

برمجة الفيچوال بيسك 2008

كتاب مترجم عن كتاب :

Programming Visual Basic 2008

Mhm76

بسم الله الرحمن الرحيم

الحمد لله رب العالمين والصلاة والسلام على سيد المرسلين سيدنا محمد وآله وصحبه وسلم.

هذا كتاب برمجة الفيچوال بيسك 2008، وهو كتاب يركز كثيراً على الجانب العملي بكل معنى الكلمة وقد حاولت فيه قدر الإمكان تسهيل استخدامه للقارئ العربي، مع العلم أن مشروع الكتاب مترابط من البداية إلى النهاية، والكتاب يحوي 25 فصل بحيث يغطي تقريباً معظم مواضيع البرمجة تحت الفيچوال بيسك 2008، ولا بد من استخدامه أن يكون لديك دراية نوعاً ما في البرمجة تحت الفيچوال بيسك لأنه لا يبدأ من البداية ولكن بإمكانك إذا اصلت القراءة المتسلسلة حتى ولو كنت مبتدئاً أن تتعلم الكثير وربما توفق بمعرفة الكثير ولا أقول أنه ليس للمبتدئين ولكنه بشكل خاص للمحترفين.

حاولت في هذا الكتاب، عمل مشروع الكتاب باللغة العربية ولكنني لم أوفق بترجمة كل شيء لضخامته وخاصة التعليقات على الكود وحيث أنها مكررة كثيراً فبإمكانك فهمها من خلال ما عملت على ترجمته. وكوني مترجم أكثر منه مبرمج، وكون الكتب التي تحتاج إلى ترجمة كثيرة، وحتى الآن، استعين بالله في ترجمة كتاب آخر وأعمل به وقد ترجمة حوالي ستة فصول، هذا غير كتاب "حتى الاحتراف فيچوال بيسك 2008" والذي لم أنهيه بعد، لأن كاتب آخر ربما يكون قد عمل كتاب كامل بخصوصه حتى ولو كان بخصوص الفيچوال بيسك 2005، فرأيت أن من الواجب عليّ توفير كتاب آخر، فكان هذا الكتاب.

عملت على تضمين قاعدة بيانات سكول سرفر الضرورية للمشروع مع مشروع الكتاب وطبعاً، لن أنشر برامج جميع المشروع فما سأشره هو المشروع بشكله النهائي، مع العلم أنني قد عطلت بعض الميزات مثل التقارير وتسطيع تفعيلها والتعديل عليها، وبسبب انخفاض سرعة الانترنت عندنا والمشاكل التي نعاني بها فيه، اختصرت النشر على المشروع بشكله الأخير.

هذه محاولة على طريق تطور أمتنا، أرجو أن تستفيد منها الأمة. هذا الكتاب مجاني للجميع، وثمانه الدعاء لي بظهر الغيب.

جميع الحقوق محفوظة للكاتب تيم باتريك، والمترجم mhm76 وأرجوا أن لا يستغله أحد. والحمد لله رب العالمين.

كائنات الدوت نت .NET Object

من أجل الفهم الكامل لتطوير البرمجيات في الدوت نت، يجب أن تفهم ما هي الكائنات. (إذا كنت على دراية بالبرمجة الموجهة بالكائنات، بإمكانك الذهاب إلى المقطع التالي، ولكن سنتقد بعض من المحتوى الهام). وحيث أن بعض من هذه المعلومات في هذا المقطع ستظهر في الفصل الثامن، يبقى من الهام شرح بعض من مفاهيم الدوت نت هنا. **الكائنات والبيانات. Objects and Data.** من وجهة نظر البرمجة، يقوم الكمبيوتر بعمل أربع مهام أساسية:

- ✓ إنه يخزن البيانات *data* في المنطقة الخاصة بذاكرة الكمبيوتر.
 - ✓ إنه يدعم معالجة هذه البيانات *data* من خلال العمليات الأساسية المتضمنة الجمع، الطرح، والجبر المنطقي، ومعالجة المتغيرات النصية للنصوص.
 - ✓ إنه يسمح للمستخدم من التفاعل مع البيانات *data* المخزنة في الذاكرة.
 - ✓ إنه يوفر طريقة ما لجلب البيانات *data* وإدخالها بالإضافة إلى إخراجها من الذاكرة، من خلال وسائل الإدخال والإخراج مثل لوحة المفاتيح (إدخال) والطابعات (وسائل إخراج) وكذلك وسائل التخزين المختلفة مثل السواقات.
- إن جوهر هذه المهام الأربعة هو البيانات *data*. يوجد الكمبيوتر لمعالجة البيانات، توفر أنظمة التشغيل حجر الأساس لهذه المهام. ولكنها تطبيقات برمجية تنشئ القدرة لهذه الميزات *features—the ability* لكي تعالج بيانات حقيقية *data—real* ذات معنى للمستخدم. إن لغات البرمجة العالية المستوى *High-level programming languages* هي الأدوات الأساسية المستخدمة لتطوير هذه التطبيقات، وتستخدم كل منها بعض الطرق العامة لجعل ميزات معالجة البيانات متاحة للمبرمج.

بالعودة إلى الأيام الذهبية للبرمجة بلغة التجميع *assembly*، وإذا علمت قسم عنوان الذاكرة *memory address* لقطعة من البيانات، فبإمكانك الوصول إليها ومعالجتها مباشرة. في الأيام الأولى للغة البيسك وفي أغلب لغات البرمجة (الإجرائية *procedural*) الأخرى، كان الوصول للبيانات يتم من خلال المتغيرات *variables*. وبما أن اللغات قد تطورت في التعقيد والأهداف، وعليه فقد تطور عرضها للبيانات. فيما يخص لغة *LISP* (اختصار ل *List Processing* أو *Lots of Irritating Silly Parentheses*)، فإنه تتواجد أي قيمة للبيانات ضمن قائمة *list* أو مجموعة *set* كبيرة من البيانات. لكن في لغات الدوت نت، فإنه يتم عرض البيانات من خلال الكائنات *object*.

فالكائنات هي مجموعة قيم البيانات ومصحوبة بالكود المصدري. بينما في لغات البيسك القديمة، كان كل عنصر من البيانات أكثر أو أقل استقلالية من خلال متغيره المحجوز (أو المخصص)، المرتبط بقيم البيانات في اللغات الكائنية التوجه (أو الموجهة بالكائنات *OOP languages*) التي يمكن تصنيفها إلى كائنات. وغالباً ما تتضمن تنفيذات الكائن كود مصدري مصمم لمعالجة قيم البيانات لذلك الكائن.

تمثل الكائنات بشكل عام شيء ما، وغالباً ما يكون لهذا الشيء مثيل (نظير) في العالم الحقيقي *real-world*، إما مادياً أو ظاهرياً (معنوياً). على سبيل المثال، من المحتمل أن يتضمن كودك كائن المنزل الذي له حقول *fields* بيانات أو خصائص *properties* للعنوان، كلون الدهان الخارجي، وعدد الناس القاطنين في المنزل. فالكود المصدري المساعد يمكنه أن يدير تلك البيانات: بطريقة *method* الطلاء تغير قيمة اللون المستخدمة للدهان الخارجي. تدعى البيانات وعناصر الكود ضمن كائن ما بالأعضاء *members*. وإن بعض الأعضاء مخفي داخل الكائن ويتم الوصول إليها فقط بالكود المصدري للكائن. الأعضاء الأخرى أكثر عمومية: حيث يمكن لأي كود في تطبيقك أن يستخدمها، وهي ليست مجرد مجموعة جزئية من كود تطبيق موجود داخل الكائن. اعتبر التلفاز ككائن ما، فالأعضاء العامة للتلفاز على العموم هي سهلة الاستخدام: زر الطاقة جهاز التحكم، التحكم بالصوت، وهكذا. وهذه المسالك التي من خلالها يتحكم المستخدم بقيم بيانات التلفاز (مخرجات الفيديو والايديو). توجد أيضاً أعضاء مخفية داخل التلفاز: بإمكانك استخدام هذه الأعضاء لتؤثر على الصورة وجودة الصوت، على الرغم من أن هذا سيكون فكرة سيئة لأغلب المستخدمين. وبنفس الطريقة لا يحتاج الكائن لكود خارجه للعبث بمكوناته الداخلية في ماعدا الأعضاء العامة. لأهتم كيف يعمل التلفاز داخلياً، طالما أنني أستطيع الحصول على الصورة والصوت الخارج منه باستخدام المتحكمات (الأدوات) التي تم عرضها (الطاقة، القناة، الصوت).

الكائنات والواجهات Objects and Interfaces

تمثل المكونات العامة للكائن واجهته *interface*. فإذا ما أراد كود من خارج الكائن معالجة البيانات المنتمية لذلك الكائن، فإنه يستخدم أعضاء الواجهة. ليس عليه أن يفهم الأعضاء المخفية أو كيفية عملها، وذلك جيد. وخاصة إذا تغيرت هذه الأعضاء الداخلية لأي سبب كان، من الممكن أن يحدث غالباً أكثر مما تعتقد. تعمل الكائنات في البرمجة الموجهة بالكائنات بنفس الطريقة (طريقة تطور التلفاز أنه كان يملك واجهة كبيرة ويشغل حيزاً واسعاً أما الآن فإنه يملك أدوات تحكم عن بعد ويشغل حيزاً أقل ويحتوي على قنوات ووظائف أكثر ولكن الواجهة تقريباً نفسها). طالما أن الواجهة العامة بقيت تقريباً كما هي، فإن الكود الفعلي للكائن ونظام تخزين بياناته الداخلية معروف أيضاً بتنفيذ الكائن يمكن تغييره دون التأثير على كامل التطبيق.

الكائنات والحالات (النسخ أو الصور عنها) Objects and Instances

بالحقيقة إن الواجهة و التنفيذ لكائن تمثلاًن هيكله فقط، وهذه هي الأجزاء التي ينشئها المبرمج من خلال الكود المصدري. وحتى أنها توجد قبل أن ينصب ويترجم البرنامج على كمبيوتر المستخدم. في الحقيقة، في هذا المستوى، حتى أن الكائنات لا تُعرف بشكل حقيقي بالكائنات. في معظم اللغات (ومن ضمنها الفيجوال بييسك)، فإن الكلمة فئة *class* تشير إلى عملية تنفيذ واجهة كائن ما. حالما يتم تنصيب تطبيقك على الكمبيوتر وتشغيله، فإن الكود ينشئ نسخة *instances* من الفئة ليخزن البيانات الفعلية في الذاكرة. هذه النسخ (الصور) هي الكائنات الحقيقية من تطوير البرمجة الموجهة بالكائنات *OOP*. اعتماداً على طريقة كتابة كودك، فربما يتم استخدام تنفيذ فئة وحيدة لإحداث واحدة أو حتى مئات من الكائنات في الذاكرة بنفس الوقت. في الدوت نت، تظهر جميع قيم كودك وبياناتك داخل الكائنات. وجميل جداً أن يكون كل شيء تراه في برنامج التنفيذ هو كائن ما: حيث أن نموذج ويندوز هو كائن ما: أداة صندوق القائمة *list box* التي على ذلك النموذج هي كائن ما: وبنء مفرد في صندوق القائمة ذاك هو كائن ما.

أجزاء إطار عمل الدوت نت The Parts of the .NET Framework

يوجد هناك الكثير من أجزاء إطار عمل الدوت نت التي ما زالت قيد المناقشة. وتظهر هذه الأجزاء كثيراً في توثيق الدوت نت، وكل منها لديه كلمة مركبة من ثلاثة حروف *three-letter acronym* (ALT)، أو ما شابه ذلك.

لغة التنفيذ المشتركة The Common Language Runtime

تقع لغة التنفيذ المشتركة *Common Language Runtime (CLR)* موقع القلب بالنسبة لإطار عمل الدوت نت، فكل شيء تعمله في برنامج الدوت نت تتم إدارته *managed* بواسطة لغة التنفيذ المشتركة *CLR*. عندما تنشئ متغيراً ما، اشكر لغة التنفيذ المشتركة *CLR* ونظام إدارة بياناتها

management، عندما تعمل على حذف قطعة من البيانات ، اشكر لغة التنفيذ المشتركة CLR وكيف تدير **manage** التخلص من البيانات من خلال نظام تجميع النفايات التابع لها. هل لاحظت كم يتكرر ظهور الكلمة **manage** في تلك الجمل؟ ولكن التعبير المناسب ، هو ماذا تعمل لغة التنفيذ المشتركة . في الحقيقة ، تدعى البرامج المكتوبة من أجل إطار عمل الدوت نت الكود المدار **managed code**. أي كود يحدث خارج سيطرة لغة التنفيذ المشتركة ، المتضمنة مكونات COM (ActiveX) المستعملة من قبل تطبيقك الدوت نت ، المعروف بالكود الغير مدار **unmanaged code** . تشبه لغة التنفيذ المشتركة كثيراً مطار لوس انجلوس الدولي . فالطائرات تصل و تغادر في كل دقيقة . السيارات بالآلاف تدخل وتغادر طريق الأوتوستراد و مركز المياني العامة . يتحرك الناس واللموص باستمرار بين المحطات الرئيسية الثمانية والمحطة الدولية الضخمة . يوجد أحداث كثيرة ، ولكن تتم إدارة الكثير منها. فالطائرات لا يمكنها الإقلاع أو الهبوط بدون موافقة على ضبط التحليق . تدير البوابات و نقاط الوصول الطرق و الكراجات العامة . بصدق ، تدير وكالات TSA الجيدة طيران المسافرين و للصوص داخل وخارج المناطق المأمونة للمحطات . تؤكد هياكل الإدارة والضبط في مكان ما في LAX بشكل مرتب و مضمون طيران الناس بين طائراتهم ومدنهم من لوس انجلوس . وتؤكد هياكل الضبط والإدارة في لغة التنفيذ المشتركة CLR بشكل مرتب و مضمون على تدفق البيانات بين كود الدوت نت و بقية الكمبيوتر أو الشبكة المتصلة . و من المحتمل أنك تود أن تعرف عن سر كيفية كون لغة التنفيذ المشتركة قادرة على معالجة البرامج المكتوبة في أي لغة من لغات الدوت نت ، المتضمنة الفيجوال بيسك و سي شارب والفورتران . وهذه هي إرادة المتنافسين في الميكروسوفت . بالواقع إنهم يعرفونه بالفعل، لأنه لا يوجد سر. تحول كل لغات الدوت نت الممكنة (بمعنى تترجم) كودك المصدري إلى اللغة ميكروسوفت الوسيطة **Microsoft Intermediate Language (MSIL)** ، وبشكل عام اختصرت **IL** وهي اللغات المعروفة من قبل لغة التجميع **assembly language** التي تبدو غير مفهومة . على سبيل المثال ، يوجد هنا بعض الكود المصدري للفيجوال بيسك لتطبيق الكونسول الذي يطبع بشكل مبسط "Hello, World!" ، من إجرائية الكود المسماة Main :

```
Module Module1
Sub Main()
    Console.WriteLine("hell world")
    Console.Read()
End Sub
End Module
```

وهو برنامج الدوت نت الكامل . عندما يحوله مترجم الفيجوال بيسك إلى لغة الميكروسوفت الوسيطة MSIL ، تبدو الإجرائية Main مشابهة لهذه (نوعاً ما معدلة لتناسب مع هذه الصفحة)

```
.method public static void Main( ) cil managed
{
    .entrypoint()
    .custom instance void [mscorlib]System.
STAThreadAttribute::.ctor( ) = ( 01 00 00 00 )
// Code size 11 (0xb)
    .maxstack(8)
IL_0000: ldstr("Hello, World!")
IL_0005: call
void [mscorlib]System.Console::WriteLine(string)
IL_000a: ret()
} // end of method Module1::Main
```

نعم، إنه مبهم ولكنه مناسب، تفهمه لغة التنفيذ المشتركة CLR ، وهذا ما نعتبره فعلاً في الدوت نت . طالما أنك تستطيع أن تحصل على كودك داخل اللغة الوسيطة IL ، فإن الدوت نت سوف تعالجه . يحدث مترجم الفيجوال بيسك تماماً ليولد اللغة الوسيطة IL من أجلك . وكذلك مترجمات لغة الدوت نت الأخرى، المتضمنة سي شارب ، واللغة الوسيطة IL الهدف . حتى أنه يمكنك أيضاً أن تكتب كود اللغة الوسيطة IL الخاص بك، لكن من المحتمل أنك تقرأ الكتاب الخطأ من أجل ذلك . فقط طمئن نفسك ، سيكون هذا الجزء الأخير من اللغة الوسيطة IL التي تراها في هذا الكتاب .

توصيف اللغة المشتركة The Common Language Specification

اللغات التي تدعي دعم الدوت نت لا يمكنها فقط أن تقول ذلك لأي سبب قديم . في الواقع عليها أن تكون منسجمة مع الدوت نت و أعمالها . و قد تم عمل ذلك من خلال توصيف اللغة المشتركة (CLS) Common Language Specification . يعرف توصيف اللغة المشتركة CLS أصغر مجموعة من الميزات التي يجب على اللغة أن تنجزها قبل أن تعتبر خاضعة للدوت نت ، أو بدقة ، خاضعة لتوصيف اللغة المشتركة CLS-compliant . يمكن أن تذهب اللغة إلى ما وراء الحد الأصغر إذا أرادت ذلك ، و تتضمن الدوت نت العديد من الميزات الإضافية فوق ميزات اللغة المحددة التي يمكن أن تبنى . من الممكن أن لا تكون اللغة التي تنفذ الحد الأصغر فقط لتوصيف اللغة المشتركة المحددة قادرة على التفاعل الكامل بين مكونات اللغات التي تجاوزت توصيف الحد الأصغر . الفيجوال بيسك هي ، بالطبع ، تابعة لتوصيف اللغة المشتركة ، و بالحقيقة إنها تبتعد خلف الحد الأصغر .

نظام الأنواع المشتركة The Common Type System

بما أن لغة التنفيذ المشتركة CLR تتحكم بكودك المصدري بأي حال ، فإن الميكروسوفت اعتقدت أنه من الجيد أن تملك السيطرة على بيانات الكود المصدري أيضاً . يعمل إطار عمل الدوت نت هذا من خلال نظام الأنواع المشتركة (CTS) Common Type System ، الذي يعرف أنواع البيانات الأساسية وآلية البيانات المستخدمة في برامج الدوت نت . وهو يتضمن جميع أنواع القيم: العددية، السلاسل النصية والأنواع المنطقية . وهو يعرف أيضاً الكائن **object**، وهو وحدة تخزين البيانات الجوهرية في الدوت نت .

يقسم نظام الأنواع المشتركة CTS جميع الكائنات إلى مقدارين كبيرين . يدعى المقدار الأول: الأنواع ذات القيمة **value types**، ويخزن البيانات الفعلية الصحيحة في الخانات . إذا كان لديك قيمة عددية صحيحة **integer** بـ 32 بت **32-bit**، فإنها تدخل التعديل في خانة نوع القيمة ، الجاهزة لاستخدامك المباشر . ويحتوي المقدار الآخر: الأنواع المرجعية **reference types** . عندما تنظر في هذا المقدار، فإنك ترى خريطة تخبرك أين تجد البيانات الفعلية

بطريقة ما في ذاكرة الكمبيوتر. ومن الواضح أن الأنواع ذات القيمة هي المرجحة وهي أبسط للاستخدام ، وهي كذلك، ولكن تصاحبها العديد من القيود، التي لا تُفرض على الأنواع المرجعية.

يمكن للمكونات و البرامج المكتوبة باستخدام نظام الأنواع المشتركة القياسي أن تتبادل البيانات مع بعضها البعض بدون أي عوائق hindrances أو قيود. (يسقط القليل من أنواع بيانات الدوت نت خارج الجوهـر "core" لأنواع نظام الأنواع المشتركة ، لكن عليك أن تتجنبها فقط و خاصة عندما تريد أن تتفاعل مع المكونات التي يمكن أن تستخدم جوهر الأنواع لنظام الأنواع المشتركة فقط.) عندما تكتب تطبيقاتك في الفيجوال بيسك ، فإن معظم كودك سيظهر في فئات classes. والفئات هي أنواع مرجعية وهي تتضمن كلاً من قيم البيانات والكود المصاحب. إن قيم البيانات الموجودة في فئة ما معظمها على الأغلب الجوهر لأنواع بيانات نظام الأنواع المشتركة ، لكن يمكن أن تحتوي أيضاً على الكائنات التي تصممها في مكان آخر في تطبيقاتك. وتتضمن الفيجوال بيسك أيضاً التراكيب، structures، و أنواع قيمة تنفيذ التراكيب، وتتضمن أيضاً كل من البيانات و الكود.

الفئات و التراكيب هي تماماً نوعان من البيانات / الكود المتوفرة في الفيجوال بيسك . الواجهات هي فئة وهياكل تركيب structure skeletons وتتضمن الواجهات تفاصيل تصميمية لما يجب أن يظهر في تركيب أو فئة مترابطة ، ولكن لا تتضمن أي تنفيذ فعلي أو كود تشغيل. تعرف التقاويض إجراء ما بدون تنفيذ (توقيع "signature")، ويتم استخدامه لدعم الأحداث events ، هذه الأعمال (الممهـد لها من قبل المستخدم ، ونظام التشغيل ، أو كودك) التي تخبر كودك "Get to work now!". الوحدات البرمجية Modules هي مقاطع من الكود والبيانات ، ولكن لا تشبه الفئات والتراكيب ، وإنه ليس بإمكانك أن تنشئ كائنات مستقلة منها . مجموعة العدادات Enumerations هي مجموعة مرتبطة بالقيم الصحيحة ، و هي عموماً من أجل الاستخدام كقائمة من الاختيارات .

في لغة الدوت نت ، إن جميع هذه المصطلحات (فئة class ، تركيب structure ، واجهة interface ، تفويض delegate ، وحدة برمجية module ، العداد enumeration) تعرف إجمالاً بالأنواع types. من المحتمل أنك عرفت للتو أن للدوت نت عناصر مشوشة فيها: لذلك لن تشتري كتاب يتعلق بها إن كانت سهلة. ولكن على الرغم أن جميع التقنيات معقدة ، فإن الكلمة البسيطة، نوع type، هي التي تسبب الإرباك الأكبر. إنك ستلاقي بعض الخوف طيلة هذا الكتاب عند قراءته كل مرة . المشكلة : ليست فقط أن النوع يشير إلى تلك العناصر الجوهرية لنظام الأنواع المشتركة CTS لكن يتم استخدامه عند التحدث عن أنواع القيمة الخاصة بالفيجوال بيسك (تدعى أكثر الأحيان الفيجوال بيسك بأنواع البيانات "data types") . واللقب للتركيب هو أنواع معرفة بالمستخدم "user-defined types" ومع ذلك يوجد التباس آخر لاستخدام النوع "type". يذكر المبرمجون الذين استخدموا الفيجوال بيسك قبل تجسيده للدوت نت أيضاً أن النوع "Type" كعبارة لغوية اعتادت أن تنشئ أنواع معرفة بالمستخدم user-defined types . على الميكروسوفت أن تستخدم بعض الكلمات الأخرى للعالم بدلاً من نوع type كالفئات classes ، الواجهات interfaces ، العدادات enumerations ، وهكذا . سيكون اختيار أي كلمة أفضل بما أنه سيتم استخدامها أحياناً لتناقش البرمجيات . لكن "type" هو الكلمة المختارة، لذلك من الأفضل لك أن تعتاد على رؤيتها .

تتألف عادة أعضاء النوع من حقول بيانات بسيطة و إجرائيات كود ، لكن يمكنك أن تدخل أيضاً أنواع أخرى كأعضاء . حيث أنه ، يمكن لفئة ما أن تتضمن فئة ما متداخلة nested إذا دعت الحاجة لذلك . تدعم أشكال محددة فقط التداخل . أتحديث أيضاً عن مستويات الوصول access levels في هذا القسم . كل عضو يملك مستوى دخول و الذي يقول ما الكود الذي يستطيع استخدامه ذلك العضو . يوجد خمسة مستويات دخول access ، تتراوح من العام (شخص ما و إخوته يمكن أن يستخدموا الأعضاء) إلى الخاص private (عليك أن تدخل النوع كي تعرف أنه موجود هناك) . الفصل السادس يناقش نظام الدوت نت بأجل التفاصيل ، المتضمنة المعلومات التي تحتاجها عن الفئات ، والتراكيب ، إلخ.

