم أول صف معروض حاليا على الشاشة.. ويمكنك أيضا أن تضع فيها رقم الصف الذي تريد الانزلاق إليه ليصير أول صف معروض.

هل رؤوس الصفوف مرئية RowHeadersVisible:

إذا جعلت قيمة هذه الخاصية False، فلن يظهر العمود الذي يحتوي رؤوس صفوف الجدول.. والقيمة الافتراضية هي True.

عرض رءوس الصفوف RowHeadersWidth:

تقرأ أو تغير عرض العمود الذي يحتوي رؤوس صفوف الجدول.

طريقة تغيير عرض رؤوس الصفوف RowHeadersWidthSizeMode:

توضح كيف يتم ضبط عرض رءوس الصفوف، وهي تأخذ إحدى قيم المرقم DataGridViewRowHeadersWidthSizeMode التالية:

EnableResizing	يمكن للمستخدم تغيير عرض رؤوس الصفوف بسحبها بالفأرة.
DisableResizing	لا يستطيع المستخدم تغيير عرض رؤوس الصفوف.

AutoSizeTo AllHeaders	ضبط عرض رؤوس الصفوف تلقائيا لتناسب محتوياتها.
AutoSizeTo DisplayedHeaders	ضبط عرض رؤوس الصفوف الظاهرة على الشاشة تلقائيا لتناسب محتوياتها.
AutoSizeTo FirstHeader	ضبط عرض عمود رؤوس الصفوف ليناسب محتوى أول خانة فيه.

طراز حافة رؤوس الصفوف RowHeadersBorderStyle:

تتحكم في شكل إطار رؤوس الصفوف، وهي تأخذ إحدى قيم المرقم DataGridViewHeaderBorderStyle الذي تعرفنا عليه من قبل.

الطراز الافتراضي لخانات رؤوس الصفوف RowHeadersDefaultCellStyle:

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل رؤوس الصفوف.

الطراز المتقدم لحواف رؤوس الصفوف AdvancedRowHeadersBorderStyle:

تعيد كائن الطراز المتقدم DataGridViewAdvancedBorderStyle الذي يتحكم في شكل إطار رؤوس الصفوف.

الطراز الافتراضي لخانات الصفوف RowsDefaultCellStyle:

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل خانات صفوف الجدول.

الطـــــــراز الافتراضـــــــي التبـــــــادلي لخانـــــــات الصـــــــفوف AlternatingRowsDefaultCellStyle:

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في شـكل خانات الصفوف الفردية في الجدول.. لاحظ أنك لو وضعت قيمة في هذه الخاصـية، فستتحكم الخاصية RowsDefaultCellStyle في شكل خانات الصـفوف الزوجيـة فقط، بينما تـتحكم الخاصـية AlternatingRowsDefaultCellStyle فـي شـكل خانات الصفوف الفردية.. لكي ترى تأثير هذا، افتح نافذة الخصـائص وحـدد هـذه الخاصية، واغط زر الانتقال الموجود في خانـة قيمتهـا، لعـرض بـاني الطـراز، واستخدمه لجعل لون الخلفية BackColor للصفوف التبادلية فضيا Silver.. سيؤدي هذا إلى أن يعرض الجدول صفا خلفيته بيضاء يليه صف خلفيته فضـية ثـم صـف خلفيته بيضاء وهكذا، كما هو موضح في الصورة:

ID	Author	CountryID	Phone	About
١٢	توفيق الحكيم	٢١		....
١٣	عباس العقاد	٢١		...
١٤	فاروق جويدة	٢١		شاعر مصري معاصر

عرض أخطاء الصفوف ShowRowErrors:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فستظهر أيقونة الخطـأ في خانة رأس الصف الذي توجد فيه أخطاء.

عرض أيقونة التحرير ShowEditingIcon:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فستعرض خانـة رأس الصف أيقونة على شكل قلم، عندما يحرر المستخدم قيمة أي خانة في الصف.

نظام الترتيب SortOrder:

تحدد اتجاه ترتيب صفوف جدول العرض، وهي تأخذ إحدى قيم المرقم SortOrder التي تعرفنا عليها سابقا.

كما يمدك جدول العرض بالوسائل التالية للتعامل مع الصفوف:

تحجيم الصف تلقائيا AutoResizeRow:

تضبط ارتفاع الصف الذي ترسل إليها رقمه كمعامل، ليناسب محتويات خاناته.

وتوجد صيغة أخرى لهذه الوسيلة، تستقبل إحدى قيم المرقم DataGridViewAutoSizeRowMode، التي توضح كيف يتم تغيير ارتفاع الصف، وقد تعرفنا على هذا المرقم سابقا.

تحجيم الصفوف تلقائيا AutoResizeRows:

تضبط ارتفاع جميع صفوف جدول العرض، لتناسب محتويات خاناتها.

وتوجد صيغة أخرى لهذه الوسيلة، تستقبل إحدى قيم المرقم DataGridViewAutoSizeRowMode، التي توضح كيف يتم تغيير ارتفاع كل صف، وقد تعرفنا على هذا المرقم سابقا.

تحجيم عرض رؤوس الصفوف تلقائيا AutoResizeRowHeadersWidth:

تضبط عرض العمود الذي يحتوي على رؤوس الصفوف، وهي تستقبل إحدى قيم المرقم DataGridViewRowHeadersWidthSizeMode، التي توضح كيف يتم تغيير عرض رؤوس الأعمدة، وقد تعرفنا عليه سابقا.

وتوجد صيغة أخرى لهذه الوسيلة، تزيد على الصيغة السابقة بمعامل أول، يستقبل رقم الصف الذي تريد ضبط العرض تبعا لمحتويات خانته الرئيسية.

عدد الصفوف المعروضة DisplayedRowCount:

تعيد عدد الصفوف التي يراها المستخدم على الشاشة في هذه اللحظة، ولها معامل منطقي، إذا جعلت قيمته True فسيدخل ضمن العدد الصفوف التي يظهر جزء منها فقط.

معرفة مستطيل عرض الصف GetRowDisplayRectangle:

تعيد كائن مستطيل Rectangle يحتوي على موضع وأبعاد الصف المطلوب، وهي تستقبل معاملين:

- رقم الصف المطلوب.
- معامل منطقي إذا جعلت قيمته True، فستعيد هذه الوسيلة المستطيل المحيط بالجزء المعروض من الصف على الشاشة، وإذا جعلته False، فستعيد المستطيل المحيط بكامل الصف حتى لو كان جزء منه غير ظاهر على الشاشة.

إبطال الصف InvalidateRow:

أرسل إليها رقم الصف، لتقوم بإبطال رسمه في الجدول، مما يجبره على إعادة رسم نفسه من جديد.

كما يمدك جدول العرض بالأحداث التالية للتعامل مع الصفوف، علما بأن المعامل الثاني e في معظم هذه الأحداث من النوع DataGridViewRowEventArgs، وهو يمتلك الخاصية Row التي تعيد كائن العمود DataGridViewRow الذي سبب انطلاق الحدث:

تغير نص خطأ الصف RowErrorTextChanged:

ينطلق عندما تتغير قيمة الخاصية ErrorText في أحد صفوف الجدول.

تغير عرض رؤوس الصفوف RowHeadersWidthChanged:

ينطلق عند تغيير عرض العمود الذي يحتوي على رؤوس الصفوف، سواء بواسطة المستخدم أو من الكود.

تغير ارتفاع الصف RowHeightChanged:

ينطلق عندما تتغير قيمة الخاصية Height الخاصة بأحد صفوف الجدول، سواء بواسطة المستخدم، أو من الكود.

تغير حالة الصف RowStateChanged:

ينطلق عند تغير حالة الصف، مثلما يحدث عند استقباله المؤشر الضوئي Focus أو فقدانه له.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowStateChangedEventArgs، وله الخاصيتان التاليتان:

Row	تعيد كائن الصف DataGridViewRow الذي تغيرت حالته.	
StateChanged	تخبرك بحالة الصف التي تغيرت، وهي تعيد إحدى قيم المرقم DataGridViewElementStates التي تعرفنا عليها من قبل.	

على سبيل المثال، لو كان الصف الثاني محددا وغادرته لتحدد الصف الأول، فإن الحدث RowStateChanged سينطلق مرتين كالتالي:

١- المرة الأولى بسبب تغير حالة تحديد الصف الثاني، وستشير الخاصية e.Row إلى الصف الثاني، وستكون للخاصية e.StateChanged القيمة Selected.

٢- المرة الثانية بسبب تغير حالة تحديد الصف الأول، وستشير الخاصية e.Row إلى الصف الأول، وستكون للخاصية e.StateChanged القيمة Selected.

ضغط رأس الصف RowHeaderMouseClick:

ينطلق عندما يضغط المستخدم رأس الصف بالفأرة.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellMouseEventArgs الذي تعرفنا عليه سابقا.. وقد استخدمنا هذا الحدث في المشروع لعرض القائمة الموضعية المناسبة عند ضغط المستخدم لرأس الصف.. لاحظ أننا لا نستطيع استخدام قوائم موضعية ثابتة في وقت التصميم، لأننا نعرضها فقط إذا حدث خطأ في حفظ الصف الحالي في قاعدة البيانات، كما أن نوع القائمة يختلف تبعا لنوع الخطأ.. وقد استخدمنا المعامل e.RowIndex للحصول على كائن الصف من مجموعة الصفوف:

```
var R = DgAuthors.Rows[e.RowIndex];
```

ومن ثم استخدمنا الوسيلة GetRowDisplayRectangle الخاصة بجدول العرض لمعرفة موضع هذا الصف، وذلك لاستخدامه في تحديد موضع القائمة الموضعية:

```
var Pos = DgAuthors.GetRowDisplayRectangle(
    e.RowIndex, false).Location;
```

بعد هذا فحصنا نص الخطأ الخاص بالصف.. ونظرا لأن نص الخطأ الذي نضعه في الخاصية ErrorText الخاصة بصف مجموعة البيانات ينتقل كما هو إلى الخاصية ErrorText الخاصة بصف جدول العرض، فقد فحصنا بعض الكلمات التي كتبناها لشرح الخطأ، لنعرف منها نوع هذا الخطأ:

```
if (R.ErrorText.Contains("حذفه"))
   InsertCntxt.Show(DgAuthors, Pos + e.Location);
else if (R.ErrorText.Contains("بتعديله"))
   UpdateCntxt.Show(DgAuthors, Pos + e.Location);
```

لاحظ أن موضع مؤشر الفأرة الذي تعيد الخاصية e.Location يكون منسوبا إلى نقطة رأس الصف.. لهذا علينا أن نجمع عليه موضع رأسي الصف ليكون منسوبا إلى النموذج ككل.

النقر المزدوج على راس الصف RowHeaderMouseDoubleClick:

ينطلق عندما ينقر المستخدم رأس الصف مرتين بالفأرة.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellMouseEventArgs.

النقر المزدوج على فاصل الصف RowDividerDoubleClick:

ينطلق عندما ينقر المستخدم مرتين بالفأرة فوق الخط الفاصل بين صفين، لتحجيم ارتفاع الصف تلقائيا ليناسب محتويات خاناته.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowDividerDoubleClickEventArgs، وهو يمتلك الخصائص التالية:

RowIndex	تعيد رقم الصف الذي نقره المستخدم بالفأرة.	
Button	تعيد إحدى قيم المرقم MouseButtons التي تخبرك بزر الفأرة الذي ضغطه المستخدم.	
Clicks	تعيد عدد مرات ضغط زر الفأرة.	
Delta	تعيد عدد حركات عجلة الفأرة.	
X	تعيد الموضع الأفقي لمؤشر الفأرة.	
Y	تعيد الموضع الرأسي لمؤشر الفأرة.	
Location	تعيد كائن نقطة Point، به موضع مؤشر الفأرة.	
Handled	إذا جعلت قيمتها True، فلن تتخذ أية خطوات إضافية لمعالجة الحدث.. هذا معناه إلغاء عملية التحجيم التلقائي للصف.	

قبل رسم الصف RowPrePaint:

ينطلق قبل رسم أحد صفوف جدول العرض، ليتيح لك التدخل في طريقة رسمه.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowPrePaintEventArgs وهو يمتلك الخصائص التالية:

	RowIndex	تعيد رقم الصف.
	RowBounds	تعيد كائن المستطيل Rectangle الذي يحمل موضع وأبعاد الصف.
	ClipBounds	تقرأ أو تغير كائن المستطيل Rectangle الذي يحمل موضع وأبعاد المساحة التي يجب إعادة رسمها من جدول العرض.. لاحظ أن هذه المساحة قد تختلف عن مساحة الصف، فمثلا قد تخفي نافذة أخرى جزءا من جدول العرض، وعند اختفاء هذه النافذة يحتاج الجزء الذي غطته إلى إعادة رسمه لإنعاشه.
	ErrorText	تعيد نص الخطأ الخاص بالصف.
	Graphics	تعيد كائن الرسوم Graphics الذي سيستخدم لرسم الصف.
	Inherited RowStyle	تعيد طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل خانات الصف.. وهي تماثل الخاصية InheritedStyle الخاصة بكائن الصف، لكن استخدام كائن الصف في حدث الرسم يؤدي إلى بطء تنفيذ البرنامج، لذا يفضل استخدام الخاصية InheritedRowStyle لضمان أحسن أداء.
	IsFirst DisplayedRow	تعيد True إذا كان الصف المراد رسمه هو أول صف ظاهر على الشاشة.
	IsLast VisibleRow	تعيد True إذا كان الصف المراد رسمه هو آخر صف مرئي في جدول العرض.
	State	تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح حالة الصف المراد رسمه.

	PaintParts	تحدد الأجزاء التي يجب على جدول العرض رسمها في كل خانة من خانات الصف، وهي تأخذ إحدى قيم المرقم DataGridViewPaintParts التالية: - None: لا يتم رسم الخانة. - All: رسم كل أجزاء الخانة.. هذه هي القيمة الافتراضية. - Background: رسم خلفية الخانة. - Border: رسم إطار الخانة. - ContentBackground: رسم خلفية محتوى الخانة. - ContentForeground: رسم لون محتوى الخانة. - ErrorIcon: رسم أيقونة الخطأ. - Focus: رسم المستطيل الذي يشر إلى أن الخانة بها العلامة الضوئية Focus. - SelectionBackground: رسم خلفية التحديد. وتستطيع دمج أكثر من قيمة من هذه القيم معا باستخدام المعامل OR.
	Handled	اجعل قيمتها True لتخبر جدول العرض بأن حدث الرسم قد تمت الاستجابة له كليا، ولن ينطلق الحدث RowPostPaint ولا الحدث CellPainting.. لا تستخدم هذه القيمة إلا إذا أردت إلغاء رسم الصف بواسطة جدول العرض، وفي هذه الحالة عليك أن ترسمه أنت بنفسك من داخل هذا الحدث.

كما يمتلك المعامل e الوسائل التالية، لتتيح لك التحكم في رسم مكونات الصف بنفسك:

DrawFocus	ترسم مستطيلا حول المنطقة التي تريدها، وهي تستقبل المعاملين التاليين: - كائن المستطيل Rectangle الذي سيتم رسم الإطار حوله. - معامل منطقي، إذا جعلته True يتم تلوين المستطيل بلون خلفية التحديد SelectionBackColor، وإذا جعلته False يتم تلوين المستطيل بلون خلفية الصف BackColor.. وكلا اللونين يحددهما طراز الصف DataGridViewRow.InheritedStyle.
PaintCells	ترسم خانات الصف، وهي تستقبل معاملين: - كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد رسم خاناتها. - إحدى قيم المرقم DataGridViewPaintParts التي توضح الأجزاء المراد رسمها من الخانات.
PaintCells Background	ترسم خلفية خانات الصف، وهي تستقبل معاملين: - كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد تلوين خاناتها. - معامل منطقي، إذا جعلته True يتم تلوين المستطيل بلون خلفية التحديد SelectionBackColor، وإذا جعلته False يتم تلوين المستطيل بلون الخلفية BackColor.
PaintCells Content	ترسم محتويات خانات الصف، وهي تستقبل كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد رسم محتويات خاناتها.

	PaintHeader	ترسم خانة رأس الصف، ولها صيغتان: ١- الأولى تستقبل معاملا منطقيا، إذا جعلته True يتم تلوين خلفية رأس العمود بلون خلفية التحديد، وإذا جعلته False يتم تلوينه بلون الخلفية. ٢- الصيغة الثانية تستقبل إحدى قيم المرقم DataGridViewPaintParts التي توضح الأجزاء المراد رسمها من خانة رأس الصف.

بعد رسم الصف RowPostPaint:

ينطلق بعد رسم أحد صفوف جدول العرض.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowPostPaintEventArgs وهو يمتلك نفس الوسائل والخصائص كما في الحدث السابق ما عدا الخاصيتين PaintParts و Handled.

ويمكنك استخدام هذا الحدث لرسم مستطيل حول الصف الحالي Current Row في جدول العرض.

لاحظ أن جدول العرض يطلق هذا الحدث كلما كانت هناك ضرورة لإنعاش رسم الصف (كأن يختفي جزء من النافذة، أو يتم تكبيرها أو تصغيرها، أو يتحرك المنزلق فيعرض أو يخفي جزءا من الصف... إلخ).. ونظرا لأن هذا الحدث ينطلق بعدد

الصفوف الظاهرة على الشاشة كلما دعت الحاجة لرسمها، فعليك أن تفحص معاملات الحدث لتتأكد من أن رقم الصف المرسوم هو رقم الصف الحالي:

```
var Row = DataGridView1.CurrentRow; // الصف الحالي
if (Row == null)
   return; // لا يوجد صف محدد حاليا
if (Row.Index != e.RowIndex)
   return;
```

بعد هذا يمكنك أن ترسم مستطيلا حول الصف.. لفعل هذا استخدم الخاصية e.RowBounds لمعرفة إحداثيات المستطيل المحيط بالجزء الظاهر على الشاشة من الصف.. واستخدم الخاصية e.Graphics للحصول على كائن الرسوم الخاص بالصف، لتقوم بواسطته بعملية الرسم.

إلى هنا وكل شيء بسيط.. لكن هناك بعض اللمسات التي يجب وضعها حتى يظهر المستطيل بشكل صحيح:

- فمن الأفضل ألا يحتوي المستطيل على خانة رأس الصف Header.. لهذا سنطرح من عرض المستطيل عرض هذه الخانة، ويمكن معرفته باستخدام الخاصية DataGridView.RowHeadersWidth.
- ليست هناك مشكلة إن كان عرض الصف أكبر من عرض جدول العرض (في هذه الحالة يظهر المنزلق الأفقي)، فكائن الرسوم سيرسم المستطيل داخل حدود جدول العرض، حتى لو حاولت أن تعطيه عرضا كبيرا جدا.. لكن المشكلة تحدث في الحالة العكسية، حينما يكون عرض الصف أصغر من عرض جدول العرض، ففي هذه الحالة سيظهر المستطيل بامتداد جدول العرض كله، وسيكون أكبر من عرض الصف!.. لحل هذه المشكلة علينا أن نطرح من عرض المستطيل الفارق بين عرض الصف وعرض جدول العرض.. لتنفيذ هذا، يجب أن نتأكد أن آخر عمود في الجدول (وهو العمود رقم DataGridView.Columns.Count - 1) معروض على الشاشة حاليا باستخدام الخاصية Displayed الخاصة بكائن العمود.. ثم نستخدم الوسيلة

DataGridView.GetColumnDisplayRectangle لنحصل على المستطيل الذي يحمل أبعاد هذا العمود.. هذه الوسيلة تستقبل رقم العمود، ولها معامل ثان إذا جعلته True، فإنها تعيد أبعاد الجزء المعروض من العمود وتستبعد المساحة المختفية من العمود.. هذا هو ما نريده هنا:

```
var X1 = 0;
var I = DataGridView1.Columns.Count - 1;
if (DataGridView1.Columns[I].Displayed)
{
   var ColRect = DataGridView1.GetColumnDisplayRectangle(
       I, true);
   X1 = ColRect.Left;
}
```

القيمة X1 التي حصلنا عليها في الكود السابق، سنطرحها من عرض المستطيل الذي سنرسمه.

- نظرا لأننا في المشاريع العربية نتعامل مع جدول عرض يظهر من اليمين إلى اليسار، فسنحتاج أيضا إلى تعديل موضع الحافة اليسرى للمستطيل ليبدأ من الحافة اليسرى للعمود الأخير.. أي القيمة X1 التي حصلنا عليها في الكود السابق!
- نظرا لأن جدول العرض قد يحتوي على منزلق رأسي، فيجب أن نطرح عرض هذا المنزلق من X1.. يمكننا معرفة عرض المنزلق الرأسي من معلومات نظام التشغيل باستخدام SystemInformation.VerticalScrollBarWidth، لكن قبل أن نطرح هذه القيمة، يجب أن نعرف أولا إن كان المنزلق الرأسي معروضا أم لا.. يمكننا أن نعرف هذا إذا مررنا على جميع صفوف الجدول، لنجمع ارتفاعاتها فإن كانت أكبر من ارتفاع جدول العرض، فهذا معناه أن هناك حاجة لعرض المنزلق الرأسي.. لكن علينا أيضا أن نفحص قيمة الخاصية DataGridView1.ScrollBars لنتأكد أن عرض المنزلق الرأسي مسموح به.. هذا هو الكود الذي يفعل هذا:

```
var X2 = 0;
var RowsHeight = 0;
if (DataGridView1.ScrollBars == ScrollBars.Both ||
      DataGridView1.ScrollBars == ScrollBars.Vertical)
{
   foreach (DataGridViewRow R in DataGridView1.Rows)
        RowsHeight += R.Height;

   if (RowsHeight > DataGridView1.Height)
      X2 = SystemInformation.VerticalScrollBarWidth + 4;
}
```

والآن دعنا نعرّف القلم الذي سنرسم به، وليكن لونه بنيا:

```
var pen = new Pen(Color.Brown);
var penWidth = 2;
pen.Width = penWidth
```

والآن دعنا نحسب أبعاد المستطيل الذي سنرسمه.. لاحظ أنا سنأخذ سمك خط الرسم في حساباتنا:

```
var X = X1 == 0 ? Rect.Left + (int)(penWidth / 2) : X1;
var Y = Rect.Top + (int)(penWidth / 2);
var W = Rect.Width - penWidth - (X1 - X2) -
          DataGridView1.RowHeadersWidth;
var H = Rect.Height - penWidth;
```

أخيرا لم يبقَ إلا أن نرسم المستطيل حول الصف:

```
e.Graphics.DrawRectangle(pen, X, Y, W, H);
```

لو جربت هذا الكود، فسترى المستطيل يظهر حول الصف الحالي.. لكنك كلما انتقلت من صف إلى آخر، رأيت أجزاء من المستطيل ما زالت حول الصف السابق، بينما قد لا يظهر المستطيل حول الصف الجديد!

نحتاج إذن إلى طريقة لمحو المستطيل تماما من الصف السابق.. يمكن فعل هذا في الحدث RowEnter، الذي ينطلق قبيل دخول صف جديد.. في هذا الحدث تشير الخاصية DataGridView.CurrentRow إلى الصف السابق، بينما تشير الخاصية e.RowIndex إلى رقم الصف الذي سيصير الصف الحالي.. كل ما سنفعله هو

إنعاش كلا الصفين باستدعاء الوسيلة DataGridView.InvalidateRow، التي تستقبل رقم الصف المراد إنعاش رسمه.. هذا هو كود هذا الحدث:

```
var LastRow = DataGridView1.CurrentRow;
if (LastRow != null)
   DataGridView1.InvalidateRow(LastRow.Index);
DataGridView1.InvalidateRow(e.RowIndex);
```

يمكنك الآن تجربة الكود.. ستجده يعمل على ما يرام.

لكن تتبقى مشكلة واحدة فقط، تحدث عند تحريك المنزلق الأفقي (إن كان ظاهرا)، فهذا يؤدي إلى تكرار رسم الإطار، ما يجعل الحافة اليسرى له ترسم أكثر من مرة داخل خانات الصف مع استمرار التحرك.. نحتاج إذن إلى إنعاش المستطيل كلما تحرك المنزلق الأفقي.. يمكن فعل هذا في الحدث DataGridView.Scroll كالتالي:

```
if (e.ScrollOrientation == ScrollOrientation.HorizontalScroll &&
   DataGridView1.CurrentRow != null)
      DataGridView1.InvalidateRow(
         DataGridView1.CurrentRow.Index);
```

لاحظ أن الطريقة التي استخدمناها لمعرفة ظهور المنزلق الرأسي فيها مشكلة، فهي لا تأخذ في الاعتبار مساحة المنزلق الأفقي إن كان ظاهرا.. في الحقيقة أنا أستخدم طريقة مختلفة، فقد أنشأت أداة جديدة ترث الأداة DataGridView، وهذا أتاح لي استخدام الوسائل والخصائص المحمية Protected في فئة جدول العرض، ومنها الخاصية DataGridView.VerticalScrollBar التي تعيد كائن المنزلق الرأسي، ومن خلاله يمكن استخدام الخاصية Visible لمعرفة إن كان ظاهرا أم لا، كما يمكن استخدام الخاصية Width لمعرفة عرضه.

إضافة صفوف RowsAdded:

ينطلق عند إضافة صفوف جديدة إلى جدول العرض برمجيا، أو بواسطة المستخدم (عندما يكتب في الصف الجديد الموجود في نهاية الجدول).

ويمكنك استخدام هذا الحدث لترتيب الصفوف المضافة، وذلك باستدعاء الوسيلة Sort الخاصة بجدول العرض.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowsAddedEventArgs، وهو يمتلك الخاصيتين التاليتين:

RowIndex	تعيد رقم أول صف من الصفوف التي أضيفت إلى الجدول.
RowCount	تعيد عدد الصفوف التي أضيفت إلى الجدول.

إضافة صف بواسطة المستخدم UserAddedRow:

ينطلق عندما يضيف المستخدم صفا جديدا إلى جدول العرض.

حذف صفوف RowsRemoved:

ينطلق عند حذف صف أو مجموعة صفوف من جدول العرض، والمعامل الثاني e الخاص به من النوع DataGridViewRowsRemovedEventArgs وهو يمتلك الخاصيتين التاليتين:

RowIndex	تعيد رقم أول صف من الصفوف التي حذفت من الجدول.
RowCount	تعيد عدد الصفوف التي حذفت من الجدول.

المستخدم يحذف صفا UserDeletingRow:

ينطلق عندما يحاول المستخدم حذف صف، وقبل أن يتم حذف الصف فعليا، ليتيح لك عرض رسالة تحذير للمستخدم أو إلغاء عملية الحذف.. والمعامل الثاني e لهذا الحدث من نوع الفئة DataGridViewRowCancelEventArgs، وهي ترث الفئة CancelEventArgs، مما يعني أنك تستطيع وضع القيمة True في الخاصية e.Cancel لإلغاء حذف الصف.. كما يملك هذا المعامل الخاصية e.Row التي تعيد كائن الصف DataGridViewRow الذي يريد المستخدم حذفه.. وقد فعلنا هذا في المشروع DataGridViewAuthorBooks بالكود التالي:

```
if (MessageBox.Show("هل تريد حذف هذا الصف", "تأكيد الحذف",
    MessageBoxButtons.OKCancel) == DialogResult.Cancel)
    e.Cancel = true;
```

كما استخدمنا هذا الحدث في المشروع DataSetSample.. في هذا المشروع لو حذف المستخدم أحد المؤلفين من الجدول العلوي، فإن كتب هذا المؤلف ستظل في جدول الكتب، ولو حاول حفظ التغييرات في قاعدة البيانات فسيتم رفضها بسبب قيد المفتاح الفرعي Foreign Key Constraint المفروض على العلاقة بين المؤلفين وكتبهم، والتي تمنع وجود كتاب مرتبط بمؤلف تم حذفه!

ويمكنك حل هذه المشكلة على مستوى قاعدة البيانات أو مجموعة البيانات، بضبط خصائص القيد لحذف السجلات الفرعية تتابعيا بمجرد حذف السجل الأصلي Cascade Delete.. لكننا استخدمنا حلا آخر في هذا المشروع، وذلك باستخدام الحدث UserDeletingRow الخاص بجدول عرض المؤلفين، لحذف السجلات المعروضة حاليا في جدول عرض الكتب.

المستخدم حذف صفا UserDeletedRow:

ينطلق بعد حذف المستخدم لأحد صفوف جدول العرض.. لاحظ أن الحدث RowEnter ينطلق أولا قبل انطلاق هذا الحدث، بسبب انتقال المؤشر إلى صف آخر بعد حذف الصف الحالي.. هذا قد يسبب لك مشكلة في بعض المواقف.. مثلا: لا يمكنك استخدام هذا الحدث بدلا من الحدث UserDeletingRow في المشروع DataSetSample، وذلك لأن حذف المؤلف يجعل المؤشر ينتقل إلى مؤلف آخر فينطلق الحدث RowEnter، الذي يعرض كتب المؤلف الحالي في جدول عرض الكتب، ثم ينطلق الحدث UserDeletedRow فيحذف كتب مؤلف موجود، ولا يحذف كتب المؤلف المحذوف!.. لهذا تذكر دائما أن ترتيب انطلاق هذه الأحداث هو:

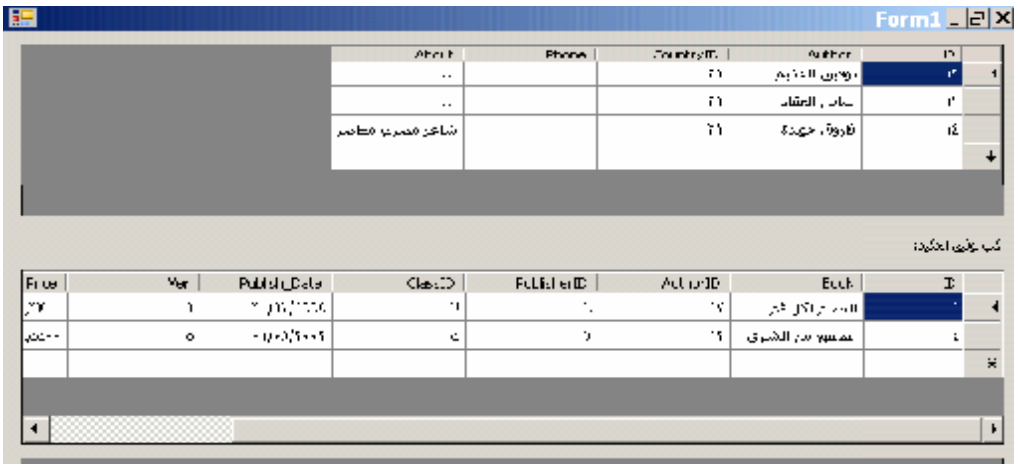
١- UserDeletingRow. ٢- RowEnter. ٣- UserDeletedRow.

دخول الصف RowEnter:

ينطلق عند استقبال أحد صفوف جدول العرض للمؤشر Focus، لكن قبل أن يصير هو الصف الحالي Current Row.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellEventArgs، وهو يمتلك الخاصيتين التاليتين:

RowIndex	تعيد رقم الصف الذي استقبل المؤشر.	
ColumnIndex	تعيد رقم العمود الذي استقبل المؤشر.	

وقد استخدمنا هذا الحدث في المشروع DataGridViewMasterDetails.. في هذا المشروع وضعنا على النموذج جدولي عرض، أحدهما يعرض جدول المؤلفين، والثاني يعرض سجلات الكتب التابعة للمؤلف المحدد حاليا في جدول العرض الأول، كما في الصورة:



ولفعل هذا، اتبعنا الخطوات التالية:

١- استخدمنا الحدث RowEnter لمعرفة الصف الذي تم تحديده:

```
var R = DGAuthors.Rows[e.RowIndex];
```

٢- استخدمنا الخاصية DataGridViewRow.DataBoundItem لنحصل على كائن عرض الصف DataRowView المناظر له في جدول المؤلفين:

```
var DRv = (DataRowView) R.DataBoundItem;
```

```
if (DRv == null)
   return;
```

٣- استخدمنا الوسيلة DataRowView.CreateChildView لإنشاء كائن عرض DataView يحتوي على سجلات الكتب التابعة لسجل هذا المؤلف.. لاحظ أن اسم العلاقة في هذا المشروع بالاسم "كتب المؤلف":

```
var Dv = DRv.CreateChildView("كتب المؤلف");
```

٤- استخدمنا كائن العرض DataView كمصدر بيانات لجدول العرض الثاني، وبهذا يعرض كتب المؤلف المحدد في جدول العرض الأول:

```
DGAuthorBooks.DataSource = Dv;
```

وستجد هذا الكود كاملا في المشروع DataGridViewMasterDetails.

وهناك مشروع آخر يعرض المؤلفين وكتبهم بنفس الطريقة، وهو المشروع DataSetSample، لكن الكود الموجود في الحدث RowEnter في هذا المشروع مختلف قليلا، فهو يعتمد على أن رقم الصف في جدول العرض، هو نفس رقم الصف في كائن العرض الافتراضي DefaultView لجدول المؤلفين، لهذا يحصل على كائن عرض الصف كالتالي:

```
var TblAuthors = Ds.Tables["Authors"];
var DRv = TblAuthors.DefaultView[e.RowIndex];
```

وبعد هذا لا يوجد اختلاف في الكود، فهو يحصل على كائن عرض كتب المؤلف، ويستخدمه كمصدر بيانات لجدول عرض الكتب:

```
var Dv = DRv.CreateChildView("AuthorsBooks");
DgBooks.DataSource = Dv;
```

لاحظ أن هذا الكود سيعمل بشكل صحيح، حتى لو ضغط المستخدم رأي أي عمود لإعادة ترتيب الصفوف رغم أننا نعرف أن تغيير الترتيب يغير رقم كل صف في مجموعة الصفوف.. السبب في عدم حدوث أية مشاكل، أن ترتيب جدول العرض يؤدي إلى ترتيب كائن العرض الافتراضي DefaultView الخاص بجدول المؤلفين، مما يحافظ على نفس ترقيم الصفوف في كل من جدول العرض وكائن العرض، وبالتالي يعمل الكود بشكل صحيح دائما.

مغادرة الصف RowLeave:

ينطلق عندما يفقد أحد صفوف العرض المؤشر، بسبب الانتقال إلى صف آخر، أو بسبب الانتقال من جدول العرض إلى أداة أخرى!.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellEventArgs كما في الحدث السابق.

إجازة الصف RowValidating:

ينطلق عندما يحاول المستخدم مغادرة الصف الحالي في جدول العرض.. هذا يتيح لك فحص القيم التي أدخلها في هذا الصف والتأكد من صحتها.

والمعامل الثاني e لهذا الحدث من نوع الفئة DataGridViewCellCancelEventArgs، وهي ترث الفئة CancelEventArgs، مما يعني أنك تستطيع وضع القيمة True في الخاصية e.Cancel لإجبار المؤشر على البقاء في الصف الحالي، وذلك عندما تكتشف أن به قيمة خاطئة وتريد إجبار المستخدم على تصحيحها أولا.. كما يملك هذا المعامل الخاصيتين e.RowIndex و e.ColumnIndex كما في الحدث RowEnter.

تمت إجازة الصف RowValidated:

ينطلق بعد إجازة بيانات الصف الحالي في جدول العرض.. لاحظ أن مغادرة المستخدم للصف الحالي يؤدي إلى انطلاق الأحداث التالية بالترتيب:

١- الحدث RowEnter الخاص بالصف الذي فقد المؤشر.. في الحقيقة هذا أمر عجيب وغير مبرر، لكنه يحدث!

٢- الحدث RowLeave الخاص بالصف الذي فقد المؤشر.

٣- الحدث Validating الخاص بالصف الذي فقد المؤشر.

٤- الحدث Validated الخاص بالصف الذي فقد المؤشر.

٥- الحدث RowEnter الخاص بالصف الجديد الذي استقبل المؤشر.

إلغاء مشاركة الصف RowUnshared:

ينطلق عندما يتحول صف مشترك Shared Row إلى صف غير مشترك Unshared Row.. ويمكنك استخدام هذا الحدث أثناء اختبار أداء برنامجك، لمعرفة الكود الذي يتسبب في جعل الصفوف تفقد خاصية المشاركة.. هذا مفيد عندما تريد توفير الذاكرة في البرامج التي تعرض كما هائلا من البيانات في جدول العرض.

اللهم ارحم أبي واغفر له وكفر عنه سيئاته

التعامل مع خانات جدول عرض البيانات:

يمنحك جدول العرض الخصائص التالية للتعامل مع خاناته:

المفهرس Indexer:

يعيد كائن الخانة DataGridViewCell الموجودة في العمود المرسل كمعامل أول، والصف المرسل كمعامل ثان.. والمثال التالي يعرض قيمة الخانة الموجودة في العمود الثاني والصف الثالث:

```
MessageBox.Show(DataGridView1[1, 2].Value.ToString( ));
```

كما توجد صيغة أخرى، يمكنك أن ترسل إلى معاملها الأول اسم العمود بدلا من رقمه، مثل:

```
MessageBox.Show(DataGridView1["Author", 2].Value
      .ToString( ));
```

الخانة الحالية CurrentCell:

تعيد كائن الخانة DataGridViewCell المحددة حاليا في جدول العرض.. ويمكنك أيضا أن تضع في هذه الخاصية كائن الخانة التي تريد تحديدها، حيث سينزلق جدول العرض لجعلها ظاهرة على الشاشة.

لاحظ أنك لا تستطيع أن تضع في هذه الخاصية خانة رأس الصف Header أو خانة معطلة Disabled أو خانة موجودة في صف مختفي Hidden، وإلا حدث خطأ.

وتعيد هذه الخاصية Nothing إن لم تكن هناك خانة محددة حاليا، ولو وضعت فيها Nothing فسيزول مربع التحديد من الخانة الحالية.

عنوان الخانة الحالية CurrentCellAddress:

تعيد كائن نقطة Point يحتوي على موضع الخانة الحالية، حيث تمثل الخاصية X رقم الصف الذي توجد به الخانة، والخاصية Y رقم العمود:

```
// رقم الصف الحالي
MessageBox.Show(Dgv.CurrentCellAddress.X.ToString( ));
```

رقم العمود الحالي //
MessageBox.Show(Dgv.CurrentCellAddress.Y.ToString( ));

هل الخانة الحالية في وضع التحرير IsCurrentCellInEditMode:

تعيد True إذا كانت الخانة الحالية في وضع التحرير حاليا.

طريقة التحرير EditMode:

تحدد كيف يمكن للمستخدم بدء تحرير الخانة، وهي تأخذ إحدى قيم المرقم DataGridViewEditMode التالية:

EditOnEnter	تحرير الخانة بمجرد دخولها.. هذا مفيد عند التنقل بين خانات الصف بضغط زر الجدولة TAB، أو التنقل بين خانات العمود بضغط الزر Enter.
EditOnKeystroke	تحرير الخانة عند ضغط أي حرف أبجدي.
EditOn KeystrokeOrF2	تحرير الخانة عند ضغط حرف أبجدي أو الزر F2.. هذه هي القيمة الافتراضية.
EditOnF2	تحرير الخانة عند ضغط الزر F2.
Edit Programmatically	تحرير الخانة برمجيا باستدعاء الوسيلة BeginEdit.

لاحظ أن كل القيم السابقة ما عدا EditProgrammatically تسمح للمستخدم ببدء تحرير الخانة بمجرد نقرها مرتين بالفأرة.

أداة التحرير EditingControl:

تعيد كائن الأداة Control الذي يحمل أداة التحرير المستضافة في الخانة الحالية، إن كانت في وضع التحرير.. وإذا لم تكن الخانة الحالية في وضع التحري، تعيد هذه الخاصية null.

لوحة التحرير EditingPanel:

تعيد كائن اللوحة Panel التي تستخدم لعرض أداة التحرير في الخانة الحالية.. ولهذه الخاصية قيمة دائما حتى لو لم تكن الخانة في وضع التحرير.

هل الخانة الحالية قذرة IsCurrentCellDirty:

تعيد True إذا كانت قيمة الخانة الحالية قد تغيرت ولم يتم حفظ التغيير في مصدر البيانات بعد.

أول خانة معروضة FirstDisplayedCell:

تعيد كائن الخانة DataGridViewCell الذي يمثل أول خانة عادية (ليست رأس صف أو عمود) ظاهرة على الشاشة في جدول العرض.. هذه الخانة تكون أعلى يسار الجدول المعروض من اليسار إلى اليمين، وأعلى يمين الجدول المعروض من اليمين إلى اليسار.. ويمكنك أيضا أن تضع في هذه الخاصية كائن الخانة التي تريد أن ينزلق جدول العرض إليها ليجعلها أول خانة معروضة فيه.

الخانة العلوية اليسرى TopLeftHeaderCell:

تقرأ أو تغير كائن الخانة الرئيسية DataGridViewHeaderCell الذي يمثل الخانة الموجودة في الركن العلوي الأيسر من جدول العرض المعروض من اليسار إلى اليمين، أو الموجودة في الركن العلوي الأيمن في الجدول المعروض من اليمين إلى اليسار.. ويمكنك الاستفادة من هذه الخاصية في التحكم في خصائص هذه الخانة، ككتابة نص بها، أو وضع قامة موضعية لها... إلخ.

الخانات المحددة SelectedCells:

تعيد مجموعة من النوع DataGridViewSelectedCellCollection، التي ترث الفئة BaseCollection وتمثل الواجهة IList، وهي تحتوي على الخانات المحددة حاليا في جدول العرض.

عرض أخطاء الخانات ShowCellErrors:

إذا جعلت قيمة هذه الخاصية True، فستعرض الخانات التي بها أخطاء أيقونة حمراء لتنبيه المستخدم.

عرض تلميحات الخانات ShowCellToolTips:

إذا جعلت قيمة هذه الخاصية True، فسيظهر تلميح على الشاشة عندما يحلق المستخدم بالفأرة فوق أي خانة من خانات الجدول.. لاحظ أن هذا التلميح هو النص الموجود في الخاصية ToolTipText الخاصة بالخانة.

طريقة النسخ إلى لوحة القصاصات ClipboardCopyMode:

تتحكم في كيفية نسخ الخانات المحددة إلى لوحة القصاصات، وهي تأخذ إحدى قيم المرقم DataGridViewClipboardCopyMode التالية:

Disable	لا يسمح بنسخ محتويات الخانات إلى لوحة القصاصات.
EnableWith AutoHeaderText	يتم نسخ محتويات الخانات المحددة، وإن كانت هناك رؤوس أعمدة أو رؤوس صفوف محددة يتم نسخ محتوياتها أيضا.
EnableWithout HeaderText	يتم نسخ محتويات الخانات المحددة فقط، مع تجاهل رؤوس الصفوف والأعمدة.
EnableAlways IncludeHeaderText	يتم نسخ محتويات الخانات المحددة، ونسخ رؤوس الأعمدة ورؤوس الصفوف التي توجد فيها هذه الخانات، بغض النظر عما إذا كانت هذه الرؤوس محددة أم لا.

طراز حواف الخانات CellBorderStyle:

تحدد شكل إطار خانات الجدول، وهي تأخذ إحدى قيم المرقم DataGridViewCellBorderStyle التالية:

None	لا توجد إطارات.
Single	إطار يتكون من خط مفرد.
SingleVertical	حافة رأسية تتكون من خط مفرد.
SingleHorizontal	حافة أفقية تتكون من خط مفرد.
Sunken	إطار غائر.
SunkenVertical	حافة رأسية غائرة.
SunkenHorizontal	حافة أفقية غائرة.
Raised	إطار بارز.
RaisedVertical	حافة رأسية بارزة.
RaisedHorizontal	حافة أفقية بارزة.
Custom	إطار مخصص.. هذه القيمة للقراءة فقط، ولا يمكنك وضعها بنفسك، وإنما تتغير تلقائيا عند تغيير الخاصية AdvancedCellBorderStyle.

الطراز المتقدم لحواف الخانات AdvancedCellBorderStyle:

تعيد كائن الطراز المتقدم DataGridViewAdvancedBorderStyle الذي يتحكم في شكل إطار خانات جدول العرض.

الطراز الافتراضي للخانات DefaultCellStyle:

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في مظهر خانات الجدول.

كما يمدك جدول العرض بالوسائل التالية للتعامل مع الخانات:

تحديد الكل SelectAll:

تحدد كل خانات جدول العرض.

هل جميع الخانات محددة AreAllCellsSelected:

تعيد True إذا كانت جميع خانات جدول العرض محددة، وهي تستقبل معاملا منطقيا، إذا جعلته False فسيتم فحص الخانات المرئية Visible فقط، أما إذا جعلته True فستدخل الخانات الخفية في الاعتبار.

إزالة التحديد ClearSelection:

تجعل كل خانات جدول العرض غير محددة.

ولهذه الوسيلة صيغة ثانية، تتيح لك استثناء خانة معينة من هذه العملية، وهي تستقبل المعاملات التالية:

- رقم العمود الذي توجد به الخانة المستثناة.
- رقم الصف الذي توجد به الخانة المستثناة.
- معامل منطقي، إذا جعلته True فسيتم تحديد الخانة المستثناة إن لم تكن محددة فعلا، وإذا جعلته False فستترك الخانة على حالتها الأصلية كما كانت، سواء كانت محددة أم لا.

وقد استخدمنا هذه الخاصية في الزر "تحديد الصف الثاني" في المشروع DataGridViewAuthorBooks لإزالة تحديد كل الخانات قبل تحديد الصف الثاني:

```
DGAuthors.ClearSelection( );
DGAuthors.Rows[0].Selected = true;
```

لاحظ أن هذا الكود أكفأ من كود الزر "تحديد الصف الأول" في نفس المشروع، ليس فقط لأنه أكثر اختصارا وسهولة، ولكن لأن إزالة تحديد الصفوف المحددة لا يزيل تحديد الخانات المتفرقة المحددة.. جرب أن تضغط الزر CTRL وتضغط أكثر من خانة متفرقة بالفأرة.. لو ضغطت الزر "تحديد الصف الأول" فسيحدد الصف الأول

دون إزالة تحديد هذه الخانات، بينما لو ضغطت الزر الثاني، فسيزيل تحديد جميع هذه الخانات ويحدد الصف الثاني.

الحصول على محتوى لوحة القصاصات GetClipboardContent:

تعيد كائن بيانات DataObject، يحتوي على الخانات المحددة حاليا في جدول العرض، ليمكنك وضعه في لوحة القصاصات مباشرة دون أن تشغل ذهنك بنسخ محتويات الخانات بتنسيق خاص بك.. ولقد شرحنا كائن البيانات DataObject بالتفصيل عندما تعرفنا على لوحة القصاصات Clipboard في كتاب "برمجة نماذج الويندوز".

وتسبب هذه الوسيلة خطأ في البرنامج إذا كانت للخاصية ClipboardCopyMode القيمة Disable.