مكتبات فئات الدوت نت .NET Class Libraries

إن الكمبيوترات بدقة مبهمة تماماً . في حين أنني أستطيع العد بجميع الطرق حتى العدد 17 ، أما الكمبيوتر فأعظم قيمة للعد هي العدد 1 : إنه يعرف فقط الأرقام 0 و 1 . تتضمن وحدة المعالجة المركزية CPU مجموعة من العمليات البسيطة المستخدمة لمعالجة الأرقام 0 و 1، و عمليات أكثر قليلاً تقارن بين 0s و 1s بطرق معقدة . والحيلة العظيمة الأخيرة للكمبيوتر هي قدرته على نقل 0s و 1s من و إلى الذاكرة . تأكد أنه يعمل هذه الأشياء بسرعة الضوء تقريباً ، ولكن هل يستطيع أن يحسب 3.14 حتى 3 مليون خانة عشرية ؟ حسناً بالفعل إنه يستطيع. لا تعرف الكمبيوترات أي شيء عن حروف الأبجدية ، وهي بحق يمكنها معالجة الأرقام 0 و 1 فقط ، ومع ذلك أستطيع استخدام الكمبيوتر هنا لكتابة كتاب ربح جائزة award-winning. إنها القدرة على تصنيف قيم البيانات البسيطة 1 بت و المعاملات داخل مكتبات libraries معقدة بشكل متزايد للتخصص الوظيفي التي تجعل فائدة الكمبيوتر ممكنة. يبني عمل إطار الدوت نت على عشرات من التخصص الوظيفي المعقد بشكل متزايد. عندما تنصب إطار عمل الدوت نت ، فإن وحدة المعالجة المركزية ونظام النوع المصاحب لها يمثلان الجوهر لإطار العمل . يتضمن إطار العمل بحد ذاته جميع التخصص الوظيفي الأساسي الذي يحتاجه لتمكين من أن تضيف 2 إلى 2 وتحصل بشكل صحيح على 4 . وبصفتك مطور تطبيق عمل ما، ستقضي وقتاً طويلاً لعمل ذلك فقط. لكن ماذا لو أردت أن تعمل شيء ما أكثر تعقيداً ، شيء ما والذي تعرف بعض البرامج التي نفذته ، كتصنيف قائمة من الأسماء أو رسم دائرة ملونة على نموذج ما؟ لتحصل على الجواب ، اذهب إلى class libraries، مكتبات فئات الدوت نت . تتضمن هذه المكتبات ، المنصبة مع إطار العمل ، الكثير من (المعقدة بشكل متزايد) التخصص الوظيفي المكتوب سابقاً والذي ليس عليك أن تكتبه من الخريشة.

يوجد مكتبتا فئة في الدوت نت : مكتبة الفئة القاعدية Base Class Library (BCL) ومكتبة فئة إطار العمل Framework Class Library (FCL). مكتبة الفئة القاعدية أصغر ، وتحتوي الميزات الأساسية الأعظم التي لا يستطيع البرنامج العمل بدونها . وهي تتضمن فقط تلك الفئات التي هي أساس واجب لدعم التطبيقات على إطار العمل إذا دعمت الميكروسوفت ذلك، تقول، نفذ إطار العمل إلى نظام تشغيل الكمبيوترات الشخصية Linux . إن مكتبة فئة إطار العمل أكبر ، و تحتوي كل شيء آخر اعتقدت الميكروسوفت أنك تريد أن تملكه في برامجها ، ولكنه ليس أساسياً بشكل جوهري ليُدخل في مكتبة الفئة القاعدية . ويوجد فئات كثيرة جداً في مكتبة الفئة القاعدية ، بالألاف من الفئات (نعم ، بالألاف !) ، العدادات ، الواجهات ، وأنواع أخرى محتواة في مكتبة الفئة القاعدية ومكتبة فئة إطار العمل ، ستعتقد أنه من الصعب أن تجد فقط الفئة التي تحتاجها . لكن ليس ذلك صعباً ، على الأقل ليس صعباً جداً . يتضمن إطار عمل الدوت نت ميزة ما مسماة فضاءات الأسماء namespaces . تظهر جميع الأنواع في الدوت نت كهـرم - تركيب مثل الشجرة - مع القليل فقط من المدخلات الدنيا عند الجذر . كل عقدة node في الهرم هي فضاء اسم . وتعرف أي فئة أو نوع آخر في المكتبات بتسمية جميع فضاءات الاسم بشكل وحيد ، من أسفل الجذر إلى فضاء الاسم المحلي الذي يحوي الفئة ، بفصل كل عقدة بنقطة (.) . بخلاف معظم الأهرام التي لديها جميع الفروع المبتدئة من عقدة جذر وحيدة ، لدى هرم فضاء اسم الدوت نت عقد جذور متعددة . يسمى فضاء اسم الجذر الأكبر النظام System . ويتضمن العديد من الفئات ، ولكنها تتضمن أيضاً عقد هرمية مباشرة متعددة (فضاءات اسم) . بما أن إطار العمل يتضمن ميزات لكل من تطوير تطبيق

قاعدة الوندوز Windows based و قاعدة الانترنت web-based ، فإنه يوجد فضاءات اسم تحتوي ميزات متطورة لويندوز محدد- Windows specific و لانترنت محدد web-specific . تظهر فضاءات الاسم فقط ضمن فضاء اسم النظام System ، وتدعى Windows و Web . وجميع الكود المرتبط بالنماذج المعروضة على الشاشة في فضاء الاسم Windows يظهر في فضاء الاسم Forms ، وضمن فضاء الاسم هذا الفئة الفعلية التي تنفذ نموذج ما، مسمى Form . في الفيجوال بيسك ، تعرف فئة ما بتحديد كل فضاءات الاسم التابعة لها ، مبتدئاً من فضاء اسم الجذر التابع لها . فئة النموذج Form لديها الاسم المحدد الكامل التالي : System.Windows.Forms.Form

توجد جميع الفئات و الأنواع في مكان ما في الهرم ، رغمًا من أنه لا تتحد كل فئة من النظام System . تظهر العديد من الميزات الداعمة المحددة للفيجوال بيسك في فضاء الاسم Microsoft.VisualBasic ، الذي لديه "Microsoft" كعقدة جذر خاصة به بدلاً من "System" عندما تنشئ مشاريع جديدة في الفيجوال بيسك ، الاسم للمشروع هو ، بقيمة ابتدائية، عقدة المستوى الأعلى top-level في الهرم . إذا أنشأت تطبيق Windows جديد مسمى WindowsApplication1 ، فإن نموذج القيمة الابتدائية "Form1" لديه الاسم المحدد الكامل التالي :

WindowsApplication1.Form1

اسم الفضاء للتطبيق الجديد ليس مجرد فئة ثانية second-class ملحقة منحدره من فضاء الاسم System . يتم دمجها بشكل كامل في هرم فضاء اسم الدوت نت الكامل: فضاء الاسم WindowsApplication1 هو عقدة جذر ، فقط مثل عقد الجذر System و Microsoft ، يتضمن الفيجوال بيسك الميزات التي تدعك تغيير القيمة الابتدائية لفضاء اسم تطبيقك ، أو مكان واحد من فئات التطبيق في فضاء اسم ما خاص . يمكنك حتى أن تضع فئات تطبيقك في فرع فضاء اسم النظام System . تغيير WindowsApplication1 إلى النظام System.MySuperApp يتحرك من Form1 إلى:

System.MySuperApp.Form1

إذا كان تطبيقك بشكل فعلي عنصر أو مكتبة موجهة للاستخدام في البرامج ، فإن فئات تطبيقك ستظهر في فضاء الاسم الذي حددته عندما حمل البرنامج الآخر عنصرك في منطقة التطبيق الخاصة به. إن كودك سيبدو كجزء من فضاءات الاسم Microsoft-supplied . على الرغم من أنك تستطيع أن تضيف فئاتك إلى فضاء اسم النظام System ، فإنك سوف تتعرض لغضب ميرمجي الدوت نت الآخرين . يتم فرض فضاء الاسم System ليكون ل"system" (اقرأ مزود الميكروسوفت) ميزات ، وهو ذلك . توجد أيضاً فرصة ما وهي أنه ربما يستخدم بائعان نفس مسار فضاء الاسم . لذلك ، لتجنب تناقضات فضاء الاسم المحتملة و النظرات الدنيئة من المبرمجين الآخرين ، عليك أن تسمي فئات تطبيقك مثل :

CompanyName.ApplicationName.ClassName

كفئة وحيدة أو نوع آخر لا يمكن أن يتم إفساده من خلال فضاءات الاسم المتعددة ، حتى ضمن نفس فرع الهرم . على أية حال ، يمكن أن يتقاسم نوعان أو فئتان الاسم المشترك في فضاءات اسم مختلفة ، حتى ضمن نفس الفرع .

تظهر جميع فئات مكتبة الفئة القاعدية ومكتبة فئة إطار العمل متمازجة في كل مكان في هرم فضاء الاسم الكامل . هذا يعني أنه لا يمكنك بالضرورة أن تقول فيما إذا كانت فئة مخصصة هي من مكتبة الفئة القاعدية أو من مكتبة فئة إطار العمل . بصراحة ، لا توجد مشكلة : فلن يهتم كودك من أي مكتبة تأتي فئة ما ، طالما أنها متاحة للاستخدام على شبكة أجهزة المستخدم .

فقط قيل إصدار الفيجوال استديو 2008 ، أعلنت ميكروسوفت أنها ستجعل الكود المصدري أكثر توافقاً مع مكتبة فئة إطار العمل النسخة 3.5 المتاحة لمراجعة المطورين . وهذا يعني أن المبرمجين الذين يريدون معرفة كيف تصنف الميكروسوفت قائمة من أسماء ما في الذاكرة أو ترسم دائرة ملونة على نموذج ما والذي سيحصل على الأقل على نظرة جزئية عن كيفية عمل ذلك .

الكشوف والمجمعات Assemblies and Manifests

التجميع هو "وحدة نشر" لأجزاء تطبيق أو مكتبة الدوت نت . في % 99.9 من الحالات ، التجميع هو ببساطة ملف دوت نت قابل للتنفيذ (an.exe file) أو فئات مكتبة الدوت نت أو أنواع أخرى (a.dll file) . من الممكن أن تقسم التجميع بين ملفات متعددة ، لكن عادة هو ملف واحد من أجل مجمع واحد . إن ما يجعل ملف قابل للتنفيذ أو ملفات أخرى تجميع ما هو وجود كشف manifest . من أجل تجميعات ملف وحيد single-file ، فإن الكشف manifest يظهر مباشرة في الملف : ويمكنه أن يظهر أيضاً في ملف ما بحد ذاته . الكشف هو قطعة ما كبيرة من البيانات التي تسجل تفاصيل هامة حول التجميع ، المتضمن اسمها ، معلومات إصدارها ، قيم ثقافتها المبدئية ، ومعلومات عن أنواعها و مجتمعاتها الخارجية المرجعية ، و قائمة ما من جميع الملفات الموجودة في التجميع . لن تميز لغة التنفيذ المشتركة CLR تجميع ما بدون كشفه ، لذلك لا تفقده . يمكن أن تتضمن المجمعات اسم قوي strong name اختياري . وهذا يساعد على ضمان سلامة و أصل تجميع ما خلال توقيع رقمي متعلق بالكشف . يستخدم الاسم القوي مفتاح عام للكتابة السرية ليضمن أن التجميع منقطع النظير و لا يمكن التلاعب به . تتضمن الفيجوال استديو و إطار عمل الدوت نت أدوات تتيح لك إضافة اسم ما قوي للتجميع . عندما تنشر تطبيقك ، سوف تعرض بشكل عادي جميع ملفات التجميع ، والملفات التشكيلية ، و أي ملفات متعلقة خاصة بتطبيقك في فهرس تنصيب التطبيق ، تماماً كما في الأيام الجوراسية القديمة قبل الدوت نت . صممت المجمعات المشاركة لكي يتم استخدامها من قبل أكثر من تطبيق واحد على آلة مفردة يمكن أن يتم تخزينها في ذاكرة التجميع العالمي المخفية (Global Assembly Cache (GAC) . يجب أن تملك جميع المجمعات المتركرة في ال GAC أسماء قوية . يمكن أن تسمح بعض الأنظمة فقط لمدير النظام أن يضيف التجميعات إلى ال GAC .

توصيف البيانات والمواصفات Metadata and Attributes

يتم إحضار المجمعات لك بالحرف m . بالإضافة إلى الكشوف و أعضاء نوع ما، تحتوي المجمعات أيضاً توصيف البيانات metadata . إن كود التطبيق و مكونات البيانات مخزنة في تجميع ما مطابق لبنود الكود و البيانات الموجودة في الفيجوال بيسك المتعلقة بالكود المصدري : ومن أجل كل نوع و عضو في كودك المصدري ، يوجد كود مساعد منفذ في التجميع المنشور . وهذا يحدث الإدراك ، و لا يوجد كثير من التغيير قبل انتشار الدوت نت المتطورة pre-.NET . ما الاختلاف الذي يجعل مترجم الفيجوال بيسك الآن يربط المعلومات الإضافية - توصيف البيانات - بكل نوع و عضو في التجميع . يُوثق توصيف البيانات الاسم من المحتوى المساعد ، والمعلومات حول أنواع البيانات المطلوبة ، ومعلومات عن وراثه فئة لعنصر ، و رخص التأمين المطلوبة قبل أن يُستخدم العنصر من قبل المستخدم أو البرمجيات الأخرى . يعزز كودك المصدري للفيجوال بيسك توصيف البيانات من أجل أي عنصر لتجميعك من خلال المواصفات attributes . ينتج توصيف البيانات بوساطة مواصفة ما هي أكثر من مجرد عدد ID ما . تنفذ المواصفات فئات الدوت نت كاملة، مع المنطق المساعد و قيم البيانات الخاصة بها . يمكن لأي كود دوت نت والذي يعرف كيف يعالج المواصفات أن يفحص المواصفات من أجل عضو أو نوع ما و يتخذ حدث كما تدعي حاجته . وهذا يتضمن الفيجوال استديو ، ومترجم الفيجوال بيسك ، وتطبيقاتك التقليدية الخاصة بك . كيف يكون مثال ما عادي : يتضمن إطار عمل الدوت نت مواصفة ما مسماة ObsoleteAttribute تتيح لك هذه المواصفة أن تحدد أعضاء أو أنواع تجميعك وكأنها

مهجورة أو غير مدعومة منذ زمن طويل . (تستخدم الفيچوال استديو هذه المواصفة لتعرض تحذير ما كلما حاولت أن تستخدم ميزة ما خارجة عن التاريخ out-of-date لكل من مكتبة الفئة القاعدية أو مكتبة فئة إطار العمل). ولتستخدم المواصفة ، أضفها إلى عضو ما من تطبيقك مستخدماً أقواس زاوية الشكل (على الشكل <>):

```
Class MyClassWithOldMembers
  <ObsoleteAttribute()> Sub DoSomeWork()
    End Sub
End Class
```

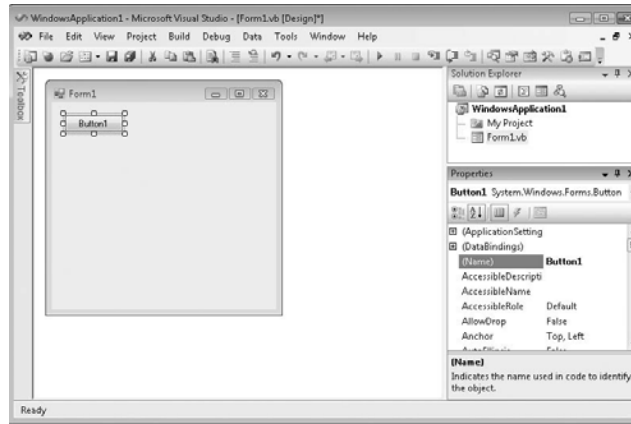
يعرف هذا الكود فئة ما وحيدة (MyClassWithOldMembers) بإجراء ذو عضو ما وحيد (DoSomeWork)، إجراء ما والذي يعمل بوضوح عمل ما. يتم ربط الإجراء بمواصفة ObsoleteAttribute. بواسطة الزبون ، تنتهي جميع أسماء المواصفة بالكلمة Attribute. يمكنك أن تهمل هذا الجزء من الكلمة إذا أردت ، طالما أن الكلمة الناتجة لا تتناقض مع أي كلمة محجوزة في لغة الفيچوال بيسك :

```
Class MyClassWithOldMembers
  <Obsolete()> Sub DoSomeWork()
    End Sub
End Class
```

عندما تترجم الفئة و تخزنها في تجميع ما ، فإنه يتم تخزين المواصفة <ObsoleteAttribute> كجزء من تعريف DoSomeWork . يمكنك الآن أن تكتب تطبيق فيچوال بيسك منفصل والذي يسمح بتجميع ما و مخرجات الاسم و حالات لكل نوع و عضو يجده ، وعندما يصادف ذلك البرنامج التحليلي العضو القديم ، سيكتشف ObsoleteAttribute في توصيف البيانات ، ويطبع الحالة :

DoSomeWork Procedure: Obsolete, don't use it!

يتم تصميم معظم المواصفات مع غرض ما خاص في العقل. تأمر بعض المواصفات الفيچوال استديو أن يعرض الأعضاء لفئة ما بطرق محددة . من المحتمل أنك عملت سابقاً مع ميزات تحرير النموذج form-editing للفيچوال استديو لتصمم تطبيق بسيط لسطح مكتب ويندوز عندما تضيف تحكم ما (مثل زر ما أو صندوق قائمة) لنموذج ما و تختار ذلك التحكم ، فإن الفيچوال بيسك ستترك تحرر التفاصيل من ذلك التحكم من خلال منطقة لوحة الخاصيات Properties (انظر الشكل التالي).



يتم تنفيذ زر التحكم Button كفئة ما ، ويظهر العديد من أعضاء فئته في لوحة الخاصيات Properties ، ولكن ليس جميعها . عندما تم تصميم فئة الزر Button ، فإنه تتم إضافة المواصفات إلى أعضائه التي تخبر الفيچوال استديو أي الأعضاء يجب أن يظهر في لوحة الخاصيات Properties ، وأي منها يجب أن لا يظهر . من الواجب على الفيچوال استديو أن تفحص هذه المواصفات ، و تعرض الخاصيات المطلوبة فقط.

التوافقية Versioning

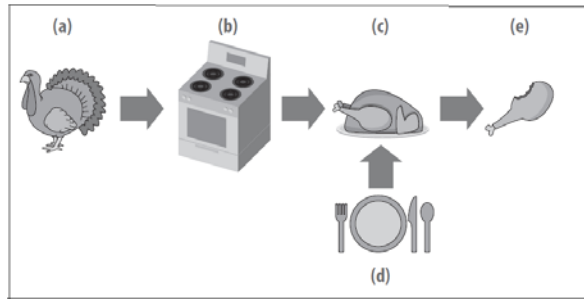
كما لديك ، تطبيقاتي سليمة من تحريراتها الأولية ، ولا أجد سبب لأعدلها أو أضيف ميزات لها . ولكن يوجد تنظيمات تطوير برمجة متضمنة شركة ضخمة واحدة لعمل ذلك ، ولكي لا أسبب إحراج ، سوف أشير إليها فقط بواسطة حرفها الأولي M الذي يلتصق الحاجة إلى تفوق "one-up" المتنافسين بواسطة الخروج مع التوافقية المحسنة "improved" عن سابقتها متخلفة عن عروض البرامج . دعنا نقول أنه حدث الحرف "M" ليملك معالج كلمة شائعة ما والتي تتضمن توافقية 1.0 من فحص تهجئة عنصر ما. يحدث أيضاً الحرف "M" ليبيع أداة بريد ما والتي تعتمد بشكل محدد على التوافق 1.0 من نفس ذلك العنصر المشارك . إذا حرر الحرف "M" ، في عرض رجل منافس ما ، تحديث ما لمعالج الكلمة و فحص تهجئة العنصر (الآن توافق 2.0)، فماذا يحدث لقدرة فحص تهجئة أداة البريد ؟

رغم أن هذا لا يحدث دائماً في الحياة الواقعية . ولكن إذا حدث ، التبدل لأساس عنصر ما متشارك مع عنصر أحدث. لكن يمكن أن يسبب توافق متضارب إلى حد ما مشاكل حقيقية . المشكلة المسروقة هي الانتشار لتوافقيات متعددة لعنصر ما منفرد على نفس شبكة الأجهزة ، جميعها في اتجاهات مختلفة . هل يمكن أن تحذف أي منها بشكل آمن ؟

تحل الدوت نت هذه المشاكل من خلال التوافقية versioning . تعرف جميع المجمعات التي تستخدم العناصر المشاركة بالضبط أي توافقية لعناصر مشاركة تطلبها . على الرغم من أنه يمكن أن يتم تصوير تطبيق ما ليستخد توافقية فيما بعد ، فإنه سيستخدم التوافقية المحددة بشكل أصلي فقط من عنصر مشارك ما افتراضي . يمكن أن تتم إضافة التوافقيات المتعددة من عنصر منفرد مشارك ما إلى ذاكرة التجميع العالمي المخفية GAC، وتدعى ميزة ما جنباً إلى جنب side-by-side بالانتشار deployment. تؤكد لغة التنفيذ المشتركة أن التطبيق الصحيح يرتبط مع العنصر الصحيح. حتى أنه يمكنك أن تنفذ بنفس الوقت التطبيقات التي تستخدم توافقيات مختلفة لنفس العنصر.

من كود مصدري إلى ملف تنفيذي From Source Code to EXE

تعرف الآن بشكل جيد أن كل شيء موجود لتعرف أكثر حول الدوت نت لولا موضوع البرمجة المزعج. قبل التفيتيش في كود فعلي ما ، دعنا نأخذ استراحة صغيرة ونفحص مدى عمر تطبيق ما ، من البداية إلى النهاية (انظر الشكل التالي):



1- إنك ، كمبرمج ، مسؤول عن تحضير العناصر الأساسية (a) من التطبيق . من أجل برامج الفيجوال بيسك ، هذا يعني إنشاء واحد أو أكثر من ملفات الكود المصدري مع امتداد (.vb). يمكن أن تتضمن عناصرك أيضاً ملفات دعم أخرى ، مثل ملفات مصدرية (ملفات نصوص و رسومات ، وغالباً ما تستخدم دعم اللغات المتعددة).

2- يتم التلاعب بتطبيقك بواسطة مترجم الفيجوال البيسك كما في (b) . النتيجة هي تجميع ما ، يكمل مع كشف manifest ما و توصيف البيانات metadata. إن النتائج بالفعل هي لغة ميكروسوفت وسيطة MSIL شبه مترجمة وتتضمن الإصدار الجاهز للتنفيذ ready-to-execute من أنواع وأعضاء الكود المصدري الأصلي ، المتضمن كافة أسماء النوع و العضو . يمكن أن تكون كافة هذه المحتويات ناقصة الترجمة "decompiled" (معادة إلى اللغة الوسيطة الكاملة ، ورغماً من عدم إشباع الفيجوال بيسك) مستخدماً أداة ما مسماة ildasm.exe (مفكك Disassembler لغة الميكروسوفت الوسيطة) ، التي يتم إدخالها مع إطار عمل الدوت نت . بما أنه من المحتمل أنك لا تريد لأحد ما من أن يفكك تطبيقك وينظر إلى الكود ، فإن الميكروسوفت (أو أي طرف آخر) تدعم أيضاً obfuscator ، الذي يمزج بدرجة كافية المحتوى من كودك ليحجبه صعباً بما فيه الكفاية حتى يستطيع إعاقه نظرات المتطفلين .

3- يتم نشر المجمع (c) إلى شبكة أجهزة المستخدم . يتم استخدام طرق مختلفة قليلة لتنشر التطبيق ، المتضمن (1) إنتاج حزمة تنصيب مثبت الويندوز القياسي ، (2) إنتاج توزيع ما ClickOnce : أو (3) إنجاز تنصيب نسخ ما xcopy ، والذي لا يتضمن شيء أكثر من نسخ المجمع التنفيذي نفسه إلى الجهاز الهدف . لا يوجد مشكلة في أي طريقة نشر تختارها ، وإن وقت التنفيذ NET runtime الخاص بالدوت نت في (d) يجب أن يتم تثبيته على شبكة أجهزة المستخدم .

4- المستخدم يأكل -أقصد أنه يشغل - البرنامج كما في (e) . تعمل لغة التنفيذ المشتركة لمجمع لغة ميكروسوفت الوسيطة ترجمة نهائية على الفور-just-in-time (JIT)، لتحضرها من أجل الاستخدام على المنصة المحلية . ومن ثم تعرض التطبيق للمستخدم ، وتدير جميع سمات التطبيق عندما يكون مشغل . يختبر المستخدمون مستوى المتعة والرضا المصادف بشكل قليل عند استخدام تطبيقات برمجية أخرى .

تقديم الفيچوال بيسك Introducing Visual Basic

قواعد المنطق و البيانات The Basics of Logic and Data

كمثال ما، دعنا نقول أنك تلقيت طلب ما من قسم بيع البرامج لعمل برنامج يقلب جميع الحروف في أي قطعة كبيرة من نص مجهز للبرنامج. لذلك أولاً اكتشف المنطق، و ثم نفذه في الفيچوال بيسك. باستخدام شبه الكود (الكود المزيف) pseudocode لغة برمجة اصطناعية (مفتعلة) التي تركيبها بنفسك لتساعدك على كتابة البرامج، بإمكانك أن تضع مخطط (أو مسودة أولية sketch) خارج هذه المهمة الأساسية (مع ترقيم الأسطر الرئيسية):

- 1- احصل على النص الأصلي (أو سلسلة حرفية) من المستخدم .
- 2- إذا لم يزودك المستخدم بأي محتوى ، عندئذ اخرج حالياً .
- 3- حضر مكان ما للسلسلة الحرفية المعكوسة ، فارغ حالياً .
- 4- كرر التالي حتى تكون السلسلة الحرفية فارغة :
- 5- انسخ الحرف الأخير من السلسلة الحرفية الأصلية المتبقية ،
- 6- ضع ذلك الحرف على نهاية السلسلة المقصودة .
- 7- قصّر السلسلة الأصلية ، بإسقاط الحرف الأخير .
- 8- [أنهي قسم التكرار]
- 9- دع المستخدم يرى السلسلة الحرفية المقصودة .

يمكنك أن تكتب هذا المنطق بطرق متعددة : وهذا مجرد مثال واحد فقط . يمكنك الآن أن تحول الشبه شفرة هذه إلى لغة من اختيارك: في هذه الحالة ، الفيچوال بيسك (لا تقلق حول تفاصيل القواعد الغوية حتى الآن):

```
1: originalText = InputBox("Enter text to reverse.")
2: If (Len(originalText) = 0) Then Return
3: finalText = ""
4: Do While (originalText <> "")
5:     oneCharacter = Right(originalText, 1)
6:     finalText &= oneCharacter
7:     originalText = Left(originalText,
8:         Len(originalText) - 1)
9: Loop
10: MsgBox("The reverse is: " & finalText)
```

- إن هذا الكود المصدري *source code* جاهز الآن ليتم استخدامه في برنامج الفيچوال بيسك . و يوضح أيضاً السمات الأساسية المتعددة لكتابة الكود :
- تدعى الخطوات المفردة من التعليمات التدريجية *step-by-step* عبارات *statements* . و في الفيچوال بيسك، تظهر كل عبارة على سطر ما بنفسها . يمكنك قطع العبارات الطويلة إلى سطور متعددة بوصل السطور بفراغ مزدوج تحت كلمة ما *space-underscore*، كما هو واضح في السطر 7 من الكود . عندما يتم نشر عبارة مفردة عبر سطور متعددة في هذه الحالة ، فإن العبارة الكاملة تدعى أحياناً بسطر منطقي *logical line*. بما أنه غالباً ما يتضمن سطر منطقي مفرد ما عمل فيچوال بيسك ابتدائي مفرد ما فقط (مثل عمل *If* أو *Do* ، أو استخدام أعمال الإسناد المختلفة لإشارة المساواة [=]) ، ويشار أيضاً إلى هذه الأعمال كعبارات.
 - تتم معالجة عبارات الكود كل واحدة بدورها، من الأعلى إلى الأسفل . على أي حال ، تبدل عبارات محددة التدفق العادي للبرنامج من الأعلى إلى الأسفل *top-to-bottom* ، كما تم عمله بالكتلة *Do While...Loop* على السطر 4 و السطر 8 من الكود النموذج. وهكذا تدعى العبارات بعبارات ضبط التدفق *flow control* ، و تتضمن الحلقات (تكرار قطعة الكود)، والشروط (بشكل اختياري معالجة قطعة ما من الكود المؤسس على مقارنة ما أو نتيجة محسوبة) ، والقفزات (تحريك بشكل مباشر إلى قسم آخر من الكود) .
 - يتم تخزين البيانات في المتغيرات *variables* ، والتي تسمى حاويات لقيم البيانات . تتضمن قطعة الكود النموذج ثلاث متحولات وهي : *originalText*، *oneCharacter*، *finalText* ، تخزن جميع هذه المتغيرات نص (سلسلة حرفية) بيانات . يسمح لك نظام النوع المشترك للدوت نت (CTS) بأن تنشئ متحولات لأربع أنواع رئيسية من قيم البيانات الأساسية وهي: نصية *text* (كل من الحروف المفردة و السلاسل الحرفية الطويلة) ، أعداد *numbers* (كل من القيم الصحيحة والعشرية) ، التواريخ *dates* (و الأوقات) ، المنطقية *Booleans* (قيم صح أو خطأ) . يمكنك أيضاً أن تنشئ أنواع معقدة أكثر من البيانات بتجميع الأنواع الأساسية .
 - تُخزن البيانات في متحول ما من خلال إسناد ما *assignment* . يتضمن هذا بشكل عام توظيف اسم متغير ما على جهة اليسار من عملية إسناد ما *assignment operator* (=) ، و وضع البيانات أو الحسابات لتخزن في ذلك المتغير على جهة اليمين من نفس إشارة المساواة تلك . تخزن العبارة *finalText = ""* على السطر 3 سلسلة حرفية ما فارغة ("") في المتغير *finalText*. وتعرض عبارة الإسناد *=* على السطر 6 قواعد إسناد لغوية مختلفة نوعاً ما .
 - يمكن أن تتضمن العبارات استدعاءات الدالة *function calls* ، وقطع من التخصص الوظيفي المسبق الكتابة ، وتقع جميعها تحت اسم مفرد . تعمل استدعاءات الدالي حزمة عمل ، و بعدئذ تعيد *return* نتيجة نهائية ، وقيم بيانات . يتم إتباع أسماء الدالي بمجموعة من الأقواس ، التي يمكن أن تتضمن الصفر أو معاملات نسبية *arguments* أكثر ، وقيم بيانات شرطية مدعومة بكود الاستدعاء الذي يستخدمه الدالي ليولد نتائجه . يتضمن الكود النموذج أمثلة عديدة من استدعاءات الدالي ، والمتضمنة الدالي *Right* على السطر 5 . يعيد هذا الدالي نسخة ما من الحروف التي على أقصى اليمين *rightmost* من سلسلة نصية أخرى . إنه يوافق وسيطين اثنين : السلسلة الأصلية من التي استخرجت الحروف من أقصى اليمين ، و قيمة صحيحة ما تشير إلى عدد الحروف اللازمة لتعود . يعيد الكود *Right(originalText, 1)* نسخة ما من الحرف المفرد على أقصى اليمين (1) من *originalText*.

عندما تستخدم دالي ما في كودك المصدري ، عندئذ يمثل الدالي شيء صغير كمتغير: جميع النصوص لاستدعاء الدالي ، من البداية لأسماؤها إلى النهاية بأقواسها المغلقة ، يمكن تُستبدل بمتغير ما والحاوي على نفس البيانات الناتجة. لا تظهر استدعاءات الدالي على جانب الجهة اليسرى من عبارة إسناد ما ، ولكن يمكن أن تظهر تقريباً في أي مكان آخر والذي يمكن أن يظهر فيه المتغير . على سبيل المثال ، يمكن أن يستخدم السطران التاليان لاستبدال السطر 2 في النموذج :

```
' Replacing --> If (Len(originalText) = 0) Then Return
lengthOfText = Len(originalText)
If (lengthOfText = 0) Then Return
```

• بالإضافة إلى الدوال ، تتضمن أيضاً الفيچوال بيسك إجراءات *Procedures*. تحزم الإجراءات *Procedures* الكود المكتوب مسبقاً في حزمة معرفة ، تماماً كما في الدوال *functions*. ولكنها لا تعيد قيمة. يجب أن يتم استخدامها كعبارات مستقلة : ولا يمكن أن تستخدمها حيث ستستخدم استدعاء متغير ما أو دالي ما . الاستدعاء إلى MsgBox على السطر 9 هو مثال نموذجي لاستدعاء الإجرائية في الاستخدام . (MsgBox هو دالي فعلي . يمكن أن يتم جعل الكود النموذجي المدرج سابقاً أكثر فعالية قليلاً . في الحقيقة ، من المحتمل بشكل كامل أن الميكروسوفت حصلت على مسودة أولية من هذا الكتاب ، بما أنها تضمنت ميزة سلسلة الحروف المقلوبة string-reversal اليمينية في الفيچوال بيسك ، و سميتها StrReverse :

```
originalText = InputBox("Enter text to reverse.")
If (Len(originalText) = 0) Then Return
finalText = StrReverse(originalText)
MsgBox("The reverse is: " & finalText)
```

ذلك صحيح : تضمنت الفيچوال بيسك سابقاً ميزة السلسلة الحرفية المقلوبة ، والبعض من كود المكتبة المكتوب مسبقاً الذي أناقشه .تتضمن الفيچوال بيسك الكثير مثل الدوال الجوهرية *intrinsic functions* التي تعتبر جزء من اللغة ، و التي تحزم التخصص الوظيفي المفيد المكتوب مسبقاً. يظهر العديد من هذه الدوال في فضاء الاسم Microsoft.VisualBasic ، والذي يتم صنعه بشكل أوتوماتيكي ميسر لكودك المصدري الخاص بالفيچوال بيسك عندما تنشئ مشروع فيچوال بيسك جديد .

المتغيرات و أنواع البيانات Data Types and Variables

عملياً ما أطلبه من تطبيق الفيچوال بيسك الخاص بي ليعمل هو التالي : خذ البيانات من مصدر ما (لوحة المفاتيح ، القرص الصلب ، الانترنت ، إلخ) ومثلها بطريقة مفيدة ما. ستدير جميع البرامج التي كتبها بشكل فعلي على الأقل بعض البيانات في الذاكرة . تخزن كل قيمة للبيانات في منطقة محددة ما من ذاكرة الكمبيوتر ، على الرغم من أنها محددة بواسطة لغة التنفيذ المشتركة Common Language Runtime. توجد العبارات في الفيچوال بيسك بشكل رئيسي لتدبير و تعالج هذه البيانات بطرق معقدة و مفيدة . يتم تخزين كل البيانات المدارة بواسطة لغة التنفيذ المشتركة في ذاكرة الكمبيوتر ، مع كل قيمة للبيانات منفصلة و محمية عن القيم الأخرى . وكان لدى كل قيمة بيانات كوب شاي مستقل خاص بها كما يبدو في الشكل:



تمتلك جميع قيم البيانات المدارة بواسطة لغة التنفيذ المشتركة محتوى *content* ونوع *type*. المحتوى هو البيانات الفعلية : السلسلة النصية "abc" ، العدد 5 ، فاتورة بيع ، شاي بيكو البرتقالي اللون . كل ما تضعه في كوب الشاي ، وذلك هو المحتوى. في بعض الحالات ، تسمح لك الدوت نت أن تخزن بدون شك لاشيء *nullable* في كوب الشاي (من أجل الأنواع المرجعية كما موصوفة باختصار ، أو أنواع قيمة ملغاة أو باطلة) . يشير النوع إلى النوع المحتوى المخزن في كوب الشاي . في الشكل السابق ، هذا معروض بواسطة الشكل لكل كوب شاي . لكل كوب شاي حدود على النوع من البيانات التي يمكن أن يتم سكبها في كوب الشاي : سلسلة نصية ، عدد صحيح ما ، فاتورة زبون ما .

المحارف Literals

بعض قيم البيانات الأساسية ، مثل الأعداد و السلاسل النصية، يمكن إدخالها في كودك المصدري و استعمالها كما هي . على سبيل المثال ، يعرض الإجراء MsgBox نافذة برسالة نصية مدعومة . تتضمن العبارة : MsgBox("The answer is " & 42) سلسلة محرفية *literal string* ، "The answer is" ، و قيمة صحيحة محرفية ، 42 . (الرمز "&" هو معامل يصل قيمتين مع بعضهما البعض في سلسلة حرفية جديدة .) تستخدم المحارف مرة ، و من ثم تهلك . إذا أردت أن أعرض نفس الرسالة مرة أخرى ("The answer is 42") ، علي أن أكتبها مرة ثانية بنفس قيم المحارف في جزء مختلف من الكود المصدري .

تدعم الفيچوال بيسك أنواع عديدة من المحارف الأساسية. تحاط المحارف النصية *String literals* دائماً بعلامات اقتباس *quote marks*. إذا أردت أن تدخل علامة اقتباس ما لوحدها في المنتصف لنص ما، يتضمن علامتين بدلاً من علامة واحدة : "This is ""literally"" an example."

يمكن أن تكون المحارف النصية **String literals** حقاً ، حقاً طويلة ، تصل حتى حوالي 2 بليون حرف بالطول : إذا أردت أن تكتب حرف واحد فقط بكل ثانية ، فإنه سوف يستغرق أكثر من 63 سنة ليصل إلى طول السلسلة الأعظمي . تتضمن الفيچوال بيسك أيضاً المحارف الحرفية **character literal** التي هي بالضبط حرف واحد في الطول : إذا أردت أن تكتب حرف واحد فقط بكل ثانية ، حسناً ، لا يهم . يتم تمييز هذه المحارف الحرفية بواسطة التذييل "c" بعد النص . يتم إدخال محرف الحرف "A" كما "A" "c"

يتم إحاطة محارف الوقت والتاريخ بإشارات عدد **number signs** بدلاً من علامات الاقتباس . الوقت أو التاريخ (أو كلاهما) الذي تدخل يمكن أن يتم تمييزه في أي تنسيق من قبل الميكروسوفت ويندوز في منطقتك المحددة . إذا كنت مستخدماً الفيچوال استديو **Visual Studio** ، فإنه سوف يعيد تنسيق تاريخك عندما تكتبه في المحرف : #7/4/1776#

يكون أحد عشر نوعاً مختلفاً من قيم البيانات العددية - كل من القيم العددية الصحيحة و ذات النقطة العائمة **floating-point** - جوهر "core" المجموعة من أكواف الشاي العددية . و من يحتاج إلى أكثر من 11 مع الأحد عشر كوب شاي تلك ، يمكنك أن تدير الأعداد من الصفر بجميع الطرق إلى 1×10^{300} و ما بعد ذلك أيضاً . ولتستخدم المحرف العددي ، اكتب العدد على اليمين في كودك ، مثل 27 ، أو 3.1415926535 . تتيح لك الفيچوال بيسك أيضاً أن تحدد أي من الأحد عشر كوب شاي عددي تستخدم من أجل عدد ما ، بإلحاق حرف خاص إلى نهاية العدد . و عادة 27 هو عدد صحيح 27 . ولتجعله بدقة العملة "عشري" ، "decimal" ، مزيل بإشارة **@** : 27@ . عندما أناقش أنواع البيانات بالتفصيل الكامل ، سوف أسجل في قائمة الحروف الخاصة المختلفة ، مثل **@** ، التي تدون نوع البيانات للأعداد الحرفية . النوع الرابع والأخير من محارف الفيچوال بيسك هو المحرف المنطقي **Boolean** . تمثل القيم المنطقية النوع الأبسط من بيانات الكمبيوتر : البت . القيم المنطقية هي إما صح أو خطأ ، موصول أو مقطوع ، نعم أو لا ، لذيد أو مقرف ، ققط أو كلاب ، صفر أو غير الصفر . يمثل المنطق **Booleans** دائماً أحد حالتين أو قيمتين متعاكستين . تمثل اللغة التي استخدمها العالم جورج بول العبارات المنطقية كمعادلات رياضية . الكمبيوترات تحب **love** الجبر المنطقي . لذلك جميع العمليات الأساسية للكمبيوتر ، مثل الجمع ، يتم تنفيذها باستخدام التخصص الوظيفي المنطقي .

تتضمن الفيچوال بيسك المحارف المنطقية صح **True** أو خطأ **False** . لا يوجد علامات اقتباس ، ولا إشارات أعداد ، يوجد فقط الكلمات صح **True** و خطأ **False** . في حالات محددة ، يمكنك معالجة الأعداد كقيم منطقية **Boolean values** . عليك أن تعرف الآن أن خطأ تعادل الصفر (0) ، و صح تعادل كل شيء آخر (مع أنه بشكل عام ، يستخدم 1- من أجل كل شيء آخر "everything else") .

المتغيرات Variables

جميع قيم البيانات الحرفية جيدة ، ولكنها مفيدة مرة واحدة فقط ، و بعدئذ ترحل . في كل مرة تريد أن تستخدم قيمة محرفية ، عليك أن تعيد كتابتها . و كأن قيم البيانات مخزنة في الأكواف المعدة للطرح بعد الاستخدام بدلاً من أكواف الشاي الصينية الجميلة . و بالإضافة إلى ذلك ، يدخل المبرمجون فقط القيم الحرفية ، وليس المستخدمين ، لذلك فهي محدودة الاستخدام بالنسبة لبيانات المستخدم المدارة .

ببساطة المتغيرات **Variables** ليست أكواف معدة للطرح بعد الاستخدام : يمكن إعادة استخدامها . يمكنك المحافظة على وضع نفس النوع من الشاي مرة بعد مرة في كوب الشاي . كوب الشاي متغير نصي يمكنه أن يواصل سلسلة ما من أجل إعادة الاستخدام مرة بعد مرة . على سبيل المثال ، في هذه القطعة من الكود ، يستجيب المتغير للسلاسل المختلفة المسندة إليه :

```
Dim response As String
1: response = "A"
2: MsgBox ("Give me an 'A'!")
3: MsgBox (response)
4: MsgBox ("Give me another 'A'!")
5: MsgBox (response)
6: MsgBox ("What's that spell?")
7: response = StrDup(2, "A")
8: MsgBox (response)
```

يتم إسناد المتغير **response** مرتين بنصين مختلفين : "A" (في السطر 1) وبعدها "AA" (في السطر 7) . إنه يحافظ على كل قيمة تم إسنادها إليه أخيراً : يعرض كل من السطرين 3 و 5 "A" في نافذة صندوق رسالة ما . و ليس عليك أن تسند فقط النصوص الحرفية إليه (المتغير) : يمكن لأي شيء يولد النص أن يسند نتيجته للمتغير **response** . يستخدم السطر 7 دالة فيچوال بيسك جاهزة ، **StrDup** ، ليعيد سلسلة الحرفين **two-character** "AA" و يسندها إلى المتغير **response** . إن استخدام المتغيرات هو عملية ذات خطوتين **two-step** . أولاً عليك أن تصرح **declare** عن المتغير ، و بعدئذ أن تسند **assign** قيمة ما إليه . تهتم العبارة **Dim** بقسم التصريحات **declaration** : وهو يدعك تشير إلى كل من الاسم **name** و النوع **type** للمتغير . وقواعده النحوية **syntax** صريحة إلى حد ما .

```
Dim response As String
```

حيث **response** هو الاسم للمتغير و **String** هو نوعه . يحدث التصريح باستخدام عملية المساواة = كمايلي :

```
response = "The answer"
```

يمكن أن يملك متغير مفرد قيم جديدة مسندة إليه بشكل متكرر . لهذه المرات عندما تريد لمتغيرك أن يملك قيمة محددة ما مباشرة حين التصريح ، فإنه بإمكانك أن تدمج التصريح و الإسناد داخل عبارة منفردة :

```
Dim response As String = "The answer"
```

بالطبع ، إنك غير مقيد بتصريح مفرد فقط : يمكنك أين تنشئ متحولات عديدة كما تحتاج في كودك . ويستخدم عادة كل واحد عبارة **Dim** الخاصة به :

```
Dim question As String
```

```
Dim answer As String
```

بإمكانك أيضاً أن تجمع هذين المتحولين في عبارة وحيدة :

```
Dim question As String, answer As String
```

وهذه هي تماماً البداية لما هو ممكن مع العبارة **Dim** .

Value Types and Reference Types الأنواع المرجعية و الأنواع ذات القيمة

تخزن متغيرات النوع ذو القيمة قيمة فعلية ما: الشاي في كوب شاي نوع ذو قيمة هي المحتوى (المضمون) بذاته . جميع قيم البيانات المحرفية التي ذكرتها سابقاً ، ما عدا النصية Strings ، هي أنواع ذات قيمة value types . تخزن متغيرات النوع المرجعي "reference types" مرجع ما " أو إشارة إلى البيانات الفعلية ، توجد البيانات في مكان ما آخر من الذاكرة . عندما تعمن النظر في كوب شاي قيمة مرجعية ، عليك أن تعرف أوراق الشاي في الأسفل لتحديد المكان الذي تسكن فيه البيانات . إما تملك الأنواع المرجعية البيانات أو لا تملكها . في حال غياب البيانات ، يأخذ نوع مرجعي ما القيمة لاشيء Nothing ، وهي كلمة محجوزة ما للفيجوال بيسك والتي تشير إلى أنه لا بيانات . إن الأنواع ذات القيمة ليست لا شيء Nothing: أبداً فهي تتضمن دائماً بعض القيم ، من المحتمل أن القيمة الافتراضية default لذلك النوع (مثل الصفر من أجل الأنواع العددية). يتيح لك نوع "بدون قيمة nullable" خاص أن تسند لاشيء Nothing إلى نوع ما ذي قيمة ، متيحاً لك أن تنفذ نفس المنطق " هل توجد أي بيانات هنا بأي حال من الأحوال " الذي يوجد مع الأنواع المرجعية . سوف أناقش الأنواع بدون قيمة nullable لاحقاً.

أنواع البيانات Data Types

إن نوع البيانات النصية مفيد ، ولكنه مجرد واحد من أشكال أكواب الشاي المتوفرة لديك . يعرف إطار عمل الدوت نت أنواع البيانات data types جوهرية متعددة. يتم تنفيذ كل نوع بيانات ككفة محددة ضمن فضاء اسم النظام System . نوع البيانات الأساسية الأهم ، كوب الشاي الكبير الذي يستطوع أن يملك أي نوع من البيانات ، ويسمى بالكائن Object . أكثر من مجرد كائن ما، فهذا كائن بحرف O كبير. في هرم فضاء أسماء مكتبة فئة الدوت نت ، فإنه يُحدد في System.Object . وهو الأم لجميع الفئات في الدوت نت : وجميع الفئات الأخرى ، والتراكيب ، والعدادات ، والتقويضات ، لا مشكلة في أي مكان يسكن كل منها بالنسبة لهرم فضاء الاسم ، فيما إذا تمت كتابته من قبل الميكروسوفت أو من قبلك ، فهي مشتقة في النهاية من System.Object . ليس هناك ما يستوعبه : لا يمكنك أن تنشئ نوع ما يشتق بشكل أخير من أي شيء آخر . لذلك عد إلى أنواع البيانات تلك "الجوهرية" التي أشرت إليها سابقاً . والتي تربط الأنواع الأربعة من قيم البيانات المحرفية التي جمعتها من قبل وهي : النصوص، التاريخ، الأعداد ، و القيم المنطقية . ويصنف الجدول التالي أنواع البيانات الجوهرية تلك . يملك كل نوع أيضاً اسم فيجوال بيسك محدد والذي يمكن (و يجب) أن يستخدم بدلاً منه .