وما لم تكن في حاجة إلى برمجة أوامر النسخ واللصق الخاصة بك في قائمة رئيسية أو موضعية، فلن تحتاج إلى استدعاء هذه الوسيلة بنفسك، فبمجرد ضغط المستخدم Ctrl+C من لوحة المفاتيح يتم نسخ الخانات المحددة في جدول العرض إلى لوحة القصاصات تلقائيا، حيث يمكنك لصقها في أي برنامج آخر.. على سبيل المثال: عند لصق الخانات في برنامج Notepad يوضع حرف جدولة (أربع مسافات) بين محتوى كل خانة والتي تليها، ويوضع كل صف في سطر جديد.. أما عند لصق هذه الخانات المنسوخة في برنامج Word، فإنه يعرضها في شكل جدول.

معرفة عدد الخانات GetCellCount:

تعيد عدد خانات جدول العرض التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates التي تعرفنا عليها من قبل.

معرفة مستطيل عرض الخانة GetCellDisplayRectangle:

تعيد كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد الخانة المطلوبة، وهي تستقبل المعاملات التالية:

- رقم العمود الذي توجد به الخانة.
- رقم الصف الذي توجد به الخانة.
- معامل منطقي إذا جعلته True فسيحتوي المستطيل على أبعاد الجزء الظاهر من الخانة على الشاشة، وإذا جعلته False فسيحتوي المستطيل على أبعاد الخانة كلها.

إبطال الخانة InvalidateCell:

أرسل إلى هذه الوسيلة كائن الخانة DataGridViewCell التي تريد تعطيل رسمها لتجبر جدول العرض على إعادة رسمها من جديد.

وتوجد صيغة أخرى من هذه الوسيلة تستقبل رقم العمود ورقم الصف اللذين توجد بهما الخانة بدلا من استقبال كائن الخانة.

بدء التحرير BeginEdit:

تضع الخانة الحالية في وضع التحرير.. ولهذه الوسيلة معامل منطقي، إذا جعلته True فسيتم تحديد كل محتويات الخانة في أداة التحرير.

إلغاء التحرير CancelEdit:

تلغي تحرير الخانة الحالية وتعيد الخانة إلى قيمته الأصلية.. وتعيد هذه الوسيلة True إذا نجح إلغاء التحرير.

قبول التحرير CommitEdit:

تحفظ القيمة من أداة التحرير إلى الخانة الحالية، دون إنهاء وضع التحرير.. وهي تستقبل كمعامل إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح الخطأ الذي يمكن أن يحدث أثناء حفظ القيمة، وقد تعرفنا على هذا المرقم من قبل.. وتعيد هذه الوسيلة True إذا نجحت عملية الحفظ.

إنعاش التحرير RefreshEdit:

تحدث القيمة التي تعرضها أداة التحرير، بإعادة قراءة القيمة من الخانة الحالية.. هذا مفيد عندما تتغير قيمة الخانة الحالية (نتيجة تغير مصدر البيانات) بينما يقوم المستخدم بتحرير الخانة، وتريد أنت تنبيهه إلى هذا التغيير.. وتعيد هذه الوسيلة True إذا نجح إنعاش أداة التحرير.

إنهاء التحرير EndEdit:

تنهي تحرير الخانة الحالية وتحفظ القيمة الجديدة في الخانة الحالية، وتعيد True إذا نجح إنهاء التحرير.

وتوجد صيغة أخرى لهذه الوسيلة، تستقبل كمعامل إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح الخطأ الذي يمكن أن يحدث أثناء حفظ القيمة، وقد تعرفنا على هذا المرقم من قبل.

التنبيه بأن الخانة الحالية قذرة NotifyCurrentCellDirty:

تنبه جدول العرض إن كانت الخانة الحالية قد حفظت التغييرات إلى مصدر البيانات أم لا.. وهي تستقبل معاملا منطقيا، إذا جعلته True كان هذا معناه أن الخانة الحالية لم تحفظ التغييرات إلى مصدر البيانات.

وعليك استخدام هذه الوسيلة إذا كنت تتعامل مع أنواع خانات خاصة بك.. لهذا استخدمناها في المشروع DataGridColumnTypes في الحدث الدال على تغير محتويات أدوات التحرير الجديدة التي أنشأناها:

- الحدث OnValueChanged في الفئة CalendarEditingControl.
- والحدث OnTextChanged في الفئتين TreeEditingControl و TreeComboEditingControl.

كما يمدك جدول العرض بالأحداث التالية للتعامل مع الخانات.. والمعامل الثاني e لمعظم هذه الأحداث من النوع DataGridViewCellEventHandler الذي تعرفنا عليه سابقا، وهو يخبرك برقم الصف ورقم العمود الذي توجد به الخانة:

تغير نص خطأ الخانة CellErrorTextChanged:

ينطلق عندما تتغير قيمة الخاصية ErrorText الخاصة بإحدى الخانات.

تغير حالة الخانة CellStateChanged:

ينطلق عندما تتغير حالة إحدى خانات جدول العرض، كأن يوضع بها المؤشر الضوئي أو يزول منها.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellStateChangedEventArgs، وهو يمتلك الخاصيتين التاليتين:

Cell	تعيد كائن الخانة DataGridViewCell التي تغيرت حالتها.	
StateChanged	تخبرك بالحالة الجديدة للخانة، وهي تعيد إحدى قيم المرقم DataGridViewElementStates الذي تعرفنا عليه من قبل.	

تغير الخانة الحالية CurrentCellChanged:

ينطلق عندما تتغير الخانة المحددة حاليا في جدول العرض، بسبب تغير قيمة الخاصية CurrentCell من الكود، أو بسبب انتقال المستخدم من الخانة الحالية إلى خانة أخرى.

تغير قيمة الخانة CellValueChanged:

ينطلق عندما تتغير قيمة إحدى خانات جدول العرض، وذلك بعد انتهاء وضع التحرير.. ويمكنك استخدام هذا الحدث لاتخاذ رد الفعل المناسب بعد تغير قيمة الخانة، مثل فحص القيمة الجديدة للتأكد من صحتها، أو إعادة ترتيب صفوف الجدول إذا كان الجدول مرتبا تبعا لخانات العمود الذي توجد به هذه الخانة.

تغير الحالة القذرة للخانة الحالية CurrentCellDirtyStateChanged:

ينطلق عندما تتغير قيمة الخانة بينما لا زالت في وضع التحرير ولم يتم حفظ قيمتها فعلا.. هذا مفيد في بعض الحالات، مثل الاستجابة لتغيير المستخدم لحالة مربع اختيار CheckBox موضوع في إحدى الخانات دون مغادرة الخانة، ففي هذه الحالة لن ينطلق الحدث CellValueChanged.. وقد استخدمنا هذا الحدث في المشروع DataGridColumnTypes لعرض رسالة تخبر المستخدم بحالة مربع الاختيار بمجرد تغييرها، ودون حتى أن يغادر الخانة.. لفعل هذا، فعلنا ما يلي:

- تأكدنا أولا أن الخانة الحالية في جدول العرض CurrentCell هي خانة مربع اختيار DataGridViewCheckBoxCell، لأن هذا الحدث ينطلق مع أي نوع من أنواع الخانات.
- استخدمنا الخاصية EditedFormattedValue الخاصة بالخانة الحالية لعرض قيمة مربع الاختيار للمستخدم.. ونظرا لأن هذه الخاصية تعيد كائنا Object، فقد حولناه أولا إلى نوع المرقم CheckState لنستطيع عرض اسم حالة الاختيار

باستخدام الوسيلة ToString، ولو لم تفعل هذا، فستظهر أرقام تدل على الحالة مثل (٠ و ١ و ٢).

- ستواجهنا مشكلة هنا، وهي أن هذا الحدث ينطلق بعد أول مرة تتغير فيها حالة الخانة، ومهما وضع المستخدم علامة الاختيار أو أزالها فلن ينطلق هذا الحدث، إلا إذا غادر المستخدم الخانة الحالية أولا ليتم حفظ التغيير.. ولحل هذه المشكلة يمكننا أن نجبر الخانة على حفظ التغييرات باستخدام الوسيلة CommitEdit الخاصة بجدول العرض.
- لكن استدعاء الوسيلة CommitEdit سيؤدي إلى انطلاق الحدث CurrentCellDirtyStateChanged في الحال وقبل تنفيذ باقي الإجراء الحالي، وهو ما سيجعل الرسالة تظهر مرتين.. ولحل هذه المشكلة، عرّفنا متغير منطقيا Boolean اسمه ExitCurrentCellDirtyStateChanged، واستخدمناه كمؤشر Flag، بحيث نجعل قيمته True قبل استدعاء الوسيلة CommitEdit ونعيدها إلى False بعدها مباشرة، وفي بداية الحدث CurrentCellDirtyStateChanged نفحص قيمة هذا المتغير، فإن كانت True نغادر الإجراء في الحال.. وبهذا تظهر الرسالة مرة واحدة فقط.

وستجد هذا الكود كاملا في المشروع DataGridColumnTypes.

بدء تحرير الخانة CellBeginEdit:

ينطلق عند بدء تحرير إحدى خانات جدول العرض.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellCancelEventArgs الذي تعرفنا عليه من قبل، وهو يتيح لك إلغاء عملية التحرير بوضع القيمة True في الخاصية e.Cancel.

ظهور أداة التحرير EditingControlShowing:

ينطلق عند بدء تحرير الخانة الحالية في جدول العرض، وظهور أداة التحرير بها.. والمعامل الثاني e لهذا الحدث من النوع

DataGridViewEditingControlShowingEventArgs، وهـــــو يمتلـــــك الخاصيتين التاليتين:

Control	تعيد كائن الأداة Control الذي يمثل أداة التحرير التي يـتم عرضها حاليا.. ويمكنك تحويل هذا الكـائن إلـى نـوع أداة التحرير الفعلي، واستخدامه لتغيير خصائصها أو وضع القـيم الابتدائية بها.
CellStyle	تعيد كائن طراز الخانة DataGridViewCellStyle، الذي يمكنك من خلاله التحكم في شكل وتنسيق الخانة التـي يـتم تحريرها حاليا.

ومن فوائد هذا الحدث، منحك القدرة على الاستجابة لأحداث أداة التحرير.. على سبيل المثال: إذا أردت أن تمنع المستخدم من كتابة أي شيء ما عدا الأرقـام فـي خانـات العمود رقم ١ في جدول العرض (افترض أن اسمه Dgv)، فيمكنـك اسـتخدام هـذا الحدث لفعل هذا كالتالي:

```
e.Control.KeyPress -= PhoneColumn_KeyPress;
if (Dgv.CurrentCell.ColumnIndex == 1 && e.Control != null)
    e.Control.KeyPress += PhoneColumn_KeyPress;
```

كل ما فعلناه هو إضافة معالج للحدث KeyPress الخاص بـأداة التحريـر (سـتكون مربع نص في الأعمدة النصية).. لاحظ أن نفس الأداة قد تستخدم في تحريـر أعمـدة أخرى، لهذا قمنا بإزالة معالج الحدث في بداية الكود، حتى تعمل باقي الأعمدة بشـكل طبيعي.. وحتى لو لم يكن لديك سوى عمود واحد، فعليك فعل نفس الأمر، لأن إضافة معالج للحدث KeyPress في كل مرة تظهر فيها أداة التحرير، سيؤدي إلـى تكـرار انطلاق الحدث KeyPress عدة مرات، مما سيبطئ البرنامج بمرور الوقت!

أخيرا: هذا هو كود الإجراء المعالج للحدث KeyPress:

```
private void PhoneColumn_KeyPress(object sender,
        KeyPressEventArgs e)
{
   if (!(char.IsDigit(e.KeyChar)) &&  e.KeyChar != (char)Keys.Back)
        e.Handled = true;
}
```

إنهاء تحرير الخانة CellEndEdit:

ينطلق عند انتهاء تحرير الخانة الحالية في جدول العرض.

ضغط الخانة CellClick:

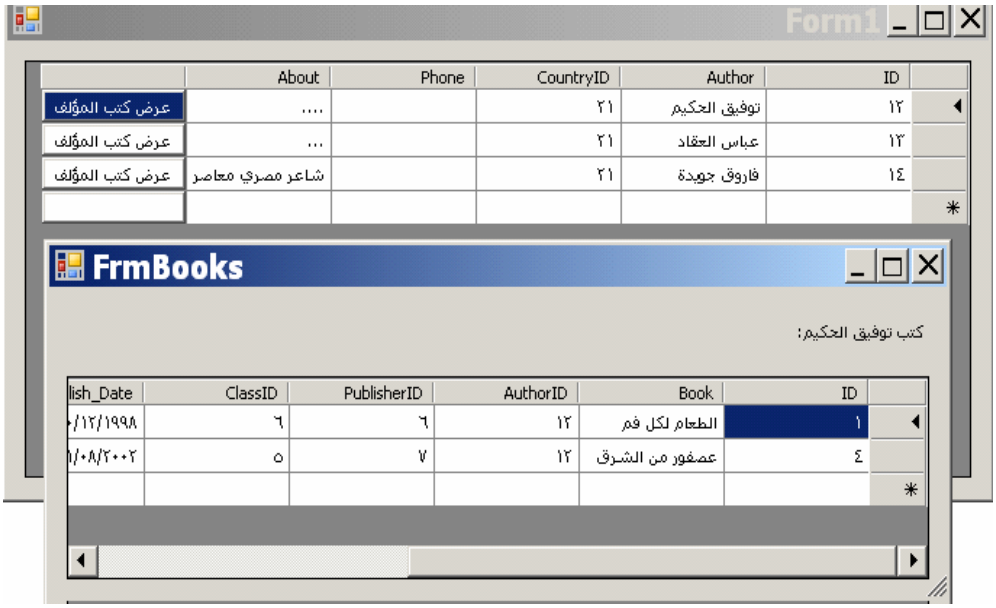
ينطلق عند ضغط أي جزء من الخانة بزر الفأرة بما في ذلك إطارها وهامشها ومحتوياتها وأي أداة موضوعة داخلها.. وينطلق أيضا عند ضغط زر المسافة من لوحة المفاتيح عندما يكون بالخانة زر أو مربع اختيار.

ضغط الخانة بالفأرة CellMouseClick:

مماثل للحدث السابق، ولكنه يتميز عنه بأن المعامل الثاني e من النوع DataGridViewCellMouseEventArgs الذي تعرفنا عليه سابقا، وهو يعطيك معلومات وافية عن موضع الفأرة وحالة أزرارها.

ضغط محتويات الخانة CellContentClick:

ينطلق عند ضغط الأداة التي تستضيفها الخانة.. وقد استخدمنا هذا الحدث في التطبيق DataGridColumnTypes لعرض رسالة للمستخدم عندما يضغط زرا في أحد خانات عمود الأزرار، كما استخدمناه في المشروع DataGridViewAuthorBooks، لعرض نافذة جديدة بها كتب المؤلف الذي تم ضغط الزر الخاص به.



النقر المزدوج على الخانة CellDoubleClick:

ينطلق عند النقر مرتين بالفأرة على أي جزء من الخانة، بما في ذلك إطارها وهامشها ومحتوياتها وأي أداة موضوعة داخلها.

النقر المزدوج على الخانة بالفأرة CellMouseDoubleClick:

مماثل للحدث السابق، ولكنه يتميز عنه بأن المعامل الثاني e من النوع DataGridViewCellMouseEventArgs الذي تعرفنا عليه سابقا، وهو يعطيك معلومات وافية عن موضع الفأرة وحالة أزرارها.

النقر المزدوج على محتويات الخانة CellContentDoubleClick:

ينطلق عند النقر مرتين بالفأرة على الأداة التي تستضيفها الخانة.

هبوط زر الفأرة فوق الخانة CellMouseDown:

يحدث مباشرة بعد ضغط المستخدم لأحد أزرار الفأرة، بينما مؤشرها فوق الخانة، وقبل أن يترك الزر.. والمعامل الثاني e من النوع DataGridViewCellMouseEventArgs.

ارتفاع زر الفأرة فوق الخانة CellMouseUp:

ينطلق بعد ترك المستخدم لزر الفأرة المضغوط، ومؤشرها فوق الخانة.. والمعامل الثاني e من النوع DataGridViewCellMouseEventArgs.

دخول الفأرة إلى الخانة CellMouseEnter:

ينطلق عندما يدخل مؤشر الفأرة إلى حدود الخانة، وينطلق لمرة واحدة فقط إلى أن يغادر المؤشر حدود الخانة.

مغادرة الفارة للخانة CellMouseLeave:

ينطلق عندما يغادر مؤشر الفأرة حدود الخانة.

تحرك الفأرة فوق الخانة CellMouseMove:

ينطلق أثناء تحرك مؤشر الفأرة فوق الخانة.. والمعامل الثاني e من النوع DataGridViewCellMouseEventArgs، وهو ينطلق عدة مرات في كل ثانية إلى أن يتوقف المؤشر عن الحركة، أو يغادر حدود الخانة.

دخول الخانة CellEnter:

ينطلق عندما تتغير الخانة الحالية في جدول العرض بالانتقال إلى خانة أخرى، أو عندما ينتقل المؤشر الضوئي من أداة أخرى إلى جدول العرض.

ويمكن أن ينطلق هذا الحدث مرتين متتاليتين، وذلك إذا كان المؤشر الضوئي في أداة أخرى، وضغط المستخدم خانة غير الخانة التي كانت محددة في جدول العرض.

مغادرة الخانة CellLeave:

ينطلق عندما يغادر المستخدم الخانة الحالية في جدول العرض، وتفقد المؤشر الضوئي.

تجري إجازة الخانة CellValidating:

ينطلق عندما يحاول المستخدم مغادرة الخانة الحالية.. والمعامل الثاني e لهذا الحدث من نوع الفئة DataGridViewCellValidatingEventArgs، وهي ترث الفئة CancelEventArgs، مما يعني أنك تستطيع وضع القيمة True في الخاصية e.Cancel لإجبار المؤشر على البقاء في الخانة الحالية، وذلك عندما تكتشف أن بها قيمة خاطئة وتريد إجبار المستخدم على تصحيحها أولا.. وإضافة إلى هذا، يمتلك هذا المعامل الخصائص التالية:

	ColumnIndex	تعيد رقم العمود الذي توجد به الخانة.
	RowIndex	تعيد رقم الصف الذي توجد به الخانة.
	FormattedValue	تعيد القيمة المنسقة الموجودة في الخانة، والتي عليك التأكد من صحتها.

تمت إجازة الخانة CellValidated:

ينطلق بعد التأكد من صحة القيمة الموجودة في الخانة.

تنسيق الخانة CellFormatting:

ينطلق عندما تحتاج الخانة إلى عرض قيمتها، لتسمح لك بتنسيقها بالشكل الذي تريده.. ويمكنك استخدام الخاصية DataGridViewCellStyle.Format مباشرة لتحديد صيغة

تنسيق الخانة، لكن أحيانا قد لا تجد صيغة مباشرة للتنسيق الذي تريده، وفي هذه الحالة يمكنك استخدام هذا الحدث لوضع القيمة بالشكل الذي تريده في الخانة.. كما أن هذا الحدث يتيح لك تغيير شكل الخانة وليس محتوياتها فقط.

لاحظ أن هذا الحدث ينطلق كلما تم إنعاش رسم الخانة في الجدول، وكلما أردت قراءة قيمتها المنسقة.. هذا معناه أن هذا الحدث ينطلق كثيرا، لهذا عليك ألا تكتب فيه أي كود طويل يستهلك وقتا ملموسا، حتى لا تؤثر على سرعة وأداء البرنامج.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellFormattingEventArgs، وهو يمتلك الخصائص التالية:

ColumnIndex	تعيد رقم العمود الذي توجد به الخانة.
RowIndex	تعيد رقم الصف الذي توجد به الخانة.
CellStyle	تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في مظهرها وتنسيقها.
DesiredType	تعيد كائن النوع Type، الذي يمثل نوع القيمة المنسقة المراد عرضها في الخانة.
Value	تحتوي القيمة الأصلية للخانة، وعليك أن تضع بدلا منها القيمة المنسقة التي تريد عرضها.
Formatting Applied	إذا جعلت قيمتها true، فستخبر الخانة بأن القيمة الموجودة في الخاصية Value هي القيمة المنسقة، وعليها عرضها مباشرة بدون أي عمليات تنسيق إضافية.. والقيمة الافتراضية لهذه الخاصية هي False، وهذا معناه أن على الخانة استخدام خصائص التنسيق الموجودة في الخاصية CellStyle.

تحويل قيمة الخانة CellParsing:

ينطلق بعد انتهاء تحرير قيمة الخانة، وذلك ليسمح لك بتحويل القيمة المنسقة التي أدخلها المستخدم إلى النوع الأصلي لبيانات الخانة.. هذا مفيد عندما تتعامل مع تنسيق خاص بك باستخدام الحدث السابق، وتريد إجراء عملية التحديث العكسية.. ولو لم تستخدم هذا الحدث، فستقوم الخانة بالتحويل التلقائي من النوع المنسق إلى النوع الأصلي.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellParsingEventArgs، وهو يمتلك الخصائص التالية:

ColumnIndex	تعيد رقم العمود الذي توجد به الخانة.
RowIndex	تعيد رقم الصف الذي توجد به الخانة.
InheritedCellStyle	تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في مظهرها وتنسيقها.
DesiredType	تعيد كائن النوع Type، الذي يمثل نوع القيمة الأصلية المراد التحويل إليها.
Value	تحتوي القيمة المنسقة للخانة، وعليك أن تضع بدلا منها القيمة التي تريد حفظها في الخانة.
ParsingApplied	إذا جعلت قيمة هذه الخاصية True، فستخبر الخانة بأن القيمة الموجودة في الخاصية Value هي القيمة الأصلية، وعليها حفظها مباشرة بدون أي عمليات تحويل إضافية.. والقيمة الافتراضية لهذه الخاصية هي False، وهذا معناه أن على الخانة استخدام خصائص التنسيق الموجودة في الخاصية InheritedCellStyle لتحويل القيمة المنسقة إلى القيمة الأصلية.

رسم الخانة CellPainting:

ينطلق عندما يحتاج جدول العرض إلى إعادة رسم إحدى خاناته.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellPaintingEventArgs، وهو يمتلك الخصائص التالية:

ColumnIndex	تعيد رقم العمود الذي يحتوي على الخانة.
RowIndex	تعيد رقم الصف الذي يحتوي على الخانة.
CellStyle	تعيد كائن طراز الخانة DataGridViewCellStyle، الذي يتحكم في شكل وتنسيق الخانة المراد رسمها.
Advanced BorderStyle	تعيد كائن الطراز المتقدم DataGridViewAdvancedBorderStyle الذي يتحكم في شكل إطار الخانة.
CellBounds	تعيد كائن المستطيل Rectangle الذي يحمل موضع وأبعاد الخانة المراد رسمها.
ClipBounds	تقرأ أو تغير كائن المستطيل Rectangle الذي يحمل موضع وأبعاد المساحة التي يجب إعادة رسمها من الخانة.
ErrorText	تعيد نص الخطأ الخاص بالخانة.
Graphics	تعيد كائن الرسوم Graphics الذي سيستخدم لرسم الخانة.
State	تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح حالة الخانة المراد رسمها.
Value	تعيد قيمة الخانة.
Formatted Value	تعيد القيمة المنسقة للخانة.

	PaintParts	تحدد الأجزاء التي يجب على جدول العرض رسمها في الخانة، وهي تأخذ إحدى قيم المرقم DataGridViewPaintParts.
	Handled	إذا جعلت قيمة هذه الخاصية True فستخبر جدول العرض بأن حدث الرسم قد تمت الاستجابة له كليا.. وعليك ألا تستخدم هذه القيمة إلا إذا أردت إلغاء رسم جدول العرض للخانة، وفي هذه الحالة عليك أن ترسمها أنت بنفسك من داخل هذا الحدث.

كما يمتلك المعامل e الوسائل التالية:

	Paint	ترسم جزءا من الخانة، وهي تستقبل معاملين: – كائن المستطيل Rectangle الذي يحمل موضع وأبعاد المساحة التي سيعاد رسمها من الخانة. – إحدى قيم المرقم DataGridViewPaintParts تحدد أجزاء الخانة التي يجب رسمها.
	Paint Background	ترسم خلفية الخانة، وهي تستقبل معاملين: – كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد رسم خلفيتها. – معامل منطقي، إذا جعلته True فسيتم تلوين المستطيل بلون خلفية التحديد SelectionBackColor، وإذا جعلته False فسيتم تلوين المستطيل بلون الخلفية BackColor.
	PaintContent	ترسم محتويات الخانة، وهي تستقبل كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد رسمها.. لاحظ أن عليك رسم خلفية الخانة أولا قبل رسم محتوياتها، ولو فعلت العكس فسيمحو رسم الخلفية محتويات الخانة!

التعامل مع جدول العرض:

يمنحك جدول العرض الخصائص التالية للتحكم في مظهره وأدائه:

لون الخلفية BackgroundColor:

تتحكم في لون خلفية جدول العرض، وهو اللون الظاهر في الجزء الذي لا توجد به خانات.

طراز الحواف BorderStyle:

تحدد شكل إطار جدول العرض، وهي تأخذ إحدى قيم المرقم BorderStyle التالية:

None	جدول العرض بدون أي إطار.
FixedSingle	يحيط بجدول العرض مستطيل أسود يتكون من خط مفرد السماكة.
Fixed3D	إطار مجسم (ثلاثي الأبعاد).

لون الشبكة GridColor:

تتحكم في لون شبكة الخطوط التي تفصل بين الصفوف والأعمدة.. وتؤثر هذه الخاصية فقط على الإطار المفرد، لكنها تكون بلا تأثير عند استخدام إطارات مجسمة ففي هذه الحالة تستخدم الألوان الخاصة بنظام التشغيل.

متعددة التحديد MultiSelect:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيستطيع المستخدم تحديد أكثر من خانة أو صف أو عمود معا في نفس الوقت.

للقراءة فقط ReadOnly:

إذا جعلت قيمة هذه الخاصية True، فلن يستطيع المستخدم تحرير خانات جدول العرض.. والقيمة الافتراضية هي False.

المنزلقات ScrollBars:

توضح أيا من المنزلقين سيعرضه جدول العرض، وهي تأخذ إحدى قيم المرقم ScrollBars التالية:

None	لا تظهر أية منزلقات.
Horizontal	المنزلق الأفقي فقط.
Vertical	المنزلق الرأسي فقط.
Both	المنزلقين الأفقي والرأسي معا.. هذه هي القيمة الافتراضية.

إزاحة الانزلاق الأفقي HorizontalScrollingOffset:

تتحكم في الموضع المبدئي للمنزلق الأفقي.. والكود التالي يجعل المنزلق الأفقي يتحرك ليجعل العمود الثاني في الجدول أول عمود ظاهر على الشاشة:

```
Dgv.HorizontalScrollingOffset = Dgv.Columns[0].Width;
```

إزاحة الانزلاق الرأسي VerticalScrollingOffset:

تعيد موضع المنزلق الرأسي لجدول العرض.. وعلى عكس الخاصية السابقة، لا يمكنك تغيير موضع المنزلق الرأسي بنفسك، وهذا أمر عجيب!

طريقة التحديد SelectionMode:

تتحكم في طريقة تحديد خانات جدول العرض، وهي تأخذ إحدى قيم المرقم DataGridViewSelectionMode التالية:

CellSelect	يمكن تحديد خانة أو أكثر.
FullRow Select	يؤدي ضغط رأس الصف أو أية خانة موجودة به إلى تحديد الصف كله.
FullColumn Select	يؤدي ضغط رأس العمود أو أية خانة موجودة به إلى تحديد العمود كله.. وتسبب هذه القيمة خطأ في البرنامج إذا كانت للخاصية SortMode الخاصة بالعمود القيمة Automatic.
RowHeader Select	يؤدي ضغط رأس الصف إلى تحديد الصف كله، بينما يؤدي ضغط أي خانة إلى تحديد هذه الخانة بمفردها.. هذه هي القيمة الافتراضية.
ColumnHeader Select	يؤدي ضغط رأس العمود إلى تحديد العمود كله، بينما يؤدي ضغط أي خانة إلى تحديد هذه الخانة بمفردها.. وتسبب هذه القيمة خطأ في البرنامج إذا كانت للخاصية SortMode الخاصة بالعمود القيمة Automatic.

وستواجه مشكلة لو غيرت قيمة هذه الخاصية إلى ColumnHeaderSelect أو FullColumnSelect في وقت التصميم.. تعال نجرب هذا بأنفسنا:

- ابدأ مشروعا جديدا اسمه SelectionMode.
- ضع على النموذج جدول عرض، واضغط F4 لعرض نافذة الخصائص.
- اضغط الرابط Add Column الموجود في الهامش السفلي لنافذة الخصائص.. سيفتح هذا نافذة إنشاء عمود جديد مباشرة.. غير خصائص العمود، واضغط Ok لإغلاق النافذة.
- اضغط الرابط Edit Columns الموجود في الهامش السفلي لنافذة الخصائص، لفتح نافذة محرر مجموعة الأعمدة.. غير قيمة الخاصية SortMode الخاصة بالعمود الذي أنشأته إلى NotSortable حتى لا يحدث خطأ عند السماح بتحديد العمود كله، واضغط Ok.

- حدد الخاصية SelectionMode في نافذة الخصائص، ومن القائمة المنسدلة اختر ColumnHeaderSelect.
- اضغط F5 لتشغيل البرنامج.. ستجد أن خطأ حدث في البرنامج منع عرض النموذج!

لعلك مندهش، وتهتف متعجبا: لقد فعلنا كل ما هو مطلوب، فلماذا حدث هذا الخطأ؟!

السبب في هذا، يكمن في الكود الذي يحفظ قيم الخصائص التي تغيرها في وقت التصميم.. هذا الكود مكتوب في الملف From1.Designer.cs، وهو مرتب بحيث يكتب خصائص جدول العرض أولا، تليها خصائص أعمدته.. هذا معناه أن الخاصية SelectionMode تتغير أولا إلى ColumnHeaderSelect، قبل أن تتغير الخاصية SortMode الخاصة بالعمود عن قيمتها الافتراضية، مما يسبب هذا الخطأ.. ولا أنصحك بتغيير الكود الموجود في هذا الملف يدويا، لأن ما ستفعله سيضيع هباء عند أول تعديل تجريه على النموذج في وقت التصميم، لأنه سيؤدي إلى إعادة إنتاج كود ملف التصميم!

لهذا ليس أمامك إلا حل واحد: أن تستخدم نافذة الخصائص لإعادة قيمة الخاصية SelectionMode إلى RowHeaderSelect، وتستخدم حدث تحميل النموذج لتغيير قيمة هذه الخاصية كالتالي:

```
DataGridView1.SelectionMode =
    DataGridViewSelectionMode.ColumnHeaderSelect;
```

الآن لو شغلت البرنامج فسيعمل بشكل صحيح، وسيمكنك تحديد أي عمود بضغط رأسه بالفأرة.

حرف الجدولة القياسي StandardTab:

إذا جعلت قيمة هذه الخاصية True، فسيؤدي ضغط زر الجدولة TAB من لوحة المفاتيح إلى الانتقال من جدول العرض إلى الأداة التالية له على النموذج.. والقيمة الافتراضية لهذه الخاصية False، لهذا يؤدي ضغط زر الجدولة إلى الانتقال بين

خانات جدول العرض، وفي هذه الحالة يمكن أن يضغط المستخدم Ctrl+TAB للانتقال من جدول العرض إلى الأداة التالية له على النموذج.

المؤشر المستخدم UserSetCursor:

تعيد كائن مؤشر الفأرة Cursor.. وهي تختلف عن الخاصية Cursor الموروثة من الأداة الأم Control، في أنها تعيد قيمة مؤشر الفارة الأصلية، مهما تغير شكل المؤشر نتيجة مروره فوق بعض المناطق الخاصة من جدول العرض.

كما يمتلك جدول عرض البيانات الوسيلتين التاليتين:

ترتيب Sort:

ترتب صفوف جدول العرض، ويمكنك استدعاءها في حدث ضغط رأس العمود ColumnHeaderMouseClick، للتحكم في كيفية ترتيب الصفوف.. ولهذه الوسيلة صيغتان:

١- الصيغة الأولى تستقبل كائن واجهة المقارنة IComparer الذي تريد استخدامه في عملية الترتيب (راجع كتاب برمجة إطار العمل).

٢- والصيغة الثانية تستقبل معاملين:

- كائن العمود DataGridViewColumn الذي سيتم ترتيب الصفوف تبعا لقيم خاناته.
- إحدى قيمتي المرقم ListSortDirection التي توضح إن كان الترتيب تصاعديا Ascending أم تنازليا Descending.

اختبار الضغط HitTest:

تخبرك بمعلومات عن موضع معين في جدول العرض، ولها معاملان:

- الإحداثي الأفقي X للموضع.
- الإحداثي الرأسي Y للموضع.

وتعيد هذه الوسيلة كائنا من نوع فئة "معلومات اختيار الضغط" HitTestInfo، وهي فئة معرفة داخل الفئة DataGridView، تمتلك الخصائص التالية:

	Nowhere	تعيد كائن معلومات اختبار الضغط HitTestInfo، يشير إلى نقطة موجودة في منطقة فارغة من جدول العرض (ليست بها خانات عادية أو خانات عناوين).
	Column Index	تعيد رقم العمود الذي توجد فيه نقطة الاختبار.
	ColumnX	تعيد الموضع الأفقي لحافة العمود الذي توجد فيه نقطة الاختبار.
	RowIndex	تعيد رقم الصف الذي توجد فيه نقطة الاختبار.
	RowY	تعيد الموضع الرأسي لحافة الصف الذي توجد فيه نقطة الاختبار.
	Type	تعيد إحدى قيم المرقم DataGridView.HitTestType، لتخبرك بنوع المنطقة التي توجد بها نقطة الاختبار: - None: منطقة فارغة. - Cell: خانة. - ColumnHeader: رأس عمود. - RowHeader: رأس صف. - TopLeftHeader: الخانة الرئيسية العلوية اليسرى. - HorizontalScrollBar: المنزلق الأفقي. - VerticalScrollBar: المنزلق الرأسي.

وقد استخدمنا هذه الوسيلة في حدث حركة الفأرة MouseMove الخاص بجدول العرض في المشروع SelectionMode، لنعرض في اللافتة LbInfo معلومات عن النقطة التي تتحرك فوقها الفأرة.

كما يمتلك جدول عرض البيانات الأحداث التالية:

اكتمال ربط البيانات DataBindingComplete:

ينطلق بعد حدوث تغير في مصدر البيانات، أو بعد تغير قيمة أي من الخصائص التالية: DataSource, DataMember, BindingContext.. هذا مفيد عندما تريد أداء بعض المهام تناسب التغير الذي حدث في البيانات التي يعرضها جدول العرض، مثل تغيير عرض بعض الأعمدة وغير ذلك.

والمعامل الثاني e من النوع DataGridViewBindingCompleteEventArgs، وهو يملك الخاصية ListChangedType التي تخبرك بنوع التغير الذي حدث في مصدر البيانات، وهي تعيد إحدى قيم المرقم ListChangedType، التي تعرفنا عليها عند شرح الحدث IBindingList.ListChanged في فصل عروض البيانات Data Views.

خطأ في البيانات DataError:

ينطلق عند حدوث خطأ في مصدر البيانات، أو عند حدوث خطأ في تنسيق أو تحويل قيمة إحدى خانات الجدول.. والمعامل الثاني e لهذا الحدث من نوع الفئة DataGridViewDataErrorEventArgs، وهي ترث الفئة DataGridViewCellCancelEventArgs التي تعرفنا عليها سابقا، والتي تخبرك برقم العمود ورقم الصف اللذين توجد بهما الخانة التي سببت الخطأ.. وفي حالة حدوث الخطأ في مصدر البيانات الخارجي، فإن الخاصيتين e.ColumnIndex و e.RowIndex تخبرانك بموضع الخانة الحالية في جدول العرض، حتى لو لم تكن مسئولة عن الخطأ أو مرتبطة به بشكل مباشر.. كما يمكنك أيضا استخدام الخاصية

e.Cancel لإلغاء مغادرة الخانة التي سببت الخطأ لو كان الخطأ حدث بسبب تحرير إحدى الخانات.

وإضافة إلى هذه الخصائص الموروثة، يمتلك المعامل e الخصائص التالية:

	Context	تعيد إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح سبب الخطأ، وقد تعرفنا عليها سابقا.
	Exception	تعيد كائن الاستثناء Exception الذي يحتوي معلومات عن الخطأ الذي حدث.
	Throw Exception	إذا جعلت قيمة هذه الخاصية True، فسيتم إطلاق الخطأ في البرنامج بعد انتهاء تنفيذ كود الحدث الحالي.. والقيمة الافتراضية لهذه الخاصية هي False.

ويتصرف جدول العرض كالتالي إذا لم تكتب إجراء يستجيب لهذا الحدث:

١- عند مغادرة صف توجد أخطاء بإحدى خاناته، يعرض جدول العرض رسالة خطأ افتراضية للمستخدم فيها تفاصيل أكثر من اللازم عن الخطأ، وهي رسالة قبيحة حقا ومنفرة!

٢- يلغي القيم التي سببت الخطأ، وينقل المؤشر إلى الصف الذي أراده المستخدم.. هذا مستفز جدا لأنه يزيل القيم التي أدخلها المستخدم ولو كان الخطأ في صف جديد فإنه يحذفه بالكامل!

وللتخلص من هذا الأداء الشنيع، استخدم الكود التالي في هذا الحدث:

```
e.Cancel = true;
```

هذا سيحقق لك فائدتين:

١- منع عرض رسالة الخطأ التلقائية، وتشغيل نغمة تحذير بدلا منها.

٢- إجبار المستخدم على البقاء في نفس الصف دون إلغاء أي قيم أدخلها، مما يتيح له تعديل أخطائه.

انزلاق Scroll:

ينطلق عندما يحرك المستخدم أحد المنزلقين.. والمعامل الثاني e لهذا الحدث من النوع ScrollEventArgs، وهو يمتلك الخصائص التالية:

OldValue	تعيد عددا يدل على موضع المنزلق قبل تحريكه.
NewValue	تعيد عددا يدل على موضع المنزلق بعد تحريكه.
Scroll Orientation	تعيد إحدى قيمتي المرقم ScrollOrientation لتخبرك بوضعية المنزلق الذي سبب الحدث.. وهاتان القيمتان هما: - HorizontalScroll: المنزلق الأفقي. - VerticalScroll: المنزلق الرأسي.
Type	تعيد إحدى قيم المرقم ScrollEventType التي تخبرك بسبب انطلاق الحدث، وهذه القيم هي: - SmallDecrement: تحرك المنزلق خطوة صغيرة إلى الخلف. - SmallIncrement: تحرك المنزلق خطوة صغيرة إلى الأمام. - LargeDecrement: تحرك المنزلق قفزة كبيرة إلى الخلف. - LargeIncrement: تحرك المنزلق قفزة كبيرة إلى الأمام. - ThumbPosition: تحرك مؤشر الانزلاق إلى الأمام أو الخلف. - ThumbTrack: مؤشر الانزلاق يتحرك إلى الأمام أو الخلف. - EndScroll: توقفت عملية الانزلاق. - First: وصل المنزلق إلى أقل قيمة له. - Last: وصل المنزلق إلى أقصى قيمة له.

تغير التحديد SelectionChanged:

ينطلق عندما تتغير الخانات أو الصفوف أو الأعمدة المحددة.

مقارنة الترتيب SortCompare:

ينطلق عند مقارنة قيمتي خانتين في أحد الأعمدة أثناء عملية ترتيب الصفوف، وهو لا ينطلق إذا كان جدول العرض مرتبطا بمصدر بيانات، أو كان جدول العرض في الوضع الافتراضي VirtualMode.

والمعامل الثاني e لهذا الحدث من النوع ، وهو يمتلك الخصائص التالية:

	Column	تعيد كائن العمود DataGridViewColumn الذي يتم ترتيبه.
	CellValue1	تعيد قيمة الخانة الأولى.
	CellValue2	تعيد قيمة الخانة الثانية.
	RowIndex1	تعيد رقم الصف الذي توجد فيه الخانة الأولى.
	RowIndex2	تعيد رقم الصف الذي توجد فيه الخانة الثانية.
	SortResult	ضع في هذه الخاصية نتيجة المقارنة، وهي تأخذ ثلاث قيمة: - عدد أصغر من الصفر إذا كانت الخانة الأولى تسبق الثانية في الترتيب. - عدد أكبر من الصفر إذا كانت الخانة الثانية تسبق الأولى في الترتيب. - صفرا إذا كانت الخانتان متساويتين.
	Handled	اجعل قيمة هذه الخاصية True، إذا أردت ألا يقوم جدول العرض بأية عمليات مقارنة للخانتين بعد هذا.

ويسمح لك هذا الحدث بالتحكم في كيفية مقارنة خانات الجدول إذا كانت تحتوي على قيم مركبة تصعب مقارنتها مباشرة.. لكن لأهمية الفعلية لهذا الحدث تنبع من أنه يتيح لك الفرصة لترتيب صفوف الجدول تبعا لأكثر من عمود.. لكي تفعل هذا، اتبع

الخوارزمية التالية:

١- قارن القيمتين CellValue1 و CellValue2، فإن كانت إحداهما أكبر من الأخرى، فضع ناتج المقارنة في الخاصية SortResult.

٢- إذا كانت القيمتان CellValue1 و CellValue2 متساويتين، فيمكنك استخدام عمود آخر للترتيب على أساسه، وليكن اسمه Col2.

٣- استخدم الخاصيتين RowIndex1 و RowIndex2 لقراءة قيمتي الخانتين الموجودتين في العمود Col2، وقارن بينهما، وضع ناتج المقارنة في الخاصية SortResult.

٤- إذا تساوت القيمتان من العمود Col2 أيضا، فيمكنك استخدام عمود ثالث للترتيب بنفس الطريقة، إذا كنت ترى هذا ملائما.

والكود التالي يوضح لك هذه الخطوات:

```
int Comp = e.CellValue1.ToString( ).CompareTo(
    e.CellValue2.ToString( );
if (Comp != 0)
   e.SortResult = Comp;
else // سنقارن خانتين من عمود آخر
{
   string V1 = DataGridView1["Col2",
           e.RowIndex1].Value.ToString( );
   string V2 = DataGridView1["Col2",
          e.RowIndex2].Value.ToString( );
   e.SortResult = V1.CompareTo(V2);
}
e.Handled = true;
```

تم ترتيبه Sorted:

ينطلق بعد انتهاء ترتيب صفوف جدول العرض.

التعامل مع جدول العرض في الوضع الافتراضي VirtualMode:

يتيح لك جدول العرض التحكم في عرض البيانات به بطريقتك الخاصة، وهو ما يعرف بالوضع الافتراضي VirtualMode، وهو مماثل للوضع الافتراضي الخاص بقائمة العرض ListView التي تعرفنا عليها في كتاب "برمجة نماذج الويندوز".

وفي الوضع الافتراضي، لا يحتفظ جدول العرض إلا بجزء محدود من البيانات جاهزة في الذاكرة، بينما يترك لك مسئولية إمداده بقيم الخانات التي يحتاج لعرضها على الشاشة كلما تحرك المستخدم بالمنزلق الأفقي أو الرأسي.. هذا مفيد في الحالات التالية:

١- عند التعامل مع مصادر بيانات لا يمكن ربطها مباشرة بجدول العرض، كالملفات الثنائية أو مجموعة مختلفة من المصفوفات أو غير ذلك.

٢- عند الحاجة إلى عرض كم هائل من البيانات، ففي الوضع الافتراضي لا يستهلك جدول العرض إلا جزءا محدودا من الذاكرة مهما زاد حجم البيانات التي تريد عرضها، لأنه فعليا لا يحتفظ إلا بالجزء الذي يعرضه على الشاشة.. لهذا يكون الوضع الافتراضي أكفأ وأسرع في عرض البيانات الضخمة.

٣- عندما لا يكون هناك مصدر بيانات، وإنما يتم توليد البيانات بناء على معادلة أو شرط أو ما شابه.. في هذه الحالة يوفر عليك الوضع الافتراضي عناء ملء مصفوفة أو قاعدة بيانات بالبيانات المولدة، ثم ربطها بجدول العرض كمصدر بيانات، ففي الوضع الافتراضي يمكن توليد قيمة الخانة مباشرة في الحدث CellValueNeeded.. والمشروع VirtualModeSample يريك مثالا على هذا، ففيه يعرض الجدول خمسة أعمدة ومليون صف (أي خمسة ملايين خانة)، حيث يحتوي العمود الأول على الأعداد من ١ إلى مليون، بينما يحتوي العمود الثاني على مربع قيم العمود الأول، ويحتوي العمود الثالث على قيم العمود الأول أس ٣ وهكذا، وهو ما يمكن توليده بالمعادلتين التاليتين:

قيمة أي خانة في العمود الأول = رقم الصف الذي توجد به + ١.

قيمة أي خانة في أي عمود آخر = (رقم الصف الذي توجد به + ١) أس (رقم العمود الذي توجد به + ١).

والآن لو جربت تشغيل هذا المشروع، فستجد أنه لن يستغرق وقتا قبل أن يظهر النموذج، وعليه جدول العرض وقد احتوى على الخانات مليئة بالأرقام.. في الحقيقة لم يستغرق ذلك وقتا لأن جدول العرض حسب فقط قيم الخانات التي تراها أمامك، وعندما تسحب المنزلق الرأسي إلى أسفل، فسيقوم بحساب قيم الخانات الجديدة التي ستظهر لك.. هذا يجعل التعامل مع خمسة ملايين خانة عملية في غاية السرعة والكفاءة!

ويمنحك جدول العرض العناصر التالية للتعامل معه في الوضع الافتراضي:

الوضع الافتراضي VirtualMode:

إذا جعلت قيمة هذه الخاصية True، فسيصير جدول العرض في الوضع الافتراضي، وعليك التحكم في كيفية عرض وتحديث قيم خاناته، وكيفية إضافة وحذف صفوفه.

تحديث نص خطأ الصف UpdateRowErrorText:

تجبر جدول العرض على إطلاق الحدث RowErrorTextNeeded لتحديث نص الخطأ الخاص بصف معين، مما يتيح لك تغيير نص الخطأ بنفسك.. ولهذه الوسيلة صيغتان:

١. الصيغة الأولى تستقبل رقم الصف المراد تحديث نص خطئه.. ويمكنك استخدام – ١ للإشارة إلى صف رؤوس الأعمدة.
٢. الصيغة الثانية تحدث نصوص الخطأ لنطاق من الصفوف، لهذا فهي تستقبل معاملين: رقم أول صف ورقم آخر صف في النطاق.

تحديث معلومات ارتفاع الصف UpdateRowHeightInfo:

تجبر جدول العرض على إطلاق الحدث RowHeightInfoNeeded لتحديث ارتفاع صف معين، مما يتيح لك تغيير ارتفاع الصفوف بنفسك.. وتستقبل هذه الوسيلة معاملين:

- رقم الصف المراد تحديث ارتفاعه.. ويمكنك استخدام -١ للإشارة إلى صف رؤوس الأعمدة.
- معامل منطقي، إذا جعلته True فسيتم تحديث ارتفاع كل الصفوف التالية للصف الذي أرسلت رقمه إلى المعامل الأول.

تحديث قيمة الخانة UpdateCellValue:

تجبر جدول العرض على إطلاق الحدث CellValueNeeded، لتحديث قيمة إحدى خانات جدول العرض.. وتستقبل هذه الوسيلة معاملين:

- رقم العمود الذي توجد به الخانة.. ويمكنك استخدام -١ للإشارة إلى عمود رؤوس الصفوف.
- رقم الصف الذي توجد به الخانة.. ويمكنك استخدام -١ للإشارة إلى صف رؤوس الأعمدة.

تحديث نص خطأ الخانة UpdateCellErrorText:

تجبر جدول العرض على إطلاق الحدث CellErrorTextNeeded، لتحديث نص خطأ إحدى خانات جدول العرض.. ولها معاملان:

- رقم العمود الذي توجد به الخانة.. ويمكنك استخدام -١ للإشارة إلى عمود رؤوس الصفوف.
- رقم الصف الذي توجد به الخانة.. ويمكنك استخدام -١ للإشارة إلى صف رؤوس الأعمدة.

قيمة الخانة مطلوبة CellValueNeeded:

ينطلق عندما يرسم جدول العرض إحدى خاناته، ويحتاج منك إلى إمداده بقيمتها.. في هذه الحالة عليك حساب قيمة الخانة كما فعلنا في التطبيق VirtualModeSample، أو الحصول على قيمة الخانة من الملف أو المصفوفة أو المصفوفة القائمة ArrayList التي تحتفظ فيها ببيانات الجدول.

والمعامل الثاني e لهذا الحدث من النـــوع DataGridViewCellValueEventArgs، وهو يمتلك الخصائص التالية:

	ColumnIndex	تعيد رقم العمود الذي يحتوي على الخانة.
	RowIndex	تعيد رقم الصف الذي يحتوي على الخانة.
	Value	تعيد القيمة الموجودة حاليا في الخانة.. ويمكنك أن تضع في هذه الخاصية القيمة الجديدة التي تريد أن تعرضـــها الخانة.

دفع قيمة الخانة CellValuePushed:

ينطلق عندما يغير المستخدم قيمة إحدى خانات الجدول في وضعه الافتراضي، ليمكنك من حفظ القيمة الجديدة في وسيط التخزين الخاص بك (ملف أو مصـــفوفة.. إلـــخ).. والمعامل الثاني e من النوع DataGridViewCellValueEventArgs، كمـــا فـــي الحدث السابق.

صف جديد مطلوب NewRowNeeded:

ينطلق بمجرد انتقال المستخدم إلى الصف الجديد (الصف الأخير) في جدول العـــرض في وضعه الافتراضي.. هذا يتيح لك وضع القيم الافتراضية في هذا الصف.. لاحـــظ أن تحريـــر المســـتخدم لأيـــة خانـــة فـــي هـــذا الصـــف سيســـتدعي الحـــدث CellValueNeeded لكل خانـــات الصـــف، ولـــو جربـــت هـــذا فـــي المشـــروع VirtualModeSample، فستجد أن خانات الصف الجديد قد امتلأت بكـــل الأرقـــام المحسوبة.

القيم الافتراضية مطلوبة DefaultValuesNeeded:

ينطلق عند إضافة صف جديد فارغ إلى نهاية جدول العرض، في وضعه الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.

والمعامل الثاني لهذا الحدث من النوع DataGridViewRowEventArgs الذي تعرفنا عليه سابقا، ويمكنك استخدام الخاصية e.Row لملء خانات الصف بالقيم الافتراضية، مع ملاحظة أن هذه القيم ستحفظ مباشرة في مصدر البيانات، أو ستؤدي إلى انطلاق الحدث CellValuePushed في الوضع الافتراضي لتتيح لك حفظها في وسيط التخزين الخاص بك.

إلغاء تحرير الصف CancelRowEdit:

ينطلق عند إلغاء تحرير أحد صفوف جدول العرض في وضعه الافتراضي.

والمعامل الثاني e لهذا الحدث من النوع QuestionEventArgs، وهي يمتلك الخاصية المنطقية Response، التي إذا جعلتها True كان هذا معناه الموافقة على تنفيذ الحدث (إلغاء التحرير)، إما إذا جعلتها False كان هذا معناه رفض تنفيذ الحدث (وهذا معناه استمرار عملية التحرير وعدم إلغائها).

الحالة القذرة للصف مطلوبة RowDirtyStateNeeded:

ينطلق عندما يريد جدول العرض معرفة إن كان الصف الحالي قد حفظ التغييرات التي حدثت في خاناته أم لا.. والمعامل الثاني e لهذا الحدث من النوع QuestionEventArgs، مع ملاحظة أنك إذا جعلت للخاصية e.Response القيمة True (وهي القيمة الافتراضية)، كان هذا معناه أن الصف لم يحفظ التغييرات بعد، لهذا سينطلق الحدث CancelRowEdit إذا حاول المستخدم إلغاء عملية التحرير بضغط الزر Esc.. إما إذا جعلت قيمة هذه الخاصية False، كان هذا معناه أن الصف قد حفظ التغييرات، ولن ينطلق الحدث CancelRowEdit.

نص خطأ الصف مطلوب RowErrorTextNeeded:

ينطلق عندما يحتاج جدول العرض إلى النص الذي يحتوي على الأخطاء التي حدثت في الصف الحالي، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. والمعامل الثاني e لهذا الحدث من النوع

DataGridViewRowErrorTextNeededEventArgs، وهو يمتلك الخاصيتين التاليتين:

	RowIndex	تعيد رقم الصف.
	ErrorText	ضع في هذه الخاصية نص الخطأ الخاص بالصف.

نص خطأ الخانة مطلوب CellErrorTextNeeded:

ينطلق عندما يحتاج جدول العرض إلى النص الذي يحتوي على الأخطاء التي حدثت في الخانة الحالية، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellErrorTextNeededEventArgs، وهو يمتلك الخصائص:

	ColumnIndex	تعيد رقم العمود الذي توجد به الخانة.
	RowIndex	تعيد رقم الصف الذي توجد به الخانة.
	ErrorText	ضع في هذه الخاصية نص الخطأ الخاص بالخانة.

نص تلميح الخانة مطلوب CellToolTipTextNeeded:

ينطلق عندما يحتاج جدول العرض إلى نص التلميح الخاص بإحدى خاناته، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellToolTipTextNeededEventArgs، وهو يمتلك الخصائص:

	ColumnIndex	تعيد رقم العمود الذي توجد به الخانة.
	RowIndex	تعيد رقم الصف الذي توجد به الخانة.
	ToolTipText	ضع في هذه الخاصية نص تلميح الشاشة الخاص بالخانة.

شريط القائمة الموضعية للصف مطلوب RowContextMenuStripNeeded:

ينطلق عندما يضغط المستخدم بزر الفأرة الأيمن على أحد الصفوف طالبا عرض القائمة الموضعية، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. هذا مفيد عندما تريد تغيير بعض عناصر القائمة الموضعية تبعا لمحتويات الصف أو حالته.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowContextMenuStripNeededEventArgs، وهو يمتلك الخاصيتين التاليتين:

	RowIndex	تعيد رقم الصف.
	Context MenuStrip	ضع في هذه الخاصية كائن رق القائمة الموضعية ContextMenuStrip التي تريد عرضها.

شريط القائمة الموضعية للخانة مطلوب CellContextMenuStripNeeded:

ينطلق عندما يضغط المستخدم بزر الفأرة الأيمن على إحدى الخانات طالبا عرض القائمة الموضعية، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. هذا مفيد عندما تريد تغيير بعض عناصر القائمة الموضعية تبعا لمحتويات الخانة أو حالتها.

ولو لم تستجب لهذا الحدث، فإن الجدول سيعرض القائمة الموضعية التي استخدمتها في الحدث RowContextMenuStripNeeded.

والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellErrorTextNeededEventArgs، وهو يمتلك الخصائص:

	ColumnIndex	تعيد رقم العمود الذي توجد به الخانة.
	RowIndex	تعيد رقم الصف الذي توجد به الخانة.
	Context MenuStrip	ضع في هذه الخاصية كائن رق القائمة الموضعية ContextMenuStrip التي تريد عرضها.

معلومات ارتفاع الصف مطلوبة RowHeightInfoNeeded:

ينطلق عندما يحتاج جدول العرض إلى تغيير ارتفاع أحد الصفوف، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. هذا مفيد عندما تريد تغيير ارتفاع بعض الصفوف بعد ترتيبها.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowHeightInfoNeededEventArgs، وهو يمتلك الخصائص التالية:

	RowIndex	تعيد رقم الصف.
	Height	ضع في هذه الخاصية ارتفاع الصف.
	MinimumHeight	ضع في هذه الخاصية أقل ارتفاع مسموح به للصف.

دفع معلومات ارتفاع الصف RowHeightInfoPushed:

ينطلق عندما يغير المستخدم ارتفاع أحد الصفوف، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.

والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowHeightInfoPushedEventArgs، وهو يمتلك الخصائص التالية:

	RowIndex	تعيد رقم الصف.
	Height	تعيد ارتفاع الصف.
	MinimumHeight	تعيد أقل ارتفاع مسموح به للصف.
	Handled	إذا جعلت قيمة هذه الخاصية True، فلن يتغير ارتفاع الصف، وسيظل بنفس الارتفاع السابق قبل أن يحاول المستخدم تغييره.

تحسين أداء جدول العرض:

رأينا في هذا الفصل، كيف أن جدول العرض أداة غنية تمتلك قدرات هائلة.. للأسف، هناك عيب في هذا الأمر، يظهر عند عرض عدد ضخم من السجلات في جدول العرض، فكل تلك الفئات التي تتعامل مع الأعمدة والصفوف والخانات بكل ما تحتويه من خصائص، تحتاج إلى مساحة كبيرة في الذاكرة، مما يقلل من كفاءة جدول العرض عند التعامل مع عدد ضخم من السجلات.. في هذه الحالة مثلا، قد يكون تحريك المنزلق لعرض صفوف معينة عملية ثقيلة وبطيئة، وكذلك عرض قائمة موضعية، وما إلى ذلك، إضافة إلى بطء البرنامج ككل بسب العبء على الذاكرة!

ولحسن الحظ، لم يقف مصممو هذه الأداة العملاقة عاجزين أمام هذا العيب، فوضعوا بعض المعايير التي تساعد على تحسين أداء جدول العرض عند التعامل مع عدد ضخم من الصفوف والأعمدة.. ومنها:

١- لتغيير شكل الخانات، استخدم الخاصية DataGridView.DefaultCellStyle، فأي تغيير في هذه الخاصية ترثه جميع الخانات التي ليس لها طراز خاص بنفسها، مما يجعل استهلاك الذاكرة أقل ما يمكن مهما كان عدد الخانات هائلا.. ولا تلجأ إلى تغيير طراز كل خانة على حدة من خلال الخاصية DataGridViewCell.Style إلا في أضيق الحدود، لأن كائن طراز كل خانة يستهلك مساحة من الذاكرة.

٢- لا تغير طراز الخانة من خلال قالب الصف DataGridView.RowTemplate، لأن هذا سينشئ نسخة خاصة من كائن الطراز لكل صف في الجدول، وسيكون هذا عبئا كبيرا إذا كان في الجدول آلاف الصفوف.

٣- إذا كانت بعض الخانات تعرض البيانات بتنسيق خاص، فلا تغير التنسيق من كائن الطراز الخاص بالصفوف أو الأعمدة أو الخانات، بل استخدم الحدث CellFormatting لتغيير تنسيق الخانة عند عرضها.. هذا سيوفر مساحة الذاكرة، ولن يستهلك وقتا ملموسا في التنفيذ، لأن هذا الحدث ينطلق فقط للخانات التي تظهر للمستخدم على الشاشة، وليس كل الخانات.

٤- عند قراءة طراز الخانة، استخدم الخاصية InheritedStyle بدلا من الخاصية Style، لأن الخاصية Style تنشئ كائن طراز جديد إذا حاولت قراءتها وهي فارغة!

٥- استخدم الخاصية ContextMenuStrip لوضع قائمة موضعية لجدول العرض ككل، ولا تضع قائمة موضعية لكل صف أو عمود أو خانة على حدة.

٦- لا تضع قائمة موضعية لقالب الصف DataGridView.RowTemplate، لأن هذا سينشئ نسخة من القائمة الموضعية لكل صف في الجدول.

٧- إذا كنت تحتاج إلى قائمة موضعية مختلفة لكل صف، فاستخدم الحدث RowContextMenuStripNeeded لعرض القائمة الموضعية للصف عند الحاجة.

٨- إذا كنت تحتاج إلى قائمة موضعية مختلفة لبعض الخانات، فاستخدم الحدث CellContextMenuStripNeeded لعرض القائمة الموضعية للخانة عند الحاجة.

٩- لا تستخدم التحجيم التلقائي Auto-Sizing على مستوى جدول العرض إذا كان يحتوي على عدد هائل من الخانات، لأن حساب أنسب عرض وارتفاع للصفوف والأعمدة، يستلزم إجراء عمليات قياس لأبعاد محتويات جميع الخانات!.. وإذا كان التحجيم التلقائي مهما لك، فيمكنك أن تنفذه على الخانات المعروضة فقط، وذلك بوضع القيمة DisplayedCells أو القيمة DisplayedCellsExceptHeaders في الخاصية AutoSizeMode، وإذا أردت تحجيم رءوس الصفوف تلقائيا فاستخدم القيمة AutoSizeToDisplayedHeaders، أو AutoSizeToFirstHeader.. لكن الأفضل على الإطلاق، منع التحجيم التلقائي نهائيا، وقيامك بتنفيذه برمجيا، بكتابة الكود الذي يقيس عرض وارتفاع محتويات الخانات في الأحداث التالية:

- UpdateRowHeightInfo.
- Scroll.

- RowDividerDoubleClick.
- ColumnDividerDoubleClick.

١٠- عند التعامل مع الأعمدة والصفوف والخانات المحددة، استخدم المجموعات SelectedColumns و SelectedRows و SelectedCells بحذر، لأن كفاءتها تكون سيئة إذا كان الدول يحتوي على عدد ضخم من الخانات.

١١- استخدم الوسيلة DataGridView.GetCellCount لمعرفة عدد الخانات المحددة، بدلا من استخدام الخاصية SelectedCells.Count.

١٢- استخدم الوسيلة Rows.GetRowCount لمعرفة عدد الصفوف المحددة، بدلا من استخدام الخاصية SelectedRows.Count.

١٣- استخدم الوسيلة Columns.GetColumnCount لمعرفة عدد الأعمدة المحددة، بدلا من استخدام الخاصية SelectedColumns.Count.

١٤- تجنب السماح للمستخدم بتحديد الخانات بصورة منفردة، وبدلا من هذا ضع في الخاصية SelectionMode القيمة FullRowSelect أو FullColumnSelect.

١٥- حافظ على الصفوف المشتركة Shared Rows بقدر الإمكان.. ولكن.. ما هي الصفوف المشتركة؟.. هذا هو موضوع الفقرة التالية.

الصفوف المشتركة Shared Rows:

فكرة هذه التقنية بسيطة، فكل صف جديد يتم إنشاؤه في جدول العرض يأخذ خصائصه الشكلية من قالب الصفوف RowTemplate، لهذا لا داعي لأن نحجز له مساحة كاملة في الذاكرة لنكرر فيها نفس البيانات المشتركة مع القالب.. هذا يوفر مساحة كبيرة في الذاكرة، خاصة إذا كان عدد صفوف جدول العرض ضخما.. وتلغى مشاركة الصف إذا استخدمت أية خاصية لتغيير طريقة عرضه.. بل إن مجرد تعامل المستخدم مع أي خانة في الصف يلغي مشاركته!

لهذا لا تفيدك تقنية مشاركة الصفوف، إلا إذا كان جدول البيانات يحتوي على عدد هائل من الخانات، ولا يتوقع أن يتعامل المستخدم مباشرة إلا مع عدد قليل منها.
كما أن هذه التقنية غير مفيدة إذا كان جدول العرض لا يرتبط بمصدر بيانات، لأن وضع أي قيمة في الخانة يلغي مشاركة الصف، وهذا منطقي، لأن حفظ هذه القيمة في الذاكرة يحتاج إلى إنشاء كائن الخانة، وبالتالي كائن الصف الذي توجد به!. بينما في وجود مصدر بيانات خارجي ــ سواء من خلال تقنية الربط Binding أو من باستخدام الوضع الافتراضي Virtual Mode ــ لا يحتاج جدول العرض إلى حفظ القيم في الذاكرة، فهو يجلبها من مصدر البيانات عند الحاجة، ويرسمها في الخانات مباشرة.

وتنص قاعدة المشاركة على أنه:
يمكن مشاركة الصف فقط إذا كان من الممكن معرفة خصائص كل خانة من خاناته من خصائص الصف والعمود اللذين توجد فيهما.

ويتم إلغاء مشاركة الصف إذا تغيرت حالة خانته، بحيث لا يعود من الممكن استنتاج خصائصها من خصائص الصف والعمود.. وهذه بعض الأمثلة على الحالات التي تلغى فيها مشاركة الصف:

- تحديد خانة منفردة في الصف، دون أن تكون في عمود محدد.. لو أردت منع هذه الحالة، فلا تسمح للمستخدم بتحديد خانة منفردة.
- وضع قيمة في الخاصية ToolTipText أو ContextMenuStrip لإحدى خانات الصف.. ويمكنك تجاوز هذه الحالة، باستخدام الحدثين CellToolTipTextNeeded و CellContextMenuStripNeeded لعرض تلميح الشاشة والقائمة الموضعية عند الحاجة إليهما.
- وجود قائمة منسدلة في إحدى خانات الصف، ووضع عناصر في الخاصية Items الخاصة بها بدون استخدام تقنية الربط.

ويمكنك استخدام الإرشادات التالية لإنشاء الصفوف المشتركة والمحافظة عليها في حالة المشاركة:

١- تجنب استخدام الصيغة التي تستقبل مصفوفة من القيم عند إضافة صف جديد إلى مجموعة الصفوف Rows باستخدام الوسيلة Add أو Insert، لأن هذه الصيغة تضع القيم في خانات الصف مما يلغي مشاركته.

لاحظ أن الصف الجديد الموجود في نهاية جدول العرض هو صف غير مشترك.

٢- تجنب التعامل مع الصف من خلال مجموعة الصفوف Rows، وتجنب المرور عبر الصفوف باستخدام حلقة التكرار For Each.. وبدلا من هذا استخدم الوسائل البديلة التي تستقبل رقم الصف للتعامل معه، مثل الوسائل GetContextMenuStrip، GetErrorText، GetPreferredHeight، GetState.

٣- تجنب التعامل مع الخانة من خلال المجموعة DataGridViewRow.Cells لأن هذا يلغي مشاركة الصف الذي توجد فيه.

٤- تعامل مع خصائص الصفوف والخانات من خلال الخصائص التي يمنحها لك المعامل e في الأحداث التي تنطلق عند حدوث تغييرات في الصف أو الخانة، فهذه الخصائص لا تلغي مشاركة الصف.

٥- استخدم الخاصية CurrentCellAddress للحصول على رقم العمود ورقم الصف اللذين توجد فيهما الخانة، فهذا لا يلغي مشاركة الصف.

٦- استخدم الخاصية Rows.SharedRow للحصول على كائن الصف دون إلغاء مشاركته.. ويمكنك إجراء التعديلات على هذا الكائن، لكن مع ملاحظة أن هذه التغييرات ستؤثر على كل الصفوف المشتركة معه!

٧- استخدم الوسيلة GetContextMenuStrip للحصول على القائمة الموضعية للصف المشترك، لأن الخاصية ContextMenuStrip الخاصة بالصف المشترك ستجد أن رقمه -١ ولن تعيد القائمة الموضعية بصورة صحيحة.

٨- لا تستخدم الحدثين CollectionChanged و RowStateChanged الخاصين بمجموعة الصفوف، لأنهما يلغيان مشاركة الصفوف.

٩- إذا كانت للخاصية SelectionMode القيم FullColumnSelect أو ColumnHeaderSelect أو FullRowSelect أو RowHeaderSelect فلا تتعامل مع الصفوف المحددة من خلال الخاصية DataGridView.SelectedCells لأنها ستلغي مشاركة الصفوف المحددة في هذه الحالة!

١٠- إذا كانت للخاصية SelectionMode القيمة CellSelect فلا تستخدم الوسيلة DataGridView.SelectAll، لأنها ستلغي مشاركة الصفوف.

١١- لا تستخدم الوسيلة DataGridView.AreAllCellsSelected لأنها تلغي مشاركة الصفوف!

١٢- لا تضع القيمة False في الخاصية ReadOnly أو Selected الخاصة بالخانة، إذا كان لأي من هاتين الخاصيتين القيمة True في العمود الذي توجد فيه الخانة.

١٣- لا تستخدم الخاصية DataGridView.Rows.List لأنها تلغي مشاركة جميع الصفوف!

١٤- لا تستخدم الصيغة DataGridView.Sort(IComparer) لوسيلة الترتيب، لأن استخدام فئة مقارنة خاصة يلغي مشاركة جميع الصفوف، بسبب حاجة فئة المقارنة إلى التعامل مع نسخ من الصفوف التي تقارنها.

١٥- استخدم الحدث RowUnshared أثناء تصميم البرنامج لرصد الحالات التي تلغي مشاركة الصفوف، وحاول تجنبها.

وللتأكد من أن الصف مشترك، استخدم الوسيلة SharedRow الخاصة بمجموعة الصفوف للحصول على كائن الصف، ومن ثم افحص رقم هذا الصف، فإن كان -١ فهذا معناه أنه صف مشترك، فكما ذكرنا سابقا، الصف المشترك رقمه دائما -١!.. والمثال التالي يخبرك إن كان الصف الأول مشتركا أم لا:

```
if (DataGridView1.Rows.SharedRow[0].Index == -1)
    MessageBox.Show("هذا الصف مشترك");
```

أخيرا: أحتاج إلى تذكيرك إلى أن كل هذه المحاذير، تتعلق فقط بالحالة التي تتعامل فيها مع جدول بيانات يحتوي على عدد هائل من الصفوف يقدر بالآلاف.. وإن شئت نصيحتي،

عليك الهروب من هذه الحالة المعقدة لأنها أساسا غير عملية، فلا يمكن للمستخدم أن يستعرض آلاف السجلات دفعة واحدة.. لهذا أنصح باستخدام تقنية أخرى، هي تقنية التقسيم إلى صفحات Paging.

تقسيم جدول العرض إلى صفحات Paging:

مَن تعامل مع جدول العرض في تطبيقات مواقع الإنترنت ASP.NET Web Applications، يعرف أن جدول العرض الخاص بها يسمح بعرض البيانات في صورة صفحات، كل منها تحتوي على عدد من السجلات (٢٠ أو ٣٠ مثلا)، ويعرض الجزء السفلي من جدول العرض أرقام الصفحات المتاحة في صورة روابط، وعند ضغط أي منها، يتم عرض السجلات المناظرة لهذه الصفحة في جدول العرض.

ولا أدري لماذا لم تقدم ميكروسوفت هذه التقنية البسيطة والجميلة في جدول العرض الخاص بتطبيقات الويندوز، فهي أسهل وأكفأ وأكثر ملاءمة للمستخدم من تقنية الصفوف المشتركة!

لهذا، دعنا ننشئ نحن بأنفسنا هذه التقنية.. الأمر بسيط، فكل المطلوب هو أن ننشئ موصل جدول له معاملان: الأول يستقبل رقم السجل والثاني يستقبل عدد السجلات المطلوبة، وذلك للحصول على عدد معين من سجلات الجدول بدءا من موضع معين.. وسنضع تحت جدول العرض عددا من لافتات الوصلات LinkLable لنعرض فيها أرقام الصفحات، وعند ضغطها نحمّل السجلات المطلوبة من قاعدة البيانات إلى مجموعة البيانات، ومن ثم نعرض هذه السجلات في جدول العرض من خلال تقنية الربط.. بهذه الطريقة سنحصل على وفر هائل في الذاكرة، ليس فقط بسبب قلة عدد سجلات جدول العرض، ولكن أيضا بسبب قلة سجلات مجموعة البيانات، فحتى لو كانت تقنية مشاركة الصفوف تقلل من مساحة الذاكرة التي يستهلكها جدول العرض، إلا أنها لا تفعل شيئا حيال حجم الذاكرة التي تستهلكها مجموعة البيانات!.. هذا إضافة إلى سرعة تحميل البيانات من قاعدة البيانات، بسبب تقسيمها إلى أجزاء صغيرة.

وعليك اختيار عدد مناسب من السجلات لعرضه في كل صفحة.. ربما يكون العدد ٢٥ مناسبا لتطبيقات الويندوز، فهو عدد معقول بالنسبة للمستخدم، ولا يمثل عبئا ضخما على الذاكرة.. وعموما، لقد عرفنا الثابت RowsNo على مستوى النموذج، ويمكنك تعديله بسهولة للتعامل مع العدد الذي يناسبك من السجلات.

والآن، دعنا نرى كيف ننفذ هذه الفكرة:

- ابدأ مشروعا جديدا اسمه DataGridViewPaging.. وستجده مرفقا بأمثلة الكتاب.
- من القائمة العلوية Data اضغط الأمر Add New Data Source، واتبع خطوات المعالج السحري لإضافة جدولي المؤلفين والكتب إلى مصدر البيانات.
- افتح مخطط قاعدة البيانات واضغط موصل جدول المؤلفين AuthorsTableAdapter بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Configure، وعدل الاستعلام ليصير كالتالي:

```
SELECT * FROM dbo.Authors
WHERE ID BETWEEN @StartID AND @EndID
```

واضغط زر الموافقة.. سيعدل هذا الوسيلة Fill بإضافة معاملين لها، أحدهما اسمه StartID والآخر اسمه EndID، وبهذا يتم تحميل السجلات المحددة فقط من جدول المؤلفين.

- اضغط موصل جدول المؤلفين AuthorsTableAdapter بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Add Query، وتابع خطوات المعالج السحري لإضافة استعلام يعيد قيمة منفردة، باستخدام جملة SQL التالية:

```
SELECT MAX(ID) FROM Authors
```

وسمِّ الدالة التي تنفذ هذا الاستعلام GetMaxID.. هذه الدالة ستخبرنا برقم آخر مؤلف في جدول المؤلفين، لنستخدمه في معرفة عدد الصفحات اللازمة لعرض كل المؤلفين.

- اضغط موصل جدول الكتب BooksTableAdapter بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Configure، وعدل الاستعلام ليصير كالتالي:

```
SELECT Books.* FROM dbo.Books, Authors
WHERE AuthorId = Authors.ID
AND Authors.ID BETWEEN @StartID AND @EndID
```

واضغط زر الموافقة.. سيعدل هذا الوسيلة Fill بإضافة معاملين لها، أحدهما اسمه StartID والآخر اسمه EndID، وبهذا يتم تحميل كتب المؤلفين الذين نتعامل معهم حاليا فقط.

- لاحظ أن موصل البيانات لن ينشئ أوامر التحديث والإدراج والحذف الخاصة بجدول الكتب تلقائيا بسبب وجود عملية ربط في استعلام التحديد.. لهذا يتعين عليك إنشاء هذه الأوامر بنفسك إن كنت تريدها.
- انتقل إلى النموذج، ومن القائمة العلوية Data اضغط الأمر Show Data Sources لعرض نافذة مصادر البيانات.. اسحب جدول المؤلفين وأسقطه على النموذج.. سيضيف هذا إلى النموذج جدول عرض والأدوات اللازمة لربطه بجدول المؤلفين، مع وضع رف أدوات علوي، يسمح للمستخدم بإدخال قيمتي المعاملين StartID و EndID مع زر تنفيذ عملية الملء عنوانه Fill.. هذا جميل.. يمكنك أن تترك هذه الإمكانية أيضا للمستخدم، ليحدد بنفسه السجلات التي يريد عرضها، لكن مع تغيير عناوين اللافتات لتصير عربية كما في الصورة:

- لا تنس استخدام محرر مجموعة الأعمدة من نافذة الخصائص لإخفاء العمود RowVersion أو تغيير نوعه من عمود صورة إلى عمودي نصي، حتى لا يسبب أخطاء.. استخدم نافذة الخصائص أيضا، لإضافة عمود إلى الجدول يعرض أزرارا وامنحه الاسم ColBooks.
- أضف نموذجا آخر إلى المشروع اسمه FrmBooks، وضع عليه جدول عرض لنعرض فيع كتب المؤلف.
- انقر جدول العرض مرتين بالفأرة لعرض كود الحدث CellContentClick، واكتب فيه الكود الذي يعرض النموذج FrmBooks ويعرض كتب المؤلف الحالي فيه.. لقد فعلنا هذا من قبل في المشروع DataGridViewAuthorBooks ولن نكرر شرحه هنا.
- ضع نسخة من موصل بيانات الكتب على النموذج، وسمّه BooksTableAdapter، لاستخدامه في ملء جدول الكتب بكتب المؤلفين المعروضين في الصفحة الحالية.. لاحظ أنك تستطيع إحضار كتب كل المؤلف من قاعدة البيانات مباشرة عند الحاجة إلى عرضها، لكن هذا سيزيد من عدد مرات الاتصال بقاعدة البيانات أثناء تعامل المستخدم مع المؤلفين المعروضين في الصفحة الحالية.. لهذا من الأفضل أن نحضر كل كتب هؤلاء المؤلفين مباشرة ونحفظها في مجموعة البيانات.. ونظرا لأن كل صفحة ستحتوي على ٢٥ مؤلفا، ومع افتراض أن لكل مؤلف ١٠ كتب في المتوسط، فإننا سنضع في الذاكرة حوالي ٢٥٠ كتابا. هذا رقم معقول ولا يقلقنا.. وعلى كل حال، أنت المسئول من خلال التجربة والخطأ، عن تحديد أي من الطريقتين تجعل برنامجك يعمل أسرع دون أن يخنق الخادم أو جهاز المستخدم.
- ضع لافتة رابط LinkLabel على النموذج وامنحها الاسم LnkPage1، وضع فيها النص "١"، واعرضها أسفل الجدول كما هو واضح في الصورة السابقة.. اضبط خط هذه اللافتة وثبت حافتيها اليمنى والسفلي باستخدام الخاصية Anchor.. سنستخدم هذه اللافتة كقالب نستمد منه خصائص باقي لافتات الرابط التي سنضيفها في وقت التشغيل.

- انقر لافتة الرابط مرتين بالفأرة لكتابة كود الحدث LinkClicked.. كود هذا الحدث بسيط للغاية، فنحن نستطيع حساب رقم أول مؤلف نريد عرضه، من خلال الرقم الذي تعرضه اللافتة باستخدام المعادلة التالية:

رقم البداية = ١ + عدد الصفوف في الصفحة × (رقم الصفحة −١)

ونستطيع حساب رقم النهاية من المعادلة التالية:

رقم النهاية = رقم البداية + عدد الصفوف في الصفحة.

علما بأن رقم الصفحة، هو النص الذي تعرضه اللافتة الحالية.. لاحظ أن هذا الحدث سيستجيب لأكثر من لافتة رابط، لهذا سنستخدم المعامل Sender لمعرفة اللافتة التي ضغطها المستخدم.

الآن، صار من السهل أن نملأ جدولي المؤلفين والكتب بالسجلات.. هذا هو الكود:

```
var Lnk = (LinkLabel)sender;
int StartID = 1 + RowsNo *
              (Convert.ToInt32(Lnk.Text) - 1);
AuthorsTableAdapter.Fill(BooksDataSet.Authors,
              StartID, StartID + RowsNo);
BooksTableAdapter.Fill(BooksDataSet.Books, StartID,
              StartID + RowsNo);
```

- يتبقى لنا الآن كتابة كود حدث تحميل النموذج، لإنشاء لافتات الصفحات.. يجب أن نحسب أولا عدد الصفحات المطلوبة، وذلك بقسمة رقم آخر مؤلف على عدد الصفوف التي سنعرضها في كل صفحة، مع تقريب الكسر إلى أكبر عدد صحيح:

```
int MaxID = AuthorsTableAdapter.GetMaxID( );
double PagesNo = Math.Ceiling(MaxID / RowsNo);
```

بعد هذا سننشئ لافتات الرابط التي ستعرض هذه الصفحات، ووضع كل منها بجوار اللافتة التي تسبقها بمسافة كافية.. في البداية نحن نعرف أن اللافتة السابقة هي اللافتة LnkPage1، وبعد هذا يجب أن نحتفظ بكل لافتة نضيفها في متغير اسمه PrvLnk لنستخدمها في ضبط موضع اللافتة التالية لها.. علينا أيضا أن نضبط خط اللافتة ونثبت حوافها، ونربطها بالحدث المستجيب لضغط الرابط.. هذا هو الكود:

```
var PrvLnk = LnkPage1;
for (var I = 2; I <= PagesNo; I++)
{
   LinkLabel Lnk = new LinkLabel( );
   this.Controls.Add(Lnk);
   Lnk.Visible = true;
   Lnk.Text = I.ToString( );
   Lnk.Font = PrvLnk.Font;
   Lnk.AutoSize = true;
   Lnk.Location = Point.Subtract(PrvLnk.Location,
                        new Size(Lnk.Width + 10, 0));
   Lnk.Anchor = PrvLnk.Anchor;
   Lnk.LinkClicked += LnkPage1_LinkClicked;
   PrvLnk = Lnk;
}
```

لاحظ أن الخاصية AutoSize ستكون بلا تأثير إذا وضعت فيها true قبل أن تضيف لافتة الربط إلى أدوات النموذج.. لهذا أضفنا اللافتة إلى أدوات النموذج أولا، وضبطنا خطها، ووضعنا في الخاصية Text النص الذي ستعرضه، قبل أن نجعلها تغير حجمها تلقائيا لتناسب محتوياتها.

والآن، سيكون من الأفضل لو ضغطنا نحن رابط أول لافتة لنعرض أول صفحة من صفحات المؤلفين في جدول العرض بمجرد تشغيل البرنامج.. لكن نظرا لأن لافتة الربط لا تملك الوسيلة PerformClick الخاصة بالأزرار، فسنضطر إلى استدعاء الحدث LnkPage1_LinkClicked بأنفسنا كالتالي:

```
LnkPage1_LinkClicked(LnkPage1,
    new LinkLabelLinkClickedEventArgs(null));
```

لو شغلت البرنامج الآن، فسيعرض جدول العرض أول صفحة من صفحات المؤلفين، ويمكنك أن تضغط رابط أي صفحة أخرى لعرضها.. ولو ضغطت زر عرض كتب أي مؤلف، فسيظهر النموذج الثاني وعليه كتب هذا المؤلف.

يبدو كل شيء جيدا.. لكن للأسف هناك مشكلة صغيرة، تحدث بسبب طريقة تقسيم الصفحات التي نستخدمها!.. فنحن لا نضمن انتظام أرقام المؤلفين لأن الحقل ID مولد

تلقائيا.. لهذا قد تجد أن أول مؤلف يبدأ بالرقم ١٢ وليس ١، ويليه مؤلف رقمه ١٥ مثلا، وذلك بسبب حذف سجلات أخرى ضيعت الترقيم الوسيط!.. لهذا قد نجد صفحات تعرض عددا من السجلات أقل من العدد المطلوب، وأحيانا قد تظهر صفحات ليس فيها أية سجلات على الإطلاق!

ولحل هذه المشكلة، علينا تغيير طريقة تقسيم الصفحات، وهو ما فعلناه في المشروع DataGridViewPaging2.. في هذا المشروع تركزت معظم التغييرات على استعلامات موصلات الجداول، مع قليل من التعديلات في الكود.

دعنا نفهم فكرة التقسيم الجديدة:

في البداية لو أردنا ملء أول صفحة بعدد من المؤلفين Count@، فسنستخدم الاستعلام التالي، وهو الذي ستجده في الوسيلة FillFirstPage:

```
SELECT TOP (@Count) * FROM Authors
```

وسيتم ملء كتب هؤلاء المؤلفين بالوسيلة FillFirstPage في موصل جدول الكتب بالاستعلام التالي:

```
SELECT *
FROM  Books INNER JOIN
      Authors ON Books.AuthorID = Authors.ID
WHERE Authors.ID IN
(SELECT TOP (@Count) ID FROM Authors)
```

لاحظ أن هذا الاستعلام مركب، فهو يستخدم جملة SELECT ثانية للحصول على أرقام المؤلفين المعروضين في أول صفحة، والتأكد أن رقم المؤلف الذي نحصل على كتبه يقع ضمن هذه الأرقام.

هذا جميل.. لكن كيف نحصل على المؤلفين في الصفحات الأخرى غير الصفحة الأولى؟

المشكلة هنا أننا لا نعرف ترقيم أول مؤلف في هذه الصفحات، فكما ذكرنا من قبل، يتسم حقل الترقيم التلقائي بعدم الانتظام!

لحل هذه المشكلة، علينا معرفة ترقيم آخر مؤلف تم عرضه في الصفحة السابقة للصفحة الحالية.. افترض أن الصفحة التي سنعرضها ستبدأ بالمؤلف الحادي عشر.. هذا معناه أن

الصفحات السابقة عرضت ١٠ مؤلفين.. يمكننا إذن أن نحصل على أرقام أول ١٠ مؤلفين من الجدول كالتالي:

```
SELECT TOP (11 - 1)
ID FROM   Authors AS PrvPages
ORDER BY ID
```

ويمكننا أن نحصل على ترقيم آخر مؤلف منهم باستخدام الدالة MAX كالتالي:

```
SELECT MAX(ID)
FROM  (
          SELECT TOP (11 -1)
          ID FROM   Authors AS PrvPages
          ORDER BY ID
        ) AS MaxID
```

طبعا الرقم ١١ يفيدنا في شرح هذا المثال، لكن في البرنامج، سنضع بدلا منه معاملا اسمه @AuthorNo وهو الرقم الفعلي للمؤلف الذي يظهر في بداية الصفحة، وليس ترقيمه التلقائي الموجود في الحقل ID.. هذا الرقم يساوي:

١ + عدد الصفوف في الصفحة × (رقم الصفحة −١) كما شرحنا من قبل.

والآن، بعد أن حصلنا على ترقيم آخر مؤلف عرضناه في الصفحة السابقة، يمكننا أن نقرأ السجلات التي يزيد ترقيمها عن ترقيمه، ونأخذ منها فقط العدد @Count.. هذا هو الاستعلام الكامل:

```
SELECT TOP (@Count)
* FROM  Authors
WHERE  ID > (
                SELECT MAX(ID)
                FROM  (
                          SELECT TOP (@AuthorNo -1)
                          ID FROM   Authors AS PrvPages
                          ORDER BY ID
                        ) AS MaxID
                )
```

ملحوظة:

ستضاف جملة التحديد بعد أمر التحديث وأمر الإدراج، وهذا سيسبب أخطاء في البرنامج عند حفظ التغييرات، بسبب وجود معاملين لن يتم إرسال قيمتيهما.. لهذا عليك حذف جملة التحديد من أمر التحديث وأمر الإدراج.. يمكنك فعل هذا من الخيارات المتقدمة Advanced Options أثناء تنفيذ المعالج السحري، أو يمكنك فعله من نافذة الخصائص بعد انتهاء المعالج.. حدد موصل جدول المؤلفين، وافتح نافذة الخصائص وأسدل خصائص أمر التحديث UpdateCommand وغير قيمة الخاصية CommandText.. لاحظ أن أمر التحديد يوجد في سطر جديد، لهذا لن تستطيع التعامل معه في خانة القيمة لأنها تعرض سطرا واحدا فقط.. للتحايل على هذا، حدد السطر الظاهر (هذا هو أمر التحديث) وقصه Cut، ثم الصقه مرة ثانية Paste.. هكذا تكون قد تخلصت من أمر التحديد.. ويمكنك فعل نفس الشيء مع أمر الإدراج الموجود في الخاصية InsertCommand.

ولكي نحصل على كتب هؤلاء المؤلفين، سنكون استعلاما يضمن أن ترقيم المؤلف الذي نقرأ كتبه يقع ضمن أرقامهم كالتالي:

```
SELECT * FROM  Books
INNER JOIN Authors
ON Books.AuthorID = Authors.ID
WHERE Authors.ID IN
      (
       SELECT TOP (@Count)
       ID FROM  Authors
       WHERE  ID >
                 (
                  SELECT MAX(ID)
                  FROM  (
                           SELECT TOP (@AuthorNo -1)
                           ID FROM   Authors AS PrvPages
                           ORDER BY ID
                          ) AS MaxID
                 )
    )
```

واضح أن هذا أعقد استعلام كتبناه حتى الآن.. لكن لا تدعه يربكك، فكل ما هو بعد الكلمة IN في هذا الاستعلام هو نفس الاستعلام الذي استخدمناه في موصل جدول المؤلفين، مع فارق واحد: أننا هنا نقرأ الحقل ID فقط وليس كل حقول جدول المؤلفين، لأننا نريد استخدام الحقل ID في جملة الشرط.

يمكنك الآن تجربة المشروع DataGridViewPaging2.. ستجده يعمل بشكل رائع، فكل صفحة فعلا تعرض عدد المؤلفين المطلوب بطريقة دقيقة، ولا يستثنى من هذا إلا آخر صفحة، التي قد تعرض عددا أقل من المؤلفين، بسبب عدم وجود مزيد من المؤلفين في الجدول.. وهذا منطقي وصحيح.

مبارك.. لقد حصلت على تقنية صفحات العرض الخاصة بك.. ولا ينقص هذه التقنية إلا شيء واحد.. فمن المستحيل وضع كل أرقام الصفحات في لافتات الربط عندما يكون عدد الصفحات كبيرا (١٠٠ صفحة مثلا).. في هذه الحالة عليك أن تعرض أول عشرة أرقام فقط، مع وضع لافتة مكتوب عليها "التالي"، وعند الضغط عليها تعرض عشرة أرقام تالية، مع عرض لافتة في البداية اسمها "السابق"، عند الضغط عليها تعرض ١٠ أرقام سابقة.

دعنا نرى كيف نفعل هذا:

- أضف إلى النموذج لافتة رابط وامنحها الاسم LnkPrv واجعلها تعرض النص "السابق".
- أضف لافتة رابط أخرى اسمها LnkNext تعرض النص "التالي".
- عرف ثابتا على مستوى النموذج اسمه MaxLinks لتتحكم به في أقصى عدد يمكن عرضه من الروابط.
- عرف المتغيرات التالية على مستوى النموذج:

١- PageLinks، وهو قائمة مخصصة للتعامل مع لافتات الروابط List(Of LinkLabel)، لنضع فيها مراجع إلى لافتات الروابط التي نعرضها على النموذج.. هذا سيسهل علينا التحكم في هذه الروابط.

٢- CurLink، وهو يحمل مرجعا إلى لافتة الرابط المضغوطة حاليا، (التي تظهر الصفحة المناظرة لها في جدول العرض).. هذا سيفيدنا في التحرك إلى الأمام أو الخلف عند ضغط "التالي" أو "السابق".

٣- FirstPageNo، وهو متغير يحمل رقم أول صفحة يظهر حاليا في لافتات الروابط.. في البداية تكون قيمة هذا المتغير ١، ويمكن أن يتغير إذا عرضنا مجموعة أخرى من الروابط بسبب ضغط "التالي".

٤- PagesNo، وهو متغير يحمل عدد الصفحات الكلي الذي نتعامل معه.