الجدول 2-1 . أنواع بيانات جوهر الدوت نت و الفيجوال بيسك Visual Basic data types

وصف>Description	اسم NET	اسم VB
يدعم نوع البيانات المنطقية فقط قيم صح True وخطأ False . من الممكن أن تحول الأعداد إلى قيم منطقية : الصفر 0 يقابل خطأ و كل شيء آخر يصبح صح . وعندما تحول منطق ما بإعادته إلى عدد ، فإن خطأ يصبح 0 و صح يصبح 1- .	Boolean	Boolean
نوع بيانات عددي ما ، يخزن الباييت بايت وحيد (8- بت) أعداد صحيحة بدون إشارة ، تتراوح من الصفر إلى 255. إن نوع بيانات الباييت مفيد جداً للعمل مع بيانات غير نصية، مثل الصور الرسومية .	Byte	Byte
يخزن نوع البيانات الحرفية Char بالضبط حرف نصي واحد . تمثل كل قيمة بيانات حرفية 2 بايت (16 بت) من التخزين ، لذلك يستطيع أن يدير مجموعة حروف ذات بايت مضاعف ، مزودة الدعم للغات كاللغة اليابانية التي تملك عدد كبير من الحروف . على الرغم من أنه اعتاد أن يخزن حروف نصية مفردة ، يحفظ نوع البيانات الحرفية الحروف كقيم عددية صحيحة ، متراوحة من 0 إلى 65,535 وذلك بشكل داخلي.	Char	Char
يعالج نوع بيانات الوقت والتاريخ جميع التواريخ بين كانون الثاني 1،1 بعد الميلاد و31 كانون الأول، 9999 بعد الميلاد، في التقويم الغريغوري. يتم إدخال الوقت بشكل جيد : إذا لم يتم تحديد الوقت ، يستخدم وقت منتصف الليل. يخزن نوع بيانات التاريخ داخلياً التاريخ والوقت كعدد من " العلامات الصغيرة" منذ منتصف الليل في 1،1 كانون الثاني بعد الميلاد . كل إشارة هي جزء من 100 بليون من الثانية (نانو بالثانية).	DateTime	Date
يتم تصميم نوع البيانات Decimal بالاعتماد على العملة . إنها دقيقة جداً في الحسابات الرياضية ، ولها مجال جيد جداً ، يدعم الأعداد التي هي بعد 79- الذي يتبع ب 27 صفر ، الموجبة و السالبة ، (هل قلت 79- متبوع ب 27 صفر؟) وذلك طوله 29 رقم ، وذلك هام لتذكرك، نظراً لذلك حصلت على 29 رقم إجمالاً على كلا جانبي من الفاصلة العشرية. يأتي العدد 79- المتبوع ب 27 صفر مع التقيد بأنه لا أرقام إلى اليمين من الفاصلة العشرية . إذا أردت موقع عشري واحد ، عليك أن تترك واحد إلى اليسار (الجزء العشري من اللوغاريتم) و أن تبقى الأعداد إلى 9.7-المتبوع ب 27 صفر . إذا أردت 29 رقم بعد العدد العشري ، لقد حصلت على صفر ضخم كبير من أجل اللوغاريتم . إذا كنت قد اعتدت أن تستخدم الفيجوال بيسك 6.0 ، العدد العشري هو مشابه لنوع البيانات الثنائي Currency .	Decimal	Decimal
يعالج نوع البيانات المضاعف Double الأعداد الممكنة الكبرى من جوهر أنواع البيانات العددية . ومجاله هو حوالي 10^{-324} إلى $1.798 \times 10^{+308}$ للأعداد الموجبة ، وبمجال مشابه لقيم الأعداد السالبة . بينما أنت تفكر أنك في سماء الأعداد الضخمة giganto-number ، نوع البيانات المزدوج (المضاعف) هو جهراً غير دقيق في الحسابات المعقدة . في بعض الأوقات الحساب الذي يجب أن يؤدي إلى الصفر سوف يحسب بدقة كشيء مثل 0.00000000000005434 ، الذي تم إغلاقه . لكن مقارنة هذه الأعداد مع الصفر سوف تفشل ، بما أنها ليست صفرأ .	Double	Double
إن نوع البيانات الصحيحة Integer هو 4 بايت (32 بت) نوع صحيح ذو إشارة .	Int32	Integer

وهو يعالج الأعداد من -2,147,483,648 إلى 2,147,483,647. إذا كنت مبرمج فيجوال بيسك سابق في الدوت نت، فإن هذا نوع بيانات صحيحة جديدة مكافئ لنوع البيانات الصحيحة Long ذات الإصدار 6.0.	Int64	Long
نوع البيانات الصحيح الطويل Long أكبر من الصحيح Integer: وهو 8 - بايت (64 - بت) نوع صحيح ذو إشارة. و يعالج الأعداد من -9,223,372,036,854,775,808 إلى 9,223,372,036,854,775,808 وهو ليس نفس النوع الصحيح الطويل للفيجوال بيسك 6.0 القديم، وكما لديه سعة تخزين مضاعفة.	Object	Object
الكائن هو نوع جوهري لجميع أنواع الدوت نت. وهو يأخذ مكانه عند قمة هرم النوع و الفئة: وهو الفئة الأساسية الجوهرية لجميع الفئات الأخرى. وهو نوع مرجعي، على الرغم من أن الأنواع ذات القيمة بشكل أخير تشترك منه، أيضاً.	SByte	SByte
نوع البيانات العددية، يخزن SByte بايت وحيد (8 بايت) لأعداد صحيحة بدون إشارة، المتراوحة من -128 إلى 127. وهو إصدار ذو إشارة من نوع بيانات البايث Byte التي بدون إشارة.	Int16	Short
نوع البيانات القصيرة Short هو 2 - بايت (16 - بت) نوع أعداد صحيحة ذات إشارة. وهو يخزن الأعداد من -32,768 إلى 32,767 إذا كنت مبرمج فيجوال بيسك سابق في الدوت نت، فإن نوع البيانات Short الجديد هذا مكافئ لنوع البيانات الصحيح Integer للإصدار 6.0.	Single	Single
يشبه نوع البيانات المفرد Single كثيراً نوع البيانات المضاعف Double، فقط أصغر مجاله للأعداد الموجبة حوالي 1.4×10^{-45} إلى $3.4 \times 10^{+38}$ ، مع مجال مشابه من أجل الأعداد السالبة. مثل نوع البيانات المضاعف Double، يعاني نوع البيانات المفرد Single نوعاً ما عدم الدقة خلال العمليات الحسابية.	String	String
نوع البيانات الحرفي هو نوع مرجعي و يصل تخزينه إلى حوالي 2 بليون حرف من النص. ويخزن حروف شبه الكود، التي تكون قدرتها 2 - بايت (16 - بت) لتخزين الحروف من معظم اللغات في العالم، المتضمنة لغات ذات أبجدية ضخمة، مثل اللغة الصينية.	UInt32	UInteger
تخزن الأعداد الصحيحة UInteger بدون إشارة 4 - بايت (32 - بت) أعداد صحيحة بدون إشارة، متراوحة بين 0 إلى 4,294,967,295. وهو إصدار بدون إشارة لنوع البيانات الصحيحة Integer ذات الإشارة.	UInt64	ULong
يخزن النوع الصحيح الطويل بدون إشارة ULong 8 - بايت (64 - بايت) أعداد صحيحة بدون إشارة، متراوحة من 0 إلى 18,446,744,073,709,551,615. وهو إصدار بدون إشارة لنوع البيانات الطويلة Long ذات الإشارة.	UInt16	UShort
يخزن النوع الصحيح القصير بدون إشارة UShort 2 - بايت (16 - بايت) أعداد صحيحة بدون إشارة، متراوحة من 0 إلى 65,535. وهو إصدار بدون إشارة لنوع الصحيحة القصيرة Short ذات الإشارة.		

إن مطوري الميكروسوفت مسؤولون عن أنواع بيانات الفيجوال بيسك التي تنهي بشكل جيد تلك المهمة بما أن جميع أنواع بيانات الفيجوال بيسك الجوهرية هي ببساطة أغلفة من أجل أنواع بيانات محددة منفذة من قبل الدوت نت. أسماء الفيجوال بيسك المعطاة لكل من هذه أنواع البيانات الجوهرية هي بشكل كامل قابلة للتبادل مع أسماء الدوت نت. فعلى سبيل المثال، النوع الصحيح Integer مكافئ بشكل كامل للنظام الصحيح 32 System.Int32. في الحقيقة، عند كتابة كود الفيجوال بيسك، من الأفضل أن تستخدم مرادفات الفيجوال بيسك، بما أن معظم مطوري الفيجوال بيسك يتوقعون أسماء نوع البيانات هذه في الكود الذي قرؤوه وكتبوه.

ماعد الكائن Object والنص String، جميع أنواع البيانات هذه هي أنواع ذات قيمة. تشترك جميع الأنواع ذات القيمة من System.ValueType (الذي بدوره يشتق من System.Object). البايث ذو الإشارة SByte، الصحيح بدون إشارة UInteger، الصحيح القصير بدون إشارة UShort، الصحيح الطويل بدون إشارة ULong، وأنواع بيانات تم إضافتها إلى الفيجوال بيسك مع الإصدار 2005، على الرغم من أن مكافئات فضاء أسماء النظام System الخاص بها هي في الدوت نت منذ بدايته. على خلاف أنواع البيانات الجوهرية الأخرى، فإن هذه الأنواع الأربع ليست "خاضعة لتوصيف اللغة المشتركة CLS-compliant". وهذا يعني أنه، لا يمكن استخدامها لتتفاعل مع عناصر الدوت نت و اللغات التي تعرف نفسها بأنها ميزات الدوت نت الجوهرية فقط المطلوبة كثيراً. على العموم هذا التقييد ليس بالكثير، ولكن كن على حذر عند العمل مع مكونات أو لغات ثانوية.

التصريحات المتقدمة Advanced Declaration

عندما ذكرت الحاجة إلى تصريح *declaration* وإسناد المتحولات، كنت أركز فعلاً على الأنواع ذات القيمة. تتطلب الأنواع المرجعية خطوة إضافية واحدة: وهي الاستنساخ *instantiation*. إليك عبارات التصريح التالية:

```
Dim defaultValue As Integer
Dim nonDefaultValue As Integer = 5
Dim defaultReference As Object
```

توضح هذه السطور ثلاث متحولات منفصلة: اثنان من الأنواع ذات القيمة (النوع الصحيح Integers) و النوع المرجعي (الكائن Object). على الرغم من أن متحول واحد فقط يملك إسناد بيانات صريح، فإن جميع المتحولات الثلاث السابقة تم إسنادها بالفعل لشيء ما، إما بشكل صريح أو ضمني. لننظر إلى هذه العبارات مرة أخرى ونرى ما تم إسناده بشكل صحيح إلى المتغير.

```
Dim defaultValue As Integer = 0
```



```
' ----- This is a standalone comment, on a line by itself.
  Dim counter As Integer ' This is a trailing comment.
MsgBox ("The counter starts at " & _ ' INVALID COMMENT HERE!
counter) ' But this one is valid.
```

تبدأ التعليقات مع حرف التعليق ، حرف علامة الاقتباس المفرد القياسي ('). أي نص متبوع بحرف التعليق هو تعليق ، و يتم تجاهله عندما يتم ترجمة الكود داخل التطبيق القابل للاستخدام . يظهر أي مقطع وحيد ضمن نص محرفي لم يتم استخدامه كعلامة تعليق .

```
MsgBox ("No 'comments' in this text.")
```

يمكن أيضاً أن تنشأ التعليقات بالكلمة المحجوزة REM (كما في " REMark")، ولكن يستخدم معظم المبرمجين العلامات المفردة بدلاً منها .

العبارات الاختيارية Option Statements

بعض أمثلة الكود التي رأيتها سابقاً والتي يدعمها الفيچوال بيسك بقيمة إسناد افتراضية default assignment لمتغير ما - على الأقل من أجل الأنواع ذات القيمة - إذا أهملت تضمين واحد . في حالات محددة ، سوف يدعم الفيچوال بيسك أيضاً التصريحات *declaration* إذا تركتها خارجاً . في العبارة :

```
brandNewValue = 5
```

إذا لم يكن هناك عبارة Dim والتي تعرف brandNewValue ، فإن الفيچوال بيسك سوف يصرح عن المتغير بالنيابة عنك ، مسنداً إياه إلى نوع البيانات Object. لا تدع هذا يحدث لك ! فإنك لا تعرف أي نوع من المشاكل ستحصل إذا سمحت لمثل هذه الممارسات في كودك . سوف تجد بسرعة أن كودك مملوء بالعلل المنطقية الغامضة ، ونتائج بيانات خفية ، وهكذا .

المشكلة هي أن الفيچوال بيسك لن يحتج (يشتك) إذا أخطأت بكتابة الاسم لمتغيرك المصرح عنه ذاتياً auto-declared، المتروك بدون اختبار ، مثل هذه الممارسات يمكن أن تؤدي إلى كود مثل هذا :

```
brandNewValue = 5
```

```
MsgBox (brandNewVlaue
```

يترجم فيچوال بيسك العبارة بدون أي أخطاء ؟ و الآن يعرض صندوق الرسالة لاشيء بدلاً من الرقم 5 ؟ بإمكانك أن تتجنب مثل هذه الصدمة بحكمة باستخدام العبارات Option الداخلة في لغة الفيچوال بيسك . ويوجد أربع أنواع لهذه العبارات :

- خيار التصريح Option Explicit :

تجبرك هذه العبارة أن تصرح عن جميع المتغيرات مستخدماً Dim (أو عبارة مشابهة) قبل الاستخدام . من الممكن أن تستبدل مكان "On" ب "Off" في العبارة ، ولكن لا تفعل هذا .

- خيار التدقيق Option Strict :

سوف تعمل الفيچوال بيسك بعض تبديلات للبيانات البسيطة من أجلك عند الحاجة . على سبيل المثال ، إذا أسندت قيمة بيانات طويلة Long 64 - بت لمتغير صحيح Integer 32 - بت ، عندئذ سوف تحول الفيچوال بيسك بشكل عادي هذه البيانات إلى حجم أصغر ، محتجة فقط إذا لم تتلاءم البيانات. هذا النوع من التحويل - تحويل تضيق - ليس آمناً بما أن البيانات المصدرية سوف تفشل أحياناً لتتناسب في المكان المحدد .

(تحويل التوسيع ، كما مع تخزين البيانات الصحيحة Integer في الصحيح الطويل Long ، تعمل دائماً ، بما أن المكان المحدد دائماً يحجز القيمة المصدرية .) إن عبارة خيار التدقيق Option Strict On تتخلص من المعالجة الذاتية لتحويلات التضيق . و يتم إجبارك على أن تستخدم دوال التصريح explicit لتتجز تحويلات التضيق . وهذا جيد ، بما أنها أجبرتك أن تفكر حول نوع البيانات لمتغيراتك التي حجزتها . يمكنك أن تستبدل "On" ب "Off" في هذه العبارة ، ولكن إذا حذرتك مرة فإني أحذرك مرتين : بأن لاتعمل هذا .

- خيار الاستنتاج Option Infer :

تخبر عبارة الفيچوال بيسك 2008 الجديدة المترجم ليضع في حسابه أي نوع من البيانات التي تريد من متحول ما أن يستخدمها خاصة عندما لا تخبره . سأناقش خيار الاستنتاج في الفصل السادس ، لذلك سوف أؤخر التفاصيل الآن . عادةً هذا الخيار فعال "On."

- خيارا المقارنة الثنائي والنصي Option Compare Binary and Option Compare Text :

يوجه هذان التشكيلين من عبارة خيار المقارنة Option Compare كودك ليستخدم قواعد الفرز الخاصة من أجل ميزات مقارنة سلسلة نصية محددة . على العموم ، المقارنات الثنائية Binary هي حساسة لحالة الحروف case-sensitive ، حيث أن المقارنات النصية Text ليست كذلك . يعود الأمر إليك في أي طريقة تريد أن تستخدم القيمة الافتراضية هي الثنائية . تظهر هذه العبارات عند أعلى كل ملف كود مصدري في مشروعك ، قبل أي كود آخر :

```
Option Explicit On
```

```
Option Strict On
```

أو لتوفير سعة على قرصك الثمين ، هناك القيم الافتراضية التي تقدمها لكامل مشروعك من خلال خصائص المشروع . في الفيچوال استديو ، اختر القائمة مشروع Project ثم أمر الخصائص Properties. ثم على نافذة خصائص المشروع التي تظهر ، اختر تبويب Compile و هبئ الاختيارات الافتراضية من أجل الخيارات التالية :

“Option explicit”، “Option strict”، “Option compare”، و “Option infer” (انظر الشكل التالي):

Application	Build output path:	bin\Release\	Browse...
Compile	Compile Options:		
Debug	Option explicit:	On	Option strict:
References			Off
Resources	Option compare:	Binary	Option infer:
Services			On

معاملات أساسية Basic Operators

تتضمن الفيچوال بيسك عمليات أساسية عديدة و التي تدعك تعمل ما يريد أن يفعله كودك فعلاً : معالجة البيانات . تتيح لك معاملات الفيچوال بيسك أن تنجز دوال إدارة رياضية ، منطقية ، عددية ثنائية ، ونصية ، وجميعها بدون كلفة إضافية. المعامل الأساسي هو معامل الإسناد ، ممثل بواسطة إشارة المساواة (=) . لقد رأيت مسبقاً هذا المعامل في هذا القسم . استخدمه لتسند بعض القيم إلى متغير ما (أو ثابت) : يتم إسناد كل ما يظهر على يمين المعامل إلى متغير النوع المرجعي أو النوع ذي القيمة الذي على اليسار. العبارة :

```
fiveSquared = 25
```

هي إسناد قيمة 25 إلى المتغير fiveSquared . معظم المعاملات هي معاملات ثنائية *binary operators* - وهي تعمل على قيمتين متميزتين ، واحدة على يسار المعامل و واحدة إلى اليمين : النتيجة هي قيمة محسوبة وحيدة . وكما مع الحسابات يتم استبدالها بشكل كامل بالنتيجة المحسوبة . على سبيل المثال ، عملية الجمع :

```
seven = 3 + 4
```

تصبح

```
seven = 7
```

قبل التطبيق النهائي لمعامل الإسناد (=) . يظهر المعامل الفردي unary operator إلى اليسار فقط من معاملة . على سبيل المثال ، معامل النفي الأحادي يحول العدد الموجب إلى عدد سالب :

```
negativeSeven = -7
```

يضع الجدول التالي قائمة معاملات الفيچوال بيسك الرئيسية و يصف باختصار الغرض من كل واحدة .

المعامل	الوصف
+	يضيف معامل الجمع عددين معاً .
+	يحفظ معامل الزائد الفردي الإشارة لقيمة عدد ما . وهو ليس مفيد جداً لكي تحصل على معامل تحميل زائد
-	يطرح معامل الطرح العامل الثاني من الأول .
-	يعكس معامل النفي الفردي الإشارة لعامله العددي المقرون به
*	يضرب معامل الضرب قيمتين عدديتين ببعضهما .
/	يقسم معامل القسمة العامل العددي الأول بواسطة الثاني ، مرجعاً ناتج القسمة المتضمنة أي باقي عشري .
\	يقسم معامل قسمة الأعداد الصحيحة العامل العددي الأول بواسطة الثاني ، مرجعاً ناتج القسمة ، مع قطع الباقي العشري .
Mod	يقسم معامل الباقي العامل العددي الأول بواسطة الثاني ، و يعود بالباقي فقط كقيمة صحيحة .
^	يرفع المعامل الأسّي العامل الأول (الأساس) إلى قوة الثاني (الأس) .
&	يربط معامل توحيد السلاسل الحرفية عاملي سلسلتين حرفيتين مع بعضهما ، و يحولها إلى سلسلة جديدة بنتائج مترابطة .
And	يعود معامل الربط هذا بقيمة صح True إذا كانت كل العوامل المنطقية أيضاً صح True .
AndAlso	يشبه هذا المعامل تماماً معامل And ، ولكنه لا يفحص أو يعالج العامل الثاني إذا كان الأول خطأ False .
Or	يعود معامل الربط هذا بقيمة صح True إذا كان إحدى العاملين أيضاً صح True .
OrElse	يشبه هذا المعامل تماماً معامل Or ، ولكنه لا يفحص أو يعالج العامل الثاني إذا كان العامل الأول صح True .
Not	يعود معامل النفي بعكس العامل المنطقي .
Xor	يعود معامل "أو" الخاص بالقيمة صح True إذا كان واحد واحد فقط من العوامل صح True .
<<	يرفع معامل الرفع الأيسر البتات المستقلة في عامل عددي صحيح ما إلى اليسار باستخدام عدد البتات في العامل الثاني .
>>	يرفع معامل الرفع الأيمن البتات المستقلة في عامل عددي صحيح ما إلى اليمين باستخدام عدد مواقع البتات في العامل الثاني .
=	يعود معامل المقارنة يساوي إلى equal-to القيمة صح True إذا كانت المعاملات مساوية "equal" بعضها البعض .
<	يعود معامل المقارنة أقل من less-than القيمة صح True إذا كان العامل الأول أقل من "less than" الثاني .
<=	يعود معامل المقارنة أقل أو يساوي إلى less-than-or-equal-to القيمة صح True إذا كان العامل الأول أقل أو يساوي العامل الثاني "less than or equal to" .
>	يعود معامل المقارنة أكبر من greater-than القيمة صح True إذا كان العامل الأول أكبر من الثاني "greater than" .
>=	يعود معامل المقارنة أكبر من أو يساوي إلى greater-than-or-equal-to القيمة صح True إذا كان العامل الأول أكبر من أو يساوي العامل الثاني "greater than or equal to" .
<>	يعود معامل المقارنة لا يساوي إلى not-equal-to القيمة صح True إذا كان العامل الأول لا يساوي إلى الثاني "not equal to" .
Like	يعود معامل نموذج المقارنة القيمة صح إذا تطابق العامل الأول نموذج سلسلة حرفية محددة من قبل العامل الثاني .
Is	يعود معامل مقارنة كائن يساوي إلى القيمة صح إذا كان كلا العاملين يمثلان نفس الحالة لنوع بيانات ما في الذاكرة .

وضع العامل الثاني إلى القيمة لشيء Nothing تدعك تختبر المتغير المرجعي لترى فيما إذا يحتوي البيانات أم لا .

معامل مقارنة كائن لا يساوي إلى not-equal-to هو عكس المعامل Is .

بقدر فعالية المعاملات، فهي تستحوذ على فعالية أكبر عندما تعمل على دمجها. وهذا يعمل لأن أيًا من العوامل يمكن أن تكون تعابير معقدة بحيث تتضمن عوامل خاصة بها. الأقواس المضمنة حول شروط في عامل تضمن معالجة القيم في الترتيب الذي تتوقعه.

```
circleArea = pi * (radius ^ 2)
```

في هذه العبارة ، العامل الثاني لمعامل الضرب * هو تعبير آخر ، و الذي يتضمن المعامل الخاص به .

استخدام الدوال و الإجراءات الفرعية Using Functions and Subroutines

تسمح لك اللغات الإجرائية أن تقسم كودك داخل ما يسمى مقطع منطقي ، تدعى الإجراءات . تتيح هذه الإجراءات " التقسيم و التحكم " بالوصول إلى المحاكاة البرمجة: وأنت تكتب الإجراءات التي تنجز قسم منطقي محدد من الكود ضمن تطبيقك الكامل ، و بعدئذ الوصول إلى هذه الإجراءات من الإجراءات الأخرى . تتضمن الفيجوال بيسك Visual Basic ثلاثة أنواع من الإجراءات وهي:

- **الروتينات الفرعية Subroutines:** هذه الإجراءات ، تدعى أيضاً الإجراءات الفرعية *subprocedures* ، تعمل إجراء جزئي و بعدئذ عد إلى استدعاء الإجراء . يمكن أن يتم إرسال البيانات إلى داخل الإجراء الجزئي من خلال قائمة المعاملات النسبية *argument list* . و يمكن أن تعود بعض القيم من خلال نفس تلك القائمة ، ولكن لا يرسل الإجراء نتيجة نهائية رسمية للخلف . يعمل الإجراء الجزئي عمله ، و حالما يكتمل ، يستمر استدعاء الكود على طريقته النشيطة .
 - **الدوال Functions:** تشبه الدوال الإجراءات الجزئية تماماً ، مع ميزة إضافية واحدة : وهي أنك تستطيع أن تعيد قيمة وحيدة أو حالة كائن من الدالة كنتيجته الرسمية . وبالعادة ، يأخذ الاستدعاء هذه القيمة الراجعة بعين الاعتبار عندما يكتمل منطقها الخاص .
 - **الخصائص Properties:** عندما استعملت ، الخصائص والتي فعلاً تبدو كالمغيرات . أسندت و استرجعت قيمة إلى و من الخصائص تماماً كما عملت مع المتغيرات . على أية حال ، تتضمن الخصائص الكود المخفي ، و المستخدم غالباً لتثبيت البيانات كونها مسندة إلى الخاصية .
- الإجراءات الفرعية و الدوال و الخصائص هي أعضاء الكود لكل فئة أو نوع مشابه . سوف أؤخر نقاش الخصائص قليلاً فيما بعد في القسم . أما الآن، دعنا نستمتع بالدوال و الإجراءات ، اللذان يعرفان مع بعضهما البعض بالطرق *methods* . ولنبدأ بالإجراءات الفرعية . لكي تستدعي إجراء فرعي ما، اكتب اسمه كجملة ، متبوعاً بمجموعة قوسين . أي بيانات تحتاج لكي ترسلها إلى الإجراء تذهب في الأقواس . على سبيل المثال ، يعمل استدعاء الإجراء الفرعي التالي بعض العمل ، ممرراً العدد ID للزبون ، و `startDate`:

```
DoSomeWork(customerID, startDate)
```

يعرف كل إجراء فرعي نوع البيانات و ترتيب المعاملات النسبية التي ممررتها . تضع هذه المعاملات النسبية قائمة يمكن أن تتضمن واحد أو أكثر من المعاملات النسبية الاختيارية *optional arguments* ، التي يتم إسنادها إلى القيم الافتراضية إذا لم تتضمنها . من الممكن أن يكون الإجراء الفرعي معاد تعريفه *overloaded* . ومعرفاً قوائم معاملات نسبية ممكنة مختلفة مبنية على العدد و نوع بيانات المعاملات النسبية . سوف نصادف الكثير من هذه فيما بعد .