- في حدث تحميل النموذج، سنعدل حلقة التكرار التي تضيف اللافتات، بحيث يكون أقصى عدد نضيفه هو MaxLinks.. يتم هذا كالتالي:

```
for (int I = 2; I <= Math.Min(PagesNo, MaxLinks); I++)
{

}
```

- سيظل كود حلقة التكرار كما كان في المشروع السابق، ما عدا زيادة سطر واحد، يضيف كل لافتة ننشئها إلى المجموعة PageLinks.. لا تنس أيضا إضافة اللافتة الأولى LnkPage1 إلى القائمة:

```
PageLinks.Add(LnkPage1);
for (int I = 2; I <= Math.Min(PagesNo, MaxLinks); I++)
{
    // نفس الكود القديم
    PageLinks.Add(PrvLnk);
}
```

- استخدم الحدث LinkClicked لتعطيل الرابط "السابق" إذا كان رقم الرابط المضغوط ١، وتعطيل الرابط "التالي" إذا كان رقم الرابط المضغوط يساوي عدد الصفحات:

```
int I = Convert.ToInt32(Lnk.Text);
LnkPrev.Enabled = (I > 1);
LnkNext.Enabled = (I < PagesNo);
```

- سيكون من المفيد أيضا أن نميز الرابط المضغوط حاليا عن باقي الـروابط، وذلـك بتعطيله (فلا فائدة من ضغطه ثانية) وجعل خطه سميكا.. وعليك إعادة الرابط السابق إلى وضعه الطبيعي قبل تغيير حالة الرابط المضغوط حاليا.. أنسب مكان لفعل هـذا هو الحدث LinkClicked أيضا:

```
// إعادة الرابط السابق إلى وضعه الأصلي
CurLink.Enabled = true;
CurLink.Font = new Font(CurLink.Font,
        FontStyle.Regular);
// تمييز الرابط الحالي
Lnk.Enabled = false;
Lnk.Font = new Font(Lnk.Font, FontStyle.Bold);
CurLink = Lnk;
```

- في حدث ضغط "التالي" سنضغط الرابط الذي يزيد رقمه عن الرابط الحالي بواحـد، إن كان معروضا على الشاشة:

```
int I = Convert.ToInt32(CurLink.Text);
LnkPage1_LinkClicked(PageLinks[I - FirstPageNo + 1],
    new LinkLabelLinkClickedEventArgs(null));
```

أما إذا كان الرابط السابق هو آخر رابط معروض على الشاشة، فيجـب أن نعـرض مجموعة تالية من الروابط.. لفعل هذا لا نحتاج إلى حذف الروابط الحاليـة وإنشـاء روابط جديدة، فبإمكاننا أن نغير الأرقام المعروضة على اللافتـات ببسـاطة، وذلـك بجمع القيمة MaxLinks على كل منها.. لاحظ أن ناتج الجمع قد يتجـاوز إجمـالي عدد الصفحات في بعض الحالات، لهذا علينا إخفاء اللافتة التي يحدث لها هـذا.. ولا تنسَ تغيير قيمة المتغير FirstPageNo، لتشير إلى رقم أول رابط فـي المجموعـة الجديدة من الروابط.. هذا هو كود الحدث كاملا:

```
int I = Convert.ToInt32(CurLink.Text);
if (I == MaxLinks)
{
    foreach (var Lnk in PageLinks)
    {
      int N = Convert.ToInt32(Lnk.Text) + MaxLinks;
      Lnk.Text = N.ToString( );
      if (N > PagesNo)
        Lnk.Visible = false;
    }
    FirstPageNo += MaxLinks;
}
LnkPage1_LinkClicked(PageLinks[I - FirstPageNo + 1],
        new LinkLabelLinkClickedEventArgs(null));
```

- أخيرا، لم يتبق لنا إلا حدث ضغط "السابق".. هذا الكود مشابه لكود حدث ضغط "التالي"، مع عكس عمليات الجمع إلى طرح، وإظهار اللافتات المختفية.. كما أن شرط عرض المجموعة السابقة من اللافتات، هو أن يكون الرباط المضغوط حاليا أول رابط معروض على الشاشة.. هذا هو الكود:

```
int I = Convert.ToInt32(CurLink.Text);
if (I == FirstPageNo)
{
   foreach (var Lnk in PageLinks)
   {
     Lnk.Text = (Convert.ToInt32(Lnk.Text) - MaxLinks).ToString( );
     Lnk.Visible = true;
   }
   FirstPageNo -= MaxLinks;
}
LnkPage1_LinkClicked(PageLinks(I - FirstPageNo - 1),
        new LinkLabelLinkClickedEventArgs(null));
```

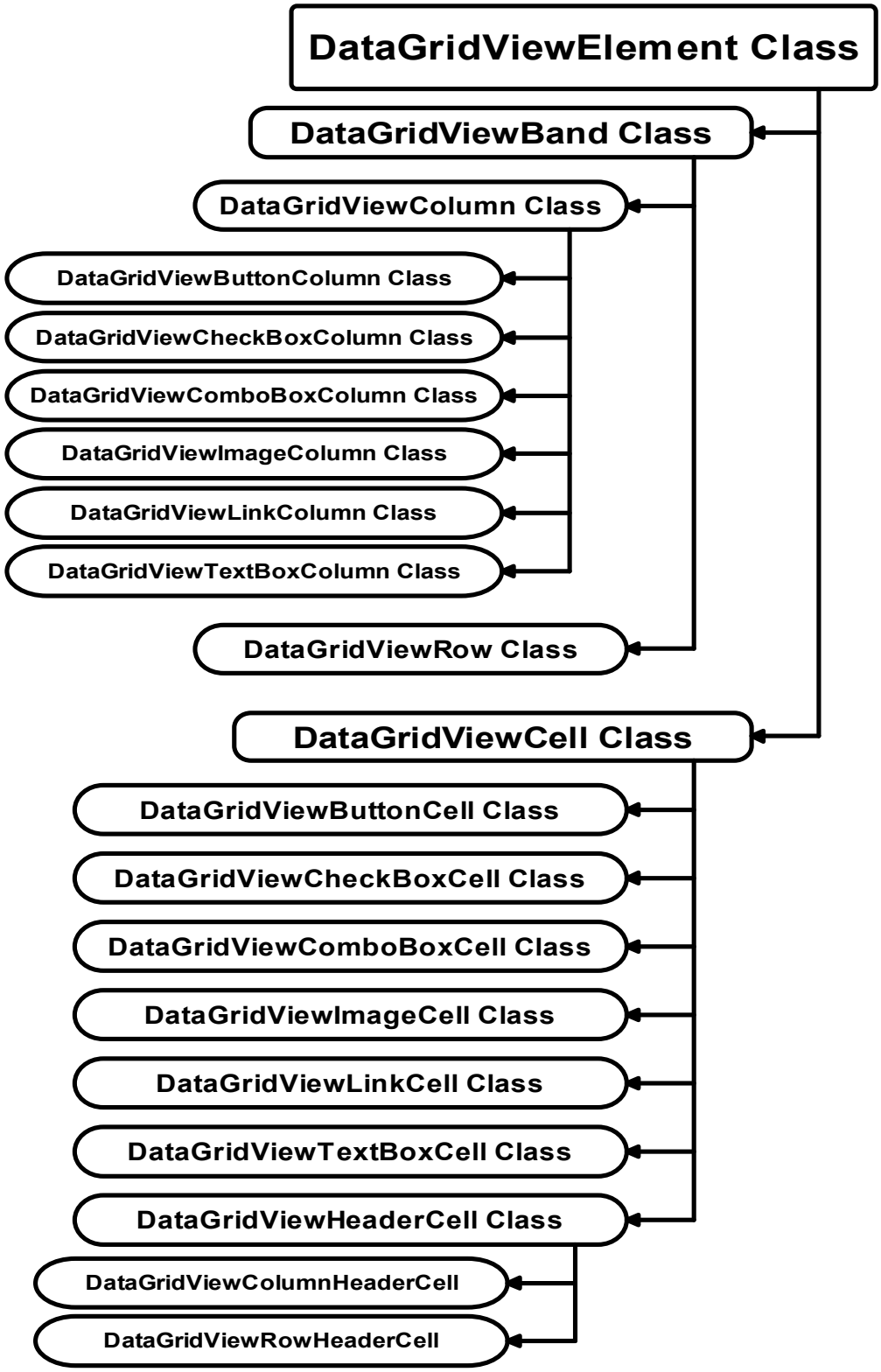
الآن، يمكنك تشغيل البرنامج والاستمتاع بتجربته.. ستجد أن لديك بالفعل جدول عرض مقسم إلى صفحات، يعمل بكفاءة تامة، وقدرات كاملة!

ملحق: ١

الفئات التي يستخدمها جدول عرض البيانات

سنشرح في هذا الملحق كل الفئات التي يحتاجها جدول العــرض DataGridView لأداء عمله، كما هو موضح في المخطط التالي:

اللهم ارحم أبي واغفر له وكفر عنه سيئاته


DataGridViewElement Class
DataGridViewBand Class
DataGridViewColumn Class
DataGridViewButtonColumn Class
DataGridViewCheckBoxColumn Class
DataGridViewComboBoxColumn Class
DataGridViewImageColumn Class
DataGridViewLinkColumn Class
DataGridViewTextBoxColumn Class
DataGridViewRow Class
DataGridViewCell Class
DataGridViewButtonCell Class
DataGridViewCheckBoxCell Class
DataGridViewComboBoxCell Class
DataGridViewImageCell Class
DataGridViewLinkCell Class
DataGridViewTextBoxCell Class
DataGridViewHeaderCell Class
DataGridViewColumnHeaderCell
DataGridViewRowHeaderCell

فئة عنصر جدول العرض DataGridViewElement Class

هذه هي الفئة الأم التي تشتق منها كل عناصر جدول العرض: الأعمدة والصفوف والخانات، وهي تملك خاصيتين فقط:

جدول العرض DataGridView:

تعيد كائن جدول العرض DataGridView الذي ينتمي إليه العنصر.

الحالة State:

تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح حالة العنصر، من بين القيم التالية:

None	العنصر في حالته الافتراضية.
Frozen	العنصر مجمد (مثبت).. هذا معناه أنه يظل ظاهرا في موضعه مهما تحرك المستخدم بالمنزلق.
ReadOnly	العنصر للقراءة فقط، ولا يستطيع المستخدم تغيير قيمته.
Resizable	يمكن للمستخدم تغيير موضع وحجم العنصر في الجدول.. لاحظ أن هذه القيمة يتم تجاهلها إذا لم يتم دمجها مع القيمة ResizableSet.
ResizableSet	العنصر مستقل في قابلية تغيير حجمه، عن العنصر الرئيسي الذي ينتمي إليه.
Selected	العنصر محدد حاليا Highlighted.
Displayed	العنصر ظاهر للمستخدم حاليا دون الحاجة إلى تحريك المنزلق لعرضه.
Visible	العنصر مرئي (غير مختفي).. هذا صحيح حتى لو كان العنصر غير معروض Displayed على الشاشة بسبب موضعه من المنزلق.

ويعمل هذا المرقم كمؤشر Flag، لهذا يمكن أن تعيد هذه الخاصية أكثر من قيمة مدمجة معا، وعليك فحص القيمة التي تريدها باستخدام المعامل &:

```
if ((DataGridView1.Rows[0].State &
        DataGridViewElementStates.Displayed) > 0)
   MessageBox.Show(DataGridView1.Rows[
        0].State.ToString( ));
```

فئة نطاق جدول العرض DataGridViewBand Class

هذه الفئة ترث الفئة DataGridViewElement، كما أنها تمثل الـــواجهتين ICloneable و IDisposable.

وتعمل هذه الفئة كفئة أم تشتق منها أعمدة وصفوف جدول العرض، ولعل هذا يوضح سبب تسمية هذه الفئة باسم النطاق Band، فهي تمثل نطاقا من الخانات تقع في صف معـــين أو عمود معين.

وتمتلك هذه الفئة الخصائص التالية:

رف القائمة الموضعية ContextMenuStrip:

تقرأ أو تغير كائن رف القائمة الموضعية ContextMenuStrip الذي يعرض القائمة الموضعية للنطاق الحالي.. هذا معناه أنك تستطيع استخدام قائمة موضعية مختلفة لكل صف، ولكل عمود!.. وتظهر القائمة الموضعية كما تعرف عند الضغط بـــزر الفـــأرة الأيمن فوق أي خانة في النطاق، ما عدا خانة رأس النطاق.

لاحظ أن أولوية القائمة الموضعية تكون كالتالي:

١- تظهر القائمة الموضعية للخانة إن وجدت.

٢- تظهر القائمة الموضعية للصف إن وجدت.

٣- تظهر القائمة الموضعية للعمود إن وجدت.

هذا معناه أن الخانة التي توجد في صف له قائمة موضعية وعمود له قائمة موضعية، ستعرض القائمة الموضعية الخاصة بالصف وليس العمود.

النوع الافتراضي لخانة العنوان DefaultHeaderCellType:

ضع في هذه الخاصية كائن النوع Type الذي يمثل نوع خانة العنوان (رأس العمود)، بشرط أن تكون قيمة هذه الخاصية من نوع الفئة DataGridViewHeaderCell أو أي فئة مشتقة منها.. وسنتعرف على هذه الفئات بالتفصيل لاحقا.

وتفيدك هذه الخاصية إذا أردت تغيير شكل ووظيفة خانة رأس الصف أو العمود.. يمكنك مثلا تعريف فئة جديدة مشتقة من الفئة DataGridViewColumnHeaderCell، ومنحها الشكل والأداء الذي تريده، ثم وضع نوعها في الخاصية DefaultHeaderCellType لكل عمود في جدول العرض، لتظهر كخانة رأس لكل منها.

مجمد Frozen:

إذا جعلت قيمة هذه الخاصية True، فسيتم تثبيت النطاق الحالي في موضعه مهما حرك المستخدم المنزلق الأفقي أو الرأسي.. هذا مفيد إذا أرت استخدام أحد الصفوف أو الأعمدة كعنوان ثابت بحيث يظل مرئيا باستمرار.

لاحظ أنك تستطيع تثبيت النطاق (الصف أو العمود) إذا كان أول نطاق أو يسبقه نطاق مثبت، وغير هذا يحدث خطأ في البرنامج.

رقم العنصر Index:

تعيد رقم النطاق الحالي في الجدول.. فإذا كان النطاق الحالي صفا، تعيد موضعه في مجموعة صفوف الجدول، وإن كان النطاق الحالي عمودا، تعيد موضعه الأصلي في مجموعة الأعمدة.

لاحظ أن تغيير المستخدم لموضع العمود بسحبه بالفأرة (إذا كنت تسمح له بترتيب الأعمدة) لا يؤثر على رقم العمود في مجموعة الأعمدة، وإنما يؤثر فقط على موضع عرضه DisplayIndex.. لكن على العكس، يتغير موضع الصف في مجموعة الصفوف، إذا ضغط المستخدم رأس أحد الأعمدة لترتيب الصفوف على أساسه.. لهذا لا تحتفظ برقم الصف في متغير طوال تشغيل البرنامج، لأنه قد يتغير في أي لحظة،

وبدلا من هذا احتفظ بمتغير من النوع DataGridViewRow يشير إلى الصف الذي تريده.

الطراز الافتراضي للخانة DefaultCellStyle:

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل خانات النطاق الحالي.. وسنتعرف على الفئة DataGridViewCellStyle بالتفصيل لاحقا.

يوجد طراز افتراضي للخانة HasDefaultCellStyle:

تعيد True إذا كنت قد وضعت قيمة في الخاصية DefaultCellStyle.. لاحظ أن الخاصية DefaultCellStyle لا تعيد Nothing أبدا، فلو كانت فارغة وحاولت قراءتها، فسيتم إنشاء طراز افتراضي ووضعه فيها!.. لهذا يمكنك استخدام الخاصية HasDefaultCellStyle أولا لمعرفة إن كانت الخاصية DefaultCellStyle فارغة أم لا.

الطراز الموروث InheritedStyle:

تعيد كائن طراز الخانة DataGridViewCellStyle الذي يتم تطبيقه على النطاق الحالي.. وتعيد هذه الخاصية الطراز الموضح في الخاصية DefaultCellStyle إذا كانت لها قيمة، وإلا فإنها تعيد الطراز الموروث من جدول العرض.

للقراءة فقط ReadOnly:

إذا جعلت قيمة هذه الخاصية True، فلن يستطيع المستخدم تغيير قيمة أي خانة في النطاق الحالي.

قابل للتحجيم Resizable:

تحدد إن كان بإمكان المستخدم تغيير حجم النطاق الحالي (تغيير عرض العمود أو ارتفاع الصف) باستخدام الفأرة.. وهي تأخذ إحدى قيم المرقم DataGridViewTriState التالية:

NotSet	لم يتم تحديد قيمة الخاصية، وسيتم استخدام القيمة الافتراضية الموروثة من جدول العرض.
True	توضع القيمة True في الخاصية.
False	توضع القيمة False في الخاصية.

محدد Selected:

إذا جعلت قيمة هذه الخاصية True، فسيتم تحديد النطاق الحالي.. كما تعيد هذه الخاصية True إذا كان النطاق الحالي محددا سواء بواسطتك أو بواسطة المستخدم.. لاحظ أن تحديد النطاق الحالي لا يلغي تحديد النطاقات الأخرى، فجدول العرض يتيح تحديد أكثر من صف أو عمود معا.

لاحظ أن هذه الخاصية تتأثر بقيمة الخاصية SelectionMode الخاصة بجدول العرض كالتالي:

- إذا كانت للخاصية SelectionMode أي قيمة غير FullRowSelect و RowHeaderSelect، فإن الخاصية Select الخاصة بالصف تكون بلا تأثير لأن جدول العرض لا يسمح بتحديد الصفوف.
- إذا كانت للخاصية SelectionMode أي قيمة غير FullColumnSelect و ColumnHeaderSelect، فإن الخاصية Select الخاصة بالعمود تكون بلا تأثير لأن جدول العرض لا يسمح بتحديد الأعمدة.

معروض Displayed:

تعيد True إذا كان النطاق الحالي ظاهرا للمستخدم على الشاشة دون الحاجة إلى استخدام المنزلق الأفقي أو الرأسي.

مرئي Visible:

إذا جعلت قيمة هذه الخاصية False، فسيتم إخفاء النطاق الحالي وعدم عرضه في الجدول.

الوسم Tag:

هذه هي الخاصية الإضافية، التي تستطيع أن تضع فيها أي كائن يحوي معلومات تهمك تتعلق بالنطاق الحالي.

فئة أساس المجموعة BaseCollection Class

هذه الفئة تمثل الواجهة ICollection، وهي تعمل كمجموعة عادية، ولكنها لا تحتوي على أية وسائل تتيح لك إضافة العناصر إليها.. كل ما تحتويه هو العناصر التالية، وهي مألوفة لنا فقد شرحناها بالتفصيل في كتاب برمجة إطار العمل، لهذا لن نعيد شرحها هنا:

Count **IsReadOnly**
IsSynchronized **SyncRoot**
CopyTo **GetEnumerator**

فئة مجموعة أعمدة جدول العرض
DataGridViewColumnCollection Class

هذه الفئة ترث الفئة BaseCollection كما أنها تمثل واجهة القائمة IList، وهي تحتوي على عناصر من نوع فئة عمود جدول العرض DataGridViewColumn Class.
ولحدث إنشاء هذه المجموعة صيغة واحدة، تستقبل كائن جدول العرض DataGridView الذي ستنتمي إليه مجموعة الأعمدة.. مثال:

```
var Cols = new DataGridViewColumnCollection(DataGridView1);
```

في الحقيقة، لا يبدو إرسال جدول العرض كمعامل إلى حدث ذا مغزى، فبعد تنفيذ الجملة السابقة لن يتغير شيء في مجموعة الأعمدة الخاصة بجدول العرض DataGridView1، ولن تكون له أية صلة بالمجموعة الجديدة Cols، والتي بدورها ستكون فارغة ولن تحتوي على أية أعمدة موجودة حاليا في جدول العرض DataGridView1!!

وتمتلك مجموعة الأعمدة العناصر التقليدية للمجموعات، والتي تستقبل كمعامل كائن العمود أو نصا يمثل اسم العمود.. لهذا دعنا نركز هنا على العناصر التالية:

إضافة Add:

تصيف عمودا إلى مجموعة الأعمدة، وتعيد رقما صحيحا يمثل موضع هذا العمود في المجموعة.. ولهذه الوسيلة الصيغتان التاليتان:

١- الصيغة الأولى تستقبل كائن العمود DataGridViewColumn وتضيفه إلى المجموعة.

٢- الصيغة الثانية تستقبل نصا يمثل اسم العمود، ونصا يمثل عنوان العمود، وتنشئ عمودا نصيا DataGridViewTextBoxColumn وتضيفه إلى المجموعة.. والمثال التالي يضيف إلى جدول العرض عمودا نصيا اسمه Col1 وعنوانه "عمود١":

DataGridView1.Columns.Add("Col1", "عمود١");

وتتسبب هذه الوسيلة في حدوث خطأ في البرنامج في الحالات التالية:

- إذا كان العمود المراد إضافته موجودا في جدول العرض من قبل.
- إذا كان الجدول يحتوي على صف أو أكثر، بينما للخاصية CellType الخاصة بالعمود القيمة Nothing.
- إذا كان العمود الجديد مثبتا Frozen، وأضفته وسط أعمدة غير مثبتة.
- إذا كان جدول العرض يقوم بتحديد كل خاناته في تلك اللحظة أو يزيل تحديدها أو يغير قيم الخاصية DisplayIndex لكل الأعمدة.
- إذا تم استدعاء الوسيلة Add من داخل أي من الأحداث التالية:
CellEnter, CellLeave, CellValidating, CellValidated, RowEnter, RowLeave, RowValidated, RowValidating.
- إذا كانت للخاصية SortMode الخاصة بالعمود القيمة Automatic، بينما للخاصية DataGridView.SelectionMode القيمة FullColumnSelect أو ColumnHeaderSelect.
- إذا كانت للخاصية InheritedAutoSizeMode الخاصة بالعمود القيمة ColumnHeader، بينما عناوين الأعمدة غير معروضة (DataGridView.ColumnHeadersVisible = False).
- إذا كانت للخاصية InheritedAutoSizeMode القيمة Fill، بينما العمود مثبتا (Frozen = True).

العنصر Item:

هذه هي الخاصية الافتراضية، وهي تعيد كائن العمود DataGridViewColumn الذي ترسل إليها اسمه أو رقمه كمعامل.. والكود التالي يعرض رقم العمود Col1 الذي أضفناه في المثال السابق:

Console.WriteLine (DataGridView1.Columns["Col1"].Index);

معرفة عدد الأعمدة GetColumnCount:

تعيد عدد أعمدة الجدول التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.. والمثال التالي يعرض عدد الأعمدة المحددة في جدول العرض:

```
Console.WriteLine(DataGridView1.Columns.GetColumnCount(
        DataGridViewElementStates.Selected));
```

معرفة عرض الأعمدة GetColumnsWidth:

تعيد مجموع عروض أعمدة الجدول التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.. والمثال التالي يخبرك بعرض الأعمدة المرئية في الجدول:

```
Console.WriteLine(DataGridView1.Columns.GetColumnsWidth(
        DataGridViewElementStates.Visible));
```

معرفة أول عمود GetFirstColumn:

تستقبل إحدى قيم المرقم DataGridViewElementStates وتعيد أول عمود له الحالة المرسلة.. وتعيد هذه الوسيلة Nothing إذا لم تجد عمودا له الحالة المطلوبة.

وتوجد صيغة أخرى لهذه الوسيلة لها معامل ثانٍ هو أيضا من نوع المرقم DataGridViewElementStates، ولكنه يستقل الحالة التي يجب ألا يكون عليها العمود.. والمثال التالي يعيد أول عمود مرئي لكنه غير معروض للمستخدم:

```
var DgCol As DataGridViewColumn =
    DataGridView1.Columns.GetFirstColumn(
    DataGridViewElementStates.Visible,
    DataGridViewElementStates.Displayed);
```

معرفة آخر عمود GetLastColumn:

مماثلة للوسيلة السابقة، ولكنها تعيد آخر عمود له الحالة الموضحة في المعامل الأول وليست له الحالة الموضحة في المعامل الثاني.

معرفة العمود التالي GetNextColumn:

تعيد كائن العمود DataGridViewColumn الذي يحقق الشروط الموضحة في المعاملات، وهي بالترتيب:

١- كائن العمود DataGridViewColumn الذي سيبدأ البحث منه، للعثور على أول عمود يليه يحقق الشروط المطلوبة.

٢- إحدى قيم المرقم DataGridViewElementStates توضح الحالة التي يجب أن يمتلكها العمود المطلوب.

٣- إحدى قيم المرقم DataGridViewElementStates توضح الحالة التي يجب ألا يمتلكها العمود المطلوب.

وتعيد هذه الوسيلة Nothing إذا لم تجد عمودا يحقق الشروط المطلوبة.

والمثال التالي يعرض أسماء كل الأعمدة الظاهرة للمستخدم والتي لا يستطيع تغيير حجمها:

```
var Cols = DataGridView1.Columns;
var DgCol = Cols.GetFirstColumn(
      DataGridViewElementStates.Displayed,
      DataGridViewElementStates.Resizable);
while ( DgCol != null)
{
 MessageBox.Show(DgCol.Name);
 DgCol = Cols.GetNextColumn(DgCol,
       DataGridViewElementStates. Displayed,
       DataGridViewElementStates.Resizable);
}
```

معرفة العمود السابق GetPreviousColumn:

مماثلة للوسيلة السابقة في معاملاتها، ولكنها تبحث عن العمود السابق للعمود المرسل للمعامل الأول، الذي يحقق الشروط المطلوبة.. دعنا نعيد كتابة المثال السابق باستخدام هذه الوسيلة مع الوسيلة GetLastColumn، لعرض أسماء الأعمدة بترتيب عكسي:

```
var Cols = DataGridView1.Columns;
var DgCol = Cols.GetLastColumn(
      DataGridViewElementStates.Displayed,
      DataGridViewElementStates.Resizable);
while (DgCol != null)
{
       MessageBox.Show(DgCol.Name);
       DgCol = Cols.GetPreviousColumn(DgCol,
           DataGridViewElementStates. Displayed,
           DataGridViewElementStates.Resizable);
}
```

المجموعة تغيرت CollectionChanged:

ينطلق هذا الحدث عند حدوث تغير في عناصر المجموعة بالحذف أو الإضافة.. والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs وقد تعرفنا عليه من قبل عند التعرف على مجموعة الجداول DataTableCollection.

فئة عمود جدول العرض DataGridViewColumn Class

هذه الفئة ترث الفئة DataGridViewBand وتمثل الواجهة IComponent.

وتعمل هذه الفئة كعمود في جدول عرض البيانات، ولحدث إنشائها صيغتان:

١- الصيغة الأولى بدون معاملات.

٢- والصيغة الثانية تستقبل كائن خانة DataGridViewCell لوضعه في الخاصية CellTemplate الخاصة بالعمود.. هذه الخانة ستعمل كقالب Template تنسخ منه كل خانة تضاف إلى العمود عند إضافة صف جديد إلى الجدول.

وتمتلك هذه الفئة الخصائص التالية:

الاسم Name:

تقرأ أو تغير اسم العمود.. هذا الاسم لا يظهر كعنوان له، وإنما يستخدم كمعرف للعمود داخل مجموعة الأعمدة، لاستخدامه مع بعض الوسائل مثل Remove و Contains.

نوع القيمة ValueType:

ضع في هذه الخاصية كائن النوع Type، الذي يمثل نوع بيانات خانات العمود الحالي.

نص تلميح الأداة ToolTipText:

ضع في هذه الخاصية النص الذي تريد عرضه للمستخدم عندما يحلق بالفأرة للحظات فوق رأس العمود.

العرض Width:

تقرأ أو تغير عرض العمود الحالي، والقيمة الافتراضية لها هي ١٠٠.

أقل عرض MinimumWidth:

تقرأ أو تغير أقل عرض ممكن للعمود، بحيث لا يمكن تصغيره عنه برمجيا أو عند سحب المستخدم لحافته بالفأرة لتغيير حجمه.. والقيمة الافتراضية لهذه الخاصية هي ٥، وغير مسموح لك بتصغيرها عن ٢.

طريقة الحجم التلقائي AutoSizeMode:

تحدد كيف يتم تغيير حجم العمود تلقائيا، وهي تأخذ إحدى قيم المرقم DataGridViewAutoSizeColumnMode التالية:

NotSet	طريقة تحجيم العمود موروثة من جدول العرض.
None	لا يتم ضبط عرض العمود تلقائيا.
AllCells	ضبط عرض العمود ليلائم محتويات كل خاناته، بما في ذلك خانة العنوان.
AllCells ExceptHeader	ضبط عرض العمود ليلائم محتويات كل خاناته، ما عدا خانة العنوان.
DisplayedCells	ضبط عرض العمود ليلائم محتويات كل خاناته الظاهرة على الشاشة حاليا، بما في ذلك خانة العنوان.
DisplayedCells ExceptHeader	ضبط عرض العمود ليلائم محتويات كل خاناته الظاهرة على الشاشة حاليا، ما عدا خانة العنوان.
ColumnHeader	ضبط عرض العمود ليلائم محتويات خانة العنوان.. وتسبب هذه القيمة خطأ في البرنامج إذا كان جدول العرض يخفي عناوين الأعمدة (ColumnHeadersVisible = False).
Fill	ضبط عرض العمود الحالي مع باقي الأعمدة لمحاولة ملء مساحة جدول العرض كلها.. وتسبب هذه القيمة خطأ في البرنامج إذا كان العمود مثبتا Frozen.

لاحظ أن المستخدم يستطيع جعل العمود يأخذ الحجم المناسب لمحتوياته، بمجرد النقر مرتين بالفأرة فوق الحافة اليمنى لخانة العنوان.

طريقة الحجم التلقائي الموروثة InheritedAutoSizeMode:

مماثلة للخاصية السابقة مع وجود اختلاف واحد، فلو كانت للخاصية AutoSizeMode القيمة NotSet، فإن الخاصية InheritedAutoSizeMode تعيد القيمة الموروثة من جدول العرض.

أولوية الملء FillWeight:

تقرأ أو تغير الوزن النسبي لكل عمود، لاستخدامه في معرفة كيفية ملء مساحة جدول العرض، وذلك عندما تكون للخاصية InheritedAutoSizeMode القيمة Fill.. والقيمة الافتراضية لهذه الخاصية هي ١٠٠، ويمكنك زيادتها أو إنقاصها، حيث يتم تكبير العمود الذي له وزن أكبر، أكثر من العمود الذي وزن أصغر.. الذي يحدث هو حساب مجموع قيم هذه الخاصية لكل الأعمدة (وليكن Sum)، ثم ضرب كل عمود في القيمة (FillWeight/Sum).. لاحظ أن أقصى قيمة للمجموع Sum يجب ألا تزيد عن ٦٥٥٣٥ وإلا حدث خطأ في البرنامج، لهذا لا تضع قيما كبيرة في هذه الخاصية، فالعبرة ليست في كبر القيمة، ولكن العبرة في كبر النسبة إلى المجموع.. فلتكن جميع القيم أصغر من ١٠٠.

طريقة الترتيب SortMode:

تقرأ أو تغير طريقة ترتيب خانات الجدول، وهي تأخذ إحدى قيم المرقم DataGridViewColumnSortMode التالية:

Automatic	هذه هي القيمة الافتراضية للجداول التي تعرض خانات نصية، وهي تسمح للمستخدم بضغط رأس العمود بالفأرة، لترتيب صفوف الجدول تبعا خانات هذا العمود.. ويظهر في خانة

	العنوان مثلث يشير رأسه إلى اتجاه الترتيب، ويمكن تغيير اتجاه الترتيب بضغط رأس العمود بالفأرة مرة أخرى.. وتسبب هذه القيمة خطأ في البرنامج إذا كانت للخاصية SelectionMode الخاصة بدول العرض القيمة FullColumnSelect أو ColumnHeaderSelect.
NotSortable	لا يمكن للمستخدم إجراء عملية الترتيب، ورغم أنه ما يزال بإمكانك إجراء الترتيب برمجيا، فلن يحتوي رأس العمود على مساحة لعرض علامة الترتيب.. وهذه هي القيمة الافتراضية للجداول التي تعرض خانات تحتوي على أزرار أو صور أو مربعات اختيار أو قوائم منسدلة أو وصلات.
Programmatic	مماثلة للقيمة السابقة، لكن رأس العمود سيحتوي على مساحة لعرض علامة الترتيب.

وتستطيع منح كل عمود في الجدول طريقة ترتيب مختلفة عن غيره.. يمكنك مثلا أن تسمح للمستخدم بترتيب الجدول عندما يضغط رأس العمود الذي يعرض المفتاح الأساسي فقط، بينما تجعل باقي الأعمدة غير قابلة للترتيب.

لاحظ أنك لو ربطت جدول العرض بقائمة List فإن ضغط رؤوس الأعمدة لن يؤدي إلى إعادة ترتيب الجدول!

السبب في هذا أن جدول العرض لا يقوم بعملية الترتيب بنفسه، وإنما يطلب من مصدر البيانات DataSource المرتبط به أن يقوم هو بتنفيذ عملية الترتيب تبعا للخاصية التي يرتبط بها العمود المضغوط.. لهذا يجب أن يكون مصدر البيانات قابلا للترتيب IsSortable = True، وهذا غير متوفر في القوائم Lists والمجموعات Collections ولا حتى في قائمة الربط BindingList.

ولحل هذه المشكلة، قمت بتتبع الأكواد التي تستخدمها ميكروسوفت، فوجدت أنها تستخدم فئة خاصة اسمها SortableBindingList لتسمح بالترتيب عند عرضها في

جدول العرض.. لكن الغريب أن ميكروسوفت جعلت هذه الفئة خاصة، ولا يمكن للمبرمج استخدامها.. لهذا عليك أن تكتب كود هذه الفئة بنفسك في مشاريعك.. وستجد هذه الفئة في المشروع Coin100Pictures ضمن أمثلة هذا الكتاب، واستخدامها بسيط جدا، فهي فئة عامة Generic Type، يمكنك أن تخصصها لأي نوع تتعامل معه من البيانات، وليس عليك أكثر من أن ترسل إلى حدث إنشائها Constructor القائمة التي تريد أن تمنحها إمكانية الترتيب.. وستجدنا نستخدمها في المشروع على الصورة:

```
var SortedList = new SortableBindingList< PictureInfo> (
      Coins.PicsInfo.Values.ToList( ));
DataGridView1.DataSource = SortedList;
```

حيث:

- PictureInfo هو نوع البيانات المخزنة في القائمة، وهي فئة خاصة بي عرّفتها في المشروع.. هذا يوضح أنك تستطيع استخدام هذه الفئة مع أي نوع، سواء كان جزءا من إطار العمل أو خاصا بك.

- Coins.PicsInfo.Values هو مجموعة Collection تحتوي على عناصر من النوع PictureInfo.. ولتحويلها إلى قائمة استخدمنا الوسيلة الإضافية ToList، وهي الوسيلة التي ستستخدمها في الغالب لإرسال القائمة إلى حدث إنشاء الفئة SortableBindingList، لتحويل المجموعات العائدة من نتائج استعلامات LinQ إلى قوائم.

هذا كل شيء.. بعد هذا جعلنا مجموعة الربط القابلة للترتيب مصدر بيانات جدول العرض بإرسالها إلى الخاصية DataGridView.DataSource.. الآن يمكنك ترتيب الجدول تبعا لأي عمود فيه بمجرد ضغط العمود بدون كتابة أي كود إضافي.

ولا تسلني مرة أخرى لماذا لم تمنحنا ميكروسوفت هذه الإمكانية مباشرة ما دامت موجودة في إطار العمل وقررت أن تخفيها عنا، فأنا لا أعلم!

ملحوظة:
في الفئة SortableBindingList ستجد أنني حولت جزءا من الكود إلى تعليق في حدث إنشاء الفئة الداخلية PropertyComparer.. هذا الجزء من الكود سبب معي خطأ عندما كنت أتعامل مع نوع بيانات موروث من نوع آخر، لأن هذا الكود اعتبر أن الصفات الموروثة ليست خاصة بهذا النوع!!.. وقد وجدت أنه لا ضرورة لهذا الكود فحذفته، وعملت الفئة بعد ذلك على ما يرام!

هل هو مرتبط بالبيانات IsDataBound:

تعيد True إذا كان العمود الحالي مرتبطا بمصدر بيانات.

اسم خاصية البيانات DataPropertyName:

تحدد اسم العمود الأصلي في مجموعة البيانات، الذي يحفظ فيه العمود الحالي بياناته.. وتأخذ هذه الخاصية قيمته تلقائيا عند ربط جدول العرض بجدول في مجموعة البيانات، وإن كان باستطاعتك تغيير قيمة هذه الخاصية يدويا لربطها بأي عمود تريده، أو أي عنصر بيانات.

لاحظ أن ضغط رأس العمود في جدول العرض لترتيب صفوفه، قد يؤدي إلى محو قيم خانات بعض الأعمدة.. فترتيب جدول العرض يؤدي إلى إعادة إنعاش الصفوف، وهذا يؤدي إلى ضياع قيم الخانات غير المرتبطة بمصدر بيانات Data Source بينما تقوم الخانات المرتبطة بمصدر البيانات بإعادة طلب القيم منه وعرضها مرة أخرى.. السبب في هذا هو أن جدول العرض مصمم لتوفير مساحة الذاكرة وتحسين الأداء، لهذا حينما يكون مرتبطا بمصدر بيانات أو يعمل في الوضع الافتراضي VirtualMode، لا يملأ كل الخانات بالبيانات، ولكنه يرسم مجموعة من الخانات مناسبة لمساحة العرض، ويحضر القيم من مصدر البيانات كلما احتاج إلى إنعاش الخانات المعروضة (عند تحريك المنزلق لعرض خانات جديدة، أو عند ترتيب الصفوف، أو عند اختفاء النافذة وإعادة عرضها.. إلخ).. لكن المشكلة تحدث حينما

تضيف بعض الأعمدة غير المرتبطة بمصدر بيانات وتملأها باستخدام الكود، أو تترك للمستخدم ملئها بنفسه، فعند ترتيب الصفوف تفقد خانات هذه الأعمدة قيمها!

ويمكنك حل هذه المشكلة بالتأكد من ربط جميع الأعمدة بمصدر البيانات.. طبعا من غير العملي إضافة أعمدة في قاعدة البيانات Database مقابلة لهذه الأعمدة، فالبيانات التي تعرضها في الغالب تكون بيانات مستنتجة أو محسوبة أو مجرد CheckBox يؤدي وظيفة معينة، أو ترقيم أو ما شابه، ومن العبث حفظ هذه البيانات في قاعدة البيانات على حساب زيادة حجمها بلا مقابل.. فما الحل إذن؟

الحل هو إضافة خاصية جديدة في الفئة التي تمثل مصدر البيانات مثل الفئات الخاصة بمجوعة البيانات محددة النوع Typed DataSet.. لا تفعل هذا في الملف المولد تلقائيا Auto Generated (الذي ينتهي اسمه بالكلمة Designer.cs) لأن أي شيء تكتبه في هذا الملف سيكون عرضة للضياع.. ولكن اضغط بزر الفأرة الأيمن على اسم الفئة في مخطط مجموعة البيانات DataSet، واضغط الأمر View Code لعرض الملف الخاص بامتداد هذه الفئة Partial Class، وأضف إليه خاصية عامة Public Property لتربط بها العمود الخاص بك.

لاحظ أن هذه الخاصية لن تظهر ضمن خصائص الكائن في نافذة مصادر البيانات Data Sources ولن تستطيع اختيار اسمها في نافذة الخصائص من ضمن خصائص مصدر الربط BindingSource، لكن رغم هذا ما زلت تستطيع ربط العمود بها بوضع اسمها يدويا في الخاصية DataPropertyName الخاصة به سواء في مصمم الأعمدة أو في الكود.

الآن يمكنك ترتيب صفوف جدول العرض دون خسارة بيانات هذا العمود.

ملحوظة:

يؤدي ترتيب جدول العرض أيضا إلى ضياع تنسيق جميع الخانات CellStyle وعودتها إلى القيم الأساسية المحفوظة في الخاصية

DataGridView.DefultCellStyle.. حل هذه المشكلة يحتاج جهـدا كبيـرا، لأن الخانة تأخذ تنسيقها من عدة خصائص مختلفة مثل:

- DataGridView.RowsDefultCellStyle .
- DataGridViewRow.DefultCellStyle .
- DataGridViewCell.CellStyle .

وغيرها من الخصائص التي يمكنك الحصول على تأثيرها النهائي من خلال الخاصـية DataGridViewCell.InheritedCellStyle

لهذا فإن محاولة حفظ قيم كل هذه الخصائص واستعادة تنسيق كل خانة بعـد ترتيـب الجدول عملية معقدة، خاصة إذا كان تنسيق الخانات يتغير أثناء تنفيذ البرنامج (كتغيـر لون خلفية أحد الصفوف عند اختيار قيمة معينة في إحدى خاناته)!

رقم العرض DisplayIndex:

تقرأ أو تغير الموضع الذي يظهر فيه العمود الحالي في جدول العرض.. هذا لا يـؤثر في شيء على ترتيب العمود في مجموعة الأعمدة، والذي توضحه الخاصية Index.

عرض الفاصل DividerWidth:

تقرأ أو تغير حجم الخط الذي يفصل العمود الحالي عن العمود التالي.. هـذا الفاصـل هو مساحة خالية تأخذ نفس لون أرضية جدول العرض، ورغم أنها تعتبر جزءا مـن العمود الحالي، إلا أنها لا تؤدي أية وظيفة من وظائفه، ما عدا العمل كفاصل شكلي.. والصورة التالية توضح تأثير وضع القيمة ١٠ في هذه الخاصية في العمود الأول:

ID	Author	CountryID	Phone	About
١٢	توفيق الحكيم	٢١		....
١٣	عباس العقاد	٢١		...
١٤	فاروق جويدة	٢١		شاعر مصري معاصر

والقيمة الافتراضية لهذه الخاصية هي صفر، لهذا لا يوجد أي فاصل بين الأعمدة، مـا عدا خطوط الشبكة العادية.

خانة العنوان HeaderCell:

ضع في هذه الخاصية كائن خانة عنوان العمود DataGridViewColumnHeaderCell الذي يحمل خصائص رأس العمود الحالي.. وسنتعرف على هذه الفئة بالتفصيل لاحقا.

ويمكنك استخدام هذه الخاصية لتغيير لون خلفية الخانة الرئيسية Header Cell لأحد أعمدة جدول العرض.. مبدئيا يجب أن تغير قيمة الخاصية DataGridView.EnableHeadersVisualStyles إلى False، فلو كانت قيمتها True فلن يكون هناك أي تأثير لو غيرت لون الخلفية في الخاصية ColumnHeadersDefaultCellStyle والخاصية RowHeadersDefaultCellStyle والخاصية Style الخاصة بالخانة الرئيسية للعمود أو الصف:

```
Dgv1.EnableHeadersVisualStyles = false;
```

بعد هذا، يمكنك أن تغير لون خلفية الخانة الرئيسية للعمود رقم X كالتالي:

```
Dgv1.Columns[X].HeaderCell.Style.BackColor = Color.Red;
```

ويمكنك تطبيق نفس الطريقة لتغيير لون النص ForeColor.

نص العنوان HeaderText:

تقرأ أو تغير النص الذي يظهر في خانة عنوان العمود الحالي.

قالب الخانة CellTemplate:

ضع في هذه الخاصية كائن الخانة DataGridViewCell الذي تريد استخدامه كقالب تستمد منه الخانات التي تضاف إلى العمود خصائصها.

نوع الخانة CellType:

تعيد كائن النوع Type، الذي يوضح نوع الخانة المستخدمة كقالب في الخاصية CellTemplate.. لاحظ أن الخاصية CellTemplate من النوع الفئة الأم

DataGridViewCell، بينما تعيد الخاصية CellType النوع الفعلي المشتق من هذه الفئة الأم.. وسنتعرف على الفئة DataGridViewCell ومشتقاتها لاحقا.

كما تمتلك هذه الفئة الوسيلة التالية:

معرفة العرض المفضل GetPreferredWidth:

تعيد أنسب عرض للعمود تبعا للمواصفات المطلوبة، وهي تأخذ معاملين:

- إحدى قيم المرقم DataGridViewAutoSizeColumnMode، تحدد طريقة الحجم التلقائي للعمود.
- معامل منطقي، إذا جعلت قيمته True، فسيتم تقدير العرض المناسب للعمود بافتراض أن ارتفاع خاناته سيظل ثابتا، أما إذا جعلته False، فسيدخل في الاعتبار إمكانية تغيير ارتفاعات الصفوف، وهل هناك التفاف لأسطر الخانات Word wrap أم لا، حيث ستتم المحافظة على النسبة بين عرض العمود وارتفاع خاناته.

والفئة DataGridViewColumn تعمل كفئة أم لكل من الفئات التالية:

١- عمود مربعات النصوص DataGridViewTextBoxColumn.

٢- عمود الأزرار DataGridViewButtonColumn.

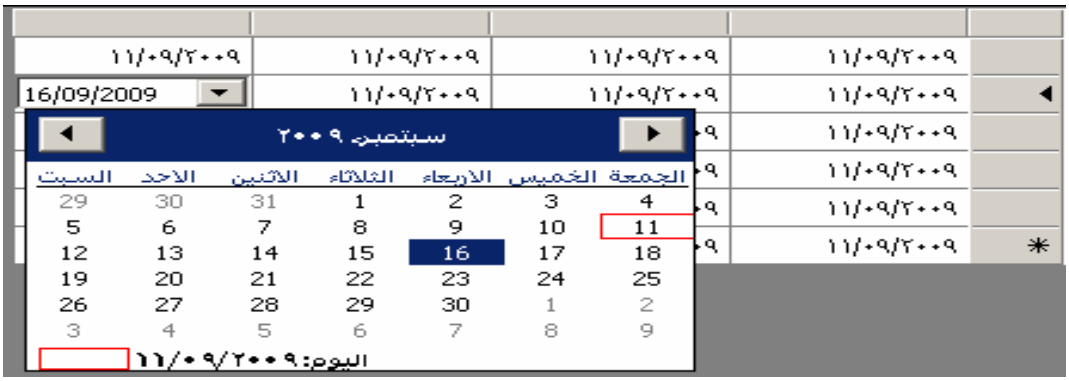
٣- عمود مربعات الاختيار DataGridViewCheckBoxColumn.

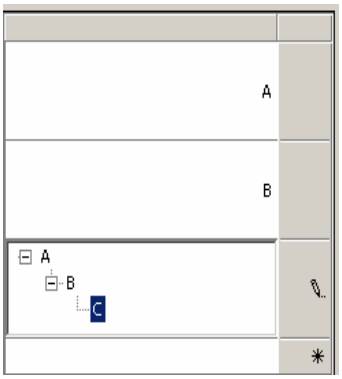
٤- عمود القوائم المركبة DataGridViewComboBoxColumn.

٥- عمود الصور DataGridViewImageColumn.

٦- عمود الوصلات DataGridViewLinkColumn.

ولكن الأمر لا يتوقف عند هذه الأنواع، فبإمكانك وراثة هذه الفئة لإنشاء أعمدة تعرض خاناتها أي أداة أخرى من أدوات الويندوز.. ولو ضغطت الزر "عمود تواريخ" في المشروع DataGridColumnTypes فسيضاف إلى جدول العرض عمود تعرض كل خانة من خاناته أداة اختيار التاريخ DateTimePicker.





ولو فتحت متصفح المشاريع، لوجدت فيه فئة اسمها CalendarColumn تتوارث الفئة DataGridViewColumn.. وكود هذه الفئة بسيط للغاية، فهي تستبدل Override عنصرين فقط من عناصر الفئة الأم: حدث الإنشاء New والخاصية CellTemplate، وذلك للتعامل مع النوع الجديد لخانات هذا العمود، وهو فئة جديدة أنشأناها بأنفسنا أيضا اسمها CalenderCell مهمتها عرض أداة اختيار التاريخ.. وسنتعرف على فكرة هذه الفئة لاحقا.

وبنفس الطريقة، أمكننا إنشاء عمود تعرض كل خانة فيه شجرة TreeView، ويمكنك إضافة هذا العمود إلى الجدول بضغط الزر "عمود أشجار" في نفس المشروع.

لاحظ أن هذه الشجرة غير عملية، فهي تظهر بكاملها داخل الخانة، وهو ما يحتاج إلى جعل مساحة الخانة كبيرة لضمان ظهور فروع الشجرة بشكل مقبول.. ويمكنك حل هذه المشكلة باستخدام شجرة منسدلة، وقد شرحنا فكرتها في كتاب "برمجة نماذج الويندوز" في المشروع TreeComboBox.. على كل حال، يمكنك إضافة هذا النوع من الأعمدة إلى جدول العرض بضغط الزر "عمود أشجار منسدلة".

عمود مربعات النصوص
DataGridViewTextBoxColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewTextBoxCell، وهي خانات تعرض لافتات، وعند تحرير أي خانة منها، فإنها تعرض مربع نص.. ويمكنك ضغط الزر "عمود النصوص" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

ويعتبر عمود مربعات النص النوع الافتراضي الذي يضيفه جدول العرض عند ربطه بمصدر بيانات.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصية التالية:

أقصى طول للمدخلات MaxInputLength:

تحدد أقصى عدد من الحروف تقبله كل خانة في العمود.. والقيمة الافتراضية لهذه الخاصية هي ٣٢٧٦٧، ولو جعلتها صفرا فهذا يعني السماح للمستخدم بكتابة الحد الأقصى من الحروف، وهو يتجاوز ٢ مليار حرف.

اللهم ارحم أبي واغفر له وكفر عنه سيئاته

فئـــة عمـــود الأزرار
DataGridViewButtonColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناتـــه مـــن النـــوع DataGridViewButtonCell، وهي خانـــات تحمـــل كـــل منهـــا زرا Button يمكـــن للمســـتخدم ضـــغطه.. ويمكنـــك ضـــغط الـــزر "عمـــود أزرار" فـــي المشـــروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

النص Text:

تقرأ أو تغير النص الافتراضي المعروض على جميع الأزرار في خانات العمود.

استخدام نص العمود لقيمة الزر UseColumnTextForButtonValue:

إذا جعلت قيمة هذه الخاصية False (وهي القيمة الافتراضية)، فلـــن يظهـــر الـــنص الموجود في الخاصية Text على أزرار خانات العمود، وستكون كل خانـــة مســـئولة عن وضع النص الخاص بالزر الذي تعرضه.

طريقة العرض المسطح FlatStyle:

تحدد طريقة عرض الزر، وهي تأخذ إحدى قيم المرقم FlatStyle التالية:

Standard	يظهر الزر مجسما بالطريقة القياسية المعتادة ثلاثية الأبعاد.. هذه هـــي القيمة الافتراضية.
Flat	يظهر الزر مسطحا، ويتغير لون خلفيته عند مرور الفأرة فوقه وعنـــد ضغطه.
Popup	يظهر الزر مسطحا، لكن عند المرور فوقه بالفأرة يبـــرز إلـــى أعلـــى ويصير مجسما.
System	يظهر الزر تبعا لاختيارات المستخدم ضمن نظام الويندوز على جهازه.

فئة عمود مربعات الاختيار
DataGridViewCheckBoxColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewCheckBoxCell، وهي خانات تحمل مربع اختيار CheckBox.. ويمكنك ضغط الزر "عمود مربعات اختيار" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.

٢- والصيغة الثانية تستقبل معاملا منطقيا Boolean، يتم إرسال قيمته إلى الخاصية ThreeState التي سنتعرف عليها بعد قليل.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

طراز العرض المسطح FlatStyle:

مماثلة لتلك الخاصة بعمود الأزرار.

ثلاثي الحالة ThreeState:

إذا جعلت قيمة هذه الخاصية True فسيكون مربع الاختيار ثلاثي الحالة (متضمنا الحالة الوسيطة غير المحددة Indeterminate).. هذا معناه أن المستخدم إذا ضغط المربع مرة فستوضع به علامة الاختيار، وإذا ضغطه مرة أخرى فسيصير في الحالة الوسيطة (بعلامة اختيار غائمة) وإذا ضغطه مرة ثالثة فستزال علامة الاختيار.

أما إذا جعلت قيمتة هذه الخاصية False (وهذه هي القيمة الافتراضية)، فسيكون لمربع الاختيار حالتان فقط (Checked – Unchecked)، وهذا معناه أن المستخدم

إذا ضغط مربع الاختيار مرة فستوضع به علامة الاختيار، وإذا ضغطه مرة أخرى فستزال منه هذه العلامة.

القيمة الخاطئة FalseValue:

تستقبل كائنا Object يحتوي على القيمة المناظرة لحالة عدم الاختيار Unchecked.

القيمة الصحيحة TrueValue:

تستقبل كائنا Object يحتوي على القيمة المناظرة لحالة الاختيار Checked.

القيمة غير المحددة IndeterminateValue:

تستقبل كائنا Object يحتوي على القيمة المناظرة للحالة الوسيطة غير المحددة Indeterminate.

ويمكنك استخدام الخصائص FalseValue و TrueValue و IndeterminateValue إذا كان العمود الحالي مرتبطا بمصدر بيانات.. افرض على سبيل المثال أن لديك مصدر بيانات يحتوي على ثلاثة أرقام هي ٠، ١، ٢، وقمت بربطها بعمود مربعات اختيار.. في هذه الحالة افعل ما يلي:

- ضع في الخاصية FalseValue القيمة ٠ لتخبر العمود أن الخانات المناظرة للقيمة ٠ في مصدر البيانات لن توضع بها علامات الاختيار.
- ضع في الخاصية TrueValue القيمة ١ لتخبر العمود أن الخانات المناظرة للقيمة ١ في مصدر البيانات ستوضع بها علامات الاختيار.
- ضع في الخاصية IndeterminateValue القيمة ٢ لتخبر العمود أن الخانات المناظرة للقيمة ٢ في مصدر البيانات ستوضع بها علامات اختيار غائمة دلالة على أنها حالة وسيطة غير محددة.

ولكن: كيف يمكنك أن تعرف أن المستخدم غير حالة الاختيار Checked في أي خانة في عمود من هذا النوع موضوع في جدول العرض؟

عندما يضغط المستخدم مربع الاختيار في أي خانة في هذا العمود، ينطلق الحدث DataGridView.CellContentClick الخاص بجدول العرض.. في هذا الحدث افعل ما يلي:

- تأكد أن الخاصية e.ColumnIndex تشير إلى رقم العمود DataGridViewCheckBoxColumn، لأن هذا الحدث ينطلق عند ضغط خانات أعمدة من أنواع أخرى.
- احصل على الخانة الحالية التي تسببت في إطلاق هذا الحدث باستخدام الخاصية DataGridView.CurrentCell.. أو يمكنك استخدام التعبير التالي للحصول على هذه الخانة (افترض أن اسم جدول العرض Dgv):

```
var Cell = Dgv.Rows[e.RowIndex].Cells[e.ColumnIndex];
```

أنا أفضل الطريقة الأخيرة، ففي بعض الأحيان يمكن أن تشير الخاصية CurrentCell إلى خانة محددة أخرى، بينما ضغط الخانة التي أطلقت الحدث ما زال لم يجعلها الخانة الحالية.. لهذا إذا وجدت رقم الصف والعمود في البيانات المرافقة لأي حدث من أحداث جدول العرض، فالآمن أن تستخدمها!

- لا تستخدم الخاصية Cell.Value لمعرفة قيمة الخانة، فهي لا تتغير إلا بعد مغادرة الصف الذي توجد فيه في جدول العرض!!.. الخدعة هنا هي استخدام الخاصية Cell.EditedFormattedValue بدلا منها:

```
if ((bool)Cell.EditedFormattedValue)
   Console.WriteLine("تم اختيار هذه الخانة");
```

ولا تحاول استخدام الحدث CellValueChanged (فكما قلنا فإن الخانة لا تحدث قيمتها إلا بعد مغادرة الصف)!!.. أيضا لا تحاول استخدام معالج للحدث CheckedChanged الخاص بمربع الاختيار CheckBox الموجود في هذه الخانة، ولكن لم أستطع، لأن هذه الخانات لا تتسبب في إطلاق الحدث DataGridView.EditingControlShowing!

كما أني أنصح بالتالي:

إذا كان العمود DataGridViewCheckBoxColumn مرتبطا بمصدر بيانات، فليس عليك أن تقلق بشأنه، فهو سيحدّث سجلات مصدر البيانات DataSource بطريقة صحيحة طبقا للخانات التي اختارها المستخدم أو أزال منها الاختيار.. أما إذا لم يكن هذا العمود مرتبطا بمصدر بيانات، وكان عليك أداء وظيفة معينة تبعا لقيم خاناته، فالحل الأسهل والأضمن هو أن تنفذ هذه الوظيفة مرة واحدة في إجراء اسمه SaveChanges مثلا، يتم استدعاؤه عندما يضغط المستخدم زر الحفظ، أو عندما يحاول إغلاق النافذة وتسأله إن كان يريد حفظ التغييرات.. في هذا الإجراء كل ما ستفعله هو المرور عبر كل صفوف جدول العرض، وفحص قيمة كل خانة في هذا العمود (افترض أنه العمود رقم I)، واتخاذ الفعل المناسب تبعا لحالتها، على الصيغة:

```
Dgv.EndEdit( );
foreach (DataGridViewRow Row in Dgv.Rows)
{
    if ((bool)Row.Cells[I].Value)
       الوظيفة الخاصة بكون الخانة مختارة //
    else
       الوظيفة الخاصة بكون الخانة غير مختارة //
}
```

لاحظ استدعاءنا للوسيلة EndEdit في بداية الكود لإنهاء تحرير أي خانة ما زالت في وضع التحرير.

فئـــة عمـــود الصـــور
DataGridViewImageColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناتـــه مـــن النـــوع DataGridViewImageCell، وهي خانات تعرض كل منها صورة.. ويمكنـــك ضـــغط الزر "عمود الصور" في المشروع DataGridColumnTypes لإضافة عمود مـــن هـــذا النوع إلى جدول العرض.

ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.

٢- والصيغة الثانية تستقبل معاملا منطقيا Boolean، يتم إرسال قيمته إلى الخاصـــية valuesAreIcons التي سنتعرف عليها بعد قليل.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

الوصف Description:

ضع في هذه الخاصية نصا يصف الصور أو الأيقونات الموجودة في خانات العمود.

القيم أيقونات ValuesAreIcons:

إذا جعلت قيمة هذه الخاصية True، فستعرض خانات العمود الأيقونة الموجودة فـــي الخاصية Icon.. أما إذا جعلتها False (وهي القيمة الافتراضية)، فستعرض خانـــات العمود الصورة الموجودة في الخاصية Image.

الأيقونة Icon:

ضع في هذه الخاصية كائن الأيقونة Icon الذي تريد عرضه في خانات العمود عندما تكون للخاصية ValuesAreIcons القيمة True.

الصورة Image:

ضع في هذه الخاصية كائن الصورة Image الذي تريد عرضه في خانات العمود عندما تكون للخاصية ValuesAreIcons القيمة False.

مخطط الصورة ImageLayout:

تحدد طريقة عرض الصورة في خانات العمود، وهي تأخذ إحدى قيم المرقم DataGridViewImageCellLayout التالية:

NotSet	القيمة غير محددة (متروكة لكل خانة على حدة).
Normal	يتم عرض الصورة كاملة في منتصف كل خانة.. هذه هي القيمة الافتراضية.
Stretch	يتم مط الصورة لتلائم عرض وارتفاع كل خانة.. هذا معناه أن الصورة ستملأ كل مساحة الخانة، ولكن هذا قد يؤدي إلى تشويهها.
Zoom	يتم تكبير أو تصغير الصورة لتلائم عرض أو ارتفاع الخانة، مع المحافظة على النسبة الأصلية بين ارتفاع الصورة وعرضها، مما يمنع تشويهها.

لاحظ أن جدول البيانات في الوضع الافتراضي يعرض عمود طابع الوقت TimeStamp تلقائيا في عمود صور، لمجرد أن هذا العمود يحمل بيانات ثنائية.. وقد رأينا كيف سبب لنا هذا مشاكل كثيرة فيما سبق.

ولحل هذه المشكلة، عليك تغير نوع هذا العمود بعد ربط جدول العرض بمجموعة البيانات.. لفعل هذا في وقت التصميم، اتبع الخطوات التالية:

- حدد جدول العرض على النموذج وافتح نافذة الخصائص.
- اضغط الرابط Edit Columns الموجود في الجزء السفلي من نافذة الخصائص.. سيفتح هذا نافذة تحرير مجموعة الأعمدة.

- حدد عمود طابع الوقت، لعرض خصائصه في الجزء الأيمن من النافذة.
- حدد الخاصية ColumnType تحت الشريط Design، واضغط زر الإسدال، واختر من القائمة النوع DataGridTextBoxColumn بدلا من النوع DataGridImageColumn.. هذا سيمنع الأخطاء التي تحدث بسبب محاولة رسم البيانات الثنائية كصورة.. لاحظ أن العمود لا يملك فعليا خاصية اسمها ColumnType، وما يفعله المصمم هو حذف العمود القديم، وإنشاء عمود جديد من النوع الذي اخترته في هذه الخاصية مع نسخ باقي خصائص العمود القديم إلى العمود الجديد.. لهذا لا تستطيع تغيير نوع العمود من الكود، إلا بحذفه وإنشاء عمود جديد.
- أو يمكنك أن تضع القيمة False في الخاصية Visible لإخفاء العمود وحل المشكلة من جذورها، وهذا ما فعلناه من الكود في المشاريع السابقة:

DGAuthors.Columns["RowVersion"].Visible = false;

فئة عمود الوصلات DataGridViewLinkColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewLinkCell، وهي خانات تعرض كل منها رابطا (وصلة) Link.. ويمكنك ضغط الزر "عمود الوصلات" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

النص Text:

تقرأ أو تغير نص الوصلة المعروضة في خانات العمود.

استخدم نص العمود كقيمة للوصلة UseColumnTextForLinkValue:

إذا جعلت قيمة هذه الخاصية False (وهي القيمة الافتراضية)، فلن يعرض النص الموجود في الخاصية Text كوصلات في خانات العمود، وستكون كل خانة مسئولة عن كتابة نص الوصلة التي تعرضها.

سلوك الرابط LinkBehavior:

تحدد كيف يبدو شكل الرابط، وهي تأخذ إحدى قيم المرقم LinkBehavior:

AlwaysUnderline	الرابط تحته خط دائما.
HoverUnderline	لا يوضع خط تحت الرابط إلا حينما يمر فوقه مؤشر الفأرة.
NeverUnderline	لا يوضح خط تحت الرابط مطلقا.
SystemDefault	خيارات متصفح الإنترنت ونظام الويندوز هي التي توضح كيف يبدو الرابط.

لون الرابط LinkColor:

تحدد لون الروابط المعروضة في العمود.. وفي الوضع التلقائي يكون هذا اللون هو الأزرق.

لون الرابط الفعال ActiveLinkColor:

تحدد لون الرابط أثناء ضغطه بالفأرة.. في الوضع التلقائي يكون هذا اللون هو الأحمر.

لون الرابط المُزار VisitedLinkColor:

تحدد لون الرابط بعد أن يضغطه المستخدم.. في الوضع التلقائي يكون هذا اللون هو الأحمر الغامق.

تتبع حالة الزيارة TrackVisitedState:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتغير لون الرابط بعد ضغطه، ليأخذ اللون المحدد في الخاصية VisitedLinkColor.

وقد استخدمنا عمود الوصلات في المشروع CustomDataSet لعرض وصلة "عرض درجات الطالب".

اللهم ارحم أبي واغفر له وكفر عنه سيئاته

فئة عمود القوائم المركبة
DataGridViewComboBoxColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewComboBoxCell، وهي خانات تحمل قوائم منسدلة Combo Boxes.. ويمكنك ضغط الزر "عمود قوائم مركبة" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

طراز العرض المسطح FlatStyle:

مماثلة لتلك الخاصة بعمود الأزرار.

عرض القائمة المنسدلة DropDownWidth:

تحدد عرض القائمة المسندلة.. والقيمة الافترضية لهذه الخاصية هي ١ ولا تقبل أقل منها، لكن عليك أن تلاحظ أن وضع أي قيمة أصغر من عرض العمود في هذه الخاصية سيجعلها بدون تأثير، لأن القائمة المنسدلة يمكن فقط أن تكون مساوية لعرض العمود أو أكبر منه!.. هذا يضمن لك أن القائمة المنسدلة ستأخذ نفس عرض العمود تلقائيا لو قام المستخدم بزيادة عرضه.. هذا معناه أن عرض القائمة المنسدلة يحسب فعليا من العلاقة التالية:

Actual Width = Max (DropDownWidth, Col Width)

إكمال تلقائي AutoComplete:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم اقتراح التكملة المناسبة للحروف التي يكتبها المستخدم في القائمة المركبة.

مصدر البيانات DataSource:

ضع في هذه الخاصية الكائن الذي سيعمل كمصدر بيانات، لاستخدامه في ملء القائمة المركبة بالعناصر، مثل اسم جدول الدول Countries:

```
Col.DataSource = Ds.Tables["Countries"];
```

لاحظ أن العمود في هذه الحالة يتعامل مع مصدرين من مصادر البيانات:

- مصدر بيانات جدول العرض نفسه (الذي يرتبط بجدول المؤلفين مثلا).
- مصدر بيانات القائمة المركبة (الذي يرتبط بجدول الدول).

عنصر العرض DisplayMember:

ضع في هذه الخاصية اسم عنصر البيانات، لعرض قيمته في القائمة المركبة.. مثل العمود Name في جدول الدول:

```
Col.DisplayMember = "Name";
```

ملحوظة:
لو استخدمت مصفوفة نصية كمصدر للبيانات، فلست بحاجة إلى الخاصية DisplayMember فعناصر المصفوفة نفسها ستكون عناصر العرض، وهذا هو ما فعلناه عند وضع القيم في عمود القوائم المركبة في المشروع DataGridColumnTypes.. لاحظ وأنت تجرب المشروع أن ضغط زر إسدال القائمة لا يعمل إلا إذا كانت الخانة التي يوجد فيها محددة أولا، لهذا فإن أول ضغطة ستعمل على تحديد الخانة، وثاني ضغطة ستسدل القائمة.

عنصر القيمة ValueMember:

تعمل هذه الخاصية مع الخاصية DataPropertyName الموروثة من فئة العمود، للربط بين جدولين.. مثلا: في المشروع UpdateErrors2 حذفنا العمود CountryID وأضفنا بدلا منه عمود قوائم مركبة يعرض أسماء الدول، بدلا من أن نعرض للمستخدم رقم الدولة التي ينتمي إليها المؤلف:

```
// حذف عمود أرقام الدول
DataGridView1.Columns.Remove("CountryID");
// تعريف عمود قوائم منسدلة
var Col = new DataGridViewComboBoxColumn( );
Col.Name = "Country";
// ملء العمود بأسماء الدول
Col.DataSource = Ds.Tables["Countries"];
Col.DisplayMember = "Name";
// إضافة العمود إلى جدول العرض
DataGridView1.Columns.Insert(2, Col);
```

هذا سهل ومفهوم، فالقائمة الآن مرتبطة بجدول الدول وتعرض قيم الحقل Name.. لكن المشكلة أن المستخدم لو اختار الدولة مصر مثلا، فعلينا وضع رقم هذه الدولة (وهو ١٢) في الحقل CountryID في جدول المؤلفين، وهذا قد يحتاج إلى كتابة بعض الكود يدويا.

لا تقلق.. لن نكتب أي كود لفعل هذا.. كل ما علينا هو إخبار القائمة المركبة أنها ستحفظ قيمة العمود ID التابع لمصدر بيانات العمود الحالي (وهو هنا جدول الدول)، في العمود CountryID التابع لمصدر البيانات جدول العرض كله (وهو هنا جدول المؤلفين)، وذلك كالتالي:

```
Col.ValueMember = "ID";
Col.DataPropertyName = "CountryID";
```

العناصر Items:

تعيد مجموعة من النوع ObjectCollection، وهي مجموعة تمثل الواجهة IList معرفة داخل الفئة DataGridViewComboBoxCell، وكل عنصر من عناصرها من النوع Object، وهي تحتوي على عناصر القائمة المركبة.. لاحظ إن استخدام الخاصية DataSource يلغي عمل الخاصية Items.. مثلا: لو أضفت بعض العناصر إلى المجموعة Items ثم وضعت مصدر بيانات في الخاصية

DataSource، فستعرض القائمة المنسدلة عناصر مصدر البيانات وتتجاهل العناصر التي أضفتها إلى الخاصية Items.. أما إذا وضعت مصدر البيانات في الخاصية DataSource أولا ثم حاولت إضافة بعض العناصر إلى الخاصية Items فسيحدث خطأ في البرنامج!

طريقة العرض DisplayStyle:

تحدد كيف تظهر القائمة المركبة في خانات العمود، وهي تأخذ إحدى قيم المرقم DataGridViewComboBoxDisplayStyle التالية:

ComboBox	حتى عندما لا تكون الخانة محددة، ستظل تعرض القائمة المركبة.
DropDownButton	عندما لا تكون الخانة محددة، ستعرض زر إسدال القائمة المركبة بمفرده.
Nothing	عندما لا تكون الخانة محددة، فإنها لا تعرض القائمة المركبة ولا حتى زر إسدالها، ولا تظهر القائمة المركبة إلا في الخانة المحددة فقط.

طريقة عرض الخانة الحالية فقط DisplayStyleForCurrentCellOnly:

إذا جعلت قيمة هذه الخاصية True، فستؤثر الخاصية DisplayStyle على الخانة المحددة في العمود الحالي فقط دون باقي خانات العمود.. والقيمة الافتراضية لهذه الخاصية هي False.

لاحظ أن وضع القيمة Nothing في الخاصية DisplayStyle مع وضع القيمة True في الخاصية DisplayStyleForCurrentCellOnly سيخفي القائمة المركبة وزر الإسدال من كل خانات العمود بما في ذلك الخانة المحددة!.. في هذه الحالة على المستخدم نقر الخانة المحددة مرتين بالفأرة لإظهار القائمة المركبة!

أقصى عدد من العناصر المسدلة MaxDropDownItems:

تقرأ أو تغير عدد العناصر التي يمكن عرضها في القائمة المندسلة بدون الحاجة إلى منزلق رأسي، أما إذا زاد عدد العناصر عن هذا العدد فسيظهر منزلق رأسي ليتيح للمستخدم عرض باقي العناصر.. وتقبل هذه الخاصية قيمة ما بين ١ و ١٠٠، وقيمتها الافتراضية ٨.

مرتبة Sorted:

إذا جعلت قيمة هذه الخاصية True، فسيتم ترتيب عناصر القائمة المرتبة أبجديا.

ملحوظة:
الخصائص السابقة تؤثر على جميع القوائم المركبة الموجودة في كل خانات العمود، كما تؤثر على القائمة المركبة الخاصة بقالب الخانة الموجود في الخاصية CellTemplate.. لكن يظل بإمكانك تغيير خصائص القائمة المركبة لكل خانة على حدة، كما سنرى لاحقا.

فئة مجموعة صفوف جدول العرض
DataGridViewRowCollection Class

هذه المجموعة تمثل الواجهة IList، وهي تحتوي على عناصر من نوع صف جدول العرض DataGridViewRow.

ويستقبل حدث إنشاء هذه الفئة كائن جدول العرض DataGridView الذي تنتمي إليه المجموعة.. لكنك لا تحتاج إلى إنشاء نسخة جديدة منها، لأنك تتعامل مع مجموعة صفوف جدول العرض من خلال الخاصية Rows الخاصة به.

وإضافة إلى العناصر التقليدية للمجموعات، تمتلك هذه الفئة الوسائل التالية:

إضافة Add:

تضيف صفا جديدا إلى جدول العرض وتعيد عددا صحيحا يمثل موضعه في مجموعة الصفوف، ولها الصيغ التالية:

١- الصيغة الأولى بدون معاملات، وهي تضيف صفا له القيم الافتراضية.

٢- الصيغة الثانية تستقبل كائن الصف DataGridViewRow لإضافته إلى المجموعة.

٣- الصيغة الثالثة تستقبل مصفوفة كائنات Object Array، وتنشئ صفا جديدا وتضع في خاناته قيم هذه المصفوفة بنفس الترتيب، وتضيفه إلى جدول العرض.

وتتسبب هذه الوسيلة في حدوث خطأ في البرنامج في الحالات التالية:

- إذا لم تكن هناك أية أعمدة في جدول العرض.
- إذا كان عدد خانات الصف أكبر من عدد أعمدة الجدول.
- إذا كان الصف الجديد مثبتا Frozen، لكنه سيضاف بعد صفوف غير مثبتة.

- إذا كان جدول العرض يقوم بتحديد كل خاناته في تلك اللحظة أو يزيل تحديدها.
- إذا تم استدعاء الوسيلة Add من داخل أي من الأحداث التالية:
CellEnter, CellLeave, CellValidating, CellValidated, RowEnter, RowLeave, RowValidated, RowValidating.
- إذا كان جدول العرض في الوضع الافتراضي (DataGridView.VirtualMode = true).
- إذا كان جدول العرض مرتبطا بمصدر بيانات (DataGridView.DataSource <> Nothing).

لاحظ أن الصف الجديد الذي أضفته لا يتم ترتيبه ضمن صفوف الجدول، ولو أردت أن يوضع في الترتيب الصحيح، فعليك استدعاء الوسيلة DataGridView.Sort في الحدث DataGridView.RowsAdded لإعادة ترتيب صفوف الجدول.

إضافة نسخة AddCopy:

تستقبل رقم صف في مجموعة الصفوف، حيث تنسخ صفا جديدا مماثلا له وتضيفه إلى المجموعة، وتعيد موضع إضافته.

إضافة نسخ AddCopies:

مماثلة للوسيلة السابقة، إلا أنها تنسخ من الصف الموضح رقمه في المعامل الأول، عدد النسخ الموضح في المعامل الثاني، وتضيف هذه الصفوف المنسوخة إلى المجموعة، وتعيد رقم آخر صف تمت إضافته.

إدراج Insert:

تدرج صفا أو أكثر في موضع معين داخل مجموعة الصفوف.. ولهذه الوسيلة عدة صيغة، كلها تشترك في أن معاملها الأول يستقبل الموضع الذي سيتم إدراج الصف فيه، وتختلف في المعامل الثاني كما يلي:

١- يستقبل المعامل الثاني للصيغة الأولى عدد الصفوف الجديدة التي سيتم إدراجها في المجموعة.
٢- يستقبل المعامل الثاني للصيغة الثانية مصفوفة كائنات Object Array، حيث يتم إنشاء صف جديد وإضافته إلى موضع الإدراج، ووضع قيم المصفوفة في خانات هذا الصف بنفس الترتيب.
٣- يستقبل المعامل الثاني للصيغة الثالثة كائن الصف DataGridViewRow المراد إدراجه.

إدراج نسخة InsertCopy:

تنسخ صفا من المجموعة وتدرجه في موضع معين فيها، ولها معاملان:

- المعامل الأول يستقبل رقم الموضع المراد إدراج الصف الجديد فيه.
- المعامل الثاني يستقبل رقم الصف المراد نسخه.

إدراج نسخ InsertCopies:

مشابهة للوسيلة السابقة، إلا أنها تزيد عليها بمعامل ثالث، يستقبل عدد النسخ التي سيتم إنشاؤها من الصف الموجود في الموضع المحدد في المعامل الثاني، ليتم إدراجها في مجموعة الصفوف بدءا من الموضع المحدد في المعامل الأول.

معرفة حالة الصف GetRowState:

تخبرك بحالة الصف الذي أرسلت إليها رقمه في المجموعة كمعامل.. وهي تعيد إحدى قيم المرقم DataGridViewElementStates الذي تعرفنا عليه من قبل عند التعرف على الخاصية DataGridViewElement.State.

معرفة أول صف GetFirstRow:

تعيد موضع أول صف له الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.. وتعيد هذه الوسيلة -١ إذا لم تجد صفا له الحالة المطلوبة.

وتوجد صيغة أخرى لهذه الوسيلة لها معامل ثانٍ هو أيضا من نوع المرقم DataGridViewElementStates، ولكنه يستقل الحالة التي يجب ألا يكون عليها الصف.. والمثال التالي يعيد رقم أول صف معروض في جدول العرض لكنه غير محدد:

```
Console.WriteLine(DataGridView1.Rows.GetFirstRow(
        DataGridViewElementStates.Displayed,
        DataGridViewElementStates.Selected));
```

معرفة آخر صف GetLastRow:

مماثلة للصيغة الأولى للوسيلة السابقة، ولكنها تعيد موضع آخر صف له الحالة المرسلة كمعامل.

معرفة الصف التالي GetNextRow:

تعيد رقم الصف الذي له حالة معينة، وهي تستقبل المعاملين التاليين:

١- رقم الصف الذي سيبدأ البحث منه.

٢- إحدى قيم المرقم DataGridViewElementStates توضح الحالة التي يجب أن يمتلكها الصف المطلوب.

وتوجد صيغة ثانية لهذه الوسيلة، لها معامل ثالث، يستقبل قيمة من قيم المرقم DataGridViewElementStates توضح الحالة التي يجب ألا يمتلكها الصف المطلوب.

والمثال التالي يعرض أرقام كل الصفوف المعروضة والتي لا يستطيع المستخدم تغيير حجمها:

```
var Rows = DataGridView1.Rows;
var Pos = Rows.GetFirstRow(
      DataGridViewElementStates.Displayed,
      DataGridViewElementStates.Resizable);
while (Pos != -1)
{
      MessageBox.Show(Pos.ToString( ));
      Pos = Rows.GetNextRow(Pos,
            DataGridViewElementStates.Displayed,
            DataGridViewElementStates.Resizable);
}
```

معرفة الصف السابق GetPreviousRow:

مماثلة للوسيلة السابقة في صيغتيها، ولكنها تبحث في المجموعة من الخلف إلى الأمام، بدءا من الموضع الذي أرسلته إلى المعامل الأول.. دعنا نكتب المثال السابق باستخدام هذه الوسيلة، لعرض أرقام الصفوف بترتيب عكسي:

```
var Rows = DataGridView1.Rows;
var Pos = Rows.Count -1;
do
{
   Pos = Rows.GetPreviousRow(Pos,
         DataGridViewElementStates.Displayed,
         DataGridViewElementStates.Resizable);
   if (Pos == -1)
      break;
   MessageBox.Show(Pos.ToString( ));
} while (true);
```

لاحظ أننا بدأنا البحث من موضع يساوي عدد الصفوف، رغم أن ترقيم الصفوف يبدأ من الصفر وينتهي عند عدد الصفوف -١.. السبب في هذا هو أن هناك صفا زائدا (هو الصف الجديد الذي تجاوره العلامة * في جدول العرض)، لهذا يمكن أن نأخذه في اعتبارنا.. أما لو حاولت أن تبدأ البحث من موضع يزيد على عدد الصفوف (مثل Count + 1) فسيحدث خطأ في البرامج.

معرفة عدد الصفوف GetRowCount:

تعيد عدد صفوف الجدول التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.. والمثال التالي يعرض عدد الصفوف المحددة في جدول العرض:

```
Console.WriteLine(DataGridView1.Rows.GetRowCount(
        DataGridViewElementStates.Selected));
```

معرفة ارتفاع الصفوف GetRowsHeight:

تعيد مجموع ارتفاع الصفوف التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.

صف مشترك SharedRow:

تعيد كائن الصف DataGridViewRow الذي يمثل الصف المشترك الموجود في الموضع المرسل كمعامل.. وسنتعرف على مفهوم الصفوف المشتركة Shared Rows بالتفصيل لاحقا.

كما تمتلك هذه المجموعة الحدث التالي:

المجموعة تغيرت CollectionChanged:

ينطلق هذا الحدث عند حدوث تغير في عناصر المجموعة بالحذف أو الإضافة.. والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs وقد تعرفنا عليه من قبل عند التعرف على مجموعة الجداول DataTableCollection.

فئة صف جدول العرض DataGridViewRow Class

هذه الفئة ترث الفئة DataGridViewBand، وهي تمثل أحد صفوف جدول العرض.. وتمتلك هذه الفئة الخصائص التالية:

كائن سهولة الوصول AccessibilityObject:

تعيد كائن تسهيل الوصول AccessibleObject المستخدم مع الصف الحالي لتسهيل تعامل ذوي الاحتياجات الخاصة (كضعاف البصر) مع بيانات هذا الصف.. هذا الموضوع خارج نطاق هذا الكتاب.

الخانات Cells:

تعيد مجموع الخانات DataGridViewCellCollection التي تحتوي على خانات الصف الحالي.. والمثال التالي يعرض قيمة الخانة الأولى في الصف الأول، مع ملاحظة أن صف رءوس الأعمدة، وعمود رءوس الصفوف لا يدخلان في الترقيم:

```
Console.WriteLine(DataGridView1.Rows[0].Cells[0].Value);
```

لكني أنصحك ألا تستخدم رقم العمود للإشارة إلى الخانة كما في المثال السابق، لأنك قد تغير موضع العمود بعد ذلك أو تضيف أعمدة أخرى قبله تؤدي إلى تغيير ترقيمه، مما يضع عليك عبء إعادة تغير كل الأكواد التي تحتوي على أرقام الأعمدة.

أنصحك أيضا ألا تستخدم اسم العمود للإشارة إلى الخانة مثل:

```
Console.WriteLine(DataGridView1.Rows[0].Cells["Col1"].Value);
```

فحتى لو لم تكن ستغير اسم العمود بعد هذا، فكتابة اسم نصي بهذه الطريقة قد يجعلك تخطئ في كتابته، فيحدث خطأ عند تنفيذ البرنامج.

إذن فما أنسب حل؟

أسهل حل لهذا الأمر، هو منح الأعمدة عند تعريفها في جدول العرض أسماء برمجية واضحة (مثل ColName)، واستخدام الخاصية Index الخاصة بكائن العمود للحصول على رقمه للإشارة إلى الخانة من خلاله مثل:

```
Console.WriteLine(DataGridView1.Rows[0].Cells(
                  ColName.Index).Value);
```

بهذه الطريقة لن يتأثر الكود بتغيير موضع العمود بعد ذلك، وفي نفس الوقت هذا الكود واضح وقابل للقراءة والفهم كما في حالة استخدام الاسم النصي للعمود، لكن بدون أي احتمال للخطأ في كتابة الاسم.

العنصر المرتبط بالبيانات DataBoundItem:

تعيد الكائن الذي يعرض الصف الحالي بياناته.. لو أخذت المشروع BindGridToArray كمثال، فإن هذه الخاصية تعيد كائن التلميذ Student الذي يعرضه الصف من المصفوفة Std.. ولو حددت أي صف في الجدول وضغطت الزر DataBoundItem في هذا المشروع، فستظهر لك رسالة تخبرك بتفاصيل كائن التلميذ المرتبط بهذا الصف.

أما لو كان جدول العرض مرتبطا بجدول من مجموعة البيانات، فستعيد هذه الخاصية كائن عرض الصف DataRowView الذي يحوي بيانات الصف المعروض حاليا.

لاحظ أن أي تغيير ستجريه على الكائن المرتبط بالصف الحالي سينتقل إلى مصدر البيانات، ومن ثم سيظهر هذا التغيير في جدول العرض مباشرة.

الارتفاع Height:

تقرأ أو تغير ارتفاع الصف الحالي.. والقيمة الافتراضية لهذه الخاصية هي الارتفاع المناسب لخط الكتابة الحالي + ٩ نقاط 9 Pixels.

أقل ارتفاع MinimumHeight:

تقرأ أو تغير أقل ارتفاع يمكن أن يقبله الصف الحالي.. وأقل قيمة لهذه الخاصية هي ٢، والقيمة الافتراضية لها هي ٣.

ارتفاع الفاصل DividerHeight:

تحدد حجم المساحة التي تفصل الصف الحالي عن الصف التالي.. هذا الفاصل هو مساحة خالية تأخذ نفس لون أرضية جدول العرض، ورغم أنها تعتبر جزءا من الصف الحالي، إلا أنها لا تؤدي أية وظيفة من وظائفه، ما عدا العمل كفاصل شكلي.. والصورة التالية توضح تأثير وضع القيمة ١٠ في هذه الخاصية في الصف الثاني:

ID	Author	CountryID	Phone	About
١٢	توفيق الحكيم	٢١		....
١٣	عباس العقاد	٢١		...
١٤	فاروق جويدة	٢١		شاعر مصري معاصر

والقيمة الافتراضية لهذه الخاصية هي صفر، لهذا لا يوجد أي فاصل بين الصفوف، ما عدا خطوط الشبكة العادية.

نص الخطأ ErrorText:

تقرأ أو تغير النص الذي يشرح الأخطاء التي حدثت في العمود، مثلما يحدث عندما يكتب المستخدم قيما غير مسموح بها في خانات العمود، أو عندما توضع قيمة في الخاصية ErrorText الخاصة بأحد صفوف مجموعة البيانات المرتبطة بجدول العرض، فهذا الخطأ ينتقل آليا إلى جدول العرض.

وتظهر أيقونة حمراء في هامش الصف لتنبه المستخدم إلى وجود خطأ، وعندما يحلق فوقها بالفأرة، سيظهر له تلميح يعرض النص الموجود في الخاصية ErrorText.

خانة رأس الصف HeaderCell:

تقرأ أو تغير كائن الخانة الرئيسية DataGridViewRowHeaderCell التي تتحكم في خانة رأس الصف الحالي.

هل هو جديد IsNewRow:

تعيد True إذا كان الصف الحالي هو الصف الجديد الموجود في آخر صف في جدول العرض وتجاوره العلامة *.. هذا الصف موجود فعلا في مجموعة صفوف الجدول، لكنه يظل صفا جديدا إلى أن يكتب فيه المستخدم أية قيمة، حيث يصبح صفا عاديا، ويضاف صف جديد بدلا منه.

كما تمتلك هذه الفئة الوسائل التالية:

ضبط طراز حافة رأس الصف AdjustRowHeaderBorderStyle:

تعدل شكل حافة رأس الصف الحالي عندما يبدأ المستخدم تحريره.. وتستقبل هذه الوسيلة المعاملات التالية:

- كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الذي سيتم تعديله.
- كائن طراز الحافة المتطور الذي سيستخدم لحفظ التغييرات البينية التي تحدث لرأس الصف.
- معامل منطقي، إذا جعلت قيمته True ستتم إضافة حافة رأسية مفردة إلى رأس الصف.
- معامل منطقي، إذا جعلت قيمته True ستتم إضافة حافة أفقية مفردة إلى رأس الصف.
- معامل منطقي، أرسل إليه True إذا كان الصف الحالي هو أول صف معروض في الجدول.
- معامل منطقي، أرسل إليه True إذا كان الصف الحالي هو آخر صف معروض في الجدول.

وتعيد هذه الوسيلة كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الذي يمثل طراز الحافة المعدل.

لاحظ أنك لست مضطرا إلى استخدام هذه الوسيلة يدويا، فجدول العرض يستدعيها تلقائيا لضبط شكل حواف الخانات الرئيسية للصفوف عند تحريرها.

إنشاء خانات CreateCells:

تحذف جميع خانات الصف الحالي، وتنشئ بدلا منها خانات جديدة، كل خانة منها مستمدة من قالب الخانة CellTemplate الخاص بالعمود الذي توجد به.. ولهذه الوسيلة صيغتان:

١- الصيغة الأولى تستقبل كائن جدول العرض DataGridView الذي سيتم استخدام قوالب الخانات الخاصة بأعمدته.

٢- والصيغة الثانية تزيد على الصيغة السابقة بمعامل ثان، يستقبل مصفوفة كائنات Object Array، بها القيم التي ستوضع في الخانات الجديدة.

معرفة رف القائمة الموضعية GetContextMenuStrip:

تعيد كائن رف القائمة الموضعية ContextMenuStrip الخاص بالصف الذي ترسل رقمه إليها كمعامل.

معرفة نص الخطأ GetErrorText:

تعيد نص الخطأ الخاص بالصف الذي أرسلت رقمه كمعامل.

معرفة الارتفاع المفضل GetPreferredHeight:

تعيد أنسب ارتفاع للصف، وهي تستقبل المعاملات التالية:

- رقم الصف المراد حساب أنسب ارتفاع له.
- إحدى قيم المرقم DataGridViewAutoSizeRowMode التي توضح كيف سيتم تغيير حجم الصف تلقائيا، وهذه القيم هي:

AllCells	يتم تغيير ارتفاع الصف ليناسب محتويات جميع خاناته، بما فيها الخانة الرئيسية.
AllCells ExceptHeader	يتم تغيير ارتفاع الصف ليناسب محتويات جميع خاناته، ما عدا الخانة الرئيسية.
RowHeader	يتم تغيير ارتفاع الصف ليناسب محتويات الخانة الرئيسية Header.

- معامل منطقي، إذا جعلت قيمته True، فسيتم تقدير الحجم المناسب على اعتبار أن عرض الصف سيظل ثابتا.. أما إذا جعلته False، فسيؤخذ في الاعتبار أن الأعمدة التي توجد فيها خانات الصف قد تغير حجمها تلقائيا لمراعاة التغير الذي حدث في ارتفاع الخانات.

معرفة الحالة GetState:

توضح حالة الصف الذي أرسلت إليها رقمه كمعامل، وهي تعيد إحدى قيم المرقم DataGridViewElementStates الذي تعرفنا عليه من قبل.

ملحوظة:
كل من الوسائل GetContextMenuStrip، GetErrorText، GetPreferredHeight، GetState تبدو عجيبة للغاية، فمعاملها يرشحها بجدارة لأن تكون وسيلة مشتركة Shared لأنها لا تتعلق بالصف الحالي تحديدا، وإنما تتعامل مع أي صف ترسل إليها رقمه، ولكن رغم هذا فهي ليست وسيلة مشتركة، فيا ترى ما هو السبب؟ في الحقيقة، هذه الوسائل مصممة للتعامل مع الصف الحالي، لهذا عليك أن ترسل إليها تحديدا رقم الصف الحالي دون غيره.. والكود التالي سيسبب خطأ في البرنامج بسبب إرسال الرقم صفر (رقم أول صف) إلى الوسيلة GetState الخاصة بالصف

رقم ٢ (الصف الثالث):

Console.WriteLine(DataGridView1.Rows[2].GetState(0));

بينما لو جربت الكود التالي فلن تحدث مشكلة!

Console.WriteLine(DataGridView1.Rows[0].GetState(0));

فيا ترى لماذا تم إنشاء هذه الوسائل بهذا الشكل العجيب؟

السبب أن الخاصية DataGridViewRow.Index التي تحمل رقم الصف تعيد -١ إذا كان الصف مشتركا Shared Row، وفي مثل هذه الحالة تناط بك مهمة إرسال رقم الصف الفعلي إلى هذه الوسائل!.. وطبعا عليك ألا تستخدم الخاصية Index لأنها لو أعادت -١ وأرسلته إلى هذه الوسائل فسيحدث خطأ في البرنامج!

لو شئت رأيي الشخصي، فتقنية مشاركة الصفوف ــ التي سنتعرف عليها بالتفصيل لاحقا ــ تسبب بعض التعقيد في الأمور، رغم أنها توفر الكثير من مساحة الذاكرة في التطبيقات الكبيرة.

وضع القيم SetValues:

أرسل إلى هذه الوسيلة مصفوفة كائنات Object Array لوضع القيم التي تحتويها في خانات الصف الحالي.. وتعيد هذه الوسيلة True إذا نجح وضع جميع القيم في جميع الخانات، أما إذا فشل وضع بعض القيم فستعيد False.. وإذا كانت المصفوفة تحتوى على عناصر أكثر من عدد خانات الصف، فإن القيم الزائدة يتم إهمالها وتوضع باقي القيم في خانات الصف، ولكن هذه الوسيلة تعيد False.. أما إذا كانت المصفوفة تحتوي على عناصر أقل من خانات الصف، فسيتم ملء بعض الخانات بالقيم الموجودة، وستترك باقي الخانات كما هي بدون تغيير.

لاحظ أن عليك إرسال مصفوفة كائنات تحديدا، لأنك لو أرسلت مصفوفة قيمية Value-Type Array (مثل مصفوفة من الأعداد الصحيحة) إلى هذه الوسيلة كمعامل، فإنها ستعتبرها قيمة واحدة وتضعها كلها في أول خانة في الصف!.. ولحل هذه المشكلة أمامك طريقتان:

- فإما أن تحول المصفوفة القيمية إلى مصفوفة كائنات، بطريقة مثل:

```
int[] Arr = { 1, 2, 3, 4 };
object[] O = new object[Arr.Length];
Arr.CopyTo(O, 0);
Row.SetValues(O);
```

- وإما أن ترسل القيم كمجموعة من المعاملات المفردة إلى هذه الوسيلة بدون وضعها في مصفوفة، وستقوم هي بجمعها في مصفوفة كائنات، مثل:

```
Row.SetValues(1, 2, 3, 4);
```

فئة خانة جدول العرض DataGridViewCell Class

هـذه الفئـة تـرث الفئـة DataGridViewElement، وهـي فئـة أساسـية مجـردة Abstract Base Class، تشتق منها كل أنواع خانات جدول العرض.
وتمتلك هذه الفئة عدة خصائص مماثلة لخصائص كائن النطـاق DataGridViewBand وكائن العمود DataGridViewColumn وكائن الصف DataGridViewRow، لهـذا لن نكرر شرحها هنا، وهي:

- **مثبتة Frozen**
- **معروضة Displayed**
- **للقراءة فقط ReadOnly**
- **قابلة لتغيير الحجم Resizable**
- **محددة Selected**
- **مرئية Visible**
- **نوع القيمة ValueType**
- **الوسم Tag**
- **نص تلميح الأداة ToolTipText**
- **نص الخطأ ErrorText**
- **الطراز Style**
- **لها طراز HasStyle**
- **الطراز الموروث InheritedStyle**
- **كائن تسهيل الوصول AccessibilityObject**
- **رف القائمة الموضعية ContextMenuStrip**

وإضافة إلى هذه الخصائص، تمتلك هذه الفئة الخصائص التالية:

رقم العمود ColumnIndex:

تعيد رقم العمود الذي توجد فيه الخانة الحالية.

العمود المالك OwningColumn:

تعيد كائن العمود DataGridViewColumn الذي توجد فيه الخانة الحالية.

رقم الصف RowIndex:

تعيد رقم الصف الذي توجد فيه الخانة الحالية.

الصف المالك OwningRow:

تعيد كائن الصف DataGridViewRow الذي توجد فيه الخانة الحالية.

الحجم Size:

تعيد كائن الحجم Size، الذي يحمل أبعاد الخانة الحالية.

الحجم المفصل PreferredSize:

تعيد كائن الحجم Size، الذي يحمل أنسب أبعاد للخانة الحالية لكي تستوعب محتوياتها استيعابا كاملا.

حدود المحتوى ContentBounds:

تعيد كائن المستطيل Rectangle، الذي يحمل موضع ومساحة محتويات الخانة الحالية.. تذكر أن الخانة قد تحتوي على نص أو صورة أو مربع اختيار أو زر أو أية أداة أخرى، لهذا فإن هذه الخاصية تعيد إليك حدود الأداة المحتواة في الخانة.

حدود أيقونة الخطأ ErrorIconBounds:

تعيد كائن المستطيل Rectangle، الذي يحمل موضع ومساحة أيقونة الخطأ التي تعرضها الخانة الحالية (إن وجدت).

القيمة الافتراضية للصف الجديد DefaultNewRowValue:

تعيد كائنا Object، يحتوي على القيمة الافتراضية للخانة الحالية إذا كانت في الصف الجديد (آخر صف في جدول العرض).

وتفيدك هذه الخاصية عندما تنشئ نوعا خاصا بك من الخانات، ففي هذه الحالة عليك أن تجعل هذه الخاصة تعيد القيمة الافتراضية للخانة الجديدة.. مثلا: لو كنت تتعامل

مع خانة تستقبل أيقونات، يمكنك أن تعرض في الخانة الجديدة أيقونة في شكل علامة استفهام.. وإن كنت تتعامل مع خانة تاريخ، فيمكنك أن تضع في الخانة الجديد التاريخ اليوم الحالي.. وهكذا.

القيمة Value:

تقرأ أو تغير قيمة الخانة الحالية، وهي من النوع Object.. مثال:

```
DataGridView1.Rows[0].Cells[0].Value = "Test";
```

القيمة المنسقة FormattedValue:

تعيد كائنا Object يحمل القيمة المنسقة للخانة الحالية.. لاحظ أن القيمة الموجودة في الخانة قد تكون نصا مثلا، بينما يتم تنسيق هذا النص كتاريخ.