الدوال أكثر متعة قليلاً بما أنها تعيد القيمة القابلة للاستخدام . و غالباً ، يتم إسناد هذه القيمة إلى متغير ما :

```
Dim balanceDue As Boolean
```

```
balanceDue = HasOutstandingBalance(customerID)
```

وبعدئذ باستطاعتك عمل شيء ما بهذه النتيجة . إذا أردت ، يمكنك أن تجاهل القيمة العائدة لدالة ما ، و تكون لدينا مسبقاً . تعيد الدالة المستعملة باكراً MsgBox المطابقة من زر منقور على الشاشة *on-screen* من قبل المستخدم لإغلاق صندوق الرسالة . إذا ضمنت فقط الكلمة OK في الزر (القيمة الافتراضية)، فمن المحتمل أنك لا تهتم بأي زر ينقر المستخدم .

```
MsgBox("Go ahead, click the OK button.")
```

ولكن بإمكانك أيضاً أن تتحكم بالنتيجة من الزر :

```
whichButton = MsgBox("Click Yes or No.", MsgBoxStyle.YesNo)
```

في هذه الحالة ، `whichButton` سوف يكون إما `MsgBoxResult.Yes` أو `MsgBoxResult.No` ، لقد تم تعريف النتيجتين الممكنتين من قبل الدالة `MsgBox` .

الشروط Conditions

في بعض الأوقات يتوجب عليك أن تصنع بعض الاختيارات ، و سوف تساعدك التعابير الشرطية أن تعمل ذلك تماماً . تتضمن الفيجوال بيسك دعماً للشروط ، التي تستخدم اختبارات البيانات لتحديد الكود التالي الذي يجب أن تتم معالجته .

عبارات الشرط If

العبارة الشرطية الأكثر شيوعاً هي عبارة `If` . وهي مكافئة للأسئلة الإنكليزية في الصيغة " إذا كان كذا و كذا صحيحاً ، عندئذ نفذ كذا و كذا " . على سبيل المثال ، يمكنها أن تعالج " إذا كان لديك \$20 ، عندئذ يمكنك أن تشتري لي طعام الغداء ، " تملك عبارة `If` قواعد لغوية والتي تقيس تعدد سطور الكود المصدرية :

```
1: If (hadAHammer = True) Then
2:     DoHammer(inTheMorning, allOverThisLand)
3:     DoHammer(inTheEvening, allOverThisLand)
4: ElseIf (hadAShovel = True) Then
5:     DoShovel(inTheNoontime, allOverThisLand)
6: Else
7:     TakeNap(allDayLong, onMySofa)
8: End If
```


تتيح لك عبارة **If** تعريف التفرعات في كودك المبني على الشروط . إنه يبني من ثلاثة عناصر رئيسية: وهي .

- الشروط **Conditions** :
إن التعبير الموجود بين الكلمة **If** (أو **Elseif**) والكلمة المحجوزة **Then** هو الشرط . يتضمن الاختبار شرطين ، على السطرين 1 و 4 . ومن الممكن أن تكون الشروط بسيطة أو معقدة ، ولكن يجب أن تنتج دائماً بقيم منطقية **Boolean** صحيحة **True** أو خاطئة **False** . ويمكن تتضمن استدعاءات لدوال أخرى و معاملات مقارنة و منطقية متعددة .

```
If ((PlayersOnTeam(homeTeam) >= 9) And(PlayersOnTeam(visitingTeam) >= 9)) Or
    (justPracticing = True) Then
    PlayBall()
Else
    StadiumLights(turnOff)
End If
```

- يتبع الشرط الأصلي دائماً الكلمة المحجوزة **If** . إذا فشلت تلك الشروط، تستطيع أن تحدد شروط إضافية تتبع الكلمة المحجوزة **Elseif** ، كما على السطر 4 . يمكنك أن تكتب عدد من عبارات **Elseif** كما تحتاج . لا يتيح لك الشرط الاختياري **Else** أن تحدد تعبير اختبار ما . بدلاً من ذلك، إنه يصل كل شيء لم يتم ضبطه بعد بواسطة عبارات **If** أو **Elseif** . يتم السماح بعبارة **Else** واحدة فقط لكل عبارة **If** .

التفرعات **Branches** :
تكون كل كلمة شرط محجوزة **Then** متبوعة بواحد أو أكثر من عبارات الفيجوال بيسك التي يتم معالجتها إذا تم تقييم الشروط المرافقة إلى صح **True** . جميع العبارات حتى تصل إلى **Elseif** ، أو **Elseif** ، أو **End If** التالية تكون مضمنة في مقطع عبارة التفرع (أي تتم معالجتها حتى تصل إلى التفرع التالي) . يمكنك أن تضمن أي عدد من العبارات في قطعة تفرع ما ، وحتى عبارات **If** ثانوية إضافية . في الكود البسيط، تتم معالجة سطور التفرع 2 و 3 إذا كان الشرط الأصلي **hadAHammer** صحيحاً . تتم معالجة السطر 5 بدلاً من ذلك إذا سقط الشرط الأصلي ، ولكن إذا ما مر الشرط الثاني **hadAShovel** . إذا لم يكن أي من الشروط صحيحاً **True** ، فإن تفرع **Else** ، على السطر 7 ، ينفذ .

الكلمات المحجوزة **Statement keywords** :
إن عبارة **If** هي واحدة من العديد من عبارات الفيجوال بيسك المتعددة الأسطر ، وجميعها تنتهي بالكلمة المحجوزة **End** متبوعة باسم الكلمة المحجوزة الأصلية (**If** في هذه الحالة) . تتضمن الكلمات المحجوزة لعبارة **If** ، والتي تمنح العبارة تركيبها : **If** ، **Then** ، **Elseif** ، **Else** ، و **End If** . إن جميع عبارات **Elseif** و **Else** والفروع ذات العلاقة هي اختيارية . تتضمن عبارة **If** الأيسر تفرع **If** فقط .

```
If (phoneNumberLength = 10) Then
    DialNumber(phoneNumber)
End If
```

من أجل الشروط ذات التفرعات وعبارة مفردة بسيطة وبدون شروط **Elseif**، فسطر واحد بديل يمكن أن يحفظ كودك أوضح.

```
If (SaveData() = True) Then MsgBox("Data saved.")
    If (TimeOfDay >= #1:00:00 PM#) Then currentStatus = WorkStatus.GoHome Else currentStatus =
WorkStatus.BusyWorking
```

إن عبارات **If** جامدة لأنها تجعل كودك أكثر من مجرد تعليمات مجموعة خطية مملّة تدريجياً والتي تنحرف لأي سبب . تتم كتابة البرمجيات لتدعم بعض عمليات العالم الحقيقي . و عمليات العالم الحقيقي نادراً ما تكون خطية . تجعل عبارة **If** إمكانية كودك ليستجيب إلى مختلف شروط البيانات ، أخذاً التفرع المناسب عند الضرورة .

حالما تدخل **If...End If** تكتمل القطعة ، وتستمر المعالجة بالعبارات التالية التي تلي العبارة **End If** .

عبارات اختيار حالة **Select Case Statements**

بعض الأحيان من المحتمل أن تكتب عبارة **If** والتي تختبر متغير على قيمة محتملة وحيدة، مع **then** أخرى، و **then** ، و **then**، وهكذا:

```
If (billValue = 1) Then
    presidentName = "Washington"
ElseIf (billValue = 2) Then
    presidentName = "Jefferson"
ElseIf (billValue = 5) Then
    presidentName = "Lincoln"
...
```

وعليه تذهب ، خلال عبارات **Elseif** كثيرة متعددة . إنها فعالة ، ولكن مملّة قليلاً ، كما أن على كودك أن يختبر بشكل خاص كل حالة . تدعم عبارة اختر حالة **Select Case** بديل أوضح من أجل مقارنات قيمة بسيطة على قائمة ما :

```
1: Select Case billValue
02 Case 1
3:     presidentName = "Washington"
4: Case 2
5:     presidentName = "Jefferson"
6: Case 5
7:     presidentName = "Lincoln"
8: Case 20
9:     presidentName = "Jackson"
```

```

10: Case 50
11:     presidentName = "Grant"
12: Case 10, 100
13:     presidentName = "!! Non-president"
14: Case Is > 100
15:     presidentName = "!! Value too large"
16: Case Else
17:     presidentName = "!! Invalid value"
18: End Select

```

على خلاف عبارة If، التي تعمل اختبار من أجل نتيجة منطقية ما، تقارن Select Case قيمة مفردة مع مجموعة ما من قيم حالة اختبار ما. في المثال، تتم مقارنة المتغير billValue مع القيم المختلفة المعرفة بواسطة كل عبارة حالة Case. جميع الكود الذي يلي عبارة حالة ما Case (حتى عبارة الحالة Case التالية) هو التفرع الذي تتم معالجته عندما تتم عملية المقارنة. بيسك شرط اختياري Case Else (السطر 16) أي شيء لا يمكن أن تتم مطابقته مع أي حالة Case أخرى. و عادة، عبارات Case تجدول قيم وحيدة من أجل المقارنة. بإمكانها أيضاً أن تتضمن قائمة ما من قيم مقارنة الفواصل المنفصلة (السطر 12)، أو تعابير مقارنة مجال بسيط (السطر 14).

If ودوال If and IIf Functions

تتضمن الفيچوال بيسك اختلافين لعبارة If من أجل الاستخدام المتوازي. خذ العبارة التالية:

```
If (gender = "F") Then fullGender = "Female" Else fullGender = "Male"
```

باستخدام الدالة IIf، يتم ضغط هذه العبارة في عبارة إسناد وحيدة مع شرط مدمج:

```
fullGender = IIf(gender = "F", "Female", "Male")
```

تملك الدالة IIf ثلاث معاملات نسبية محدودة بالفواصل comma-delimited. المعامل الأول هو الشرط، الذي عليه أن يظهر في قيم منطقية Boolean إما صح True أو خطأ False. المعامل الثاني يتم إعادته من قبل الدالة إذا كانت قيمة الشرط صح True: تعيد نتيجة الشرط الخاطئة العامل النسبي الثالث. من أجل الشروط البسيطة المخصصة لإعادة القيم إلى المتغيرات المشتركة، إنه حقاً دالة مفيد. ولكن مع أي شيء مفيد حقاً، يوجد ثلاث تحذيرات. التحذير مع IIf هو أن أي شيء يظهر داخل العبارة IIf ستم معالجته، حتى لو لم تتم إعادته كنتيجة. ولدينا هنا مثال خطير:

```
purgeResult = IIf(level = 1, PurgeSet1(), PurgeSet2())
```

ستعيد العبارة بشكل صحيح النتيجة من كل من PurgeSet1() أو PurgeSet2() المبنية على قيمة المستوى. إن المشكلة، أو المشكلة المحتملة، هي أن كلاً من الدوال، PurgeSet1()، PurgeSet2()، سوف يتم استدعاءها: إذا كان المستوى هو 1، إن كل من PurgeSet1() و PurgeSet2() سوف يتم استدعاءه، على الرغم من أن النتيجة من الدالة PurgeSet1() فقط ستم إعادتها.

للمساعدة على تجنب مثل هذه التأثيرات الجانبية، أضافت الفيچوال بيسك 2008 معامل If جديد. إنه يبدو تماماً مثل الدالة IIf، ما عدا من أجل الكلمة المحجوزة If التي تحل محل الكلمة المحجوزة IIf:

```
ClassPurgeResult = If(level = 1, PurgeSet1(), PurgeSet2())
```

سوف يتم الآن استدعاء PurgeSet1() أو PurgeSet2() فقط بناء على الشرط، و لكن ليس كلاهما. على الرغم من أن المعامل If يبدو كدالة، وهو فعلاً معامل صحيح، معروف بمعامل ثلاثي ternary. في وقت الترجمة، تعالج الفيچوال بيسك وتعالج معاملاته النسبية كمعاملات وعوامل وتولد المنطق المناسب. يأخذ تنوع للمعامل If معاملين نسبيين فقط، باستثناء المعامل النسبي المنطقي Boolean الأولي.

```
realObject = If(object1, object2)
```

في هذا الإصدار للمعامل If، إذا أخذ المعامل النسبي الأول القيمة لا شيء Nothing، فإن المعامل سيعيد المعامل النسبي الثاني. إذا لم يكن المعامل النسبي الأول لا شيء Nothing - بمعنى أنه، إذا كان فعلاً شيء ما - فإن المعامل يعيد ذلك المعامل النسبي الأول عوضاً. الهدف عدم إعادة لاشيء non-Nothing.

الحلقات Loops

تتضمن الفيچوال بيسك ثلاث أنواع رئيسية من الحلقات وهي: For...Next و For Each...Next و Do...Loop. تسمح لك بتقسيم التكرار المتعاقب لكودك من خلال التفرعات، تضيف الحلقات الفائدة لكودك حيث أنها تتيح لك أن تكرر قطعة محددة من منطق مثبت أو متغير عدد من المرات.

حلقات For...Next

تستخدم الحلقة For...Next عداد عددي والذي يزداد من قيمة بدائية إلى قيمة نهائية، معالجاً الكود ضمن الحلقة حالاً من أجل كل قيمة متزايدة.

```
Dim whichMonth As Integer
```

```
For whichMonth = 1 To 12
```

```
ProcessMonthlyData(whichMonth)
```

```
Next whichMonth
```

هذا نموذج يدور 12 مرة (من 1 إلى 12)، مرة لكل شهر. يمكنك أن تحدد أي قيم للبدائية والنهائية: يمكن أن يتم تحديد هذا المجال باستخدام متغيرات أو دوال تعيد قيم عددية. حالما يتم الحصول على قيم البدائية والنهائية، لن يتم إعادة حسابها كل مرة خلال الحلقة، حتى لو تم استخدام استدعاء الدالة للحصول حد واحد أو كلا الحدين.

```
' ----- Month(Today) كعدد
```

```
' من أجل التاريخ الحالي
```

```
For whichMonth = 1 To Month(Today)
```

```
ProcessMonthlyData(whichMonth)
```

```
Next whichMonth
```

عادة، تزداد الحلقة بواسطة (1) خلال كل مرة. يمكنك أن تبدل هذه القيمة الابتدائية بضم عبارة Step إلى نهاية سطر العبارة For:

```
For countDown = 60 To 0 Step -1
```

```
...
Next countdown
```

يسمح لك تباين قاعدي إضافي أن تصرح عن متغير عداد الحلقة ضمن العبارة ذاتها . مثل هذه المتغيرات متاحة ضمن الحلقة فقط ، ويتوقف وجودها عند خروجك من الحلقة .

```
For whichMonth As Integer = 1 To 12
    ProcessMonthlyData (whichMonth)
Next whichMonth
```

حلقات For Each. . . Next

الحلقة For Each...Next تطور عن الحلقة For ، وتعمل هذه الحلقة بحث خلال مجموعة من البنود المرتبة والمترابطة ، من البند الأول حتى الأخير . وتعمل أيضاً على كائنات المصفوفات Arrays و التجمعات collection. كما يعمل أي كائن يدعم الواجهة IEnumerable. إن التركيب مشابه تماماً للعبارة القياسية For :

```
For Each oneRecord In setOfRecords
    ProcessRecord (oneRecord)
Next oneRecord
```

حلقات Do. . . Loop

تريد أحياناً أن تكرر قطعة من الكود طالما يكون الشرط المحدد صحيحاً ، أو فقط حتى يكون الشرط صحيحاً . إن التركيب Do...Loop ينجز كلاً من هذه المهام . تتضمن العبارة Do...Loop عبارة While أو Until التي تحدد الشروط من أجل المعالجة المستمرة للحلقة. على سبيل المثال ، تعمل العبارة التالية معالجة ما لمجموعة ما من التواريخ، من بداية تاريخ ما إلى نهاية تاريخ :

```
Dim processDate As Date = #1/1/2000#
Do While (processDate < #2/1/2000#)
    'عمل معالجة للتاريخ الحالي
    ProcessContent (processDate)
    'التقدم إلى البيانات التالية
    processDate = processDate.AddDays (1)
Loop
```

سوف تستمر معالجة البيانات في هذا النموذج حتى تلاقي المتغير processDate أو تجتاز التاريخ 2/1/2000، الذي يشير إلى نهاية المعالجة . إن إصدار العبارة Until مشابه لما تقدم ، ولكن نتيجة الشرط ستكون معكوسة :

```
Do Until (processDate >= #2/1/2000#)
...
Loop
```

اجعل الشرط المضمن بسيطاً أو معقداً كما تريد . واضعاً العبارة Until أو While عند أسفل الحلقة يكفل أن العبارات التي في داخل الحلقة سوف تتم معالجتها دائماً مرة على الأقل:

```
Do
...
Loop Until (processDate >= #2/1/2000#)
```

إذا لم تتم مقابلة شرط الحلقة ، سوف تتابع الحلقة التنفيذ ولم ينتهي تنفيذها . لذلك إذا أردت أن تخرج حلقتك عند نقطة ما (و بالعادة أنت تفعل هذا) ، تأكد أن تتم مقابلة الشرط في النهاية . ويوجد حلقة أخرى مشابهة للحلقة Do...Loop ، تدعى الحلقة While...End While. وعلى أي حال ، فإنها توجد من أجل توافق تراجع فقط . استخدم بدلاً منها العبارة Do...Loop.

عبارات "خروج" Exit Statements

عادة ، عندما تدخل حلقة ما ، فإنه لديك غرض من كل تكرار للعدد الكامل من المرات المحددة بالشروط الأولية للحلقة . من أجل حلقات For ، فتوقع أن تستمر من خلال المجال العددي المدخل أو مجموعة العناصر . في حلقات Do ، فإنك تخطط لكي تحافظ على استمرار الحلقة طالما لم تقابل شرط الخروج حتى الآن . ولكن ربما هناك حلقات تريد أن تخرج منها باكراً . وتبلغ هذا باستخدام عبارة Exit. ويوجد عبارتي Exit خاصتين بالحلقات loop-specific : وهما :

- Exit For : تخرج من الحلقة For...Next أو الحلقة For Each...Next بشكل مباشر.
- Exit Do : تخرج من عبارة Do...Loop مباشرة.

تخرج كل عبارة Exit من الحلقة التي تحوي العبارة : تستمر المعالجة مع السطر الذي يلي الحلقة مباشرة :

```
For whichMonth = 1 To 12
    If (ProcessMonthlyData (whichMonth) = False) Then Exit For
Next whichMonth
```

يتم وضع كود الاستمرار هنا لأمشكلة في كيفية الخروج من الحلقة '

يتم تصميم نموذج الكود للحلقة من خلال جميع ال 12 شهر . على أية حال فشل المعالجة من أجل أي شهر من ال 12 شهر سوف يغادر الحلقة مباشرة ، تاركاً جميع أعمال معالجة الشهر المتبقي . تخرج العبارة Exit Do بشكل مشابه من حلقات Do...Loop مباشرة. في عبارة حلقة الخروج Exit ضمن حلقات متداخلة nested loops (حيث تظهر حلقة ما ضمن حلقة أخرى) ، فإن الحلقة الموافقة فقط التي تحوي العبارة مباشرة يتم خروجها:

```

For whichMonth = 1 To 12
  For whichDay = 1 To DaysInMonth(whichMonth)
    If (ProcessDailyData(whichMonth, whichDay) = False)
      Then Exit For
    Next whichDay
  ' عند الخروج من الحلقة الداخلية يأتي التنفيذ إلى هذا السطر
  ' متابعاً المعالجة من أجل الشهر التالي
Next whichMonth

```

عبارات الاستمرار Continue Statements

بما أن الخروج من حلقة ما يهجر جميع المسارات المتبقية خلال الحلقة ، فمن المحتمل أنك تتغاضى عن عملية معالجة بيانات هامة والتي سوف تحدث بطرق متتالية ، تتضمن الفيچوال بيسك عبارة Continue والتي تتيح لك أن تترك المسار الحالي فقط خلال الحلقة . يوجد تنوع لعبارات Continue مختلفة من أجل كل نوع للحلقات :

- **Continue For** :

تقفز مباشرة إلى نهاية الحلقة For...Next أو الحلقة For Each...Next و تتجهز للمسار التالي . تتم زيادة متغير الحلقة ومقارنته مع المجال أو حدود التجمع.

- **Continue Do** :

تقفز مباشرة إلى نهاية العبارة Do...Loop وتتجهز من أجل المسار التالي . تتم إعادة تقييم الشرط Until أو الشرط While . بما أنه تتم تقييم شروط الحلقة عند استخدام العبارات Continue ، فإنه توجد مواعيد عندها يمكن أن تسبب Continue إغلاق الحلقة ، كما عندما يتم إنهاء المسار خلال الحلقة مسبقاً . في هذا المثال ، تقفز العبارة Continue For معالجة الشهور التي ليس لديها بيانات للمعالجة :

```

For whichMonth = 1 To 12
  If (DataAvailable(whichMonth) = False) Then Continue For
  RetrieveData(whichMonth)
  ProcessData(whichMonth)
  SaveData(whichMonth)
Next whichMonth

```

إنشاء إجراءاتك الخاصة Creating Your Own Procedures

يجب أن تظهر جميع العبارات المنطقية في كودك ضمن إجراء ما ، إما في إجراء فرعي ، دالة ما ، أو خاصية ما . على الرغم من أنه يوجد الآلاف من الإجراءات المسبقة الكتابة من أجلك كي تختار منها في مكتبات إطار عمل الدوت نت ، فإنه بإمكانك أن تضيف إجراءات الخاص أيضاً .

الإجراءات الفرعية Subroutines

تبدأ الإجراءات الفرعية بعبارة تصريح Sub ما و تنتهي بعبارة End Sub . ويظهر جميع منطق إجراءاتك الفرعية بين هذين الفكين الراعين .

```

01 Sub ShowIngredients(ByVal gender As Char)
02:   Dim theMessage As String = "Unknown."
03:   If (gender = "M"c) Then
04:     theMessage = "Snips and snails and puppy dog tails."
05:   ElseIf (gender = "F"c) Then
06:     theMessage = "Sugar and spice and everything nice."
07:   End If
08:   MsgBox(theMessage)
09 End Sub

```

يظهر السطر 1 سطر تصريح الإجراء الفرعي بأبسط نماجه : على طوال الكتاب ، سوف تجد أن هناك كلمات محجوزة إضافية والتي تجمل تصريحات الإجراء لتغيير سلوكها . تبدأ العبارة مع الكلمة المحجوزة Sub (من أجل الإجراء الفرعي) ، متبوعاً باسم الإجراء ، ShowIngredients . تحوي الأقواس التالية لهذا الاسم وسيطات *parameters* الإجراء الفرعي . تسمح الوسيطات لمقطع آخر من الكود الذي سيستخدم هذا الإجراء أن يمرر البيانات ضمن الإجراء ، وأن يتلقى بشكل اختياري البيانات العائدة . يمكنك أن تدخل أي عدد من الوسيطات في تعريف الإجراء الفرعي : بشكل بسيط افصلها بفواصل . يعين كل وسيط الاسم بأن يتم استخدامه في الإجراء (المتغير gender في النموذج) و نوع البيانات (محرفي Char) . تتم معالجة المعاملات النسبية كمتغيرات مصرح عنها ضمن الإجراء ، كما تم عمله مع المتغير gender على السطرين 3 و 5 .

القيم المزودة بواسطة الكود المستدعي تعرف بالمعاملات النسبية *arguments* . يتم تمرير جميع المعاملات النسبية بالقيمة *by value* أو بالمرجع *by reference* . في نموذج الكود ، سيتم تمرير المعامل النسبي الممرر إلى المتغير gender بالقيمة ، كما تم تعيينها من خلال الكلمة المحجوزة ByVal . تشير الكلمة المحجوزة المرتبطة ByRef إلى معامل نسبي ما سيتم تمريره بالمرجع . وإذا لم تدخل أي كلمة محجوزة ، فإنه يتم فرض ByVal . تؤثر طريقة التمرير هذه فيما إذا حصلت تغييرات للمعامل النسبي ضمن الإجراء المحلي وتم إرسالها مرة أخرى إلى كود الاستدعاء . على أية حال ، تتأثر القدرة على تحديث البيانات الأصلية أيضاً فيما إذا كانت البيانات من النوع ذو القيمة *value type* أو من النوع المرجعي *reference type* . ويشير الجدول التالي إلى السلوك المتبع عند دمج طريقة التمرير و نوع البيانات .