نوع القيمة المنسقة FormattedValueType:

تعيد كائن النوع Type الذي يمثل نوع القيمة المنسقة الموجودة في الخانة.

القيمة المعدلة المنسقة EditedFormattedValue:

تعيد القيمة الحالية للخانة بالتنسيق المطلوب، حتى لو كانت الخانة في وضع التحرير وكتب بها المستخدم قيمة لم تقبل بعد.. وبهذا تختلف عن الخاصية FormattedValue، التي تعيد القيمة المحفوظة في الخانة فعلا.

ملحوظة:
إذا استخدمت الحدث CellContentClick الخاص بجدول العرض، لفحص قيمة خانة في عمود من النوع DataGridViewChecBoxColumn، فاستخدم الخاصية EditedFormattedValue لقراءة قيمة الخانة، ولا تستخدم الخاصية Value.. السبب في هذا أن الخانة الموجودة في عمود مربعات الاختيار، لا تغير قيمتها إلى القيمة الجديدة إلا بعد مغادرة المؤشر Focus للصف الذي توجد فيه!!

نوع التحرير EditType:

تعيد كائن النوع Type، الذي يمثل الأداة التي توضع في الخانة عند تحريرها.. مثلا: عند تحرير أي خانة في عمود نصي، يوضع فيها مربع نص من النوع DataGridViewTextBoxEditingControl لاستقبال ما يكتبه المستخدم.. وعند تحرير خانة في عمود قوائم مركبة، يوضع فيها قائمة مركبة من النوع DataGridViewComboBoxEditingControl.

وقد استبدلنا Override هذه الخاصية في الفئة CalendarCell في المشروع DataGridColumnTypes لجعلها تعيد النوع CalendarEditingControl، وهو نوع أداة تحرير خاصة بنا، أنشأناها للتعامل مع الخانات التي تعرض أداة اختيار التاريخ DateTimePicker،.

هل هو في وضع التحرير IsInEditMode:

تعيد True إذا كانت الخانة في وضع التحرير حاليا.

الحالة الموروثة InheritedState:

تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح حالة الخانة الحالية.

كما تمتلك هذه الفئة الوسائل التالية:

قياس ارتفاع النص MeasureTextHeight:

تحسب الارتفاع اللازم لرسم النص المرسل إليها.. هذا مفيد لمعرفة أنسب ارتفاع للصف يناسب هذا النص قبل وضعه في الخانة.. وتستقبل هذه الوسيلة المعاملات التالية:

- كائن الرسوم Graphics المستخدم في رسم النص.
- النص المراد قياس عرضه.
- كائن الخط Font الذي سيكتب النص به.

- أقصى عرض يمكن رسم النص فيه (عرض العمود).
- إحدى قيم المرقم TextFormatFlags توضح تنسيق النص، وهي:

Default	التنسيق الافتراضي.
Bottom	محاذاة النص إلى أسفل.
Top	محاذاة النص إلى أعلى.
Left	محاذاة النص إلى اليسار.
Right	محاذاة النص إلى اليمين.
RightToLeft	رسم النص من اليمين إلى اليسار.
HorizontalCenter	توسيط النص أفقيا.
VerticalCenter	توسيط النص رأسيا.
EndEllipsis	رسم نهايات الخطوط الحادة بحواف مستديرة.
PathEllipsis	رسم المسارات الحادة بحواف مستديرة.
ExpandTabs	توسيع حروف الجدولة.
ExternalLeading	إضافة عرض المسافة البادئة للخط إلى ارتفاع النص.
HidePrefix	إخفاء البادئة.
Internal	استخدام خط النظام في عملية القياس.
ModifyString	ليس لها تأثير.
NoClipping	عدم قص علامات التشكيل الفوقية والسفلية التي تتجاوز مستطيل الرسم.
GlyphOverhang Padding	إضافة هامش إلى مستطيل الرسم لاستيعاب علامات التشكيل الزائدة.
NoPadding	لا تضاف أية هوامش.

LeftAndRight Padding	إضافة هامش أيمن وهامش أيسر فقط.
NoPrefix	تجاهل الرمز & الدال على الحروف التذكيرية Mnemonic Characters واعتباره حرفا عاديا.
NoFullWidth CharacterBreak	عدم استخدام الكشيدة لمط الحروف لجعل النص يشغل عرض المستطيل بالكامل.
PrefixOnly	تستخدم مع ويندوز ٢٠٠٠ و XP.
SingleLine	رسم النص في سطر واحد.
TextBoxControl	تنسيق النص لعرضه في مربع نص.
WordBreak	تقسيم النص إلى سطور في نهاية الكلمات.
WordEllipsis	حذف الكلمات الزائدة عن عرض السطر، ووضع نقاط تكملة (...) بعد آخر كلمة ظاهرة.
PreserveGraphics Clipping	استخدام التقطيع الخاص بكائن الرسوم.
PreserveGraphics TranslateTransform	استخدام التحويل الإحداثي (كالتكبير والتصغير) الخاص بكائن الرسوم.

ويمكنك دمج أكثر من قيمة من هذه القيم باستخدام المعامل Or.

وتوجد صيغة أخرى لهذه الوسيلة، تزيد بمعامل سادس على الصيغة السابقة، وهو معامل منطقي مرجعي ByRef يعمل كمعامل إخراج Output، وهو يعيد True إذا كان عرض النص المرسوم أكبر من أقصى عرض مسموح به في المعامل الرابع.

قياس عرض النص MeasureTextWidth:

تحسب العرض اللازم لرسم النص المرسل إليها، ولها المعاملات التالية:

- كائن الرسوم Graphics المستخدم في رسم النص.
- النص المراد قياس عرضه.

- كائن الخط Font الذي سيكتب النص به.
- أقصى ارتفاع يمكن رسم النص فيه.
- إحدى قيم المرقم TextFormatFlags التي توضح تنسيق النص.

قياس حجم النص MeasureTextSize:

تعيد كائن حجم Size يحمل العرض والارتفاع اللازمين لرسم النص المرسل إليها، وهي تستقبل المعاملات التالية:

- كائن الرسوم Graphics المستخدم في رسم النص.
- النص المراد قياس عرضه.
- كائن الخط Font الذي سيكتب النص به.
- إحدى قيم المرقم TextFormatFlags التي توضح تنسيق النص.

قياس الحجم المفضل للنص MeasureTextPreferredSize:

تعيد كائن حجم Size يحمل أفضل عرض وارتفاع مناسبين لرسم النص المرسل إليها، وهي تستقبل المعاملات التالية:

- كائن الرسوم Graphics المستخدم في رسم النص.
- النص المراد قياس عرضه.
- كائن الخط Font الذي سيكتب النص به.
- عدد مفرد Single أكبر من صفر وأقل من ١، يحمل أقصى نسبة مسموح بها بين عرض وارتفاع النص عند رسمه.
- إحدى قيم المرقم TextFormatFlags التي توضح تنسيق النص.

ضبط شكل حافة الخانة AdjustCellBorderStyle:

مماثلة للوسيلة DataGridViewRow.AdjustRowHeaderBorderStyle، ولها نفس المعاملات، ولكنها تقوم بضبط شكل حواف الخانة الحالية عندما تكون في وضع التحرير.

تجهيز أداة التحرير InitializeEditingControl:

يقوم جدول العرض باستدعاء هذه الوسيلة مرة واحدة لإضافة أداة التحرير الخاصة بالخانة الحالية إلى أدوات التحرير التي يستخدمها.. وتستقبل هذه الوسيلة المعاملات التالية:

- رقم الصف الذي توجد به الخانة.
- كائن يحتوي على القيمة المنسقة Formatted Value التي ستوضع مبدئيا في أداة التحرير عندما تظهر في الخانة.
- كائن طراز الخانة DataGridViewCellStyle الذي يحتوي على الخصائص الشكلية للخانة، لاستخدامه في جعل أداة التحرير شبيهة بالخانة.

وقد استبدلنا Override هذه الوسيلة في كود الفئة CalendarCell في المشروع DataGridColumnTypes لجعل أداة اختيار التاريخ تعرض نفس التاريخ الموجود في الخانة عند ظهورها لأول مرة.

تحديد موضع أداة التحرير PositionEditingControl:

تحدد موضع وأبعاد أداة التحرير في جدول العرض، ولها المعاملات التالية:

- معامل منطقي Boolean، إذا جعلت قيمته True فستوضع الأداة في الموضع الذي تحدده باقي المعاملات، إما إذا جعلتها False فسيترك للأداة تحديد موضعها بنفسها.
- معامل منطقي Boolean، إذا جعلت قيمته True فسيتم تحديد حجم الأداة تبعا لباقي المعاملات، إما إذا جعلتها False فسيترك للأداة تحديد حجمها بنفسها.
- كائن مستطيل Rectangle يحمل موضع وأبعاد أداة التحرير.
- كائن مستطيل Rectangle يحمل موضع وأبعاد المساحة التي يجب ألا تتجاوزها أداة التحرير.
- كائن DataGridViewCellStyle، يحمل طراز الخانة.

- معامل منطقي، إذا جعلت قيمته True فستضاف حافة رأسية مفردة إلى رأس الصف.
- معامل منطقي، إذا جعلت قيمته True فستضاف حافة أفقية مفردة إلى رأس الصف.
- معامل منطقي، أرسل إليه True إذا كانت الخانة موجودة في أول عمود يعرضه الجدول.
- معامل منطقي، أرسل إليه True إذا كان كانت الخانة موجودة في أول صف يعرضه الجدول.

تحديد موضع لوحة التحرير PositionEditingPanel:

مشابهة للوسيلة السابقة، إلا أنها تحدد موضع اللوحة التي توضع عليها أداة التحرير داخل الخانة.. ولهذه الوسيلة نفس معاملات الوسيلة السابقة ما عدا أول معاملين فهما غير موجودين هنا.. كما أن هذه الوسيلة تعيد كائن المستطيل Rectangle الذي يحدد موضع وأبعاد أداة التحرير داخل لوحة التحرير.

إبعاد أداة التحرير DetachEditingControl:

تزيل أداة الكتابة من الخانة الحالية، وتنهي وضع التحرير.

معرفة الحالة الموروثة GetInheritedState:

تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح الحالة الموروثة من جدول العرض أو العمود أو الصف الذي توجد به الخانة الحالية.. وتستقبل هذه الوسيلة كمعامل رقم الصف الذي توجد به الخانة.

ملحوظة:
لا تستطيع استخدام الخاصية InheritedState مع خانة موجودة في صف مشترك Shared Row لأن الخاصية Index التي تشير إلى موضع هذا الصف تعيد القيمة

-١، لهذا يمكنك استخدام الوسيلة GetInheritedState بدلا منها، على أن ترسل إليها رقم الصف الفعلي بنفسك.. لاحظ أن هذا هو الحال نفسه في كل الوسائل التالية التي تستقبل رقم الصف الذي توجد به الخانة كمعامل، فهي مخصصة للتعامل مع الخانات الموجودة في صفوف مشتركة.

معرفة حدود المحتوى GetContentBounds:

تعيد كائن المستطيل Rectangle، الذي يمثل موضع وأبعاد محتويات الخانة الحالية.. ويجب أن ترسل إلى هذه الوسيلة رقم الصف الذي توجد به الخانة.

معرفة القيمة المعدلة المنسقة GetEditedFormattedValue:

تعيد القيمة المكتوبة حاليا في الخانة حتى ولو كانت في وضع التحرير، وهي تستقبل المعاملين التاليين:

- رقم الصف الذي توجد فيه الخانة.
- قيمة من قيم المرقم DataGridViewDataErrorContexts توضح محتوى الخطأ الخاص بالخانة، وهذه القيم هي:

Formatting	خطأ في تنسيق قيمة الخانة.
Display	خطأ في عرض القيمة من مصدر البيانات.
PreferredSize	خطأ في حساب أفضل حجم للخانة.. فشلت الخانة في تنسيق محتوياتها.
RowDeletion	خطأ في حذف أحد الصفوف.. يحدث هذا إذا كان الصف مرتبطا بمصدر بيانات، وأطلق مصدر البيانات خطأ عند محاولة حذف هذا الصف منه.
Parsing	خطأ في تحويل البيانات التي كتبها المستخدم أو أتت من مصدر البيانات.

Commit	خطأ في حفظ بيانات الخانة في مصدر البيانات.
InitialValue Restoration	خطأ في استعادة القيمة الأصلية للخانة عند محاولة إلغاء التحرير الحالي، وذلك بسبب تغير تنسيق الخانة.
LeaveControl	خطأ عند مغادرة جدول العرض، بسبب عدم قدرته على حفظ التغييرات في مصدر البيانات.
CurrentCell Change	خطأ عند محاولة مغادرة الخانة الحالية، بسبب وجود أخطاء فيها.
Scroll	خطأ في الانزلاق، بسبب ظهور خانة بها مشكلة.
Clipboard Content	خطأ عند نسخ محتويات الخانة إلى لوحة القصاصات Clipboard، بسبب عدم إمكانية تحويل محتويات هذه الخانة إلى نص.

لاحظ أنك تستطيع دمج أكثر من قيمة من قيم هذا المرقم معا باستخدام المعامل الثنائي OR (العلامة | في سي شارب).

معرفة رف القائمة الموضعية الموروث GetInheritedContextMenuStrip:

تعيد كائن رف القائمة الموضعية ContextMenuStrip الموروث من جدول العرض أو العمود أو الصف الذي توجد به الخانة الحالية.. وتستقبل هذه الوسيلة كمعامل رقم الصف الذي توجد به الخانة.

معرفة الطراز الموروث GetInheritedStyle:

تعيد كائن طراز الخانة DataGridViewCellStyle الموروث من الجدول أو العمود الذي توجد به الخانة.. وتستقبل هذه الوسيلة المعاملات التالية:

- كائن طراز الخانة DataGridViewCellStyle الذي ستوضع خصائص الطراز الموروث فيه.

- رقم الصف الذي توجد فيه الخانة.
- معامل منطقي إذا جعلته True، فستضاف خصائص الألوان ضمن الطراز الموروث.

هل يبدأ الزر وضع التحرير KeyEntersEditMode:

تعيد True إذا كان ضغط الحرف المرسل إليها كمعامل يبدأ تحرير الخانة الحالية، علما بأن معامل هذه الوسيلة من نوع الفئة KeyEventArgs، وهي نوع المعامل e في حدث ضغط الزر KeyPress.. والمثال التالي يخبرك إن كان ضغط حرف الإلغاء Escape يبدأ عملية التحرير أم لا (بافتراض أن Cell هو متغير يشير إلى خانة من خانات الجدول):

```
var K = new KeyEventArgs(Keys.Escape);
Console.WriteLine(Cell.KeyEntersEditMode(K)); // Fales
```

تحويل القيمة المنسقة ParseFormattedValue:

تحول القيمة المنسقة المعروضة في الخانة إلى القيمة الأصلية، وهي تستقبل المعاملات التالية:

- كائن Object يحمل القيمة المنسقة.
- كائن طراز الخانة DataGridViewCellStyle.
- كائن من النوع "محول القيمة" TypeConverter لاستخدامه لتحويل القيمة المنسقة.. ويمكنك إرسال القيمة Nothing لاستخدام المحول الافتراضي.
- كائن من النوع "محول القيمة" TypeConverter لاستخدامه للتحويل إلى القيمة الأصلية للخانة.. ويمكنك إرسال القيمة Nothing لاستخدام المحول الافتراضي.

لاحظ أن جدول العرض يستدعي هذه الوسيلة تلقائيا بعد إدخال المستخدم لقيمة جديدة في أي خانة، لهذا لست مضطرا إلى استدعائها بنفسك.

فئة خانة مربع النص
DataGridViewTextBoxCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كخانة في جدول العرض تعرض نصا، وعند تحريرها تعرض مربع نص لاستقبال نص جديد من المستخدم.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصية التالية:

أقصى طول للمدخلات MaxInputLength:

تحدد أقصى عدد من الحروف تقبله الخانة الحالية.

وفي المشروع DataGridColumnTypes، أنشأنا الفئة CalendarCell، وهي ترث الفئة DataGridViewTextBoxCell لتكون قابلة للتحرير، ولكننا استبدلنا بعض خصائصها لتسمح بعرض أداة اختيار التاريخ DateTimePicker بدلا من مربع النص.. ولو فحصت كود هذه الفئة، فستلاحظ أنه بسيط للغاية، فهو يدور حول نوع أداة التحرير المستخدمة في الخانة، ونوع القيمة التي تتعامل معها.

فئـــة خـــانـــة الـــزر
DataGridViewButtonCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي خانة تعرض زرا.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصيتين التاليتين:

طريقة العرض المسطح FlatStyle:

استخدام نص العمود كقيمة للزر UseColumnTextForButtonValue:

وهمـــا مماثلتـــان للخاصـــيتين اللتـــين تحمـــلان نفـــس الاســـم فـــي الفئـــة DataGridViewButtonColumn، لكنهما تؤثران فقط على الخانة الحالية، وليس علـــى كل خانات العمود.

واجـــهة خـــانة التحـــريـــر
IDataGridViewEditingCell Interface

تقدم هذه الواجهة إلى خانات جدول العرض التي تمثلها، القدرة علـــى تحريـــر المســـتخدم لقيمها.. وتمتلك هذه الواجهة العناصر التالية:

هل تغيرت قيمة خانة التحرير EditingCellValueChanged:

اجعل قيمة هذه الخاصية True، إذا تغيرت قيمة الخانة في وضع التحرير.

القيمة المنسقة لخانة التحرير EditingCellFormattedValue:

تقرأ أو تغير القيمة المنسقة التي تحتويها الخانة في وضع التحرير.

معرفة القيمة المنسقة لخانة التحرير GetEditingCellFormattedValue:

تعيد كائنا Object، يحتوي على القيمة المنسقة لخانة التحرير.. وتستقبل هذه الوســـيلة إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح نوع الخطـــأ الذي حدث بالخانة، وقد تعرفنا عليه من قبل.

تجهيز خانة التحرير للتحرير PrepareEditingCellForEdit:

تجهز الخانة عند بدء وضع التحرير، وهي تستقبل معاملا منطقيا، إذا جعلـــت قيمتـــه True فسيتم تحديد النص المكتوب في الخانة.

اللهم ارحم أبي واغفر له وكفر عنه سيئاته

فئة خانة مربع الاختيار
DataGridViewCheckBoxCell Class

هذه الفئة ترث الفئة DataGridViewCell، كما أنها تمثل الواجهة IDataGridViewEditingCell، وهي تعمل كخانة في جدول العرض تعرض مربع اختيار CheckBox.

ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.

٢- والصيغة الثانية تستقبل معاملا منطقيا Boolean، يتم إرسال قيمته إلى الخاصية ThreeState.

وإضافة إلى ما ترثه من الفئة الأم، وما تمثله من عناصر الواجهة IDataGridViewEditingCell، تمتلك هذه الفئة بعض الخصائص الشبيهة بخصائص الفئة DataGridViewCheckBoxColumn، وهي تقوم بنفس الوظيفة لكنها تتعامل مع الخانة الحالية فقط وليس العمود كله، لهذا لن نكرر شرحها هنا، وسنكتفي بذكر أسمائها:

- **طريقة العرض المسطح FlatStyle**
- **ثلاثي الحالة ThreeState**
- **القيمة الخاطئة FalseValue**
- **القيمة الصحيحة TrueValue**
- **القيمة غير المحددة IndeterminateValue**

فئة خانة الصور
DataGridViewImageCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كخانة في جدول العرض تعرض صورة أو أيقونة.

ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.

٢- والصيغة الثانية تستقبل معاملا منطقيا Boolean، يتم إرسال قيمته إلى الخاصية ValuesAreIcons.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة بعض الخصائص الشبيهة بخصائص الفئة DataGridViewImageColumn، وهي تقوم بنفس الوظيفة لكنها تتعامل مع الخانة الحالية فقط وليس العمود كله، لهذا لن نكرر شرحها هنا، وسنكتفي بذكر أسمائها:

الوصف Description

مخطط الصورة ImageLayout

هل القيمة أيقونة ValueIsIcon

اللهم ارحم أبي واغفر له وكفر عنه سيئاته

فئة خانة الوصلة
DataGridViewLinkCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كخانة في جدول العرض، بها وصلة Link.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة بعض الخصائص الشبيهة بخصائص الفئة DataGridViewLinkColumn، وهي تقوم بنفس الوظيفة لكنها تتعامل مع الخانة الحالية فقط وليس العمود كله، لهذا لن نكرر شرحها هنا، وسنكتفي بذكر أسمائها:

- **استخدم نص العمود كقيمة للوصلة UseColumnTextForLinkValue**
- **سلوك الرابط LinkBehavior**
- **لون الرابط LinkColor**
- **لون الرابط الفعال ActiveLinkColor**
- **لون الرابط المُزار VisitedLinkColor**
- **تتبع حالة الزيارة TrackVisitedState**

كما تمتلك هذه الفئة الخاصية التالية:

تمت زيارة الرابط LinkVisited:

إذا جعلت قيمة هذه الخاصية True، فإن الرابط يأخذ اللون المحدد في الخاصية VisitedLinkColor.. لاحظ أن لون الرابط يتغير تلقائيا عندما يضغطه المستخدم إذا كانت للخاصية TrackVisitedState القيمة True.

فئة خانة القائمة المركبة
DataGridViewComboBoxCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كخانة في جدول العرض بها قامة مركبة ComboBox.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة بعض الخصائص الشبيهة بخصائص الفئة DataGridViewCombokBoxColumn، وهي تقوم بنفس الوظيفة لكنها تتعامل مع الخانة الحالية فقط وليس العمود كله، لهذا لن نكرر شرحها هنا، وسنكتفي بذكر أسمائها:

- **تكملة تلقائية AutoComplete**
- **مصدر البيانات DataSource**
- **عنصر العرض DisplayMember**
- **عنصر القيمة ValueMember**
- **مرتبة Sorted**
- **العناصر Items**
- **طريقة العرض المسطح FlatStyle**
- **طريقة العرض DisplayStyle**
- **طريقة عرض الخانة الحالية فقط DisplayStyleForCurrentCellOnly**
- **أقصى عدد من العناصر المسدلة MaxDropDownItems**
- **عرض القائمة المنسدلة DropDownWidth**

ربط مصدر بيانات خانة بمصدر بيانات خانة أخرى:

يمكنك تغيير مصدر بيانات خانة في أحد الأعمدة، تبعا لقيمة خانة أخرى في عمود آخر في نفس الصف.. مثال: إذا كان هناك عمود تعرض خاناته أسماء المحافظات، وبجواره عمود تعرض خاناته أسماء المدن، يمكنك جعل خانة أسماء المدن تعرض فقط المدن الخاصة بالمحافظة التي اختارها المستخدم في الخانة المجاورة لها.

نحن نتكلم هنا عن أعمدة من النوع DataGridViewComboBoxColumn، وهي تمتاز بأن خاناتها تعرض قائمة مركبة ComboBox في وضع التحرير Edit Mode، لكي يختار المستخدم منها قيمة معينة.. وقد رأينا كيف يمكننا في الوضع العادي ربط العمود بمصدر بيانات باستخدام الخصائص:

- DataGridViewComboBoxColumn.DataSource
- DataGridViewComboBoxColumn.DisplayMember
- DataGridViewComboBoxColumn.ValueMember

ما يحدث فعليا هو أن العمود يضع قيم هذه الخصائص في خصائص كل خانة من خاناته.. هذه الخانات من النوع DataGridViewComboBoxCell، وهي أيضا تمتلك الخصائص DataSource وDisplayMember وValueMember.. هذا معناه أنك تستطيع تغيير مصدر بيانات كل خانة على حدة.. لكن ما يفعله العمود هو أنه يضع قيم هذه الخصائص في الخانة التي تعيدها الخاصية DataGridViewComboBoxColumn.CellTemplate.. هذه الخانة تعمل كقالب افتراضي لكل الخانات الجديدة التي تضاف في هذا العمود.. كما يقوم العمود بتغيير قيم هذه الخصائص في الخانات القديمة.

إذن، كيف يمكن تغيير قائمة أسماء المدن، عند تغيير اسم المحافظة؟

يمكن فعل هذا باستخدام الحدث GataGridView.CellValueChanged، الذي ينطلق عند تغير قيمة أي خانة من خانات جدول العرض (لا يقع التغيير إلا بعد مغادرة الخانة بالفعل).. في هذا الحدث سنفحص قيمة المعامل e.ColumnIndex للتأكد أن الخانة التي تغيرت قيمتها تقع في العمود الخاص بالمحافظات.. (وليكن رقم X):

```
if (e.ColumnIndex == X)
{
   // ……….
}
```

فإن كان هذا الشرط صحيحا، فسنتبع الخطوات التالية:

- سنحصل على الصف الحالي باستخدام رقم الصف e.RowIndex كالتالي:

```
var Row = DataGridView1.Rows[e.RowIndex];
```

- سنحصل على خانة اسم المحافظة باستخدام رقم العمود X كالتالي:

```
var Cell = Row[e.RowIndex].Cells[X]
```

- سنعرف رقم المحافظة التي اختارها المستخدم باستخدام الخاصية Value للخانة (هذا بافتراض أنك جعلت الخاصية ValueMember تشير إلى الخاصية ID في مصدر البيانات الذي يعرض المحافظات):

```
var GovID = (int)Cell.Value;
```

- اكتب الكود المناسب للحصول أسماء مدن هذه المحافظة.. افترض أن النتيجة العائدة هي مجموعة اسمها Cities.. نريد أن نستخدم هذه المجموعة كمصدر بيانات الخانة المجاورة للخانة الحالية (رقم X + 1).

عرّف متغيرا لخانة أسماء المدن.. يجب تحويل نوع هذا المتغير من نوع الخانة العامة DataGridViewCell إلى النوع المحدد DataGridViewComboBoxCell حتى يمكننا استخدام خصائص ربط البيانات الخاصة بهذا النوع من الخانات:

```
DataGridViewComboBoxCell CityCell = Row.Cells[X + 1]
CityCell.DataSource = Cities;
CityCell.DisplayMember = "Name";
CityCell.ValueMember = "ID";
```

بهذه الطريقة كلما غير المستخدم اسم المحافظة في أي صف، ستعرض خانة اسم المدن قائمة فيها أسماء مدن هذه المحافظة فقط.

لاحظ أن ترتيب صفوف جدول العرض يؤدي إلى ضياع قيم خصائص ربط البيانات من كل خانة، لأن جدول العرض يستخدم القالب CellTemplate لإعادة رسم الخانات بعد عملية الترتيب.. وهذا سيجعل أسماء المدن التي اختارها المستخدم تختفي من كل خانات

العمود (رغم أن أرقام المدن ما زالت محفوظة في الخانات، لكن لا يمكن عرض أسمائها بدون مصدر بيانات)!

لهذا إما تمنع المستخدم من ترتيب الصفوف عند ضغط الأعمدة، وذلك بتغير قيمة الخاصية SortMode في كل عمود.. وإما أن تستخدم الحدث Sorted الذي ينطلق بعد تمام عملية الترتيب، لتمر عبر كل صفوف الجدول، وتعيد وضع مصدر بيانات خانات المدن تبعا للمحافظات المحددة في كل صف!

وهناك حل ثالث، هو أن تجعل العمود يشير إلى مصدر بيانات يحتوي على كل أسماء المدن بغض النظر عن المحافظات.. في هذه الحالة سيوضع هذا المصدر في قالب الخانة CellTemplate، وعند الترتيب سيتم وضعه في كل خانات المدن، وبهذا ستظل أسماء المدن التي اختارها المستخدم ظاهرة.. لكن لو ضغط المستخدم الخانة وعرض القائمة المركبة، فسيري فيها كل أسماء المدن. لهذا عليك أن تستخدم الحدث DataGridView.CellEnter الذي ينطلق عند دخول الخانة، لتعيد للخانة مصدر البيانات الصحيح تبعا للمحافظة المحددة.

واجـــــهة أداة التحـــريـــــر
IDataGridViewEditingControl Interface

تعرّف هذه الواجهة العناصر المشتركة بين أدوات التحرير المستخدمة في خانـــات جـــدول العرض، وهي تملك الخصائص التالية:

جدول العرض EditingControlDataGridView:

تقرأ أو تغير كائن جدول العرض DataGridView الذي يعرض أداة التحرير.

القيمة المنسقة EditingControlFormattedValue:

تقرأ أو تغير القيمة المنسقة المعروضة حاليا في أداة التحرير.

رقم الصف EditingControlRowIndex:

تقرأ أو تغير رقم الصف الذي تظهر فيه أداة التحرير.

القيمة تغيرت EditingControlValueChanged:

اجعل قيمة هذه الخاصية True، إذا تغيرت القيمة المكتوبة في أداة التحرير، عن قيمة الخانة التي تعرض أداة التحرير.

مؤشر لوحة التحرير EditingPanelCursor:

تعيد كائنا، يحتوي المؤشر Cursor الذي يتم عرضه عندما تمر الفأرة فـــوق اللوحـــة Panel التي تحتوي على أداة التحرير.

تغيير موضع أداة التحرير عند تغير القيمة RepositionEditingControlOnValueChange:

تعيد True إذا كانت أداة التحرير بحاجة إلى تغيير موضعها بعد قيام المستخدم بالكتابة فيها.. على سبيل المثال: قد يكتب المستخدم نصا طويلا في أداة التحرير، مما يستلزم أن تقوم بتقسيمه على أكثر من سطر.

كما تمتلك هذه الواجهة الوسائل التالية:

تطبيق طراز الخانة على أداة التحرير ApplyCellStyleToEditingControl:

تجعل لأداة التحرير الخصائص الشكلية الخاصة بكائن طراز الخانة DataGridViewCellStyle الذي ترسله إليها كمعامل.

هل تريد أداة التحرير زر الإدخال EditingControlWantsInputKey:

تعيد True إذا كان الزر المرسل كمعامل هو أحد الأزرار التي تتعامل معها أداة الإدخال، سواء كان حرفا يمكن كتابته، أو وظيفة يمكن أن تؤديها.. أما إذا كانت أداة الإدخال لا تتعامل مع الزر، فإنها تعيد False لتوضح أن على جدول العرض التعامل مع هذا الزر.. ولهذه الوسيلة معاملان:

- المعامل الأول يستقبل إحدى قيم المرقم Keys التي تعبر عن الزر المضغوط.
- والمعامل الثاني معامل منطقي، إذا كانت قيمته True فهذا معناه أن جدول العرض يمكنه التعامل مع الزر المضغوط.

معرفة القيمة المنسقة GetEditingControlFormattedValue:

تعيد القيمة المنسقة المكتوبة حاليا في أداة التحرير، وهي تستقبل كمعامل إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح محتوى الخطأ الخاص بالخانة.

تجهيز أداة التحرير للتحرير PrepareEditingControlForEdit:

تجهز الأداة عند بدء وضع التحرير، وهي تستقبل معاملا منطقيا، إذا جعلت قيمته True فسيتم تحديد النص المكتوب في أداة التحرير.

وقد أنشأنا الفئة CalendarEditingControl في المشروع DataGridColumnTypes، وجعلناها تمثل الواجهة IDataGridViewEditingControl لكي تكون أداة تحرير نستطيع استخدامها في جدول العرض.. ولكي نجعلها تعرض أداة اختيار التاريخ في الخانة التي تستضيفها، لم نكتب أكثر من سطر واحد فقط يجعلها ترث الأداة DateTimePicker:

```
class CalendarEditingControl :
      DateTimePicker, IDataGridViewEditingControl
{
   // الكود الذي يمثل خصائص ووسائل الواجهة
   // IDataGridViewEditingControl
}
```

فئة أداة تحرير مربع النص
DataGridViewTextBoxEditingControl

هذه الفئة ترث الفئة TextBox، كما أنها تمثل الواجهة IDataGridViewEditingControl، مما يعني أنها مربع نص مخصص للظهور في خانات جدول العرض عند تحريرها، للسماح للمستخدم بالكتابة في الخانة.

ولا تمتلك هذه الفئة أية عناصر أو خصائص جديدة، غير ما ترثه من الفئة الأم، وما تمثله من عناصر الواجهة IDataGridViewEditingControl.

فئة أداة تحرير القائمة المركبة
DataGridViewComboBoxEditingControl Class

هذه الفئة ترث الفئة ComboBox، كما أنها تمثل الواجهة IDataGridViewEditingControl، مما يعني أنها قائمة مركبة مخصصة للظهور في خانات جدول العرض عند تحريرها، للسماح للمستخدم باختيار قيمة منها.
ولا تمتلك هذه الفئة أية عناصر أو خصائص جديدة، غير ما ترثه من الفئة الأم، وما تمثله من عناصر الواجهة IDataGridViewEditingControl.

فئة الخانة الرئيسية
DataGridViewHeaderCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كفئة أم تحتوي على الخصائص والوسائل المشتركة بين خانة رأس العمود وخانة رأس الصف.
ولا تمتلك هذه الفئة أية خصائص أو وسائل جديدة غير ما ترثه من الفئة الأم.. قد يبدو لك هذا غريبا، لكن فائدة هذه الفئة تتضح حينما تريد أن تنشئ خانة رئيسية خاصة بك تعرض أيقونة أو مقبضا أو لها شكل خاص، ففي هذه الحالة عليك أن تنشئ فئة ترث الفئة DataGridViewHeaderCell، وتضيف إليها القدرات الجديدة التي تريدها.

فئة خانة رأس العمود
DataGridViewColumnHeaderCell

هذه الفئة ترث الفئة DataGridViewHeaderCell، وهي تعمل كخانة تعــرض عنــوان أحد الأعمدة في جدول العرض.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصية التالية:

اتجاه الترتيب SortGlyphDirection:

تحدد اتجاه الصورة الرمزية Glyph التي يعرضها العمود لتوضح طريقــة ترتيبــه، وهي تأخذ إحدى قيم المرقم SortOrder التالية:

None	العمود غير مرتب (لا تظهر أي صورة).
Ascending	العمود مرتب تصاعديا (صورة مثلث رأسه إلى أعلى).
Descending	العمود مرتب تنازليا (صورة مثلث رأسه إلى أسفل).

فئة الخانة العلوية اليسرى
DataGridViewTopLeftHeaderCell Class

هذه الفئة ترث الفئة DataGridViewColumnHeaderCell، وهي تمثل الخانة العلويــة اليسرى في جدول العرض، والتي عند ضغطها يتم تحديد كل خانات الجدول.

ولا تمتلك هذه الفئة أية خصائص أو وسائل جديدة غير ما ترثه من الفئة الأم.

فئة خانة رأس الصف
DataGridViewRowHeaderCell

هذه الفئة ترث الفئة DataGridViewHeaderCell، وهي تعمل كخانة رأس لأحد صفوف جدول العرض، وعند الضغط عليها يتم تحديد هذا الصف.

ولا تمتلك هذه الفئة أية خصائص أو وسائل جديدة غير ما ترثه من الفئة الأم.

اللهم ارحم أبي واغفر له وكفر عنه سيئاته

فئة طراز خانة جدول العرض
DataGridViewCellStyle Class

هذه الفئة تمثل الواجهة ICloneable، وهي تحمل معلومات التنسيق والشكل الخاص بإحدى خانات جدول العرض.

ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.

٢- والصيغة الثانية تستقبل كائن طراز خانة DataGridViewCellStyle لاستخدام خصائصه كقيم مبدئية للكائن الحالي.

وتمتلك هذه الفئة بعض الخصائص الشهيرة المألوفة لنا، مثل:

- **الهامش الخارجي Padding**
- **لون الخلفية BackColor**
- **لون الكتابة ForeColor**
- **الخط Font**
- **الوسم Tag**

كما تمتلك هذه الفئة الخصائص التالية:

المحاذاة Alignment:

تحدد كيفية محاذاة محتويات الخانة، وهي تأخذ إحدى قيم المرقم DataGridViewContentAlignment التالية:

NotSet	المحاذاة غير محددة.
TopLeft	أعلى اليسار.
TopCenter	أعلى الوسط.
TopRight	يمين الوسط.

MiddleLeft	يسار الوسط.
MiddleCenter	منتصف الوسط.
MiddleRight	يمين الوسط.
BottomLeft	أسفل اليسار.
BottomCenter	أسفل الوسط.
BottomRight	أسفل اليمين.

تنسيق Format:

تستقبل نصا يمثل الصيغة التي ستستخدم لتنسيق محتويات الخانة.. لمزيد من التفاصيل عن صيغ التنسيق، راجع ملاحق كتاب برمجة إطار العمل.

مزود التنسيق FormatProvider:

هذه الخاصية من النوع IFormatProvider، وهي تستقبل كائن معلومات الثقافة CultureInfo، الذي يحوي معلومات عن اللغة والمنطقة التي ستستخدم قواعدهما في تنسيق الأرقام والتواريخ والنصوص.. وفي الوضع الافتراضي، تستخدم هذه الخاصية الثقافة المحلية الخاصة بجهاز المستخدم.

هل لمزود التنسيق القيمة الافتراضية IsFormatProviderDefault:

تعيد True إذا كانت للخاصية FormatProvider القيمة الافتراضية، وتعيد False إذا كنت وضعت قيمة أخرى في تلك الخاصية.

القيمة المنعدمة لمصدر البيانات DataSourceNullValue:

ضع في هذه الخاصية القيمة التي تريد حفظها في مصدر البيانات، إذا ترك المستخدم الخانة الحالية فارغة.. والقيمة الافتراضية لهذه الخاصية هي العدم DbNull.

هـــــل القيمـــــة المنعدمـــــة لمصـــــدر البيانـــــات افتراضـــــية IsDataSourceNullValueDefault:

تعيد True إذا كانت للخاصية DataSourceNullValue القيمة الافتراضية، وتعيد False إذا كنت وضعت قيمة أخرى في تلك الخاصية.

القيمة المنعدمة NullValue:

ضع في هذه الخاصية كائنا Object يحمل القيمة التي تريد عرضها عندما تكون الخانة فارغة أو تحتوي على القيمة المنعدمة DbNull.. كما أن قيمة هذه الخاصية ستكون هي القيمة الافتراضية للخانات التي تضاف جديدا إلى العمود الذي له الطراز الحالي.. مثال:

```
Dgv.Columns[0].DefaultCellStyle.NullValue = "....";;
```

هذا الكود سيجعل النص "..." هو القيمة المنعدمة لخانات العمود الأول في جدول العرض، لهذا سيظهر النص "..." في آخر خانة في هذا العمود (الموجودة في الصف الجديد).

لاحظ أن المستخدم يستطيع وضع القيمة المنعدمة في الخانة أثناء تحريره لها بضغط Ctrl+0 من لوحة المفاتيح.. في هذه الحالة سيتم محو كل محتويات الخانة ووضع قيمة هذه الخاصية فيها.. والقيمة الافتراضية لهذه الخاصية هي نص فارغ "".

هل القيمة المنعدمة افتراضية IsNullValueDefault:

تعيد True إذا كانت للخاصية NullValue القيمة الافتراضية، وتعيد False إذا كنت وضعت قيمة أخرى في تلك الخاصية.

لون خلفية التحديد SelectionBackColor:

تقرأ أو تغير لون خلفية الخانة المحددة Selected Cell.

لون النص المحدد SelectionForeColor:

تقرأ أو تغير لون النص في الخانة المحددة.

طريقة الالتفاف WrapMode:

توضح هل سيلتف النص على أكثر من سطر إذا تجاوز عرض الخانة أم لا، وهي تأخذ إحدى قيم المرقم DataGridViewTriState الذي تعرفنا عليه من قبل.

كما تملك هذه الفئة الوسيلة التالية:

تطبيق الطراز ApplyStyle:

أرسل إلى هذه الوسيلة كائن طراز الخانة DataGridViewCellStyle الذي تريد نسخ قيم خصائصه إلى خصائص الكائن الحالي.

فئـــة طـــراز الحافـــة المتطـــور
DataGridViewAdvancedBorderStyle Class

هذه الفئة تمثل الواجهة ICloneable، وهي تحمل معلومات عن شكل إطار إحدى خانـــات جدول العرض.

وتمتلك هذه الفئة الخصائص التالية:

كل الحواف All:

تحـــدد طـــراز جميـــع حـــواف الخانـــة، وهـــي تأخـــذ إحـــدى قـــيم المـــرقم DataGridViewAdvancedCellBorderStyle التالية:

NotSet	غير محدد.
None	بدون حافة.
Single	خط مفرد.
Inset	خط غائر.
InsetDouble	خط مزدوج غائر.
Outset	خط بارز.
OutsetDouble	خط مزدوج بارز.
OutsetPartial	خط مفرد به جزء مرتفع.

الحافة العليا Top:

مماثلة للخاصية السابقة، إلا أنها تتعامل مع الإطار العلوي للخانة.

الحافة السفلية Bottom:

مماثلة للخاصية السابقة، إلا أنها تتعامل مع الإطار السفلي للخانة.

الحافة اليسرى Left:

مماثلة للخاصية السابقة، إلا أنها تتعامل مع الإطار الأيسر للخانة.

الحافة اليمنى Right:

مماثلة للخاصية السابقة، إلا أنها تتعامل مع الإطار الأيمن للخانة.

-١٦-

شبكة البيانات DataGrid

تعمل هذه الأداة كجدول يعرض أعمدة وصفوف، ويمكن ربطه بمصادر البيانات، بنفس الطريقة التي رأيناها في الأداة DataGridView.

ورغم أن الأداة DataGridView تمتلك قدرات أكثر من هذه الأداة، وتمنحك تحكما كاملا في كل أجزائها، وتتيح لك إنشاء أنواع جديدة من الخانات على حسب احتياجك، إلا أن شبكة البيانات تمتاز بخاصيتين لا توجدان في جدول العرض:

١- وجود عدة طرازات Styles جاهزة تمكنك من اختيار شكل الجدول مباشرة في وقت التصميم.

٢- قدرة شبكة البيانات على عرض الجداول المترابطة معا بطريقة تشبه Access، حيث يعرض كل صف في الجدول الرئيسي العلامة +، وعند ضغطها يتم عرض الصفوف الفرعية التابعة له من الجدول الثانوي.. ونظرا لأن الأداة DataGridView لا تملك مثل هذه الخاصية، فإن عليك أن تستخدم أداتين منها لعرض الجدول الأصلي والجدول الفرعي، بإضافة عمود أزرار إلى الجدول الرئيسي، وعند ضغط أي زر، يتم عرض نموذج جديد به جدول عرض يحتوي على السجلات الفرعية، كما فعلنا في المشروع DataGridViewAuthorBooks.. أو يمكنك عرض الجدول الفرعي في نفس النموذج أسفل الجدول الرئيسي، وقد رأينا كيف نفعل هذا في المشروع DataGridViewMasterDetails.

ونظرا لأهمية الخاصية الثانية، والتي قد تدفعك إلى استخدام شبكة البيانات في بعض مشاريعك على سبيل التسهيل، فقد رأيت أنه من الأفضل أن نتعرف على الأداة DataGrid، وهي على كل حال، ليست بضخامة الأداة DataGridView.

وعليك إضافة هذه الأداة أولا إلى صندوق الأدوات، بالضغط بزر الفأرة الأيمن في أي موضع تحت الشرط Data، وضغط الأمر Choose Items، ومن ثم اختيار العنصر DataGrid من قائمة الأدوات، مع التأكد أن العمود الثاني في القائمة يشير إلى أن العنصر الذي اختره ينتمي إلى النطاق System.Windows.Forms، لأن هناك أداة شبكة بيانات أخرى خاصة بتطبيقات الويب.

واجهة خدمة التحرير
IdataGridEditingService Interface

تمنح هذه الواجهة مستخدم شبكة البيانات القدرة على تحرير خاناته، وهي تملك الوسيلتين التاليتين:

بدء التحرير BeginEdit:

تبدأ تحرير خانة في شبكة البيانات، وهي تستقبل المعاملين التاليين:

- كائن طراز العمود DataGridColumnStyle الذي يمثل العمود الذي توجد به الخانة المراد تحريرها.
- رقم الصف الذي توجد به الخانة المراد تحريرها.

وتعيد هذه الوسيلة true إذا نجحت في بدء التحرير.

إنهاء التحرير EndEdit:

تنهي تحرير خانة في شبكة البيانات، وهي تستقبل المعاملين التاليين:

- كائن طراز العمود DataGridColumnStyle الذي يمثل العمود الذي توجد به الخانة المراد تحريرها.
- رقم الصف الذي توجد به الخانة المراد تحريرها.
- معامل منطقي إذا جعلته true، فسيتم إلغاء القيمة الجديدة التي تم تحريرها، والعودة إلى القيمة الأصلية للخانة.. أما إذا جعلتها false، فسيتم حفظ القيمة الجديدة في الخانة.

وتعيد هذه الوسيلة true إذا نجحت في إنهاء التحرير، وتعيد false إذا فشلت في حفظ قيمة الخانة، وفي هذه الحالة تظل الخانة في وضع التحرير.

فئة طراز شبكة البيانات DataGridTableStyle Class

هذه الفئة ترث الفئة Component، وتمثل الواجهة IdataGridEditingService، وهي تتيح لك التحكم في شكل الجدول الذي ترسمه الأداة DataGrid عند عرض البيانات.

ولحدث إنشاء هذه الفئة الصيغ التالية:

١- الصيغة الأولى بدون معاملات.

٢- الصيغة الثانية تستقبل معاملا منطقيا، إذا جعلت قيمته true، فسيشير هذا إلى أن طراز الجدول الحالي هو الطراز الافتراضي.

٣- الصيغة الثالثة تستقبل مدير التداول CurrencyManager الذي يتحكم في الارتباط بمصدر البيانات، ليستخدمه طراز الجدول في إنشاء الأعمدة وعرض البيانات فيها.

شبكة البيانات DataGrid:

تقرأ أو تغير كائن شبكة البيانات DataGrid الذي يستخدم طراز الجدول.

اسم الخريطة MappingName:

ضع في هذه الخاصية اسم عنصر البيانات الذي يعرضه طراز الجدول الحالي، وفي الغالب سيكون اسم أحد جداول مجموعة البيانات، مثل "Authors".. هذا يتيح لك استخدام أكثر من طراز جدول في شبكة البيانات، كل منها يعرض جدولا من جداول قاعدة البيانات بشكل وألوان وتنسيق خاصة به.

طرازات أعمدة الجدول GridColumnStyles:

تعيد مجموعة طرازات أعمدة الجدول GridColumnStylesCollection، وهي ترث الفئة BaseCollection، كما أنها تمثل واجهة القائمة IList.. وتحتوي هذه المجموعة على عناصر من النوع DataGridColumnStyle، كل منها يمثل أحد

الأعمدة المرسومة في طراز الجدول الحالي.. وسنتعرف على الفئة DataGridColumnStyle لاحقا.