طريقة التمرير	نوع البيانات	السلوك
ByVal	Value type	التغييرات التي تحدث للإصدار المحلي من المعامل النسبي التي لا تملك التأثير على الإصدار الأصلي .
ByVal	Reference type	التغييرات التي تحدث على أعضاء <i>members</i> كائن البيانات تؤثر مباشرة على كائن البيانات الأصلي . على أية حال ، الكائن بحد ذاته لا يمكن أن يتم تغييره أو استبداله بكائن بيانات جديد بشكل

كامل .

Value type	ByRef
التغيرات التي تحدث على الإصدار المحلي يتم إعادتها إلى الإجراء المستدعي ، وتؤثر دائماً على قيم البيانات الأصلية.	
Reference type	ByRef
التغيرات التي تحدث سواء لكائن البيانات أو أعضائه تتغير أيضاً في الأصل . من الممكن أن تستبدل بشكل كامل الكائن المرسل إلى الإجراء .	

في معظم الحالات ، إذا أردت أن تستمتع في تعديل القيمة لوسيط ما و امتلاك المتغيرات التي ترجع إلى المستدعي ، استخدم ByRef، وإلا ، فاستخدم ByVal. تشمل السطور من بداية 2 إلى 8 في نموذج الكود الجسم body للإجراء ، حيث يظهر كل منطقك . سيتم استخدام أي متغيرات تم تعريفها لوحدها هنا في الإجراء أيضاً ، كما بالمتغير theMessage على السطر 2 ، وينتهي الإجراء الفرعي دائماً بعبارة End Sub.

الدوال Functions

يختلف تركيب الدالة نوعاً ما عن الإجراءات الفرعية في دعمها لقيمة عائدة فقط .

```
01 Function IsPrime(ByVal source As Long) As Boolean
2:     'تحديد فيما إذا المصدر عدد رئيسي
3:     Dim testValue As Long
4:     If (source < 2) Then
5:         Return False
6:     ElseIf (source > 2) Then
7:         For testValue = 2 To source \ 2&
8:             If ((source Mod testValue) = 0) Then
9:                 Return False
10:            End If
11:        Next testValue
12:    End If
13:    Return True
14 End Function
```

كما مع الإجراءات الفرعية ، يظهر سطر التصريح للدالة أولاً (سطر 1) ، متبوعاً بالجسم (السطور من بداية 2 حتى 13) و بإغلاق عبارة End Function (السطر 14) . يتضمن سطر التصريح تعريف نوع بيانات ما زائد بعد قائمة الوسيط . هذا هو نوع البيانات للقيمة النهائية كي تتم إعادتها إلى الكود المستدعي . استخدم قيمة العودة هذه في استدعاء الكود تماماً مثل أي قيمة أو متغير آخر . وكمثال على ذلك ، يستدعي السطر التالي الدالة IsPrime و يخزن نتيجته المنطقية Boolean في متغير ما :

```
primeResult = IsPrime(23)
```

و لتدل على عودة القيمة ، استخدم العبارة Return . يعمل نموذج الكود هذا على السطور 5 و 9 و 13 . (قاعدة الفيچوال بيسك VB 6.0 القديمة التي تتيح لك أن تسند القيمة العائدة لاسم الدالة لا تزال تعمل) .

الخصائص Properties

لقد ذكرت الحقول منذ وقت مبكر قليلاً ، والتي هي متغيرات أو ثوابت تظهر ضمن فئة ما ، ولكن خارج أي تعريف إجراء .

```
01 Class PercentRange
2:     Public Percent As Integer
03 End Class
```

إن الخصائص مشابهة للحقول : حيث يتم استخدامها مثل متغير أو ثابت على مستوى الفئة class-level. ولكن يتم برمجتها مثل الدوال ، قابلة للوسيطات ، و مالكة قيم عائدة ، و متضمنة منطق كثير كما تطلب . و غالباً ما يتم استخدام الخصائص لتحمي بيانات الفئة الخاصة مع المنطق الذي يتخلص من القيم الغير مناسبة . تعرف الفئة التالية خاصية وحيدة تدعم الوصل إلى الحقل المخفي (أو الخاص) ذو الصلة :

```
01 Class PercentRange
2:     'يخزن نسبة مئوية من 0 إلى 100 فقط
3:     Private savedPercent As Integer
04 Public Property Percent( ) As Integer
05 Get
6:     Return savedPercent
07 End Get
08 Set(ByVal value As Integer)
9:     If (value < 0) Then
10:         savedPercent = 0
11:     ElseIf (value > 100) Then
12:         savedPercent = 100
13:     Else
14:         savedPercent = value
15:     End If
16 End Set
17 End Property
18 End Class
```

تحمي الخاصية Percent (السطور من 4 إلى 17) الوصول إلى الحقل savedPercent (السطر 3)، مصححة أي قيمة مزودة للمستدعي caller-supplied التي تجتاز المجال من 0 إلى 100. تتضمن الخاصيات إسناد منفصل و استرداد المكونات ، و تستدعي محددات الوصول أيضاً . تعيد Get accessor (السطور من 5 إلى 7) قيمة مراقبة الخاصية إلى المستدعي . تدع Set accessor (السطور من 8 إلى 16) المستدعي يعدل قيمة الخاصية . تتضمن عبارة تصريح الخاصية (السطر 4) نوع بيانات والذي يلائم نوع البيانات المارة إلى داخل Set accessor (السطر 8) . هذا هو نوع البيانات لمجموعة القيم أو المستعادة من قبل المستدعي . لكي يستخدم هذا النموذج خاصية Percent ، أنشئ حالة ما من الفئة PercentRange ، بعدئذ استخدم الخاصية :

```
Dim activePercent As New PercentRange
activePercent.Percent = 107 ' An out-of-range Integer
MsgBox(activePercent.Percent) ' Displays "100", not "107"
```

يمكنك أن تنشئ خاصيات للقراءة فقط read-only أو للكتابة فقط write-only تتضمن الكلمة المحجوزة القراءة فقط ReadOnly أو الكتابة فقط WriteOnly تماماً قبل الكلمة المحجوزة Property في عبارة التصريح (السطر 4) ، و ترك محددات الوصول accessor الغير محتاج إليها ، لا تحتاج الخاصيات لأن تربط بالحقول. يمكنك أن تستخدم الخاصيات لتجمع و ترتب أي نوع قيمة ، و تخزنه و تمثل عليه بأي طريقة تشاء .

أين تضع إجراءاتك Where to Put Your Procedures

بالعودة إلى الأيام القديمة الجيدة للفيچوال بيسك 6.0 ، فإنه بإمكان الإجراءات أن تظهر تماماً في أي مكان في ملفات كودك المصدري . عليك أن تفتح ملف مصدري ما ، اكتب دالي ما ، و اذهب : لقد كان ذلك سهلاً . مع الانتقال إلى الدوت نت ، فإنه جميع إجراءات الفيچوال بيسك يجب أن تظهر الآن ضمن فئة معرفة (أو تركيب أو وحدة برمجية).

```
Class Employee
    Sub StartVacation()
    ...
    End Sub
    Function TotalVacationTaken() As Double
    ...
    End Function
End Class
```

عندما تنشئ حالات لفتتك فيما بعد في الكود ، يتم استدعاء الطرق مباشرة من خلال حالة الكائن .

```
Dim executive As New Employee
...
executive.StartVacation()
```

ميزات أخرى للتحكم بالسياق Other Flow Control Features

تتيح لك الحلقات و العبارات الإضافية المتوفرة في الفيچوال بيسك إعادة توجيه كودك المبني على البيانات ، تتضمن اللغات عبارات أخرى قليلة والتي تدعك تضبط العمل في أكثر من طريقة مباشرة.

العبارة GoTo

تتيح لك العبارة GoTo القفز مباشرة إلى موقع آخر ما ضمن الإجراء الحالي . والغرض من القفز دائماً هو عنوان السطر line label ، وهو موقع سطر مسمى في الإجراء الحالي . تظهر جميع عناوين السطور عند بداية السطر المنطقي ، و تنتهي بنقطتين .

```
PromptUser:
    GetValuesFromUser(enumerator, denominator)
    If (denominator = 0) Then GoTo PromptUser
    quotient = numerator / denominator
```

في هذا النموذج ، تقفز العبارة GoTo عائدة إلى العنوان PromptUser عندما يكتشف الكود قيمة بيانات غير مسموح بها . وتستمر معالجة البيانات مع السطر التالي مباشرة للعنوان PromptUser. لا يمكنك أن تستخدم نفس اسم العنوان مرتين في نفس الإجراء، على الرغم من أنك تستطيع أن تعيد استخدام أسماء العنوان في إجراءات مختلفة. إذا أردت ، ادخل عبارة منطقية أخرى على نفس السطر بوصفها عنوانك ، تماماً بعد النقطتين ، رغماً من أن كودك سيكون أسهل نوعاً ما حتى تقرأه إذا أبقيت العناوين على سطورها الخاصة .

```
LabelAlone:
    MsgBox("It's all alone.")
LabelAndCode: MsgBox("Together again.")
```

تستطيع كتابة عناوين متعددة في كودك كما تريد ، لكن العبارة GoTo هي واحدة من أولئك العناصر من الفيچوال بيسك التي تتم مراقبتها بشكل أقرب بواسطة وكالات البرمجة العالمية المزعجة ، مثل اللجنة العالمية للمحافظة على GoTo. تتدقق تلك المجموعة أيضاً كتب الكمبيوترات باحثة عن مراجع ازدرأ ليس لاسم تنظيماً أنها أن تجد أي شيء مثل ذلك في هذا الكتاب ، و لكن قضيته الجوهرية مفرطة في الاستخدام للعبارات التي يمكنها أن تقود إلى تغيير مسار الكود ، كالتالي :

```
Dim importantMessage As String = "Do"
    GoTo Step2
Step6: importantMessage &= "AG!"
    GoTo Step7
Step3: importantMessage &= "wit"
    GoTo Step4
Step2: importantMessage &= "wn "
    GoTo Step3
```

```
Step5: importantMessage &= "CK-G"
      GoTo Step6
Step4: importantMessage &= "h I"
      GoTo Step5
Step7: MsgBox(importantMessage)
```

يقول بعض الناس أن مثل هذا الكود صعب للقراءة . ويستدعيه آخرون لمهمة سرية . مهما تدعوه ، فإنه يصنع كود صعب جداً للمرجعة و للحفاظ عليه . من المحتمل أنه يجب عليك أن تحافظ على عين واحدة عند استخدامك لعبارات GoTo: وإذا لم تفعل ، فربما سيفعل ذلك شخص آخر . تضع الفيچوال بيسك بحد ذاتها قيود على الاستخدام GoTo. ليس بإمكانك أن تقفز إلى أو من العبارات المتشابهة المحددة التي سوف تنتج في كود ابتدائي أو قيم بيانات بشكل غير مناسب. وعلى سبيل المثال ، لا يمكنك أن تقفز إلى المنتصف من العبارة For...Next من خارج العبارة ، بما أن متغير عداد الحلقة و مجالات البداية و النهاية لن تكون مبدئية كما ينبغي.

```
' هذه العبارة "إذهب إلى"ستفشل '
  GoTo InsideTheLoop
  For counter = 1 To 10
InsideTheLoop:
  MsgBox("Loop number: " & counter)
  Next counter
```

على أية حال ، حالما تكون في داخل الحلقة ، فإنه بإمكانك القفز إلى عناوين السطر الذي يظهر أيضاً في الحلقة ، وهو مقبول أن تقفز خارج الحلقة مستخدماً GoTo. تفرض بعض التراكيب المتعددة السطور الأخرى قيود مشابهة .

عبارات العودة The Return Statement

ليس بإمكانك فقط أن تقفز في مكان قريب ضمن إجراء ما مستخدماً GoTo ، ولكن تستطيع أيضاً أن تقفز تماماً خارج الإجراء في أي وقت تريد مستخدماً العبارة Return، وعادة ، يعلق إجراء ما عندما تصل معالجة بيانات السطر الأخير للكود في الإجراء : تستمر بعدئذ معالجة البيانات مع الكود الذي استدعى الإجراء . تزود العبارة Return طريقة ما لتغلق الإجراء قبل الوصول إلى النهاية . في إجراء فرعي ، تظهر عبارة Return بحد ذاتها كعبارة ما مستقلة :

Return

في الدوال ، يجب أن تتضمن العبارة القيمة لتعاد استجابة لاستدعاء الكود : متغير ما ، محرف ما ، أو تعبير ما والذي يجب أن يصل نوع البيانات ذات القيمة العائدة المحددة للدالة .

Return 25

تحررت الدوت نت السابقة Pre-.NET من الفيچوال بيسك المستخدمة عبارة Exit لتترك بشكل مباشر إجراء ما . و لا تزال هذه مدعومة في الدوت نت . يوجد ثلاث تنوعات :

:Exit Sub

و هي تغلق إجراء جزئي ما .

:Exit Function

و هي تغلق دالة ما .

:Exit Property

و هي تغلق خاصية ما .

عند الخروج من دالة ما ، فإن العبارة Exit Function لا تتضمن طريقة ما لتخصص قيمة عودة . عليك أن ترتب قيمة العودة بشكل منفصل باسناد قيمة العودة للاسم من الدالة .

```
Function SafeDivide(ByVal numerator As Double,
  ByVal denominator As Double) As Double
  'جعل الإشارة #العدد من النوع البيانات المضاعف '
  If (denominator = 0.0#) Then
    'العودة بصفر في حال فشل القسمة '
    SafeDivide = 0.0#
    Exit Function
  End If
  Return numerator / denominator
End Function
```

عبارات التوقف و النهاية The End and Stop Statements

تقدم عبارات Stop و End توقف مباشر لتطبيق الفيچوال بيسك. تعلق العبارة End برنامجك مباشرة ، ملغية جميع الكود الإضافي و معالجة البيانات (على الرغم أنه تم توضيح مصادر مكتسبة محددة) .

توقف العبارة Stop المعالجة فقط عندما تنفذ تطبيقك ضمن المصحح debugger ، كما في بيئة تطوير الفيچوال استديو . تحول Stop التحكم إلى البيئة ، متيحة للمطور أن يفحص ويغير البيانات والكود كل ما في الإمكان قبل الاستمرار مع البرنامج. إذا لم تتم مناقشة Stop في تطبيق مستقل منقداً خارج المصحح ،إنه يحدث المستخدم كي يصحح التطبيق مستخدماً أي مصحح من نصب على شبكة الحاسوب . ليس من الضروري أن تقول ، إن المستخدم لن يتسلى .

الأحداث و معالجات الحدث Events and Event Handlers

إن الفيجوال بيسك هي لغة قيادة الحدث **event-driven**. وهذا صحيح بصورة خاصة لبرامج مكتوبة كي تنفذ على سطح مكتب ويندوز. بعد بعض الأوليات الهامة، يتحكم المستخدم على العموم بجميع الأحداث في البرنامج. من يعرف ما سوف يعمل المستخدم المولع. ربما ينقر هنا. ربما تكتب هناك. من الممكن أن يكون فوضى وهرج. ولكن كل ما يفعله المستخدم، سيعلمك برنامجك حوله من خلال الأحداث **events**. منذ الأيام الأولى للويندوز، استخدمت برامج سطح المكتب مضخة رسالة **message pump** لكي تتصل بالمستخدم وأحداث النظام لكودك. مدخلات الفأرة ولوحة المفاتيح، وأحداث مولدة للنظام **system-generated**، وإشعارات أخرى من تدفق المصادر الخارجية إلى طوابير الرسالة **message queue** المشتركة للبرنامج. تظيل مضخة الرسالة هذه الرسائل واحدة تلو الأخرى، وتفحصها، تمدها بمساحات مناسبة من كودك. في برمجة الويندوز التقليدية، إنك تقوم بإظهار مضخة الرسالة بمهارة بنفسك، مدخلاً الكود الذي يصنع استدعاءات مباشرة لإجرائية معالجة الحدث **event-handling** المبنية على نوع الرسالة. في برنامج الفيجوال بيسك (كل من الدوت نت و فيما مضى)، تدعم اللغة مضخة الرسالة من أجلك. وتحلل الرسائل وكأنها مفرغة بواسطة المضخة خارج طابور الرسالة، وتقودها إلى الكود المناسب. ويظهر في الدوت نت الكود ضمن فئات. و حالما تملك فئة ما فرصة ما لتحلل الرسالة، فإنه يمكنها أن تولد حدث ما، والذي تتم معالجته أخيراً من قبل معالج الأحداث **event handler**، إن الإجراء الفرعي الذي كتبتة كي يستجيب للفعل. وهذا الاستدعاء لمعالج يعرف بإطلاق **firing** حدث ما.

لذلك يوجد جزآن للحدث: (1) بعض الكود الذي يقرر إطلاق الحدث. (2) معالج الحدث الذي سوف يستجيب للحدث الذي تم إطلاقه. إن الأحداث بالفعل هي استدعاءات غير مباشرة تماماً للإجراء. بدلاً من امتلاك الكود الرئيسي استدعاء إجراء فرعي آخر بشكل مباشر، فإنه يطلب من الدوت نت أن تستدعي الإجراء الفرعي الآخر من أجله، ممررة معاملات نسبية خاصة ربما يرغب الكود المستدعي تضمينها. لذلك، لماذا تريد أن تعمل هذا بدلاً من صنع استدعاء مباشر لإجراء فرعي؟ من أجل شيء ما، تتيح لك هذه طريقة الغير مباشرة أن تضيف معالجات حدث لمدة طويلة بعد أن تتم كتابة كود إطلاق الحدث **event-firing** الابتدائي. هذا جيد، بما أن كود إطلاق الحدث ممكن أن يكون في مجمع ثانوي تمت كتابته منذ سنين. المنفعة الثانية هي أن حدث واحد يمكنه أن يطلق معالجات حدث متعددة. عندما يتم إطلاق الحدث، سيتم استدعاء كل معالج حدث، وكل واحد يستطيع أن ينجز أي منطق خاص موجود في معالج الإجراء الفرعي.

يمرر الكود الذي يطلق الحدث بيانات خاصة **event-specific** بالحدث إلى معالج أو معالجات الحدث الهدف من خلال قائمة وسيطات الحدث. من أجل أن يعمل استدعاء الإجراء الفرعي الغير مباشر، يحتاج معالج الحدث لأن يحوي العدد الصحيح من المعاملات النسبية، في الترتيب الصحيح، كل نوع من البيانات المحددة والمتوقعة. تعرف عبارة الحدث **Event** هذا العقد (الاتفاق) بين الحدث والمعالج.

```
Public Event SalaryChanged(ByVal NewSalary As Decimal)
```

تعرف عبارة الحدث **Event** حدث ما مسمى ب **SalaryChanged** مع معامل نسبي وحيد، ذو قيمة عشرية **Decimal**. إن رغبة أي معالج حدث في أن يراقب الحدث يجب أن يلائم توقيع المعامل النسبي.

```
Sub EmployeePayChanged(ByVal updatedSalary As Decimal)...
```

يمكن أن تحدث الأحداث **Events** من أجل أي سبب تعتبره ضرورياً: حيث لا تحتاج الأحداث أن تقييد إلى مستخدم أو أعمال نظام. في نموذج هذه الفئة، يتم إطلاق حدث ما كل مرة يتم عمل تغيير لراتب الموظف. تنجز العبارة **RaiseEvent** الإطلاق الفعلي للحدث، مخصصة الاسم للحدث ليتم إطلاقه، و مجموعة من المعاملات النسبية في الأقواس.

```
Public Class Employee
    Public Name As String
    Private currentSalary As Decimal
    Public Property Salary() As Decimal
    Get
        Return currentSalary
    End Get
    Set(ByVal value As Decimal)
        currentSalary = value
        RaiseEvent SalaryChanged(currentSalary)
    End Set
End Property
Public Event SalaryChanged(ByVal NewSalary As Decimal)
End Class
```

لا تتم إضافة معالجات الحدث بشكل مباشر إلى الفئة. عوضاً عن ذلك، تتم إضافتها إلى حالة ما للفئة. الحالة، مصرح عنها كحقل فئة ما، يجب أن يتم تعريفها استخدام الكلمة المحجوزة **WithEvents** الخاصة، التي تخبر الفيجوال بيسك أن هذه حالة سوف تعالج الأحداث.

```
Public WithEvents MonitoredEmployee As Employee
```

معالجات الحدث هي إجراءات فرعية مألوفة، ولكنها تتضمن الكلمة المحجوزة **Handles** لتشير أي حدث تتم معالجته.

```
Private Sub EmployeePayChanged(ByVal updatedSalary As Decimal)Handles
MonitoredEmployee.SalaryChanged
    MsgBox("The new salary for " & MonitoredEmployee.Name & " is " & updatedSalary)
End Sub
```

كل ما يحتاجه هو شيء ما لاستئناف الحدث.

```
Public Sub HireFred()
    MonitoredEmployee = New Employee
    MonitoredEmployee.Name = "Fred"
    MonitoredEmployee.Salary = 50000 ' Triggers event
End Sub
```


عندما يكون الراتب مجموعة ، تطلق خاصية Salary للفئة Employee الحدث SalaryChanged مستخدمة أمر الفيجوال بيسك RaiseEvent. و هذا يولد استدعاء ما لمعالج الحدث EmployeePayChanged، الذي يعرض بالنهاية الرسالة . الأحداث المبينة داخل الفئات Windows Forms في الدوت نت NET.تعمل تماماً مثل هذا ، ولكن بدلاً من أن تراقب معي زيادة الراتب . فإنها تترقب نقرات الفأرة و ضربات لوحة المفاتيح . تستخدم جميع أحداث النظام هذه توقيع معاملات نسبية ما مشترك.

`Event EventName (ByVal sender As System.Object, ByVal e As System.EventArgs)`

يطابق المعامل النسبي المرسل sender الحالة للكائن الذي يطلق الحدث ، في حال احتاج المستدعي اختبار أعضائه. إن المعامل النسبي e هو كائن ما يتيح للمستدعي أن يرسل بيانات خاصة بالحدث event-specific إلى المعالج من خلال حالة فئة وحيدة . لا تملك الفئة System.EventArgs الكثير على طول مع الأعضاء ، و لكن العديد من الأحداث تستخدم فئة بديلة و التي تشتق من System.EventArgs.

فضاءات الأسماء Namespaces

الفئات و التراكيب و الوحدات البرمجية و العدادات و الواجهات و التفويضات . أنواع الدوت نت الرئيسية . لا تعوم تماماً فقط في كود تطبيقك . و عليها جميعاً أن تصنف و تدار في فضاءات أسماء namespaces، يزود فضاء الأسماء هرم ما لأنواعك ، جزء من شقة مملوكة في بناية شجرية الشكل tree-shaped حيث يملك كل نوع بيت ، بعض هذه البيوت (أو العقد) ، مثل النظام System ، تحصل على مجموعة كبيرة مناسبة مع جميع أنواع هذه العائلات الساكنة هناك . الآخرون ، مثل System.Timers ، ربما تملك بضعة أنواع تسكن في مساكنها الكافية . ولكن كل نوع يجب أن يعيش في الهرم.

في جذر الهرم ذاته يوجد Global ، و ليس عقدة بذاته ، و لكن كلمة فيجوال بيسك محجوزة والتي تشير إلى جذر كل الجذور . بإمكانك أن تدخل Global عند إرجاع فضاءات الأسماء الخاصة بك ، ولكن استعمالها مطلوب فقط ومتى تتركها فسوف تسبب فرضي بين فرعي فضاء أسماء . بشكل مباشر تحت Global يوجد فضاءات أسماء عالية المستوى top-level قليلة ، متضمنة System و Microsoft. يحوي كل فضاء أسماء عالي المستوى top-level فضاء أسماء ثانوي ، و كل من تلك تحوي فضاءات أسماء مستوى ثالث third-level إضافية ، و هكذا . يتم إرجاع عقد فضاء الأسماء بنسبة أحدها إلى الآخر مستخدماً ترميز النقطة "dot" .