عناوين الأعمدة مرئية ColumnHeadersVisible:

إذا جعلت قيمة هذه الخاصية true (وهي القيمة الافتراضية)، فسيتم عرض صف رؤوس الأعمدة.

عناوين الصفوف مرئية RowHeadersVisible:

إذا جعلت قيمة هذه الخاصية true (وهي القيمة الافتراضية)، فسيتم عرض عمود رؤوس الصفوف.

العرض المفضل للعمود PreferredColumnWidth:

تقرأ أو تغير العرض المبدئي الذي ستأخذه الأعمدة عند إنشائها في طراز الجدول الحالي.

الارتفاع المفصل للصفوف PreferredRowHeight:

تقرأ أو تغير الارتفاع المبدئي الذي ستأخذه الصفوف عند إنشائها في طراز الجدول الحالي.

عرض عناوين الصفوف RowHeaderWidth:

تقرأ أو تغير عرض عمود رؤوس الصفوف.

للقراءة فقط ReadOnly:

إذا جعلت قيمة هذه الخاصية true، فلن يتمكن المستخدم من تغيير قيمة أي خانة في طراز الجدول الحالي.

السماح بالترتيب AllowSorting:

إذا جعلت قيمة هذه الخاصية true، فسيتمكن المستخدم من ترتيب صفوف الجدول تبعا للعمود الذي يضغط خانة عنوانه.

لون الخلفية BackColor:

تقرأ أو تغير لون خلفية الصفوف الزوجية في شبكة البيانات.

لون الخلفية التبادلي AlternatingBackColor:

تقرأ أو تغير لون خلفية الصفوف الفردية في شبكة البيانات.

لون النص ForeColor:

تقرأ أو تغير اللون المستخدم في كتابة نصوص الخانات.

لون خلفية التحديد SelectionBackColor:

تقرأ أو تغير لون خلفية الخانات المحددة.

لون نص التحديد SelectionForeColor:

تقرأ أو تغير لون نصوص خلفية الخانات المحددة.

لون خطوط الشبكة GridLineColor:

تقرأ أو تغير لون الخطوط الفاصلة بين الصفوف والأعمدة.

طراز خطوط الشبكة GridLineStyle:

تتحكم في شكل الخطوط الفاصلة بين الصفوف والأعمدة، وهي تأخذ إحدى قيمتي المرقم DataGridLineStyle التاليتين:

None	لا يتم رسم خطوط الشبكة.
Solid	يتم رسم خطوط الشبكة.

لون خلفية العناوين HeaderBackColor:

تقرأ أو تغير لون خلفية خانات عناوين الأعمدة والصفوف.

لون نصوص العناوين HeaderForeColor:

تقرأ أو تغير لون نصوص خانات عناوين الأعمدة والصفوف.

خط العناوين HeaderFont:

تقرأ أو تغير خط الكتابة المستخدم في خانات عناوين الأعمدة والصفوف.

لون الروابط LinkColor:

تقرأ أو تغير لون نصوص الوصلات Links الموجودة في خانات الجدول.

لون التحليق فوق الروابط LinkHoverColor:

تقرأ أو تغير لون نصوص الوصلات Links الموجودة في خانات الجدول، عند التحليق فوقها بالفأرة.

وتمتلك هذه الفئة عدة وسائل، لكنها غير هامة، فكلها تبدأ بالكلمة Reset متبوعة باسم إحدى الخصائص، ومهمتها إعادة قيمة تلك الخاصية إلى قيمتها الافتراضية، مثل ResetBackColor.

كما تمتلك هذه الفئة عدة أحداث، لكنها كلها تنطلق عند تغير قيمة إحدى الخصائص، مثل الحدث RowHeaderWidthChanged الذي ينطلق عندما يتغير عرض عمود رؤوس الصفوف.

واجهة التنبيه بتحرير عمود شبكة البيانات
IDataGridColumnStyleEditingNotificationService Interface

تنبه هذه الواجهة عمود شبكة البيانات بأن هناك أداة تحرير تستخدم حاليا مع إحدى خاناته، وهي تمتلك الوسيلة التالية:

بدء تحرير العمود ColumnStartedEditing:

تخبر شبكة البيانات بأن المستخدم بدأ تحرير إحدى الخانات، ولها معامل واحد من النوع Control، يستقبل أداة التحرير المستخدمة في الخانة.

فئة طراز العمود DataGridColumnStyle

هذه الفئة أساسية مجردة Abstract Base Class تجب وراثتها، وهي ترث الفئة Component، كما أنها تمثل الواجهة IDataGridColumnStyleEditingNotificationService.. وتعمل الفئات المشتقة من هذه الفئة كأعمدة في شبكة البيانات، كما أنها تتحكم في شكل وتنسيق خانات هذه الأعمدة.

وتمتلك هذه الفئة الخصائص التالية:

طراز شبكة البيانات DataGridTableStyle:

تعيد كائن طراز الجدول DataGridTableStyle، الذي يحتوي على كائن طراز العمود الحالي.

المحاذاة Alignment:

تحدد محاذاة النص في خانات العمود، وهي تأخذ إحدى قيم المرقم HorizontalAlignment التالية: Left ـ Right ـ Center.

عنوان العمود HeaderText:

تقرأ أو تغير النص المعروض في خانة عنوان العمود.

اسم الخريطة MappingName:

تحدد اسم عنصر البيانات Data Member الذي يعرضه طراز العمود.. وفي الغالب تحتوي هذه الخاصية على اسم أحد أعمدة مجموعة البيانات.

النص المنعدم NullText:

ضع في هذه الخاصية النص الذي ستعرضه خانات العمود إذا كانت فارغة.

للقراءة فقط ReadOnly:

إذا جعلت قيمة هذه الخاصية true، فلن يستطيع المستخدم تغيير قيمة أية خانة في العمود الحالي، والقيمة الافتراضية هي false.

العرض Width:

تقرأ أو تغير عرض العمود الحالي.

واصف الخاصية PropertyDescriptor:

تقرأ أو تغير كائن واصف الخاصية PropertyDescriptor الذي يحوي على سمات العمود.. هذا يمكنك من تحديد نوع البيانات التي يقبلها العمود.

وتمتلك هذه الفئة الوسيلة التالية:

تصفير عنوان العمود ResetHeaderText:

تعيد الخاصية HeaderText إلى قيمتها الافتراضية، وهي نص فارغ "".

كما تمتلك هذه الفئة مجموعة من الأحداث، أهمها الحدث التالي:

العرض تغير WidthChanged:

ينطلق إذا تغير عرض العمود.

فئة عمود النصوص DataGridTextBoxColumn Class

هذه الفئة ترث الفئة DataGridColumnStyle، وهي تمثل عمودا في شبكة البيانات تعرض خاناته نصوصا.

ولحدث إنشاء هذه الفئة الصيغ التالية:

١- الصيغة الأولى بدون معاملات.

٢- الصيغة الثانية تستقبل واصف الخاصية PropertyDescriptor المستخدم مع العمود الحالي.

٣- الصيغة الثالثة تزيد على الصيغة السابقة بمعامل منطقي، إذا جعلته true فسيتم اعتبار العمود الحالي عمودا افتراضيا Default Column.

٤- الصيغة الرابعة تزيد على الصيغة الثانية بمعامل نصي، يستقبل الصيغة المستخدمة في تنسيق النصوص في خانات العمود.

٥- الصيغة الخامسة تزيد على الصيغة السابقة بمعامل منطقي، إذا جعلته true فسيتم اعتبار العمود الحالي عمودا افتراضيا Default Column.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

التنسيق Format:

ضع في هذه الخاصية نصا يحمل الصيغة المستخدمة لتنسيق خانات العمود.

معلومات التنسيق FormatInfo:

هذه الخاصية من النوع IFormatProvider، وهي تستقبل كائن معلومات الثقافة CultureInfo، الذي يحوي معلومات عن اللغة والمنطقة التي ستستخدم قواعدهما في تنسيق الأرقام والتواريخ والنصوص.. وفي الوضع الافتراضي، تستخدم هذه الخاصية الثقافة المحلية الخاصة بجهاز المستخدم.

مربع النص TextBox:

تعيد كائن مربع النص TextBox، الذي يستضيفه العمود الحالي ويعرضه في الخانـــة التي يقوم المستخدم بتحريرها.

فئة العمود المنطقي DataGridBoolColumn Class

هذه الفئة ترث الفئة DataGridColumnStyle، وهي تمثل عمودا في شبكة البيانات تعرض خاناته مربع اختيار CheckBox، ليمثل القيم المنطقية Boolean.

ولحدث إنشاء هذه الفئة الصيغ التالية:

١- الصيغة الأولى بدون معاملات.

٢- الصيغة الثانية تستقبل واصف الخاصية PropertyDescriptor المستخدم مع العمود الحالي.

٣- الصيغة الثالثة تزيد على الصيغة السابقة بمعامل منطقي، إذا جعلته true فسيتم اعتبار العمود الحالي عمودا افتراضيا Default Column.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

السماح بالعدم AllowNull:

إذا جعلت قيمة هذه الخاصية false، فسيكون لمربع الاختيار حالتان فقط: Checked و Unchecked.. والقيمة الافتراضية لهذه الخاصية هي true، وفي هذه الحالة سيكون لمربع الاختيار الحالة الثالثة غير المحددة Indeterminate إذا كانت قيمة الخانة DbNull.

القيمة المنعدمة NullValue:

ضع في هذه الخاصية القيمة التي تعتبر عدما، والتي ستجعل مربع الاختيار في الحالة غير المحددة.. يمكنك مثلا استخدام الرقم -١ كمناظر للقيمة غير المحددة.

القيمة الخاطئة FalseValue:

ضع في هذه الخاصية القيمة التي تعتبر false، والتي ستجعل مربع الاختيار في حالة عدم الاختيار Unchecked.. يمكنك مثلا استخدام الرقم صفر كمناظر للقيمة الخاطئة.

القيمة الصحيحة TrueValue:

ضع في هذه الخاصية القيمة التي تعتبر true، والتي ستجعل مربع الاختيار في حالة الاختيار Checked.. يمكنك مثلا استخدام الرقم ١ كمناظر للقيمة الخاطئة.

سجل خانة الشبكة
DataGridCell Structure

يعمل هذا السجل كخانة في شبكة البيانات، ولحدث إنشائه معاملان:

- رقم الصف الذي توجد به الخانة.
- رقم العمود الذي توجد به الخانة.

ويمتلك هذا السجل الخاصيتين التاليتين:

رقم الصف RowNumber:

تقرأ أو تغير رقم الصف الذي توجد به الخانة.

رقم العمود ColumnNumber:

تقرأ أو تغير رقم العمود الذي توجد به الخانة.

فئة شبكة البيانات DataGrid Class

هذه الفئة ترث فئة الأداة الأم Control، وتمثل الواجهة IdataGridEditingService، وهي تمتلك الخصائص التالية:

مصدر البيانات DataSource:

ضع في هذه الخاصية الكائن الذي يعمل كمصدر للبيانات التي تعرضها شبكة البيانات، مثل مجموعة البيانات DataSet أو جدول البيانات DataTable أو عرض البيانات DataView أو أية مجموعة تحتوي على كائنات بها خصائص عامة.

عنصر البيانات DataMember:

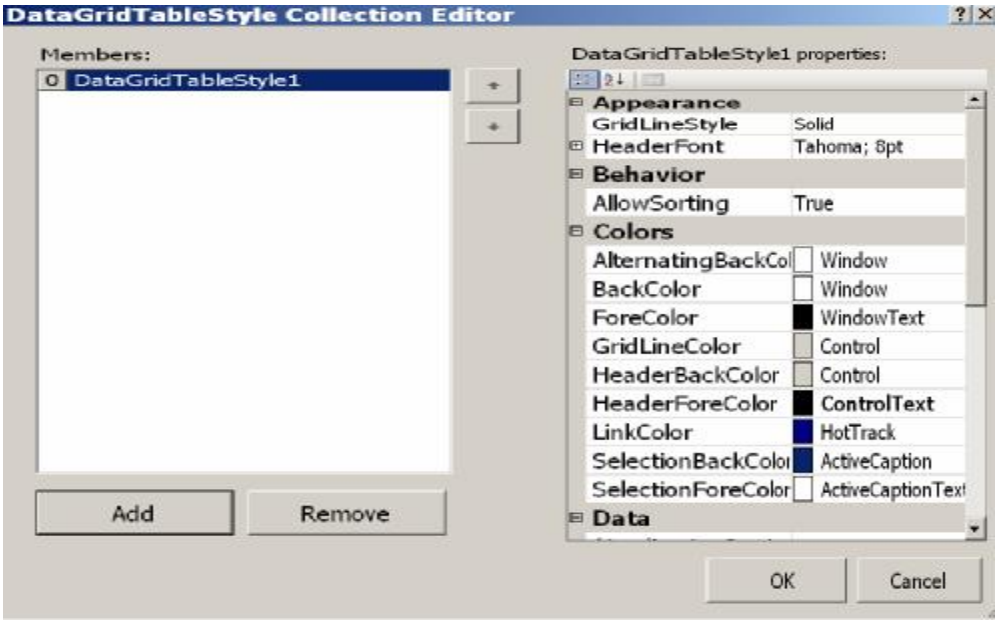
ضع في هذه الخاصية اسم عنصر البيانات المراد عرضه، مثل اسم جدول المؤلفين "Authors".

طرازات الجداول TableStyles:

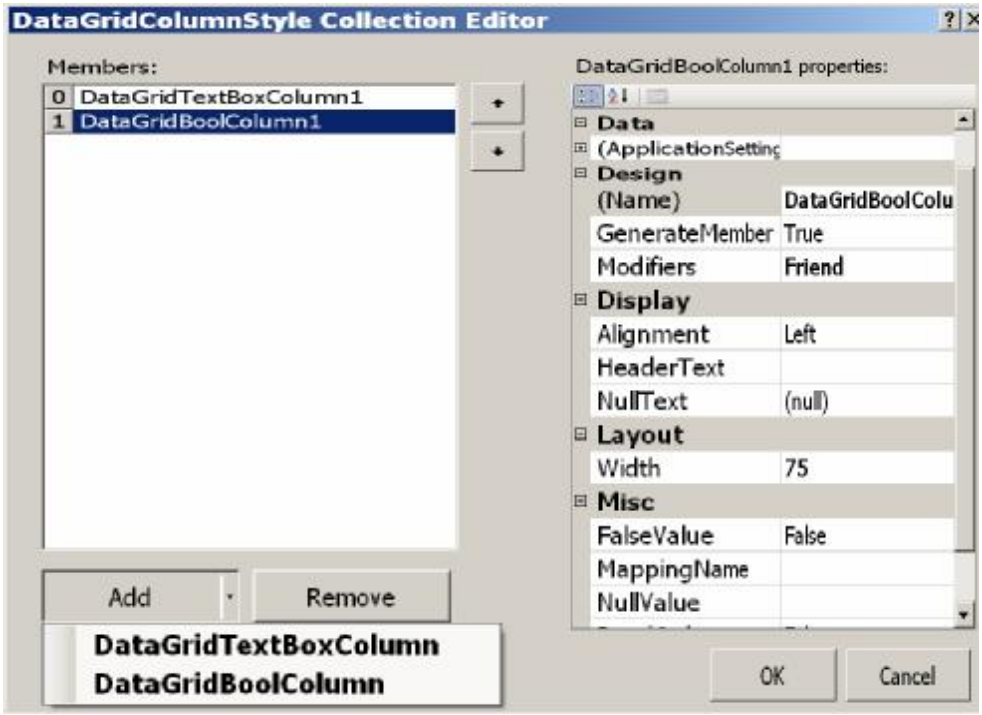
تعيد مجموعة طرازات الجداول GridTableStylesCollection، وهي ترث الفئة BaseCollection وتمثل واجهة لقائمة IList، وكل عنصر من عناصرها من النوع GridTableStyles.

ويمكنك إضافة العناصر إلى هذه المجموعة بشكل مرئي في وقت التصميم، وذلك باستخدام نافذة الخصائص، حيث يؤدي ضغط الزر الموجود في خانة هذه الخاصية إلى عرض النافذة الموضحة في الصورة:

في هذه النافذة، يمكنك ضغط الزر Add لإضافة طراز جدول جديد، واستخدام الخصائص المعروضة في النصف الأيمن من النافذة للتحكم في خصائص هذا الطراز، وربطه بجدول البيانات باستخدام الخاصية MappingName.. كما يمكنك استخدام الخاصيّة GridColumnStyles للتحكم في طراز أعمدة الجدول.

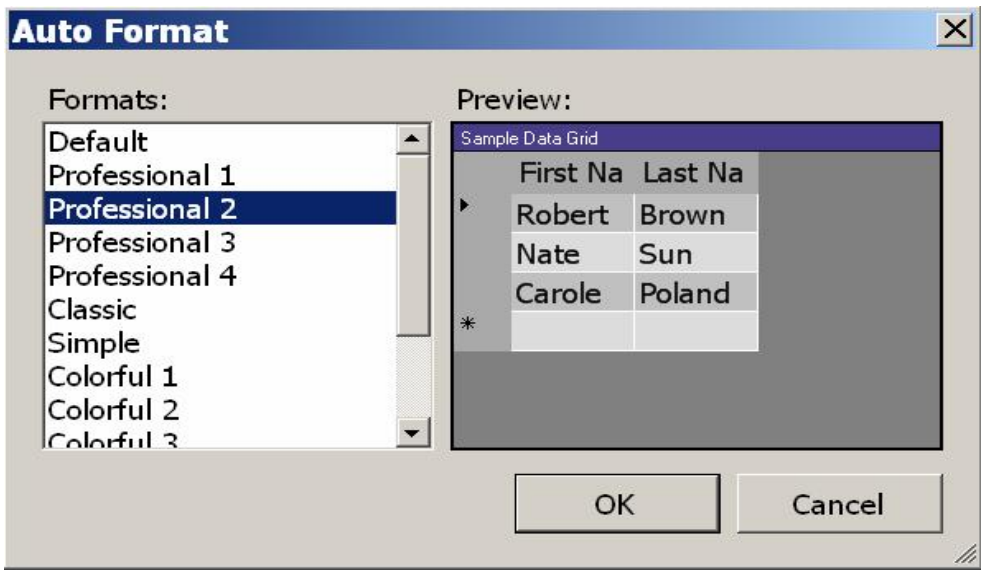


لفعل هذا اضغط زرّ الانتقال المجاور لهذه الخاصيّة، لتظهر لك نافذة محرر مجموعـــة طرازات الأعمدة:



في هذه النافذة يمكنك إضافة طراز عمود جديد بضغط الزر Add.. لاحظ وجود سهم في جانب الزر Add، ولو ضغطته بالفأرة، فستظهر قائمة موضعية، تتيح لك اختيار نوع العمود الذي تريد إضافة طراز له، سواء كان عمود نصوص DataGridTextBoxColumn، أو عمودا منطقيا DataGridBoolColumn.. ويمكنك تغيير خصائص طراز العمود من النصف الأيمن للنافذة.. ولا تنسَ أن تضع في الخاصيّة MappingName اسم العمود الأصلي الذي يعرض طراز العمود بياناته.

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



السماح بالتصفح AllowNavigation:

إذا جعلت قيمة هذه الخاصية true (وهي القيمة الافتراضية)، فستعرض شبكة البيانات الوصلات التي تتيح لك استعراض الجداول الفرعية.

ويمكنك اختبار تأثير هذه الخاصية في التطبيق DataGridNavigation، فلو شـــغّلت هذا البرنامج فلن تظهر في الجدول أيّ بيانات.. وإنمـــا ســـتظهر العلامـــة "+"، ولـــو ضغطتها فستعرض لك اسمي جدولي المؤلّفين والكتب.

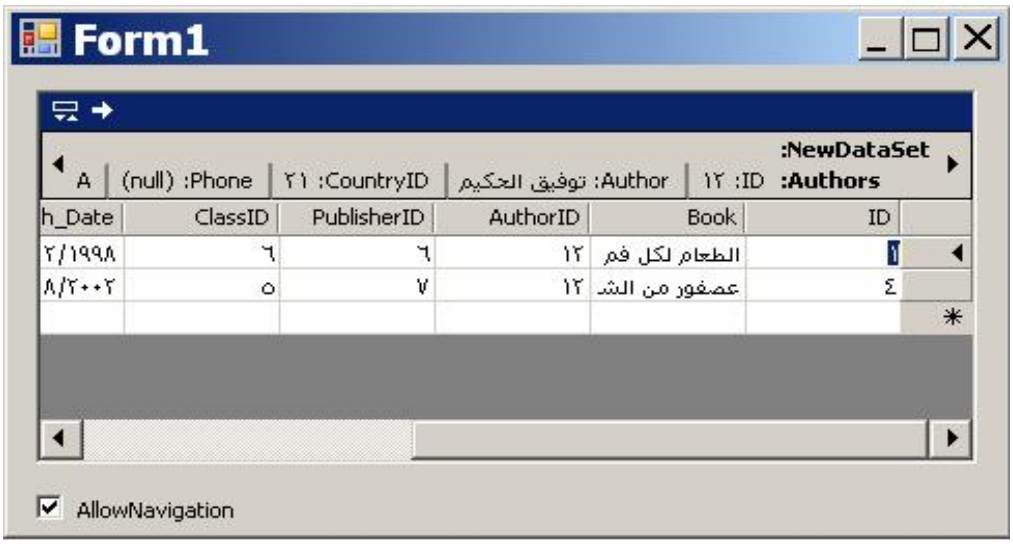
ولو أزلت علامة الاختيار من مرع الاختيار AllowNavigation، فستختفي العلامة + واسما الجدولين، وسيظل جدول العرض فارغا!

أعد وضع علامة الاختيار، واضغط اسم جدول المؤلّفين Authors.. سيمتلئ الجدول بسجلات جدول المؤلّفين، وفي الهامش العلوي لشبكة البيانات سيظهر اســـم مجموعـــة البيانات.. لهذا سيكون مفيدا أن تعطي مجموعة البيانات اسما واضحا بدلا من الاســـم الافتراضي NewDataSet.. وقد أسميناها في هذا المشروع "قاعدة بيانات الكتب".

وأمام كلّ سجلّ من سجلات المؤلفين، سترى العلامة "+".. ولو ضغطت هذه العلامة، فسيظهر لك اسم العلاقة بين الجدولين، وسيكون مفيدا أيضا لو أعطيت العلاقة اســـما واضحا للمستخدم بدلا من الاسم الافتراضي Relation1.. وقد أســـميناها فـــي هـــذا المشروع "كتب المؤلف":

وأيضا، لو أزلت علامة الاختيار من مرع الاختيار AllowNavigation، فستختفي العلامة + وستظهر سجلات جدول المؤلفين بمفردها.

اضغط اسم العلاقة بالفأرة.. سيتمّ عرض أسماء الكتب الخاصّة بهذا المؤلف، كما سيظهر سجل هذا المؤلف أعلى الجدول:



ولو أردت إخفاء سجل المؤلف المعروض أعلى الجدول، فاضغط العلامة ▤ الموجودة أعلى يسار الجدول، ولإعادة عرض هذا السجل، فاضغط نفس العلامة مرة أخرى.. ولو أردت العودة إلى جدول المؤلفين، فاضغط السهم ➡ الموجود أعلى شبكة البيانات.

لاحظ أن شبكة البيانات تعرض اسمي جدولي المؤلفين والكتب عند تشغيل البرنامج، لأننا ربطناه بمجموعة البيانات كلها، ولو كانت فيها جداول أكثر لظهرت اسماؤها كلها:

```
DataGrid1.DataSource = Ds;
```

ولو أردت عرض جدول واحد فقط كجدول المؤلفين، فاستخدم الجملة التالية:

```
DataGrid1.DataSource = Ds.Tables["Authors"];
```

في هذه الحالة ستعرض شبكة البيانات سجلات جدول المؤلفين مباشرة، مع ظهور العلامة + بجوار كل سجل من سجلاته، ليمكنك عرض السجلات الفرعية في جدول الكتب.. أما لو استخدمت الجملة التالية:

```
DataGrid1.DataSource = Ds.Tables["Books"];
```

فستعرض شبكة البيانات سجلات جدول الكتب فقط، ولن تجد أية طريقة لعرض أية سجلات من جدول المؤلفين.

لون الأرضية BackgroundColor:

تقرأ أو تغير لون أرضية شبكة البيانات (الجزء الذي لا توجد فيه خانات).

العنوان مرئي CaptionVisible:

إذا جعلت قيمة هذه الخاصية false، فلن يظهر شريط العنوان أعلى شبكة البيانات.. والقيمة الافتراضية لهذه الخاصية هي true.

نص العنوان CaptionText:

تقرأ أو تغير النص المعروض في شريط العنوان.

لون خلفية العنوان CaptionBackColor:

تقرأ أو تغير لون خلفية شريط العنوان.

لون نص العنوان CaptionForeColor:

تقرأ أو تغير لون النص المعروض في شريط العنوان.

خط العنوان CaptionFont:

تقرأ أو تغير الخط المستخدم لكتابة النص المعروض في شريط العنوان.

رقم الصف الحالي CurrentRowIndex:

تقرأ أو تغير رقم الصف المحدد حاليا في شبكة البيانات.

الخانة الحالية CurrentCell:

تعيد خانة شبكة البيانات DataGridCell المحددة حاليا.

المفهرس Indexer:

تقرأ أو تغير قيمة خانة معينة في جدول العرض.. ولهذه الخاصية صيغتان:

١- الأولى تستقبل كائن الخانة DataGridCell التي تريد التعامل معها.

٢- الثانية تستقبل رقم الصف ورقم العمود اللذين توجد فيهما الخانة.

أول عمود مرئي FirstVisibleColumn:

تعيد رقم أول عمود ظاهر على الشاشة.

عدد الأعمدة المرئية VisibleColumnCount:

تعيد عدد الأعمدة الظاهرة حاليا على الشاشة.

عدد الصفوف المرئية VisibleRowCount:

تعيد عدد الصفوف الظاهرة حاليا على الشاشة.

العرض المسطح FlatMode:

إذا جعلت قيمة هذه الخاصية true، فستظهر شبكة البيانات بشكل مسطح.. والقيمة الافتراضية لهذه الخاصية هي false.

الصفوف الرئيسية مرئية ParentRowsVisible:

إذا جعلت قيمة هذه الخاصية true (وهي القيمة الافتراضية)، فسيظهر الصف الرئيسي أعلى شبكة العرض عند عرض الصفوف الفرعية.

لون خلفية الصفوف الرئيسية ParentRowsBackColor:

تقرأ أو تغير لون خلفية الصف الرئيسي.

لون نص الصفوف الرئيسية ParentRowsForeColor:

تقرأ أو تغير لون نص الصف الرئيسي.

طراز لافتة الصفوف الرئيسية ParentRowsLabelStyle:

تتحكم في عنوان لافتة الصف الرئيسي، وهي تأخذ إحدى قيم المرقم DataGridParentRowsLabelStyle التالية:

None	لا تعرض اللافتة أي عنوان.
TableName	تعرض اللافتة اسم الجدول.
ColumnName	تعرض اللافتة اسم العمود الرئيسي (المفتاح الأساسي).
Both	تعرض اللافتة اسم الجدول واسم العمود الرئيسي.

كما يمتلك جدول العرض الخصائص التالية:

- عناوين الأعمدة مرئية ColumnHeadersVisible
- عناوين الصفوف مرئية RowHeadersVisible
- العرض المفضل للعمود PreferredColumnWidth
- الارتفاع المفصل للصفوف PreferredRowHeight
- عرض عناوين الصفوف RowHeaderWidth
- للقراءة فقط ReadOnly
- السماح بالترتيب AllowSorting
- لون الخلفية BackColor
- لون الخلفية المتبدل AlternatingBackColor
- لون النص ForeColor
- لون خلفية التحديد SelectionBackColor
- لون نص التحديد SelectionForeColor
- لون خطوط الشبكة GridLineColor
- طراز خطوط الشبكة GridLineStyle
- لون خلفية العناوين HeaderBackColor
- لون نصوص العناوين HeaderForeColor
- خط العناوين HeaderFont
- لون الروابط LinkColor
- لون التحليق فوق الروابط LinkHoverColor

كما تلاحظ، فإن هذه الخصائص موجودة بنفس الاسم والوظيفة في كائن طـراز الجـدول DataGridTableStyle، لهذا لن نعيد شرحها هنا.. عليك فقط أن تعـرف أن خصـائص طراز الجدول يكون لها الأولوية على خصائص شبكة البيانات، ولا تؤثر هذه الأخيـرة إلا على الجدول الذي تعرضه بدون إنشاء طراز جدول خاص به.

كما تمتلك شبكة العرض الوسائل التالية:

إسدال Expand:

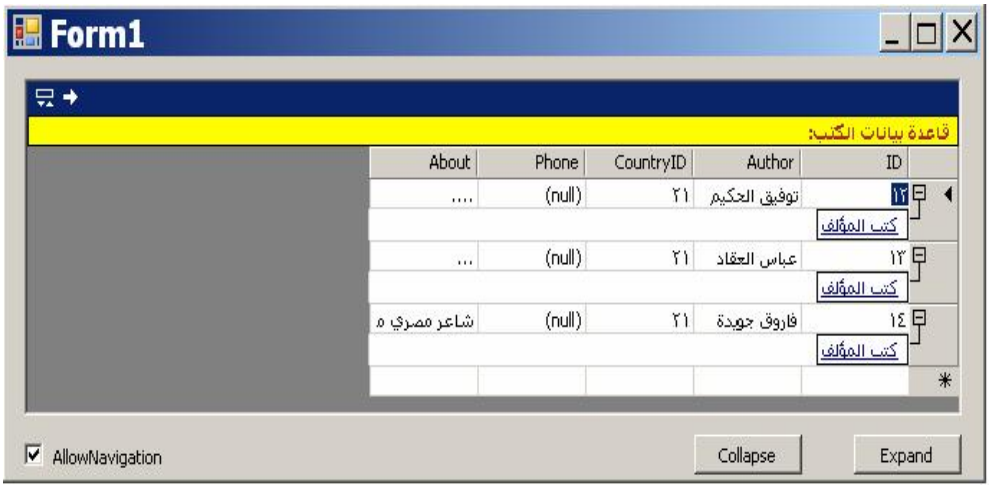
أرسل إلى هذه الوسيلة رقم الصف الذي تريد إسدال العلاقة التابعة له (كأنك ضغطت العلامة + المجاورة له).. فإذا لم تكن للصف المطلوب صفوف فرعية، فلن يحدث أي خطأ، ولن تفعل هذه الوسيلة شيئا.

ويمكنك إرسال الرقم ١- كمعامل لإسدال علاقات كل الصفوف:

```
DataGrid1.Expand(-1);
```

وستجد زرا اسمه Expand في المشروع DataGridNavigation، وعند الضغط عليه سيتم إسدال كل جداول مجموعة البيانات، أو إسدال اسم العلاقة لكل صف من صفوف جدول المؤلفين، كما في الصورة.

لاحظ أنك إذا أزلت علامة الاختيار من مربع الاختيار AllowNavigation، فستختفي العلامة + من جوار كل صف، لكن ضغط الزر Expand سيظل يسدل اسم العلاقة لكل صف، لكنها ستكون عاطلة عن العمل، ولن يؤدي ضغطها إلى عرض الصفوف الفرعية.



هل هو مسدل IsExpanded:

تعيد true إذا كانت علاقة الصف الذي أرسلت إليها رقمه مسدلة.. لاحظ أن خطأ سيحدث إذا أرسلت إلى هذه الوسيلة الرقم -١.

طي Collapse:

أرسل إلى هذه الوسيلة رقم الصف الذي تريد طي العلاقة التابعة له (كأنك ضغطت العلامة - المجاورة له).. فإذا لم تكن للصف المطلوب صفوف فرعية، فلن يحدث أي خطأ، ولن تفعل هذه الوسيلة شيئا.

ويمكنك إرسال الرقم ١- كمعامل لطي كل علاقات الصفوف:

```
DataGrid1.Collapse (-1);
```

وستجد زرا اسمه Collapse في المشروع DataGridNavigation، عند الضغط عليه يتم طي كل جداول مجموعة البيانات، أو طي اسم العلاقة لكل صف من صفوف جدول المؤلفين.

معرفة حدود الخانة GetCellBounds:

تعيد كائن المستطيل Rectangle الذي يمثل موضع وأبعاد الخانة المرسلة كمعامل.. ولهذه الوسيلة صيغتان:

١- الأولى تستقبل كائن الخانة DataGridCell.

٢- والثانية تستقبل رقم الصف ورقم العمود اللذين توجد فيهما الخانة.

معرفة حدود الخانة الحالية GetCurrentCellBounds:

تعيد كائن المستطيل Rectangle الذي يمثل موضع وأبعاد الخانة المحددة حاليا في شبكة البيانات.

هل هو محدد IsSelected:

تعيد true إذا كان الصف الذي أرسلت إليها رقمه محددا.

تحديد Select:

إضافة إلى الصيغة الموروثة من الفئة Control والتي تحدد شبكة البيانات نفسها (تنقل غليها المؤشر Focus)، توجد صيغة أخرى تحدد الصف الذي ترسل إليها رقمه كمعامل.

إلغاء التحديد UnSelect:

تزيل تحديد الصف الذي ترسل إليها رقمه كمعامل.

تصفير التحديد ResetSelection:

تزيل تحديد كل صفوف شبكة البيانات.

الانتقال إلى الخلف NavigateBack:

تعرض الجدول الرئيسي الذي كان معروضا قبل الجدول الحالي في شبكة البيانات.. فمثلا: لو كانت كتب توفيق الحكيم معروضة حاليا، فستعيد هذه الوسيلة عرض جدول المؤلفين.. أما إذا لم يكن هناك جدول سابق، فلن يحدث خطأ في البرنامج، ولن تفعل هذه الوسيلة شيئا.. هذا معناه أنها تؤدي نفس وظيفة زر التراجع الموجود أعلى شبكة البيانات.. ويمكنك تجربة هذه الوسيلة بضغط الزر NavigateBack في المشروع DataGridNavigation.

الانتقال إلى NavigateTo:

أرسل إلى هذه الوسيلة رقم الصف، واسم العلاقة، لعرض سجلاته الفرعية في شبكة البيانات، تماما كأن المستخدم ضغط العلاقة الخاصة بهذا الصف.. ولا تسبب هذه الوسيلة أي خطأ في البرنامج إذا لم يكن الصف المطلوب مرتبطا بالعلاقة المذكورة، ولكنها بدلا من هذا تقوم بالعودة إلى الجدول الرئيسي مجددا، كأنك استدعيت الوسيلة

NavigateBack.. ويمكنك تجربة هذه الوسيلة بضغط الزر NavigateTo في المشروع DataGridNavigation.

تغيير ربط البيانات SetDataBinding:

تربط شبكة العرض بمصدر البيانات، وهي تستقبل معاملين:

- الكائن الذي يعمل كمصدر للبيانات Data Source.
- اسم الجدول أو المجموعة التي تعمل كعنصر للبيانات.

اختبار الضغط HitTest:

تعيد كائن معلومات اختبار الضغط HitTestInfo الذي يحوي معلومات عن النقطة المرسلة كمعامل إلى هذه الخاصية، سواء كانت في صورة كائن نقطة Point أو في صورة الإحداثيين الأفقي X والرأسي Y.

والفئة HitTestInfo معرفة داخل الفئة DataGrid، وهي تمتلك الخصائص التالية:

	Nowhere	تعيد كائن معلومات اختبار HitTestInfo، يشير إلى نقطة موجودة في منطقة فارغة من جدول العرض (ليست بها خانات عادية أو خانات عناوين).
	Column	تعيد رقم العمود الذي توجد فيه نقطة الاختبار.
	Row	تعيد رقم الصف الذي توجد فيه نقطة الاختبار.
	Type	تعيد إحدى قيم المرقم DataGrid.HitTestType، التي تخبرك بنوع المنطقة التي توجد بها نقطة الاختبار.. وهذه القيم هي: - None: منطقة فارغة. - Cell: خانة. - ColumnHeader: رأس عمود. - RowHeader: رأس صف.

		- ColumnResize: الخط الرأسي الفاصل بين رأسي عمودين. - RowResize: الخط الأفقي الفاصل بين رأسي صفين. - Caption: شريط العنوان العلوي لشبكة البيانات. - ParentRows: الصفوف الرئيسية المعروضة أعلى شبكة البيانات.

كما تمتلك شبكة البيانات الأحداث التالية:

الخانة الحالية تغيرت CurrentCellChanged:

ينطلق عندما تتغير الخانة المحددة حاليا في شبكة البيانات.

ضغط زر الرجوع BackButtonClick:

ينطلق عندما يضغط المستخدم زر الرجوع إلى الخلف الموجود أعلى شبكة البيانات.

تصفح Navigate:

ينطلق عندما تنتقل شبكة البيانات لعرض جدول آخر.. والمعامل الثاني e لهذا الحدث من النوع NavigateEventArgs، وهو يمتلك الخاصية Forward التي تعيد true إذا كان الانتقال إلى الأمام (إلى جدول فرعي).

ضغط زر عرض تفاصيل السجل الرئيسي ShowParentDetailsButtonClick:

ينطلق عندما يضغط المستخدم زر عرض أو إخفاء تفاصيل السجل الرئيسي، المعروض أعلى شبكة البيانات.

انزلاق Scroll:

ينطلق عندما يتحرك أحد المنزلقين الأفقي أو الرأسي في شبكة البيانات.

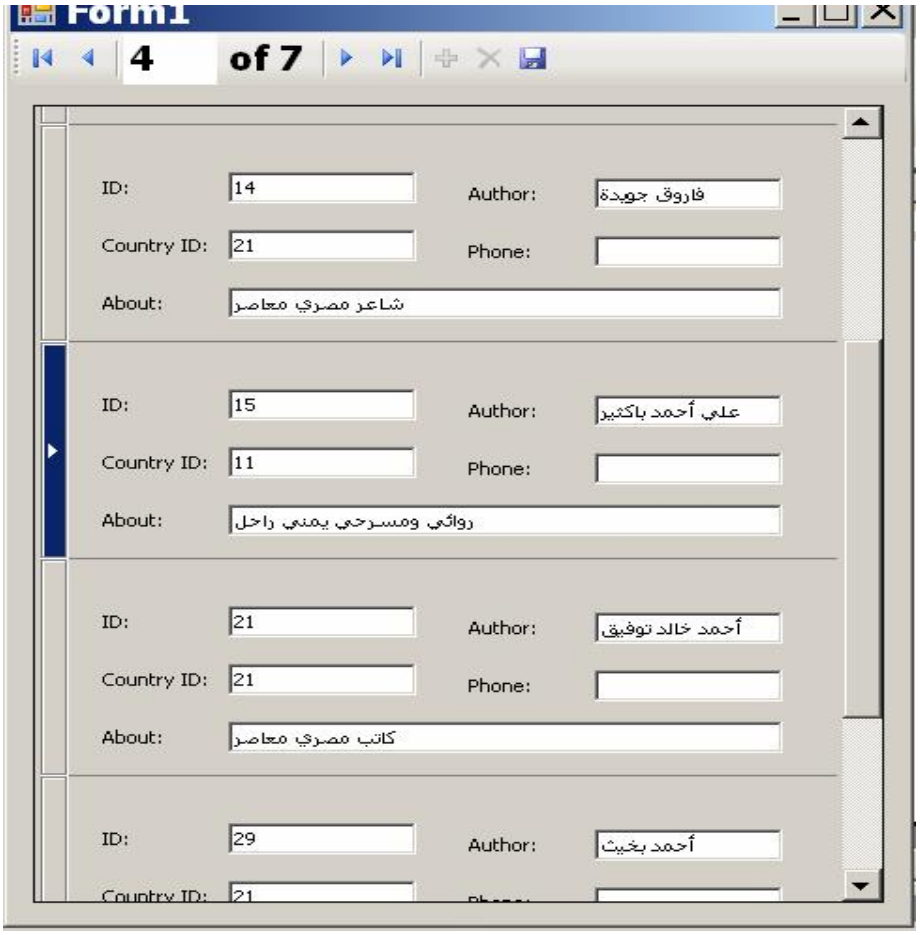
والآن، لعلك لاحظت مدى تواضع الأداة DataGrid بالنسبة إلى الأداة DataGridView، وإن كانت الأولى ما تزال صالحة للاستخدام في الحالات التي تريد فيها عرض بعض البيانات دون الحاجة إلى التحكم الكامل في الأعمدة والصفوف ومحتويات الخانات، فكما ترى، لا تقدم لك شبكة البيانات أية طريقة للتعامل مع الصفوف والأعمدة والخانات، إلا بمعرفة أرقامها، أو بتغيير طراز عرض الأعمدة، وهو ما يحد من قدرتك على برمجة هذه الأداة بشكل كبير، ولعله السبب الرئيسي الذي حدا ميكروسوفت إلى إنشاء جدول العرض!

-١٧-

مُـكـــرِّر البيـــانـــات

Data Repeater

تمنحك هذه الأداة القدرة على عرض البيانات في صورة قائمة List من العناصر بالتنســيق الذي تريده.. وتختلف هذه الأداة عن القوائم التقليدية في أنها لا تعرض العنصر على شــكل نص أو صورة، بل تتيح لك تصميم كل عنصر بأي عدد من الأدوات كما تريد، وبأي شكل تريد، كما تبين الصورة التالية:



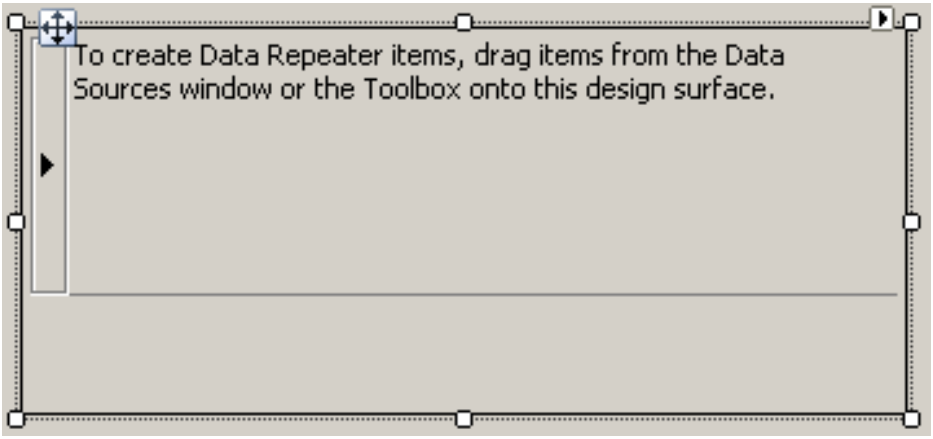
لو تأملت الصورة السابقة، فسيتضح لك أن القائمة التي نتحدث عنها تعرض سجلات جدول المؤلفين، حيث يتم عرض كل مؤلف في ٥ مربعات نصوص مع وجود اللافتات اللازمة التي تشرح وظيفة كل مربع نص.. هذا معناه أننا استخدمنا ١٠ أدوات لتصميم طريقة عرض كل عنصر في القائمة.

لكن.. هل نحن مضطرون إلى تصميم كل العناصر بأنفسنا؟

بالطبع لا، وإلا كان الأمر مستحيلا.. في الحقيقة نحن نصمم عنصرا واحدا فقط في القائمة، ومن ثم يعمل هذا العنصر كقالب Template تنسخ باقي العناصر منه.. لهذا تسمى هذه الأداة بمكرر البيانات DataRepeater، وهي ترث الفئة ContainerControl، لهذا فهي تعمل كأداة حاوية.. هذا هو ما يتيح لنا وضع أدوات أخرى عليها لتصميم عناصر القائمة.

دعنا ونرى كيف نستخدم هذه الأداة لعرض بيانات المؤلفين:

- افتح مشروعا جديدا اسمه Repeater (ستجده ضمن مشاريع هذا الكتاب).
- أضف مصدر بيانات إلى المشروع يحتوي على الجداول Authors و Books و Countries.
- افتح مخطط مجموعة البيانات، وأضف إلى جدول المؤلفين عمودا جديدا اسمه Country، وضع في الخاصية Expression النص: Parent.Name، لتجعل هذا العمود يعرض أسماء الدول التي ينتمي إليها المؤلفون، بدلا من أن نعرض للمستخدم أرقام الدول.
- انتقل إلى النموذج، وافتح صندوق الأدوات، وأسدل عناصر الشريط Visual Basic PowerPacks، واسحب الأداة DataRepeater وأسقطها على النموذج.
- استخدم مقابض الأداة لمنحها الشكل الملائم.. ولو أردت تغيير موضع الأداة فعليك سحبها من علامة الأسهم الموجودة على الركن العلوي الأيسر، وهي لا تظهر إلا إذا ضغطت إطار الأداة بالفأرة، تماما كما تفعل مع أي أداة حاوية.



- كما تلاحظ في الصورة، ينقسم سطح الأداة إلى جزئين:

١- قالب العنصر Item Template:

وهو الجزء العلوي، الذي يوجد سهم أسود على يساره.. ولو ضغطت هذا الجزء فسيتم تحديد إطاره، وسيمكنك تغيير حجمه باستخدام المقابض.. ويمكنك وضع الأدوات من صندوق الأدوات في هذا الجزء، كما يمكنك سحب العناصر من نافذة مصادر البيانات Data Sources وإلقائها عليه مباشرة لإنتاج أدوات مرتبطة بالبيانات.. لاحظ أن مكرر البيانات يشترط وجود أداة واحدة على الأقل مرتبطة بالبيانات، وغير هذا، تستطيع وضع أية أدوات أخرى تريدها، كمربع صورة يعرض صور رمزية، أو زرا ينفذ وظيفة معينة، أو لافتات تشرح وظائف مربعات النصوص.. ويقوم مكرر البيانات بعمل نسخ من هذا القالب، لعرض كل عنصر في مصدر البيانات.

٢- حاوية العرض Viewport:

هذا هو الجزء السفلي الفارغ من الأداة، وعند ضغطه يتم تحديد الأداة كلها.. ولا تستطيع إضافة أية أدوات إلى هذا الجزء، ووظيفته الوحيدة هي تحديد مساحة عرض الأداة على النموذج.. لهذا يمكنك سحب حواف هذا الجزء لضبط المسافات بينها وبين حواف النموذج.. ولا تقلق من صغر مساحة مكرر البيانات، فهو يعرض منزلقا رأسيا إذا احتاج إلى ذلك، ليستطيع المستخدم عرض العناصر

غير الظاهرة.. والأفضل أن تستخدم الخاصية Anchor من نافذة الخصائص لتثبيت حواف الأداة بالنسبة لحواف النموذج، بحيث يتم تكبير أو تصغير مكرر البيانات إذا تم تكبير أو تصغير النموذج.