`System.Windows.Forms`

هذا يحدد فضاء أسماء Forms على المستوى الثالث . بإمكانك أيضاً أن تكتب :

`Global.System.Windows.Forms`

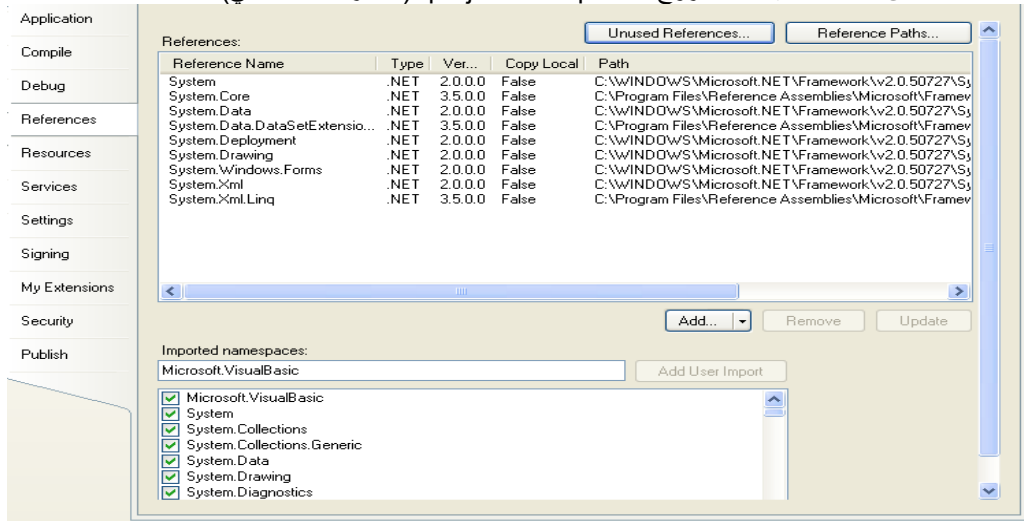
الذي يعني نفس الشيء . أيضاً يتم دعم فضاء أسماء نسبي:

`Forms`

على أية حال لاستخدام فضاءات أسماء نسبي ، عليك أن تخبر كود الفيجوال بيسك لكي يتوقعها . يوجد فضاءات أسماء عديدة هناك ، و ربما يوجد أماكن فضاءات أسماء Forms عديدة في الهرم.

إسناد مؤشر إلى فضاءات الأسماء Referencing Namespaces

قبل أن يتم استخدام فضاءات الأسماء في كودك ، كان من الواجب إسناد دليل لها أو مرجع referenced وجلبها imported بشكل اختياري . يعين إسناد دليل أو مرجع إلى فضاء الأسماء ملف المجمع DLL الذي يحوي أنواع فضاءات الأسماء تلك. اعمل كل من هذين الفعلين من خلال التتويب "مراجع References" من نافذة خصائص المشروع project's Properties. (انظر الشكل التالي)



في الحقيقة ، إنك لم تشير إلى فضاءات الأسماء في DLL ، ولكن بالأحرى الأنواع ، و جميعها يحدث لأن تقطن في فضاء أسماء خاص . على أية حال ، للنوع DLLs الرئيسي المزود بإطار عمل الدوت نت ، إنه ضمناً نفس الشيء . في الحقيقة ، حتى الميكروسوفت سمت العديد من ال DLLs لمجارات فضاءات الأسماء التي تتضمنها. يتضمن System.dll أنواع ضمن فضاء الأسماء System. تتضمن System.Windows.Forms.dll أنواع معينة لتطبيقات Windows Forms ، و تظهر كل هذه الأنواع في فضاء الأسماء System.Windows.Forms أو أحد فروعها . إذا لم ترجع DLL في مشروعك ، فإنه ولا شيء من أنواعها سيكون متوفراً لديك في كودك . تحمل الفيجوال استديو مراجع عديدة داخل مشروعك بشكل آلي مبينة على نوع المشروع الذي تنشئه أنت . و يظهر الشكل السابق المراجع الافتراضية التسعة المحتواة ضمن تطبيق Windows Forms وهي : System.Core، System.Data، System.Data.DataSetExtensions، System.Deployment، System.Drawing، System.Windows.Forms، System.Xml، System.Xml.Linq، و System.Windows.Forms.

حالما تعمل على إرجاع مكتبة فئات ما (أو أنواع أخرى) في كودك ، فإنك تتمكن من الوصول لأي من فئاتها بكتابة الاسم الكامل لفضاء أسماء تلك الفئة . على سبيل المثال ، الفئة للفرمات التي تكون على الشاشة on-screen يتم إرجاعها بواسطة System.Windows.Forms.Form. وتلك ثلاث مستويات إلى أسفل الهرم ، وبعض الفئات هي أعمق حتى .

للتجنب كتابة جميع فضاءات الأسماء الطويلة هذه عدة مرات متكررة ، فإنه تتضمن الفيجوال بيسك ميزة imports. حالما يتم جلب فضاء الأسماء ، فبإمكانك أن تصل لأي من الأنواع في فضاء الأسماء بدون تحديد اسم فضاء الاسم . إذا جلبت فضاء الأسماء System.Windows.Forms ، عليك فقط أن تكتب "Form" لتدخل الفئة Form. يظهر النصف السفلي من الشكل السابق كيفية ترتيب هذه الواردات من خلال خصائص المشروع . إن القائمة "Imported namespaces" تظهر جميع فضاءات الأسماء المرجعية المتوفرة . ببساطة ضع علامة صح على الفضاءات التي ترغب أن تجلبها لقد تم وضع علامة صح على System.Windows.Forms مسبقاً وبشكل افتراضي في تطبيقات Windows Forms. بإمكانك أيضاً أن تجلب فضاء أسماء مباشرة في كودك المصدري . استخدم العبارة Imports من بداية ملف الكود المصدري :

```
Imports System.Windows.Forms
```

تدعم العبارة Imports اختصارات فضاء أسماء ، أسماء قصيرة و التي تمثل فضاء الاسم الكامل في كودك . باستخدام العبارة :

```
Imports Fred = System.Windows.Forms
```

تتيح لك إرجاع الفئة Form مثل "Fred.Form." و ليس مثل قائمة الإيرادات في خصائص المشروع ، التي تؤثر على كامل المشروع ، تؤثر العبارة Imports في ملف كود مصدري وحيد والمكتوبة فيه فقط .

فضاءات الأسماء في مشروعك Namespaces in Your Project

بشكل افتراضي ، تظهر جميع الأنواع و الفئات في مشروعك في مستوى فضاء أسماء أعلى top-level والذي يواجه اسم مشروعك . من أجل تطبيقات Windows Forms ، و يدعى فضاء الأسماء الافتراضي هذا ب WindowsApplication1. ولتحديد مستوى فضاء أسماء أعلى مختلف top-level ، غير من خلال المؤشر Application لخصائص المشروع ، في الحقل "Root namespace" . و تظهر جميع الأنواع في مشروعك في فضاء الأسماء هذا : إذا حددت فضاء أسماء مزود من قبل ميكروسوفت Microsoft-supplied موجود كفضاء أسماء جذري لمشروعك ، سوف تظهر جميع فضاءات الأسماء في مزيج فضاء أسماء مخصص في الأنواع الموجودة مسبقاً . من أجل التطبيقات المستقلة ، هذا المزيج سوف يكون واضحاً فقط من كودك .

من فضاء الأسماء الجذر ، يمكنك أن تضع أنواع ضمن فضاءات أسماء ثانوية باستخدام العبارة Namespace و Namespace هي عبارة مقطوع ما تنتهي بالعبارة End Namespace. سوف يتم احتواء أي أنواع أنشأتها بين العبارات Namespace و End Namespace في فضاءات الأسماء الثانوية تلك. و مثلاً على ذلك ، إذا كان فضاءك الأسماء الجذر هو WindowsApplication1، فإن العبارات التالية تنشئ فئة ما و التي اسمها الكامل هو WindowsApplication1.WorkArea.BasicStuff.BusyData :

```
Namespace WorkArea.BasicStuff
```

```
Class BusyData
```

```
...
```

```
End Class
```

```
End Namespace
```

بإمكانك أن تدخل كعبارات فضاء أسماء متعددة في كودك كما تدعي الحاجة . ويتم تزويد ذلك بإدخال فضاءات أسماء أيضاً :

```
Namespace WorkArea
```

```
Namespace BasicStuff
```

```
Class BusyData
```

```
...
```

```
End Class
```

```
End Namespace
```

```
End Namespace
```

فضاء الأسماء الخاص بي The My Namespace

قدمت الفيجوال بيسك 2005 فضاء أسماء جديد "My" في مستوى أعلى top-level ، مصمماً لتبسيط مهام برمجية مشتركة . وأضافته الميكروسوفت إلى اللغة جزئياً لترسم بسالة عرض فيجوال بيسك الإصدار 6.0 داخل ملف الدوت نت . و لكن معظمه ليس حقيقياً حيث أنه تمثيلي . يجمع My عموماً ميزات مستخدمة والتي تنشر حالياً حول مكتبة فئة إطار العمل (FCL) Framework Class Library (FCL) ، و تضعها في هرم صغير mini-hierarchy للدخول المناسب . إنها بالفعل ليست أكثر تعقيداً من ذلك . و يتم تنظيم الهرم بإحكام ، مع أقسام المستخدم ، التطبيق ، و معلومات الكمبيوتر الخاصة . يتم استخدامه تماماً كأي جزء آخر من إطار العمل ، رغماً من أنك لا تستخدم الكلمة المحجوزة Imports لتدخل عناصرها بطريق نسبية . على أية حال ، إن My سهلة جداً للاستخدام ، لتظهر عدد الإصدار لتطبيقك ، على سبيل المثال ، استخدم العبارة التالية :

```
MsgBox (My.Application.Info.Version.ToString)
```

تكون بعض مناطق فضاء الأسماء My ديناميكية : و تضاف فئات أو تنقل عندما تعدل كودك المصدري . في تطبيقات Windows Forms ، يتضمن الفرع My.Forms مداخل لكل صيغة من صيغ المشروع . عندما تضيف صيغاً جديدة ، عندها تتم إضافة مدخلات جديدة بشكل أوتوماتيكي . ثم يصنع المشروع My.Forms عنوان المرجع لكل صيغة متاحة للاستخدام في كودك .

```
My.Forms.Form1.Text = "Welcome"
```

مشروع المكتبة: The Library Project

يعمل هذا المشروع على إدارة قاعدة بيانات كتب وبنود وسائط أخرى، ويتحكم بكيفية نقل هذه البنود بين رفوف الكتب والزبائن. يحتاج البرنامج أن يمتلك ميزات خاصة بالزبائن والإدارة. وسيضمن تقارير متنوعة، بما فيها طباعة فاتورة بنود المخرجات للزبون. وأهم من هذا كله، يحتاج أكواد الطباعة وقراءة تعريف الأدوات (الوسائط المركبة على الكمبيوتر)، إنه يبدو كثير على أن يقوم بعمله شخص واحد فقط، وإنه مشروع قابل للزيادة في الحجم. ولكن لن أقوم بعمله بنفسه، سوف تساعدني. فمع بعضنا البعض، ومن خلال صفحات هذا الكتاب، سوف نصمم كلانا هذا البرنامج، ونطور كوده، وسنجلب المتعة للمستخدمين. ما تبقى من مستندات هذا الفصل هي الميزات الرئيسية لتطبيق إدارة المكتبة.

ميزات بند المكتبة: Library Item Features

نظام المكتبة سيعمل على إدارة بيانات مفصلة بالكتب وبنود الوسائط الأخرى، لإيجاد هذه البيانات، وإدارة وترتيب التفاصيل والحالات بالنسبة لكل نسخة من أي بند. ولجعل هذا حقيقة، فإن برنامج المكتبة سيعمل التالي:

- ✓ يسمح للمدراء أو الزبائن من البحث عن بنود حالية في البيانات المفصلة. سيسمح البرنامج بالبحث بالاعتماد على العديد من الخصائص المختلفة لكل بند.
- ✓ دعم طرق البحث المتنوعة، من ضمنها بواسطة العنوان، بواسطة اسم المؤلف، بواسطة الموضوع أو تصنيف العنوان (مثل كتب علوم الإنسان...)، أو بواسطة كلمات مفتاحية متنوعة، أو بواسطة اسم دور النشر التي لها علاقة، أو بواسطة اسم الرقم المسلسل أو المجموعة التي تحتوي ذلك البند، أو بواسطة رقم كود التعريف المرتبط بالبند الحقيقي.
- ✓ تحديد نتائج البحث بواسطة موقع البند، أو بواسطة نوع الوسائط (كتاب، سيدي، ديفيدي، إلخ).
- ✓ دعم التعريف واستخدام المواقع المادية المميزة. فالزبون لديه كتب ووسائط تخزين في ثلاث مواقع مختلفة ضمن البناء، ومن ضمنها خزانة تخزين صغيرة للبنود النادرة الاستخدام.
- ✓ عرض تفاصيل البند المستخرج ضمن واجهة نمطية استعراضية معروفة (شائعة). على سبيل المثال، عند البحث عن كتاب بواسطة العنوان، ينقر المستخدم على اسم المؤلف للوصول إلى جميع البنود الأخرى لذلك المؤلف.
- ✓ السماح بالوصول إلى كل بند مكتبي من خلال مسح كود تعريف ما. كما هو شائع مع معظم مكتبات هذه الأيام، فالبنود في مجمع هذه المكتبة لديها ملصقات أكواد تعريفية. والتي تخدم كعرف وحيد ومفرد لكل نسخة لبند مستقل.

ميزات الزبون Patron Features

بالإضافة إلى الكتب والبنود الأخرى، يدير البرنامج قائمة من الزبائن، "زبائن" المكتبة والذين يتم الترخيص لهم بتفحص (أو مراجعة) بنود المخرجات. لدعم التفاعل مع الزبائن، سيضمن التطبيق هذه الميزات الخاصة بالزبون:

- يمكن للزبائن من مراجعة البنود دون مساعدة المدراء وأمناء المكتبة، فبإمكانهم استخدام مسح كود التعريف لمسح بنود المكتبة وكرت مكتبة الزبون وبنود المكتبة.
- يمكن أن يتم تفحص بنود الزبائن، ومراجعتها في فاتورة المكتبة.
- يتم إسناد رقم خاص "PIN" لكل زبون والذي يتصرف ككلمة مرور.
- تحدد أنواع الوسائط "لبند ما فترة (عائداته) أو مخرجاته checkout (ولاحقاً عملية إعادة تجديده أو تحديثه renewal)
- يمكن للعملاء من عرض سجل المكتبة الخاص بهم، ومن ضمنه كل الكتب المخرجة حالياً، وقائمة الغرامات المدينين بها للمكتبة.
- إذا رخص على بند معين، فبإمكان الزبون تحديث بند ما قد قام بمراجعتها حديثاً.
- المساعدة المركزية على الشبكة للزبون تكون متاحة من خلال المفتاح F1 القياسي. وملف المساعدة يتضمن معلومات غير الميزات الإدارية، مما يقلل من التجريب experimentation .
- يمكن تقسيم الزبائن ضمن "مجموعات الزبائن" من أجل التقارير والمعالجات المريحة (المناسبة) لطاقتهم الإدارية. administrative staff .

الميزات الإدارية: Administrative Features

من ضمن المدراء أمناء المكتبات، الهيئة الإدارية، وأشخاص آخرين بحاجة لإمكانية وصول متقدمة لميزات التطبيق. وهم المستخدمون الرئيسيين للنظام، وليس العملاء. يتضمن التطبيق ميزات خاصة بالمدراء التالية:

- توفر ميزة "تسجيل الدخول login" الوصول إلى الميزات الإدارية للتطبيق. يمكن فقط للمستخدمين المفوضين من تسجيل الدخول من خلال كلمة المرور المخصصة. وميزة تسجيل الدخول عادة تكون مخفية عن المشاهدة بالنسبة للعميل العادي.
- يمكن للمدراء من مشاهدة تفاصيل العملاء تماماً كما يشاهدها العميل، ولكن لديهم أيضاً إمكانية الوصول إلى تفاصيل العملاء الإضافية. وعلى وجه الخصوص بإمكان المدراء إضافة عملاء جدد وإدارة ذاتيهم وتفاصيل دراسية أو إحصائية. وبإمكان المدراء أيضاً تعطيل تسجيل عميل ما لمنع مخرجات بنود أكثر.
- يجمع ويدير المدراء الغرامات المالية fines الخاصة بالعملاء، ومن ضمنها إمكانية إضافة غرامات غير قياسية أو صرف الغرامات غير المدفوعة unpaid fines .
- يحدد المدراء سجلات كل بند والمدارة بواسطة قاعدة بيانات فاتورة النظام. وهذا يتضمن قواعد كل بند، مثل العنوان والمؤلفين. فكل بند يتضمن نسخة واحدة أو أكثر، والتي تمثل البنود المادية والتي يمكن مراجعتها ويتم إسناد أكواد التعريف للنسخ.
- إلى جانب البنود والنسخ، يحدد المدراء جميع القوائم والقيم المدعومة، ومن ضمنها أسماء المؤلفين والتصنيفات، قائمة أنواع الوسائط، الناشرين، أسماء الكتب المتسلسلة، أكواد الحالة والتي تحدد ترتيب نسخة كل بند، والموقع.
- المدراء المختارين (الخصوصيين) بإمكانهم إضافة، تحديث، أو إزالة حسابات مدراء آخرين. يتضمن كل حساب ميزة إعدادات تفويض خاصة (مجموعة الحقوق).
- بالإضافة إلى مسح أكواد التعريف، يمكن للبرنامج من مساعدة المدراء في تصميم وطباعة أكواد تعريف البنود والعملاء.
- تسمح عملية إدارة البرنامج البسيطة لهيئة الإدارة من معالجة البنود المتأخرة الدفع والغرامات بالاستناد على قواعد نظامية.

- يسمح التطبيق من أن يتم إضافة العطل وحفظها. فعندما يراجع عميل ما كتاب، يضبط البرنامج تاريخ المستحقات للبند لتجنب العطل.
- يوفر مركز التعليمات الخاص بالإدارة مساعدة لتطوير ميزات التطبيق من خلال نفس المفتاح المتاح للعملاء.
- يتضمن التطبيق بعض التقارير الإدارية الأساسية، والمقدرة على ربط "plug in" التقارير عند الحاجة في المستقبل دون الحاجة لتحديث البرنامج نفسه.

التطبيق ككل: The Application As a Whole

- إلى جانب الميزات الأساسية للبرنامج التي تم اختبارها بواسطة العملاء والمدراء يوجد العديد من المتطلبات الأخرى:
 - البرنامج سهل الاستخدام بالنسبة للمستخدم وسهل الاستعراض، وخاصة للعملاء، دون الكثير من التجريب والتدريب أو training أو المساعدة.
 - يخزن التطبيق بياناته في قاعدة بيانات مخدم سكول SQL Server database.
 - نشر التطبيق يتم عمله بواسطة هيئة الإدارة والتي يكون لديها امتيازات إدارية محلية، لذلك فإن حزمة تنصيب ويندوز تكون كافية.
 - يستخدم تركيب (إعداد) التطبيق طرق XML القياسية.
- ماعدًا هذه المتطلبات العامة والميزات الخاصة، فقد عملت على منح تصميم حر، ولكن من أين سنأتي قائمة المتطلبات؟ إنها سنأتي من المستخدمين، أصحاب التطبيق. فهو يلبي احتياجاتهم احتياجات زبائني، الذين سيستخدمون التطبيق يوم بعد يوم- وهذا ما يحدد قائمة من المتطلبات.

احتياجات المستخدمين The Needs of the Users

بالعودة إلى أيام الكمبيوتر الأولى، فلم يكن هناك مستخدمين، فمن يحتاج لمستخدمين؟ للمستخدمين الكثير من الحاجيات، ومعظمها لا يمكن سدها بواسطة الكمبيوتر. ولكن بالنسبة للتي يمكن عملها، تأتي الحاجيات في خمسة أقسام: البيانات والمعلومات، العمليات، إمكانية الاستخدام، العموميات، والحاجيات الخاصة بالمشروع. تتضمن عملية التصميم اختبار هذه الحاجيات والمراسلات الناتجة عن هذه الحاجيات ضمن البرنامج المنتج. بتفحص البيانات الحالية والإجراءات، إجراء مقابلات مع المستخدم، وإتمام طرق استخلاص الحاجات الأخرى، فأنت تجمع التفاصيل التي تحتاجها لنحت الحل المناسب.

البيانات والمعلومات Data and Information

إن مقدرتك على توفير إمكانية وصول خاصة ومناسبة للبيانات والمعلومات المطلوبة من قبل المستخدم هو ما يجعلك مبرمج محبوب جداً. معظم المستخدمين يأخذون وقت طويل لكي ينجسوا تماماً مع الكمبيوتر. فهم يحفظون معلوماتهم على 3-5 كروت تصنيف، أو لباداة أو لفافة ورقية أو مخطوطات ورقية أو ما شابه، وبالتالي فلديهم سبب ما للانتقال إلى الكمبيوتر كوسيلة تخزين. وهي أنه وسيلة مريحة. البيانات هي المعلومات الصرفة (المجردة) والمخزنة وبواسطة برنامجك: أسماء، أعداد، صور، أو أي قيمة مستقلة قائمة بذاتها. أما المعلومات فهي البيانات في سياق: سجل الزبون، أو في ترتيب معين، مظهر المنزلقات (طبقات معروضة على سطح ما). عندما توفر برنامج نوعي يرتقي بالبيانات إلى مستوى المعلومات، فأنت توفر مستوى مريح ومناسب لاحتياجات المستخدمين.

المعالجة Process

عندما يطلب المستخدم استرجاع بياناته من الكمبيوتر، فلديك ثلاث خيارات:

- ✚ إفراغ كل بايت مفرد من البيانات إلى الشاشة، الطباعة، أو الديسك، وتدع المستخدم يصنفها، عملياً، هذا هو النظام الذي لدى بعض المستخدمين قبل أن يبذروا باستخدام الكمبيوتر.
- ✚ حماية البيانات من يتمكن المستخدمين من الوصول إليها، بالتشديد أو الإصرار على تزويد كلمة مرور صحيحة أو تاريخ الانتهاء، أو أن تكون البيانات غير متاحة ولا يمكن الوصول لها. من خلال "إلغاء الأمر، إعادة المحاولة، فشل". عملياً، هذا هو النظام الذي لدى بعض المستخدمين الآخرين قبل البدء باستخدام الكمبيوتر.
- ✚ جلب البيانات كمعلومات، على هيئة قابلية الاستخدام والوصول.

على الرغم من أن كلا الخيارين الأول والثاني هما عملياً تجريبين، فالخيار الثالث هو الأفضل. وكمية البيانات الممنوحة والتي من المحتمل أن يعمل تطبيقك على إدارتها، فسيكون عليك توزيع التفاعل معها بت في كل مرة، ويتتابع مناسب. وهذه هي المعالجة process. من خلال تنفيذ معالجة محققة، فأنت تتحكم ليس فقط ببيانات المستخدم، ولكن أيضاً التفاعل المنهجي مع البيانات. معظم المستخدمين بحاجة إلى مزود أو استخلاص فقط جزء صغير من البيانات كل مرة. ولكن عندما يعملون ذلك، فسيكون هذا ضمن سياق بعض المعالجات. على سبيل المثال، فلنشغل موقع أو مكان، فالمستخدم: (1) يعمل على إدخال وتأكيد معلومات اتصال الزبون، (2) إدخال أو تحديث تفاصيل الطلب، و(3) يطبع أو يتصل إلكترونياً بمعلومات الطلب لكي يصبح تام. فتطبيقك يدير هذه المعالجة للخطوات الثلاث.

قابلية الاستخدام Usability

إذا كان برنامجك يحضر البيانات والمعلومات للمستخدم وبترتيب معين أو بطلب معين، ولكنه صعب الاستخدام فسيكرهك المستخدم. وسوف يشمنزون loathe منك. وسينسجون قصص حولك، سواء كانت حقيقية أم لا.

كمبرمج، فإن عملك أن تجعل الكمبيوتر والبرمجيات التي تعمل عليه، قابلة للاستخدام قدر الإمكان. وعلى الرغم من عدم مقدرتك على التحكم بالعديد من الميزات الأساسية للنظام، فتبقى أنت الملك عندما يتعلق الأمر بالبرمجيات الخاصة بك. كلما كانت السهولة وقابلية الاستخدام في تصميم برنامجك، كلما كان المستخدم أكثر سعادة. ولكن يجب علي أن أذكرك، فكلما كانت السهولة أكبر للمستخدم كلما كان هذا يعني العمل أكثر على التطوير (البرمجة). دائماً، أقدم لك النصائح التالية فيما يخص أي برنامج:

- ❖ لا تضع الكثير من المعلومات في نموذج واحد. وعندما ترتاب، أنقل بعض المعلومات إلى نموذج آخر.
 - ❖ قم بجلب المعلومات والبيانات الضرورية فقط للمستخدم بشكل افتراضي، واعررض المعلومات الإضافية فقط إذا طلبها المستخدم.
 - ❖ أجعله سهل من أجل المستخدم لكي يصل ويحدث البيانات، ولكن اسمح للبرنامج اسمح للبرنامج من العمل بشكل مناسب دون الحاجة لهذه البيانات.
 - ❖ استخدم النصوص، الصور، والألوان فيما ينفع المستخدم.
 - ❖ بسط التطبيق بحيث تبدو مستندات المستخدم (مستندات المساعدة أو المعلومات) غير ضرورية.
 - ❖ دائماً وفر مستندات المستخدم. وأجعلها بسيطة بشكل كافي بحيث يصبح الاتصال من أجل الدعم الفني غير ضروري.
- هذه القواعد عامة وكافية للعمل مع أي نوع من التطبيقات، وهي من وجهة نظرك كافية بالنسبة لنا كمبرمجين وهامة لهم كمستخدمين.