- افتح نافذة مصادر البيانات Data Sources Window، واضغط زر إسدال طريقة العرض المجاور للعنصر Authors واختر Details من القائمة المنسدلة.. واجعل الحقل Country يعرض بياناته في قائمة مركبة ComboBox، ثم اسحب جدول المؤلفين وألقه على قالب العنصر في مكرر البيانات.. سيؤدي هذا إلى إضافة الأدوات المناسبة إلى مكرر البيانات والنموذج.. احذف مربع النص واللافتة المرتبطتين بالحقل CountryID، ونسق شكل الأدوات كما تريد.
- أضف زرا إلى قالب العنصر، لنستخدمه لعرض كتب المؤلف.

لاحظ أن مكرر البيانات يسبب مشاكل إذا حاولت تصمم عنصر يعرض بيانات مترابطة.. مثلا: لو عرضت كتب المؤلف الحالي في جدول عرض، فسيعرض جدول العرض كتب أول مؤلف في سجلات باقي المؤلفين!!.. وهو نفس ما سيحدث لو حاولت عرض الكتب في قائمة أو قائمة مركبة!!.. بل إنك لو غيرت العنصر المحدد في القائمة المركبة في أحد السجلات، فسيتم تغييره في كل القوائم المركبة الموجودة في باقي السجلات!!.. بينما لو كانت القائمة المركبة تعرض عناصر عادية (مضافة إلى المجموعة Items دون الارتباط بمصدر بيانات)، فستعمل كل نسخة من القائمة بشكل مستقل وصحيح!.. لهذا علينا أن نكتب بعض الكود لنملأ قائمة الدول.. أضف هذا الكود إلى حدث تحميل النموذج:

```
foreach (BooksDataSet.CountriesRow R in
        BooksDataSet.Countries.Rows)
    CountryComboBox.Items.Add(R.Name);
```

لاحظ أنك لو أضفت هذا الكود بعد ملء جدول المؤلفين بالبيانات، فلن تعرض القائمة المركبة أية عناصر، رغم أن العناصر موجودة فيها فعلا!!

السبب في هذا أن ملء جدول المؤلفين بالبيانات يجعل الأدوات المرتبطة به تتلقى البيانات منه، وهذا سيجعل مكرر البيانات يعرض جميع سجلات المؤلفين، وهذا معناه أنه أنشأ نسخا من القائمة المركبة الفارغة من العناصر وعرضها.. لهذا لا يفيدك ملء القائمة الأصلية بعد

هذا، فهي ليست مرتبطة فعليا بالنسخ المعروضة للمستخدم.. هي فقط مجرد قالب Template يتم عمل نسخ منه.. لهذا يجب أن تملأ هذا القالب بالبيانات أولا وتضبط خصائص شكله ولون خطه وطريقة عرضه، قبل أن يتم عمل نسخ منه.. هذا معناه أن أفضل مكان لوضع الكود السابق هو بعد جملة ملء جدول الدول وقبل جملة ملء جدول المؤلفين!

وهناك حل آخر لهذه المشكلة، هو استخدام الوسيلتين BeginResetItemTemplate و EndResetItemTemplate كما سنرى لاحقا.

ولا تنسَ أن تستخدم نافذة الخصائص لتجعل قائمة الدول تعمل كقائمة منسدلة، وذلك بوضع القيمة DropDownList في الخاصية DropDownStyle.. هذا سيمنع المستخدم من الكتابة في مربع نص القائمة المركبة، حتى لا يكتب اسم دولة خاطئ، وبدلا من هذا سيختار الدولة التي يريدها من القائمة.. لاحظ أن الخاصية Text الخاصة بالقائمة المركبة مرتبطة بالحقل Author.Country بسبب سحبها من نافذة مصادر البيانات.. سنترك هذا كما هو، ولن يحدث خطأ، فعندما يوضع في الخاصية Text نص موجود فعلا في القائمة، فإن القائمة تحدد هذا العنصر، وهو ما سيجعل البرنامج يعمل بشكل صحيح.

أما إذا أردت عرض كتب كل مؤلف، فأفضل حل هو استخدام زر يؤدي ضغطه إلى عرض نموذج جديد عليه كتب المؤلف الحالي.. وعموما هذه هي الطريقة الأكفأ، فليس من الذكي عرض كما ضخما من البيانات في مكرر البيانات، لأنها ستلتهم مساحة عرض كبيرة وتستهلك مساحة كبيرة في الذاكرة!

لو شغلت البرنامج الآن، فسيعرض بيانات كل مؤلفين في أدوات العرض التي صممناها.. وسنرى ونحن نتعرف على خصائص ووسائل مكرر البيانات كيف نكمل وظائف هذا البرنامج.

فئة مكرر البيانات DataRepeater Class

هذه الفئة موجودة في النطاق Microsoft.VisualBasic.PowerPacks، وهي ترث الفئة ContainerControl.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الأداة الخصائص التالية:

السماح للمستخدم بإضافة عناصر AllowUserToAddItems:

إذا جعلت قيمتها true (وهي القيمة الافتراضية)، فسيتمكن المستخدم من إضافة سجل جديد إلى مكرر البيانات، وذلك بضغط زر الإضافة الموجود على شريط موجه الربط، أو بتحديد أي سجل في مكرر البيانات (بضغط الهامش الأيسر للسجل، حيث سيظهر فيه سهم يدل على أنه محدد)، وضغط CTRL+N من لوحة المفاتيح.

ويعرض السجل الجديد القيم الافتراضية للحقول، وإذا لم تكن للحقل قيمة افتراضية، فستعرض الأدوات قيم أول أو آخر سجل في الجدول.. طبعا هذا غير مرغوب، وعليك التأكد من إفراغ الحقول من هذه القيم، كما سنرى لاحقا.. لاحظ أن السجل الجديد يتم حذفه إذا غادره المستخدم دون أن يكتب فيه أية بيانات.

أما إذا جعلت قيمة هذه الخاصية false، فلن يمكن للمستخدم إضافة سجل جديد بضغط CTRL+N من لوحة المفاتيح، لكن سيظل زر إضافة سجل جديد الموجود على موجه الربط فعالا، وسيكون عليك تعطيله بنفسك.

السماح للمستخدم بحذف العناصر AllowUserToDeleteItems:

إذا جعلت قيمتها true (وهي القيمة الافتراضية)، فسيتمكن المستخدم من حذف السجل المحدد حاليا في مكرر البيانات، بضغط زر الحذف الموجود على شريط موجه الربط، أو ضغط الزر DELETE من لوحة المفاتيح.

عدد العناصر ItemCount:

تعيد عدد السجلات المعروضة حاليا في مكرر البيانات.

ويمكنك أن تضع في هذه الخاصية عدد العناصر التي تريد عرضها عند استخدام مكرر البيانات في الوضع الافتراضي Virtual Mode كما سنرى لاحقا.. لكن محاولة وضع أي قيمة في هذه الخاصية في الوضع العادي ستؤدي إلى حدوث خطأ في البرنامج.

قالب العنصر ItemTemplate:

تعيد كائنا من النوع DataRepeaterItem، يمثل العنصر المستخدم كقالب في مكرر البيانات.. وسنتعرف على الفئة DataRepeaterItem بعد قليل.

ويمكنك استخدام هذه الخاصية لتغيير خصائص عناصر مكرر البيانات.. لاحظ أنك تستطيع فعل هذا في وقت التصميم، وذلك بضغط قالب العنصر بالفأرة لتحديده، ثم ضغط F4 لعرض خصائصه في نافذة الخصائص.. هذا يتيح لك تغيير الخط ولون الخلفية والعديد من الخصائص الأخرى التي تؤثر على المساحة التي تعرض السجلات في مكرر البيانات.. بينما لو ضغطت جزء العرض Viewport فستظهر خصائص مكرر البيانات نفسه في نافذة الخصائص.

العنصر الحالي CurrentItem:

تعيد كائنا من النوع DataRepeaterItem، يمثل العنصر المحدد حاليا في مكرر البيانات.. ويمكنك أيضا أن تضع في هذه الخاصية، كائن العنصر الذي تريد تحديده.. ولا توجد طريقة اتحديد أكثر من عنصر في نفس الوقت.

لاحظ أن مكرر البيانات لا يمتلك الخاصية الافتراضية Items.. السبب في هذا أن مكرر البيانات هو أداة حاوية، لهذا تستطيع أن تتعامل مع عناصره من خلال الخاصية الموروثة Controls، التي تستطيع أن ترسل إليها رقم العنصر لتعيد إليك الكائن الذي يمثله.. مثال:

```
var Itm = (DataRepeaterItem) DataRepeater1.Controls[0];
```

والمثال التالي يتيح لك المرور عبر عناصر مكرر البيانات:

```
foreach (DataRepeaterItem Itm in DataRepeater1.Controls)
    MessageBox.Show(Itm.ItemIndex.ToString( ));
```

ولا تنسَ استخدام جملة الاستخدام التالية أعلى صفحة الكود قبل تجربة المثال:

```
using Microsoft.VisualBasic.PowerPacks;
```

لكني لا أنصحك باستخدام هذه الطريقة، لأنها ستمر على بعض عناصر مكرر البيانات فقط وبترتيب عشوائي!!.. السبب في هذا أن مكرر البيانات يعرض فقط العناصر الظاهرة للمستخدم على الشاشة، ولا يعرض باقي العناصر إلا إذا حرك المستخدم المنزلق الرأسي.. لذا إذا أردت إجراء أي تغيير على العناصر، فاستخدم الحدث DrawItem لفعل هذا، فهو ينطلق قبل عرض كل عنصر.

رقم العنصر الحالي CurrentItemIndex:

تعيد رقم السجل المحدد حاليا في مكرر البيانات.. ويمكنك إرسال رقم أي سجل ليتم تحديده.. والمثال التالي يحدد السجل الثاني في الأداة:

```
DataRepeater1.CurrentItemIndex = 1;
```

عدد العناصر المعروضة DisplayedItemCount:

تعيد عدد السجلات الظاهرة للمستخدم حاليا في مكرر البيانات بدون تحريك المنزلق الرأسي.. ولهذه الخاصية معامل منطقي، إذا جعلته true فسيدخل ضمن الحساب السجلات التي تظهر أجزاء منها فقط:

```
MessageBox.Show(DataRepeater1.DisplayedItemCount(
        true).ToString( ));
```

أما إن جعلته false، فسيتم حساب عدد السجلات الظاهرة بصورة كاملة:

```
MessageBox.Show(DataRepeater1.DisplayedItemCount(
        false).ToString( ))
```

رقم أول عنصر معروض FirstDisplayedItemIndex:

تعيد رقم أو سجل ظاهر للمستخدم في مكرر البيانات.

رأس العنصر مرئي ItemHeaderVisible:

إذا جعلت قيمتها false، فسيتم إخفاء الهاش الأيسر الذي يعرض رءوس العناصـــر.. والقيمة الافتراضية true.

حجم رأس العنصر ItemHeaderSize:

تقرأ أو تغير عرض الهامش الأيسر الذي يعرض رءوس العناصر.

لون التحديد SelectionColor:

تقرأ أو تغير لون الخلفية الذي يعرض في خانة رأس السجل المحدد حاليا.

طراز المخطط LayoutStyle:

تقرأ أو تغير طريقة عرض مكرر البيانـــات، وهـــي تأخـــذ إحـــدى قيمتـــي المـــرقم DataRepeaterLayoutStyles التاليتين:

Vertical	يتم تكرار العناصر رأسيا (من أعلى إلى أسفل) في شكل صـــفوف.. هذا هو الوضع الافتراضي.
Horizontal	يتم تكرار العناصر أفقيا (من اليسار إلى اليمين) في شـــكل أعمـــدة، ويظهر هامش علوي يحمل رءوس هذه الأعمدة.. ويمكنك رؤية هذا في المشروع RepeaterItemColor.

وتمتلك هذه الأداة الوسائل التالية:

إضافة جديد AddNew:

تضيف سجلا إلى نهاية مكرر البيانات.. وتسبب هذه الوسيلة خطأ إذا كانت للخاصـــية AllowUserToAddItems القيمة false.

حذف من موضع RemoveAt:

أرسل إلى هذه الوسيلة رقم السجل الذي تريد حذفه من مكرر البيانات.

إلغاء التحرير CancelEdit:

تلغي البيانات التي أدخلها المستخدم في السجل الحالي، وتعيد وضع القيم الأصلية في الأدوات.. هذا مفيد إذا أردت أن تمنح المستخدم القدرة على ضغط الزر Esc من لوحة المفاتيح لإلغاء التغييرات التي أجراها في السجل الحالي.. في هذه الحالة عليك أن تكتب إجراء يستجيب للحدث KeyDown لجميع الأدوات التي تعرض بيانات السجل، وتكب فيه الكود الذي يستدعي هذه الوسيلة إن كان الزر المضغوط هو الزر Esc.. وستجد الكود التالي في الإجراء UserCancelsEdit في المشروع Repaeter، مع ملاحظة أن هذا الإجراء يستجيب للحدث KeyDown لكل مربعات النص والقائمة المركبة أيضا:

```
if (e.KeyCode == Keys.Escape)
   DataRepeater1.CancelEdit( );
```

لاحظ أن التغييرات التي يدخلها المستخدم في أي أداة في السجل الحالي، يتم قبولها بمجرد مغادرة الأداة إلى أية أداة أخرى، في نفس السجل أو في سجل آخر.. هذا معناه أن ضغط الزر ESC سيلغي التغييرات التي حدثت في الأداة الحالية فقط ولن يؤثر على أية أداة أخرى.. ولو غادر المستخدم الأداة التي أجرى فيها التغييرات، ثم عاد إليها وضغط ESC فلن يحدث شيء!

تحريك العنصر إلى مجال الرؤية ScrollItemIntoView:

أرسل إلى هذه الوسيلة رقم السجل الذي تريد تحريك المنزلق إليه ليصير مرئيا للمستخدم.

وتوجد صيغة أخرى، تستقبل معاملا ثانيا، إذا جعلته true، فسيتم تحريك المنزلق بحيث يصير السجل هو أول سجل معروض في مكرر البيانات، مع محاذاة الحافة العلوية للسجل بالحافة العلوية لمكرر البيانات.

بدء تغيير قالب العنصر BeginResetItemTemplate:

كما أشرنا من قبل: أي تغيير تجريه على خصائص الأدوات الداخلة في تكوين قالب العنصر بعد عرض عناصر مكرر البيانات يكون بلا تأثير.. لهذا لو أردت تغيير خصائص أية أداة، أو أردت إجراء تعديلات على القالب نفسه بإضافة أو حذف أدوات من خلال الخاصية ItemTemplate، فعليك أولا أن تستدعي الوسيلة BeginResetItemTemplate لتنبيه مكرر البيانات إلى أن هناك تغييرات ستحدث في طريقة العرض.

إنهاء تغيير قالب العنصر EndResetItemTemplate:

استدع هذه الوسيلة في نهاية الكود الذي يجري تعديلات في قالب العنصر، لإجبار مكرر البيانات على إنعاش العناصر التي يعرضها لتظهر عليها التغييرات التي حدثت.. والكود التالي يغير لون خلفية القائمة إلى الأصفر، ويمكنك تجربته بضغط الزر "تغير لون الخلفية" في المشروع Repeater:

```
DataRepeater1.BeginResetItemTemplate( );
CountryComboBox.BackColor = Color.Yellow;
DataRepeater1.EndResetItemTemplate( );
```

جرب وضع علامة التعليق // أمام السطرين الأول والأخير في الكود السابق واضغط الزر.. ستجد أن لون القائمة لن يتغير.

كما تمتلك هذه الفئة الأحداث التالية:

تغير رقم العنصر الحالي CurrentItemIndexChanged:

ينطلق عندما تتغير قيمة الخاصية CurrentItemIndex من الكود، أو بسبب انتقال المستخدم من سجل إلى آخر في مكرر البيانات.

خطأ البيانات DataError:

ينطلق عند حدوث خطأ في قراءة البيانات من مصدر البيانات، أو في نقل البيانات المحدثة من مكرر البيانات إليه.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterDataErrorEventArgs، وله الخصائص التالية:

DataRepeaterItem	تعيد عنصر مكرر البيانات DataRepeaterItem الذي يمثل السجل الذي حدث فيه الخطأ.	
Control	تعيد الأداة التي حدث فيها الخطأ.	
PropertyName	تعيد اسم خاصية الأداة، التي سببت الخطأ.. بمعنى آخر: تعيد عنصر العرض.	
Exception	تعيد كائن الاستثناء Exception الذي يحمل معلومات الخطأ.	
ThrowException	إذا جعلت قيمة هذه الخاصية true، فسيحدث الخطأ في البرنامج بعد انتهاء هذا الحدث.. والقيمة الافتراضية هي false.	

أضاف المستخدم عناصر UserAddedItems:

ينطلق بعد أن يضغط المستخدم CTRL+N، وقبل أن يضاف العنصر الجديد إلى مكرر البيانات.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsEventArgs، وله الخاصيتان التاليتان:

ItemIndex	تعيد رقم العنصر الجديد.
ItemCount	تعيد عدد العناصر التي تمت إضافتها.

ويعتبر هذا الحدث أنسب مكان لإفراغ خانات السجل من أية قيم غير مرغوبة، فكما ذكرنا سابقا، يعرض مكرر البيانات قيم السجل الأول أو الأخير في السجل الجديد،

لهذا يمكنك أن تمحوها، أو تضع بدلا منها القيم الابتدائية المناسبة.. وقد استخدمنا هذا الحدث لفعل هذا في المشروعين Repeater و RepeaterItemColor.

يجري حذف عناصر DeletingItems:

ينطلق عند حذف سجل من مكرر البيانات، سواء من الكود أو بواسطة المستخدم.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsCancelEventArgs، وهو مماثل لمعامل الحدث السابق، إلا أنه يزيد عنه بامتلاك الخاصية Cancel، وإذا وضعت فيها true يتم إلغاء حذف السجل.. لهذا يعتبر هذا الحدث ملائما لتعرض رسالة للمستخدم ليؤكد رغبته في حذف السجل:

```
if (MessageBox.Show("هل تريد حذف هذا السجل فعلا؟",
    "تأكيد الحذف", MessageBoxButtons.OKCancel  ) ==
    DialogResult.Cancel )
  e.Cancel = true;
```

لاحظ أنك ضغط زر الحذف الموجود على شريط موجه الربط سيحذف العنصر من مصدر البيانات مباشرة، ولن تظهر رسالة التحذير.. لو أردت تغيير هذا الأداء، فضع في الخاصية DeleteItem الخاصة بموجه الربط القيمة null، واكتب ما يلي في حدث ضغط زر الحذف:

```
int I = DataRepeater1.CurrentItem.ItemIndex;
DataRepeater1.RemoveAt(I);
```

وستجد هذا الكود في المشروع Repeater.

المستخدم يحذف عناصر UserDeletingItems:

مماثل للحدث السابق في كل شيء، ما عدا أنه ينطلق فقط عندما يضغط المستخدم الزر Delete لحذف السجل المحدد في مكرر البيانات، ولا ينطلق بسبب حذف السجل من الكود.

المستخدم حذف عناصر UserDeletedItems:

ينطلق بعد أن يحذف المستخدم سجلا من مكرر البيانات، ولا ينطلق لو تم الحذف من الكود.. والمعامل الثاني e من النوع DataRepeaterAddRemoveItemsEventArgs، وقد تعرفنا عليه.. ولا تستطيع إلغاء الحذف فلا توجد الخاصية e.Cancel.

يجري نسخ العنصر ItemCloning:

ينطلق قبيل عمل نسخة من قالب العنصر.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterItemCloneEventArgs، وهو يمتلك الخصائص التالية:

	Source	تعيد عنصر مكرر البيانات DataRepeaterItem الذي سيتم نسخه.
	Target	تقرأ أو تغير عنصر مكرر البيانات DataRepeaterItem الناتج من عملية النسخ.. هذا يتيح لك التحكم في عملية النسخ كما تريد، فالكائن الذي تضعه في هذه الخاصية يكون هو ناتج النسخ.
	Handled	اجعل قيمتها true، لمنع مكرر البيانات من أداء عملية النسخ الخاصة به.. وعليك أن تأخذ نسخة من الكائن الموضح في الخاصية Source، وتجري عليها التعديلات التي تريدها، ثم تضعها في الخاصية Target.

تم نسخ العنصر ItemCloned:

ينطلق بعد نسخ عنصر من قالب العناصر.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterItemEventArgs، وهو يمتلك الخاصية DataRepeaterItem التي تعيد العنصر المنسوخ.. لاحظ أن هذا العنصر لم يعرض بعد في مكرر البيانات، لهذا لا تحاول استخدام رقمه في أي عملية، فسيكون صفرا دائما!.. كما أن الأدوات الموجودة على العنصر ما زالت فارغة ولم ترتبط بمصدر البيانات بعد، لهذا لا تحاول قراءة قيمها.. كل ما يمكنك فعله هو تغيير خصائص هذه الأدوات بالطريقة التي تناسبك، كأن تملأ قائمة بمجموعة من العناصر مثلا.

رسم عنصر DrawItem:

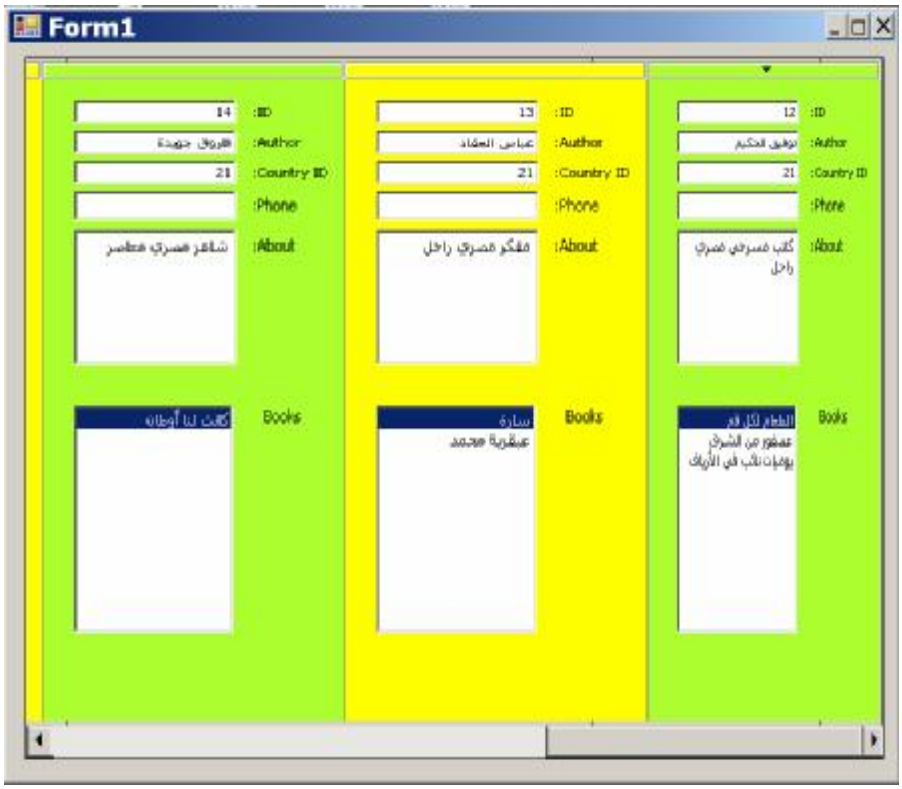
ينطلق عند رسم عنصر في مكرر البيانات.. لاحظ أن رسم العنصر يتكـــرر مـــرات عديدة، حيث يعاد رسم العنصر كلما ظهر في مساحة العرض مع حركة المنزلق.

والمعامل الثاني e لهذا الحدث من النوع DataRepeaterItemEventArgs كمـــا فـــي الحدث السابق.

ويعتبر هذا الحدث أفضل حدث يمكنك استخدامه للتحكم في العناصر المعروضة، فهو الحدث الوحيد الذي ينطلق بعد إضافة العنصر إلى مكرر البيانات فعلا وبعد إتمام ربط أدواته بالبيانات.. لهذا استخدمنا هذا الحدث فـــي المشـــروع RepeaterItemColor لأداء الوظيفتين التاليتين:

١- إذا كان العنصر زوجيا نلونه بالأخضر، وإذا كان فرديا نلونه بالأصفر:

```
if (e.DataRepeaterItem.ItemIndex % 2 == 0)
   e.DataRepeaterItem.BackColor = Color.Green;
else
   e.DataRepeaterItem.BackColor = Color.Yellow;
```



٢- نقرأ رقم المؤلف المعروض في مربع النص، ونستخدمه لنحصل على كائن عرض DataView يحتوي على كتبه، ونجعله مصدر البيانات للقائمة لكي تعرض كتب المؤلف الحالي.. لاحظ أن هذه هي الطريقة الصحيحة الوحيدة لربط القائمة بمصدر البيانات، حيث يجب أن نربط كل نسخة من القائمة بمصدر بياناتها على حدة، وقد رأينا من قبل كيف تفشل محاولة ربط القائمة الموضوعة على قالب العنصر بالبيانات.. لكن عيب هذه الطريقة هو أنك مضطر إلى إعادة ربط القائمة بمصدر البيانات في كل مرة يتم فيها رسم العنصر.. ولو جربت الكود التالي، فسيؤدي إلى نتائج خاطئة، وستعرض بعض القوائم كتب مؤلفين آخرين:

```
if (BksLst.DataSource == null)
{
   BksLst.DataSource = BooksView;
   BksLst.DisplayMember = "Book";
}
```

السبب في هذا أن الشرط سيكون صحيحا مرة واحدة فقط عند رسم القائمة لأول مرة، لكن بعد هذا كلما تحرك المنزلق وأعيد رسم العنصر، فسيكون الشرط خاطئا، ولن يتم ربط القائمة بمصدر البيانات، مما سيجعلها تعرض نتائج خاطئة.. لست أعرف يقينا سبب هذا، ولكني أخمن أن مصممي مكرر البيانات يحسّنون أداءه بتحريك القوائم من العناصر التي اختفت مع حركة المنزلق، لعرضها على العناصر التي ظهرت على الشاشة!.. لهذا لو لم تقم بتحديث محتويات كل قائمة بنفسك عند رسم العنصر، فإنها تظل تحتفظ بنتائج تخص سجلات أخرى!

لاحظ أن خطأ سيحدث في الكود الذي كتبناه عند رسم العنصر الجديد، لأنه غير مرتبط بعد بصف في مجموعة البيانات.. لهذا علينا إضافة شرط لإنهاء الكود إذا كان العنصر جديدا.. يمكننا أن نعرف هذا إذا كان مربع النص IDTextBox يحمل رقما سالبا لأنه لم يأخذ رقما تلقائيا بعد:

```
int AuthorID = int.Parse(Itm.Controls["IDTextBox"].Text);
if (AuthorID < 0)
   return;
```

ولإكمال وظيفة البرنامج، سمحنا للمستخدم بالنقر المزدوج بالفأرة على القائمة، واستخدمنا الحدث DoubleClick الخاص بها لنعرض تفاصيل كتب المؤلف الحالي في جدول عرض على نموذج مستقل.. في الحقيقة هذا الكود في منتهى البساطة، فكل ما نفعله فيه هو جعل مصدر بيانات جدول العرض، هو نفسه مصدر بيانات القائمة:

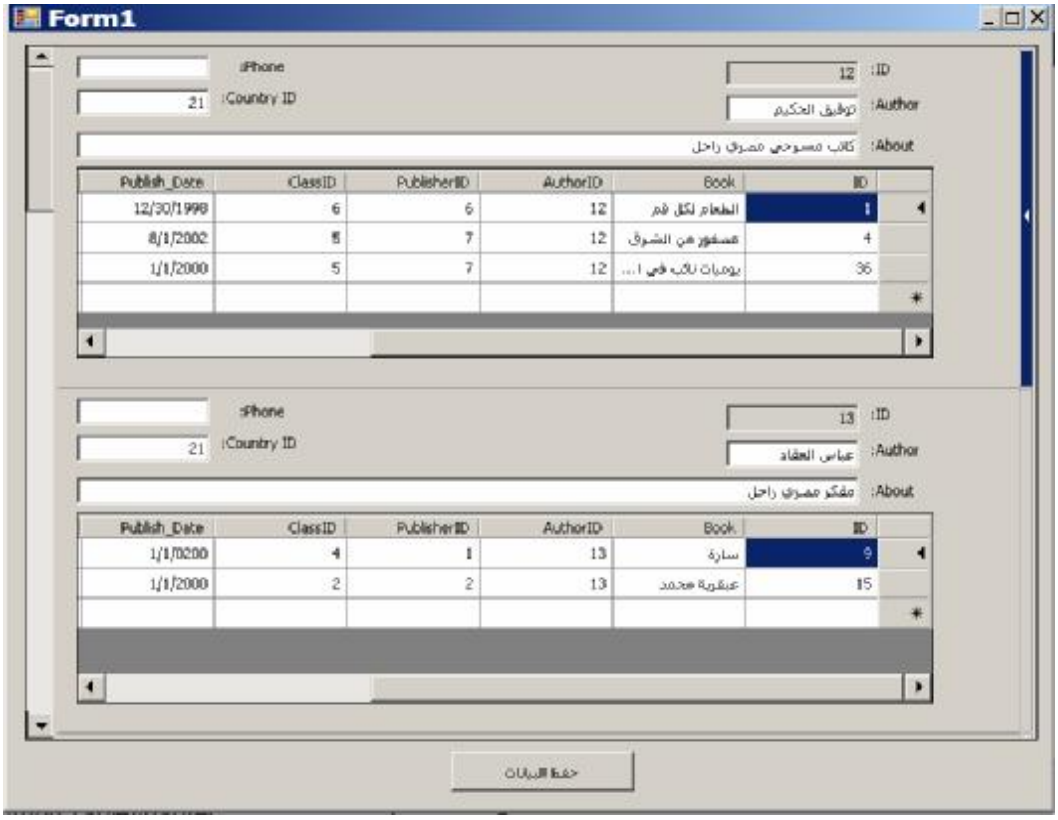
```
FrmBooks.GrdBooks.DataSource = BksLst.DataSource;
```

النقطة الوحيدة الهامة هنا، هي أننا لا نستخدم القائمة BooksList الموضوعة على قالب العنصر، وإنما نستخدم نسخة القائمة الخاصة بالعنصر الحالي في مكرر البيانات.. وسنعرف لاحقا كيف نحصل على هذه النسخة.

استخدام مكرر البيانات في الوضع الافتراضي Virtual Mode:

رأينا من قبل كيف نستخدم قائمة العرض ListView وجدول العرض DataGridView في الوضع الافتراضي.. وبالمثل يمكننا استخدام مكرر البيانات في الوضع الافتراضي.. هذا مفيد إذا كنت تولد البيانات بناء على معادلة دون الحاجة إلى مصدر بيانات، أو إذا كان حجم البيانات ضخما، وتريد التدخل في طريقة عرضها لتحسين أداء البرنامج.

ويريك المشروع VirtualRepeater كيف يمكن عرض بيانات المؤلفين في مكرر البيانات بطريقة افتراضية، مع عرض كتب كل مؤلف في جدول عرض في نفس السجل.. في هذه الحالة يحتفظ مكرر البيانات في الذاكرة ببيانات المؤلفين الظاهرين على الشاشة فقط، وكلما تحرك المستخدم بالمنزلق سيطلب منا مكرر البيانات إمداده ببيانات المؤلفين المراد عرضهم.



دعنا نتعرف على الخصائص والوسائل والأحداث التي يمنحها لنا مكرر البيانات للتعامل مع الوضع الافتراضي، لنرى كيف نستخدمها في كتابة هذا المشروع:

الوضع الافتراضي VirtualMode:

إذا جعلت قيمة هذه الخاصية true، فسيعمل مكرر البيانات في الوضع الافتراضي.. والقيمة الابتدائية لهذه الخاصية هي false.. وقد استخدمنا نافذة الخصائص في المشروع VirtualRepeater لجعل قيمة هذه الخاصة true.. ونظرا لأن الخاصية ItemCount لا تظهر في نافذة الخصائص، فقد استخدمنا حدث تحميل النموذج لنضع فيها عدد المؤلفين المراد عرضهم:

```
DataRepeater1.ItemCount = BooksDataSet.Authors.Count;
```

لو شغلت المشروع الآن، فسترى عناصر بعدد المؤلفين معروضة في مكرر البيانات.. ورغم أن هذه العناصر ستعرض الأدوات التي وضعتها على قالب العنصر في وقت التصميم، فستكون فارغة، لأن تقنية الربط Binding لا تعمل في الوضع الافتراضي للأسف!!.. لهذا عليك كتابة الكود الذي يعرض البيانات في هذه الأدوات بنفسك، كما سنرى بعد قليل.

قيمة العنصر مطلوبة ItemValueNeeded:

ينطلق هذا الحدث عندما تحتاج أداة موجودة في أحد السجلات إلى عرض قيمتها.. هذا يشمل اللافتات ومربعات النصوص، لهذا عليك أن تتحقق من الأداة قبل أن تضع فيها القيمة.. ويعتبر هذا الحدث المكان الملائم لعرض البيانات في الأدوات في الوضع الافتراضي.. والمعامل الثاني e لهذه الحدث من النوع DataRepeaterItemValueEventArgs، وله الخصائص التالية:

ItemIndex	تعيد رقم العنصر في مكرر البيانات.	
Control	تعيد الأداة التي تحتاج إلى عرض البيانات.	
Property Name	تعيد اسم خاصية الأداة التي ستعرض البيانات (عنصر العرض).	

	Value	ضع في هذه الخاصية القيمة التي تريد عرضها في الأداة.. لاحظ أن هذه الخاصية حساسة جدا لنوع البيانات، لهذا عليك إجراء عمليات التحويل المناسبة قبل وضع القيمة فيها.. مثلا: لو وضعت الرقم ١ في هذه الخاصية لعرضه في مربع النص IDTextBox الذي يعرض رقم المؤلف، فلن يظهر في مربع النص أي شيء!.. بينما لو وضعت النص "١" في هذه الخاصية فسيظهر في مربع النص!.. السبب في هذا أن الخاصية Text تقبل نصوصا لا أعدادا صحيحة، والخاصية e.Value لا تقوم بالتحويل المطلوب!.. لهذا عليك استخدام الوسيلة ToString لتحويل الحقول الرقمية إلى نصوص قبل وضعها في هذه الخاصية.

وقد استخدمنا هذا الحدث في المشروع VirtualRepeater لعرض القيم في الأدوات.. هذا الكود بسيط للغاية، فهو يستخدم الجملة الشرطية Select ليفحص اسم كل أداة، ويضع فيها القيمة المناسبة.. ولا تحتاج قراءة القيم من جدول المؤلفين إلى كود معقد، فرقم العنصر في مكرر البيانات، هو نفسه رقم السجل في جدول المؤلفين.. سيكون هذا الكود على الصورة التالية:

```
var Authors = BooksDataSet.Authors;
switch (e.Control.Name)
{
   case "IDTextBox":
      e.Value = Authors[e.ItemIndex].ID.ToString( );
      break;
   case "AuthorTextBox":
      e.Value = Authors[e.ItemIndex].Author;
      break;
}
```

ونظرا لأن بعض الحقول قد تسبب مشاكل إذا كانت فارغة DbNull، لذا عليك استخدام المقطع Try Catch للاحتراز.. وستجد هذا الكود كاملا في المشروع VirtualRepeater.

لاحظ أن جدول العرض لا يطلق الحدث ItemValueNeeded، لهذا عليك استخدام الحدث DrawItem لربط جدول العرض بكتب المؤلف.. كل ما سنفعله، هو الحصول على كائن عرض View Object يحتوي على كتب المؤلف الحالي، ووضعه كمصدر بيانات لجدول العرض:

```
var Itm = e.DataRepeaterItem;
var Authors = BooksDataSet.Authors;
// الحصول على كائن عرض الصف الخاص بالمؤلف الحالي
var Rv = Authors.DefaultView[Itm.ItemIndex];
var Rl = BooksDataSet.Authors.ChildRelations[0];
// الحصول على نسخة جدول العرض الحالية
var GrdBooks = (DataGridView)
        Itm.Controls["BooksDataGridView"];
// الحصول على كائن عرض كتب المؤلف الحالي من خلال العلاقة
// واستخدامه كمصدر بيانات لجدول العرض
GrdBooks.DataSource = Rv.CreateChildView(Rl);
```

عنصر جديد مطلوب NewItemNeeded:

ينطلق هذا الحدث عندما يطلب المستخدم إضافة سجل جديد إلى مكرر البيانات بضغط CTRL+N.. هذا يتيح لك إضافة سجل جديد إلى مصدر البيانات، حتى يمكن حفظ البيانات التي يدخلها المستخدم فيه.. وقد استخدمنا هذا الحدث في المشروع VirtualRepeater لإضافة صف جديد إلى جدول المؤلفين كالتالي:

```
var R = BooksDataSet.Authors.NewAuthorsRow( );
R.Author = " ";
R.CountryID = 12;
BooksDataSet.Authors.AddAuthorsRow(R);
```

لاحظ أننا وضعنا مسافة في حقل اسم المؤلف، لأن جدول المؤلفين لا يسمح بتركه فارغا، كما وضعنا الرقم ١٢ مبدئيا في حقل رقم الدولة لنفس السبب.. لو لم نفعل هذا، فسيحدث خطأ في البرنامج.. ويمكنك التخلص من المسافة قبل عرضها في مربع النص، باستخدام الوسيلة Trim في الحدث ItemValueNeeded.

إضافة عنصر ItemsAdded:

ينطلق هذا الحدث بعد إضافة السجل الجديد إلى مكرر البيانات، ليمكنك قراءة رقم العنصر الجديد.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsEventArgs الذي تعرفنا عليه سابقا.. لاحظ أن ترتيب استدعاء الأحداث عند إضافة عنصر جديد كالتالي:

- NewItemNeeded.
- ItemValueNeeded.
- DrawItem.
- ItemsAdded.

دفع قيمة العنصر ItemValuePushed:

ينطلق هذا الحدث عندما يغير المستخدم قيمة إحدى الأدوات الموضوعة على السجل الحالي، ثم ينتقل منها إلى أداة أخرى.. هذا يتيح لك كتابة الكود المناسب لحفظ قيمة هذه الأداة في مصدر البيانات.. ولا تنسَ فحص القيمة والتأكد من أنها مناسبة قبل محاولة نقلها إلى مصدر البيانات، كي لا يحدث خطأ.. وسيكون من الجيد أن تمنع الخطأ من المنبع، كالتالي:

١- استخدام الخاصية MaxLength لتحديد أقصى طول لمربعات النصوص التي تستقبل نصوصا.. لقد وضعنا الرقم ٣٠ في هذه الخاصية في مربع النص الذي يستقبل اسم المؤلف.

٢- وضع القيمة true في الخاصية ReadOnly لجعل مربع النص الذي يعرض رقم المؤلف ID للقراءة فقط.

٣- كتابة الكود المناسب في الحدث KeyPress في مربعات النص التي تستقبل أرقاما، لمنع كتابة أية حروف.

٤- استخدام أداة التاريخ والوقت DateTimePicker لاستقبال التاريخ بدلا من مربعات النصوص.. كما يمكنك استخدام عمود مخصص لعرض التواريخ في جدول العرض، بالطريقة التي تعلمناها في الفصل الخاص بجدول العرض.

٥- استخدام قائمة منسدلة لعرض أسماء الدول بدلا من السماح للمستخدم بكتابة رقم الدولة.. سأترك لك فعل هذا بنفسك، فقد فعلناه من قبل.

٦- استخدام مربع نص مقنن MaskedTextBox لاستقبال رقم الهاتف بالصيغة الصحيحة (راجع مرجع برمجة الويندوز).

لاحظ أنك لا تحتاج إلى حفظ التغييرات التي تحدث في سجلات جدول العرض، لأنها تحفظ تلقائيا بسبب ربطه بمصدر البيانات.

والمعامل الثاني e لهذا الحدث من النوع DataRepeaterItemValueEventArgs كما في الحدث ItemValueNeeded.

هل العنصر الحالي قذر IsCurrentItemDirty:

تعيد true إذا كان المستخدم قد أجرى تعديلات على السجل الحالي في مكرر البيانات، دون أن تحفظ بعد في مصدر البيانات.. يحدث هذا إذا غير المستخدم قيمة إحدى الأدوات دون أن يغادرها.

تم حذف العنصر ItemsRemoved:

ينطلق هذا الحدث بعد حذف عنصر من مكرر البيانات، ليتيح لك حذف العنصر المناظر له من مصدر البيانات.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsEventArgs، وقد سبق أن تعرفنا عليه.. وقد استخدمنا هذا الحدث في المشروع VirtualRepeater لحذف المؤلف من جدول المؤلفين كالتالي:

```
BooksDataSet.Authors.Rows.RemoveAt(e.ItemIndex);
```

لاحظ أننا لا نحتاج على حذف كتب المؤلف ولن تحدث أية أخطاء لهذا.. السبب في هذا أننا عرفنا قيد المفتاح الفرعي التالي في حدث تحميل النموذج:

```
var Fkc = new ForeignKeyConstraint(
        BooksDataSet.Authors.IDColumn,
        BooksDataSet.Books.AuthorIDColumn);
Fkc.UpdateRule = Rule.Cascade;
Fkc.AcceptRejectRule = AcceptRejectRule.Cascade;
Fkc.DeleteRule = Rule.Cascade;
BooksDataSet.Books.Constraints.Add(Fkc);
```

كما ترى، فقد عرفنا قاعدة الحذف المتتالي، لحذف كتب المؤلف تلقائيا بمجرد حذف المؤلف نفسه، وهذا يمنع حدوث أية أخطاء، ويوفر علينا كتابة كود الحذف.. كما عرفنا قاعدة التحديث المتتالي أيضا لمنع أية مشكلة عند حفظ بيانات المؤلف الجديد وتغير رقمه التلقائي.. والحقيقة أن المستخدم يجب ألا يدخل الكتب قبل حفظ المستخدم، وإلا فقد يخسرها بسبب عدم قبول قيمة الحقل AuthorID بعد تغير الرقم التلقائي للمؤلف.

فئة عنصر مكرر البيانات DataRepeaterItem Class

هذه الفئة ترث فئة اللوحة Panel، لهذا تستطيع احتواء أدوات أخرى، وهي تعمل كعنصر موضوع على مكرر البيانات، سواء كان العنصر المعروض في وقت التصميم (القالب)، أو العناصر المنسوخة منه في وقت التشغيل.
وإضافة إلى ما ترثه من خصائص الأدوات التقليدية وخصائص الأداة الحاوية وخصائص اللوحة، تمتلك هذه الفئة الخصائص التالية:

هل هو العنصر الحالي IsCurrent:

تعيد true إذا كان هذا العنصر هو العنصر الحالي (المحدد) في مكرر البيانات.

هل هو قذر IsDirty:

تعيد true إذا كان المستخدم قد غير بعض بيانات العنصر ولم تحفظ التغييرات بعد في مصدر البيانات.

رقم العنصر ItemIndex:

تعيد رقم العنصر في مكرر البيانات.

وأهم ما يعنينا هنا، هو كيفية التعامل مع الأدوات الموضوعة على العنصر.. كما ذكرنا من قبل، فإن العنصر هو لوحة Panel، وهذا معناه أنه أداة حاوية، لهذا يمكنك استخدام الخاصية Controls للتعامل مع الأدوات الموضوعة عليه سواء بأرقامها أو بأسمائها.. ويكون التعامل مع الأدوات بأرقامها مناسبا إذا أردت المرور عبر كل الأدوات، بينما يكون التعامل مع الأدوات بأسمائها أكثر ملائمة للكود الذي يقرأ قيم الأدوات أو يغيرها، لأنه يجعل الكود أكثر وضوحا وسهولة.. لاحظ أن الاسم الذي تمنحه للأداة في قالب العنصر في وقت التصميم، هو نفسه الاسم الذي ستستخدمه للتعامل مع نسخة الأداة الموضوعة على

أي عنصر.. السبب في هذا أن كل عنصر هو نسخة طبق الأصل من القالب، وهذا معناه أن كل أداة موضوعة عليه تأخذ نفس خصائص نسختها الأصلية الموضوعة على القالب بما في ذلك الاسم.. وعليك ألا ترتبك بين الاسم الموضوع في الخاصية Name، واسم المتغير الذي يشير إلى الأداة.. مثلا: الكود التالي يغير نص مربع النص الأصلي الموضوع على القالب:

```
AuthorTextBox.Text = "Test";
```

ولن يؤثر هذا الكود على نسخ مربع النص الموضوعة على الأدوات، إلا إذا استخدمته بين الوسيلتين BeginResetItemTemplate و EndResetItemTemplate كما أوضحنا من قبل.

أما إذا أردت تغيير اسم المؤلف الحالي فقط، فيمكنك استخدام الكود التالي:

```
var Itm = DataRepeater1.CurrentItem;
var AutherTxtBx = (TextBox) Itm.Controls["AuthorTextBox"];
AutherTxtBx.Text = "Test";
```

وقد استخدمنا الكود التالي في المشروع RepeaterItemColor لعرض كتب المؤلف في القائمة:

```
var BksLst = (ListBox) Itm.Controls["BooksList"];
BksLst.DataSource = BooksView;
BksLst.DisplayMember = "Book";
```

كما استخدمنا الكود التالي في حدث ضغط زر عرض كتب المؤلف الموضوع على قالب العنصر:

```
var Itm = DataRepeater1.CurrentItem;
// الحصول على كائن عرض صف المؤلف الحالي
var Rv = BooksDataSet.Authors.DefaultView[Itm.ItemIndex];
// الحصول على كائن عرض يحتوي على كتب هذا المؤلف
var Rl = BooksDataSet.Authors.ChildRelations[0];
var BooksView = Rv.CreateChildView(Rl);
// عرض الكتب في جدول العرض
var FrmBooks = new FrmBooks( );
FrmBooks.GrdBooks.DataSource = BooksView;
```

```
FrmBooks.Text = "كتب " +
                Itm.Controls["AuthorTextBox"].Text;
FrmBooks.ShowDialog( );
```

لاحظ أن كل نسخ الزر تستجيب أيضا لهذا الحدث.. ألم أقل لـــك إن نســـخة الأداة مماثلـــة للأداة الأصلية في كل شيء؟.. هذا يشمل الإجراءات المستجيبة لأحداث الأداة، لهذا تستطيع برمجة أحداث الأدوات الموضوعة على قالب العنصر مباشرة، وستكون بذلك قد برمجـــت أحداث كل النسخ المنسوخة من هذه الأداة.

اللهم ارحم أبي واغفر له وكفر عنه سيئاته

تم الكتاب بحمد الله

أخي الفاضل:

إذا كنت قد استفدت ببعض أو كل ما قرأته في هذا الكتاب، فلا تنسني من دعائك بالهداية والتوفيق والسداد والرزق وحسن الخاتمة.

وادع لأبي رحمه الله بالرحمة، فقد نشرت هذا الكتاب مجانا كصدقة جارية له.. وإذا كنت في الحرم أو على مقربة منه، فلا تبخل بعمل عمرة سريعة له والدعاء له في الحرم المكي والحرم النبوي الشريفين.