العموميات (الخصائص المشتركة) Commonality

تتطلع touts ميكروسوفت باستمرار لتجديد (الابتكار innovation)، والمقدرة على الابتكار قد نقلت المنتجات البرمجية للأمام بخطوة هائلة. ولكن لسوء الحظ، على المستخدمين إمكانية معاملة فقط الكثير من الابتكار كل مرة. بالنسبة للبرمجيات: فحتى مع البرامج المبتكرة والجديدة يجب المحافظة على بعض العموميات مع نظام التشغيل، وبما يتوافق مع البرامج الأخرى التي يتم تثبيتها على الجهاز، وكما تتحدث مع المستخدمين حول احتياجاتهم وتفكر بأكثر تقدم في تقنية البرمجيات والتي ستعمل على توفيرها، فلا تنسى العموميات. كما أنك لا تنسى المتطلبات الجوهرية للمستخدمين: فلا تغمر المستخدمين بالطرق الجديدة لعمل المهام التي يفكرون بأنه يتم عملها بطرق تقليدية دائماً يقومون بها بنفس الأسلوب، فما يحتاجه المستخدمون هو الاستقرار consistency في طريقة الأداء.

المشروع Project

يحتوي دليل هذا الفصل الملفات التالية:

مستند اتفاقية المشروع Project Agreement.doc

هذا هو مستند المشروع الرئيسي والذي يحدد ميزات المشروع الكامل. وتحظى الاتفاقية بتمثيل كل من المطور والمستخدم. والخروج عن هذه الوثيقة يحدث من خلال معالجة "طلب التغيير Change Order" فقط.

مستند طلب التغيير Change Order.doc

يتم استخدام هذا المستند لتعديل المشروع الأصلي من خلال معالجة "طلب التغيير". عند استخدام هذه الوثيقة، فهي تتضمن وصف للتغيير الذي يجب عمله على المشروع، وأي جدولة وتكاليف سوف تتأثر.

مستند قبول المشروع Project Acceptance.doc

يتم استخدام هذه الوثيقة عندما يكتمل المشروع. والمستخدم جاهز لقبول المنتج النهائي. وهذا المستند يضم عناصر "اختبار معيار القبول Acceptance Criteria Testing" و"قبول المشروع" المشروحة مسبقاً في هذا الفصل. ولك الحرية التامة في استخدام هذه الوثائق لدعم مشروعك الخاص.

تصميم قاعدة البيانات Designing the Database

إذا كان لديك بيانات فتحتاج لوضعها في مكان ما. وهل هناك مكان أفضل من وضعها في قاعدة بيانات؟ قاعدة البيانات هي تجمع من البيانات المنظمة لسهولة استخراجها بواسطة الكمبيوتر.

قواعد البيانات العلائقية: Relational Databases

تعمل قواعد البيانات العلائقية على تجميع البيانات في جداول tables، وكل منها يخزن مجموعة خاصة من السجلات records غير المرتبة. من أجل الراحة، يتم إحصار الجداول كشبكة من قيم البيانات، حيث أن كل صف row يمثل سجل وحيد وكل عمود column يمثل حقل field ثابت والذي يظهر في كل سجل. يمثل الجدول التالي جدول طلبيات، مع سجلات منفصلة السطور فكل سطر يمثل بند طلبية.

Record ID	Order ID	Customer ID	Name	Product ID	Product	Price	Quantity
92231	10001	AA1	Al Albertson	BEV01COF	Coffee	3.99	3
92232	10001	AA1	Al Albertson	BRD05RYE	Rye bread	2.68	1
92233	10002	BW3	Bill Williams	BEV01COF	Coffee	3.99	1
92234	10003	BW3	Bill Williams	BEV01COF	Tea	3.99	2
92235	10004	CC1	Chuck Charles	CHP34PTO	Potato chips	0.99	7

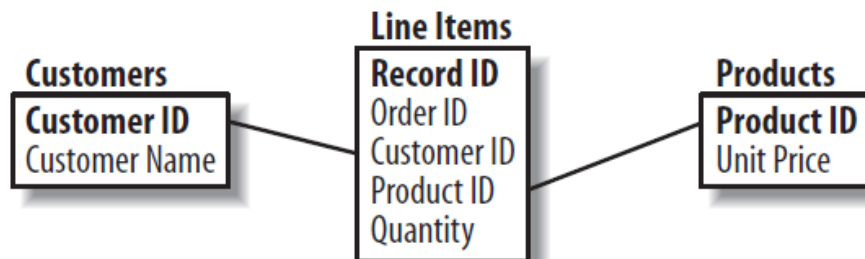
إن وضع جميع معلوماتك في جدول مناسب تماماً. فالبيانات الهامة تظهر من خلال نظرة سريعة في ترتيب مناسب وجميل، ومن السهل ترتيب النتائج بالاعتماد على عمود معين. لسوء الحظ، هذا الجدول من الطلبيات لديه الكثير من أسماء الزبائن Customer ID المكررة وأسماء المنتجات Product تتكرر أيضاً عدة مرات. وبالإضافة إلى معرف المنتج product ID، فيشير BEV01COF إلى قهوة، وواحد من أسطر القائمة يشير إلى شاي، والعديد من المشاكل تكون ملازمة للبيانات الموضوعية في ملف flat file وحيد لجدول قاعدة بيانات database table. وبالتالي كان الحل فيما يسمى بالتطبيع "normalization". وذلك بتقسيم البيانات ضمن جداول منفصلة مع مجموعات جزئية من البيانات، يؤدي إلى إسناد معرف وحيد ومفرد لكل record/row سجل/صف في كل جدول (مفتاح رئيسي a primary key) وعمل العديد من التعديلات الأخرى، ويمكن أن يتم تطبيع البيانات من أجل كل من فعالية المعالجة وتكامل البيانات data integrity. فمن أجل الطلبيات البسيطة في الجدول السابق، يمكن أن يتم تطبيع البيانات ضمن ثلاث جداول منفصلة: واحد من أجل بنود أسطر الطلبيات، وجدول آخر من أجل الزبائن، وواحد آخر من أجل المنتجات كما في الجداول التالية وعلى الترتيب respectively. ففي كل جدول قمت بوضع علامة نجمة asterisk بجانب عنوان العمود الذي يتصرف كعمود مفتاح رئيسي.

Record ID *	Order ID	Customer ID	Product ID	Quantity
92231	10001	AA1	BEV01COF	3
92232	10001	AA1	BRD05RYE	1
92233	10002	BW3	BEV01COF	1
92234	10003	BW3	BEV01COF	2
92235	10004	CC1	CHP34PTO	7

Customer ID *	Customer Name
AA1	Al Albertson
BW3	Bill Williams
CC1	Chuck Charles

Product ID *	Product Name	Unit Price
BEV01COF	Coffee	3.99
BRD05RYE	Rye bread	2.68
BEV01COF	Coffee	3.99
CHP34PTO	Potato chips	0.99

فلضم البيانات من جداول متعددة مباشرة، اربط (أوصل) الحقول المتطابقة. على سبيل المثال، بإمكانك وصل حقل Customer ID في جدول البنود المسطرة مع حقل المفتاح الرئيسي في جدول الزبائن. حالما يتم الربط، فتفاصيل سجل بند سطر مفرد مربوط يمكن جلبه بواسطة الربط الكامل لأسم الزبائن customer name. وهو نفسه بالنسبة للربط المباشر مع أي جدولين والذين لديهما حقول قابلة للربط أو الوصل. يبين الشكل التالي العلاقات الخطية بين جداول الزبائن customer، والمنتجات product، و الطلبية order line.



يبين الشكل ثلاث جداول مرتبطة ولكن ما تزال تعمل كأنها جدول مفرد.

لربط الجداول ببعضها، تنفذ قواعد البيانات العلائقية لغات الاستعلام *query languages* التي تسمح لك من معالجة البيانات باستخدام الجبر العلائقي *relational algebra* (والذي اشتق منه مصطلح قاعدة البيانات العلائقية)، والأكثر شيوعاً لهذه اللغات هي، السكول SQL، والتي تستخدم اللغة الانكليزية البسيطة مثل الجمل للربط، والترتيب، التلخيص، واستخلاص قيم البيانات التي تحتاجها تماماً. والعبرة الرئيسية، هي "اختر SELECT"، والتي توفر اختيار بيانات قاعدية واستخلاص الميزات. ويوجد أيضاً ثلاث عبارات أخرى مشتركة وهي، إدراج INSERT، تحديث UPDATE، وحذف DELETE، تسمح لك هذه العبارات من معالجة السجلات المخزنة في كل جدول، فمع بعضها كل من هذه العبارات الأربع تكون: أوامر لغة معالجة البيانات *Data Manipulation Language (DML)* الرئيسية لسكول SQL.

تحتوي سكول أيضاً على عبارات لغة تعريف البيانات *data definition language (DDL)* والتي تسمح لك بتصميم الجداول المستخدمة لحفظ البيانات، بالإضافة إلى ميزات قاعدة البيانات الأخرى. وسأريك أمثلة عن عبارات سكول المتنوعة فيما بعد في هذا الفصل. تبسط الأنظمة الخاصة للبيع *Vendor-specific systems* مثل خادم ميكروسوفت سكول *Microsoft's SQL Server*، وأوراكل *Oracle's*، وميكروسوفت أكسس *Microsoft's Access* الميزات الجوهرية لكل من DDL و DML من خلال تحليل إضافي للبيانات وأدواتها الإدارية.

وتكافح هذه الأنظمة مرةً أخرى على ميزات هامة مثل تكرار البيانات *data replication*، تكامل البيانات المقاومة للانهار *crash-proof* و *data integrity*، السرعة التي تعمل بها الاستعلامات المعقدة على استرجاع النتائج المطلوبة، ومن لديه أكبر تدفق خاص.

سكول سرفر 2005 . SQL Server 2005

إن أداة قاعدة البيانات الرئيسية على مستوى العمل لميكروسوفت هي سكول سرفر *SQL Server*. وهي على خلاف ميكروسوفت أكسس (منتج آخر معروف لقواعد البيانات العلائقية من إصدارات ميكروسوفت والذي يأتي مع مجموعة أوفيس)، يتضمن سكول سرفر إدارة بيانات وميزات تحليل متقدمة.

إن الصفات الغالبة الآن على جميع إصدارات الفيچوال أستوديو احتواءها نسخة ما من سكول سرفر، حتى تلك الإصدارات المنخفضة (المتما). في إمكانك استخدام سكول سرفر ضمن تطبيقات الدوت نت الخاصة بك، ولكن سكول سرفر 2005 اليوم يتيح لك نحت (صناعة) الإجراءات المخزنة المضمنة باستخدام كود الدوت نت، على طول مع لغة المعالجة الإنشائية *T-SQL scripting* language الأصلية التقليدية.

اسم السكول سرفر *SQL Server* يدل ضمناً كمنتج "خادم server". فهو يعمل في خلفية النظام، ويتصل معك أنت المستخدم، فقط بامتلاكك على شبكة عمل *network* قياسية يتم تأسيسها لأول مرة *first establish* متصلة مع محرك الخدمة *server engine*. وهذا صحيح حتى ولو كان محرك خادم سكول مشغل على جهازك الخاص. فترقب منتج السرفر مثير كما هو الحال بالنسبة لقراءة كتب التدريس المتخصصة *tutorial books* بالفيچوال بيسك 2008 الأخرى والتي تتجنبها على نحو واسع، لذلك فإن ميكروسوفت توفر أدوات عميل متنوعة تتيح لك إدارة قواعد البيانات، الجداول، وخصائص قاعدة البيانات الأخرى، بالنسبة لي لم أثبت أي من إصدارات السكول ما عدا الأدوات التركيبية التي يتم تنصيبها مع نسخة الفيچوال أستوديو 2008 الاحترافية.

على الرغم من ذلك ماتزال ميكروسوفت مستمرة في تحديث وبيع ميكروسوفت أكسس، ومن الموصى به أكثر وأكثر من أجل المطورين المحترفين استخدام وتوزيع قواعد البيانات في هيئة مخدم سكول. وحتى أن ميكروسوفت ترخص لك إعادة توزيع سكول سرفر 2005 بطبعته السريعة مع تطبيقك. ولعمل هذا، عليك أولاً امتلاك "ترخيص سكول سرفر 2005 بطبعته السريعة والقابلة لإعادة التوزيع" من ميكروسوفت. ولحسن الحظ فهي مجانية ويمكنك امتلاكها من موقع الانترنت التالي:

<http://www.microsoft.com/sql/express>

سكول (لغة الاستعلام الهيكلية) *structured query language (SQL)*

تولي العمل في اليابان سهل جداً، بمجرد إتقانك اللغة. ونفس الكلام صحيح فيما يخص خادم سكول: فهو سهل جداً من أجل معالجة وإمكانية الوصول للبيانات، حالما تتقن اللغة. ولكن في هذه الحالة، اللغة هي سكول SQL، أو لغة الاستعلام الهيكلية *Structured Query Language*. فلقد أصبحت سكول المعبر القياسي في صناعة قاعدة البيانات.

يشرح هذا الفصل عبارات *DDL* و *DML* والتي هي أكثر منفعة بالنسبة لتطوير برنامج المكتبة. وستكون مسرور لمعرفة أن سكول ليست بهذا الحرص *picky* الكبير فيما يتعلق بهيئة العبارات المتنوعة. حيث أن الفروقات في حالة الأحرف الكبيرة والصغيرة *Upper- and lowercase distinctions* يتم تجاهلها، ف *SELECT* هي نفس *select* ونفس *SeLeCt*. (كود *T-SQL*) *Traditional SQL* سكول التقليدي يكون على الأغلب في الحالة الكبيرة) وإنني استخدم الحالة الكبيرة من أجل كل الكلمات المحجوزة، والحالة المدمجة من أجل الجداول *tables*، الحقول *fields*، والبنود الخاصة الأخرى.

مهما يكن فإن من المهم الثبات على منوال واحد. وإن توظيف المحارف الخاصة *whitespace* (مثل الفراغ، أو أي محارف لا تظهر في الطباعة) يعود إليك وكما تراه مناسب. فيمكنك وضع عبارات في سطر واحد طويل. أو وضع كل كلمة في سطر مفرد. الحالة الوحيدة التي تكون فيها المحارف الخاصة وحالة الأحرف (كبيرة أو صغيرة) مسألة مهمة في المتغيرات النصية للبيانات النصية الفعلية، فما تكتبه، هو ذلك الذي يبقى. تنتهي عبارات سكول عادة بالفاصلة المنقوطة *semicolon*، ولكن بعض الأدوات لا تتطلب إدخال الفاصلة المنقوطة *semicolon*، وأدوات أخرى تتطلب أن تضمنها. عندما تستخدم أدوات العميل المرئية لسكول سرفر (إدارة أستوديو وإدارة أستوديو السريع) فإن الفاصلة المنقوطة اختيارية، ولكن من الجيد تضمينها عندما تستخدم عدة عبارات مع بعضها البعض، عبارة بعد عبارة. وعبارات سكول المستخدمة في كود الفيچوال بيسك لا تتضمن أبداً الفواصل المنقوطة. فيما بعد عندما تنظر إلى صيغ سكول *SQL script* التي كتبتها، ستلاحظ الكلمة *GO* مرة بعد مرة، في السكول سرفر، هذا الأمر يقول "من أجل جميع العبارات الأخرى التي تظهر حتى الآن، تقدم للأمام وعالجها الآن *For all of the other statements that appeared so far, go ahead and process them now.*"

عبارات "لغة تعريف البيانات" *DDL Statements*

من المحتمل أن تكون هذه صدمة لك، ولكن قبل أن تخزن أي بيانات في جدول، عليك أن تعمل على إنشاء ذلك الجدول. سكول لديها الأداة التامة لعمل هذا: عبارة "إنشاء جدول *CREATE TABLE*". فهي عبارة من العديد من العبارات. والقواعد الأساسية *basic syntax* بسيطة جداً:

```
CREATE TABLE tableName
```

```
(
fieldName1 dataType options,
fieldName2 dataType options,
and so on...
)
```

فقط اعمل على ملئ الأجزاء وستكون جاهز لتعبئة (البيانات ، وهذا كل ما في الأمر). أسماء الحقول والجدول يتم بناءها من الحروف والأرقام، بإمكانك تضمين فراغات وبعض المحارف الخاصة، ولكن يتم عمل هذا من أجل كتابة كود فقط فيما بعد سترى. فكل بائع لديه مجمع خاص من أنواع لبيانات *data types* ، سأقترح إصدارات سكول سرفر هنا. يسمح لك *options* من تعيين أشياء مثل فيما إذا يتطلب الحقل بيانات *data* أم لا، وفيما إذا يمثل المفتاح الرئيسي للجدول أم لا، وقيود *constraints* أخرى مشابهة. الامتدادات *Extensions* للتركيب تتيح لك تثبيت القيود *constraints* التي تطبق على كامل الجدول، الفهارس *indexes* (التي تسمح لك ترتيب أو البحث بعمود معين بسرعة أكبر)، وتفصيل تخزين البيانات. إليك مثال عن عبارة "إنشاء جدول" والتي يمكن استخدامها لجدول بنود اسطر الطليبة *order line* .

```
CREATE TABLE LineItems
(
RecordID bigint IDENTITY PRIMARY KEY,
OrderID bigint NOT NULL,
CustomerID varchar(20) NOT NULL
REFERENCES Customers (CustomerID),
ProductID varchar(20) NOT NULL,
Quantity smallint NOT NULL
)
```

الكلمة المحجوزة *IDENTITY* تسمح لخدم سكول SQL Server من تولي مسؤولية تعبئة حقل *RecordID* بالبيانات، وسوف تستخدم العداد التتابعي *sequential counter* لتزويد قيمة مميزة ومفردة لـ *RecordID* بالنسبة لكل سجل جديد. التعبير "مفتاح رئيسي *PRIMARY KEY*" يعرف الحقل *RecordID* كقيمة تعريف مفردة ومميزة بالنسبة لكل سجل في الجدول. أنواع البيانات *bigint* و *smallint* تشير إلى الحقول التي تحفظ قيم عددية ذات حجم مناسب، والنوع *varchar* يوفر مساحة لأجل النصوص، بالطول الأعظمي المخصص في الأقواس (20 characters). يعرف التعبير *REFERENCES* العلاقات بين جدول *LineItems* و جدول *Customers*. (لاحظ تركيب "النقطة dot" لفصل أسماء الجداول والحقول، وهي تظهر في كل مكان في السكول) (المرجعيات (العلاقات *References*) بين الجداول تعرف أيضاً علاقات (مرجعيات) أجنبية (أو ما يطلق عليه مفتاح ثانوي) *foreign references* . إذا كنت تريد جعل هيكل أو خيار التغيرات لجدول أو حقله بعد أن يتم إنشائه، تتضمن سكول عبارة *ALTER TABLE* والتي يمكنها تغيير تقريباً كل شيء في الجدول. ويوجد أيضاً عبارة *DROP TABLE* التي تستخدم للتخلص من الجدول وكل بياناته. ومن المحتمل أنك تتجنب هذه العبارة على البيانات الموجودة في المنتج، وكما تستخدم تميل للغضب قليلاً عندما تختفي فجأةً جميع بياناتك. يلخص الجدول التالي أنواع البيانات المتاحة في مخدم سكول SQL Server.

Data type	شرح	Description
Bigint	8-بايت (64-بت) لحقل عددي صحيح للقيم التي تتراوح من -9,223,372,036,854,775,808 إلى 9,223,372,036,854,775,807	
binary	بيانات ثنائية ثابتة الطول، حتى 8,000 بايت في الطول. وإنك تخصص الطول من خلال الوسيط، مثل في <i>binary(100)</i>	
Bit	تدعم ثلاث قيم ممكنة: 1,0، أو بدون قيمة <i>NULL</i> . بشكل عام تستخدم للقيم المنطقية <i>Boolean</i> . ذاتياً، يخزن السكول سرفر عدة حقول بت لسجل مفرد في حقل عددي صحيح مدمج.	
char, nchar	(حرفي <i>char</i>) قياس ثابت الطول أو نصوص (س. حرف <i>nchar</i>) ليوني كود <i>Unicode</i> ، حتى 8,000 حرف في الطول. وإنك تخصص الطول من خلال الوسيط في القوس مثل <i>char(100)</i>	
Cursor	يستخدم هذا النوع من البيانات ضمن الإجراءات المخزنة <i>stored procedures</i> ، ولا يتم استخدامه لإنشاء عمود.	
Datetime	لحقول الوقت والتاريخ العام التي تتراوح من <i>January 1, 1753</i> إلى <i>December 31, 999</i> ، ودقة الوقت <i>Time accuracy</i> لأي قيمة ثابتة ضمن 3.33 ميلي ثانية. يعمل السكول سرفر 2008 على إضافة العديد من البيانات المتعلقة بأنواع البيانات: <i>date</i> (تواريخ دون وقت)، ووقت <i>time</i> (أوقات دون تواريخ)، تاريخ ووقت <i>datetime2</i> (نفس الحالي <i>datetime</i> ، ولكن بمجال أعلى ودقة إلى 100 بلليون من الثواني <i>nanoseconds</i> 100)، و <i>datetimeoffset</i> (مجالات للوقت والتاريخ)	
decimal, numeric	حقل بتدريج عشري وثابت الدقة. إنك تحدد العدد الأكبر من الأرقام التي تظهر على جانبي النقطة العشرية (الدقة <i>precision</i>) والعدد الأكبر لهذه الأرقام التي يمكن أن تظهر على الجهة اليمنى للنقطة أو الفاصلة العشرية <i>decimal point</i> (التدريج <i>scale</i>). على سبيل المثال، وضع <i>decimal(10,4)</i> يعمل على إنشاء حقل حتى عشرة أرقام بالكامل، وأربع منها يمكن أن تظهر بعد الفاصلة أو النقطة العشرية. وقيمة الدقة الأعلى هي 38. و <i>numeric</i> مرادف <i>decimal</i> - <i>synonym</i> ، مثل <i>dec</i> .	
Float	حقل النقطة العشرية العائمة (الكسرية) مع التخزين المتغير. بإمكانك تخصيص عدد الوحدات (البتات <i>bits</i>) المستخدمة لتخزين قيمة، حتى 53. بشكل افتراضي، جميع الوحدات الـ 53 يتم استخدامها، لذلك وضع <i>float</i> مكافئ لـ <i>float(53)</i>	

float(24)مكافئ لـ. والقيم المخزنة على هيئة 1.0×10^{38} ، تفاوت الدقة ونفس المجال بالوحدات bits المستخدمة لتخزين نوع البيانات هذه حساس لأقل عدد من الأخطاء المحسوبة.	
نوع البيانات هذا جديد، في السكول سرفر 2008، ويدعم الاستعلام الهرمي (التسلسلي) والبيانات الشجرية الشكل (الهيئة tree-shaped data). وهي غير متاحة في سكول سرفر 2005.	Hierarchyid
لا تستخدم أنواع البيانات هذه، حيث أنه سيتم إزالتها في النهاية من سكول سرفر.	image, text, ntext
حقل عددي صحيح 4-بايت (32-بت) للقيم ضمن المجال من -2,147,483,648 إلى 2,147,483,647.	Int
حقل عالي الدقة high-accuracy 8بايت (64بت) لتخزين القيم النقدية (عملة currency values)، حتى أربع أرقام للفاصلة العشرية (النقطة العشرية). ويخزن مجال لقيم البيانات من -922,337,203,685,477.5808 إلى 922,337,203,685,477.5807.	Money
يتم استخدام نوع البيانات هذه لتسجيل أحداث التعديل المطبقة على السجلات. ويوجد قيود على استخدامها، وهي غير مضمونة لأن تكون مفردة ومميزة ضمن الجدول. timestamp هو مكافئ قليل الأهمية deprecated synonym لـ rowversion، استخدم rowversion بدلاً عنه.	rowversion, timestamp
حقل للوقت والتاريخ العام للتواريخ ذات المجال من January 1, 1900 AD إلى June 6, 2079 AD. ودقة الوقت Time accuracy لأي قيمة ثابتة هي ضمن دقيقة واحدة.	Smalldatetime
حقل (متغير) عددي صحيح 2بايت (16بت) للقيم التي تتراوح من -32,768 إلى 32,767.	Smallint
حقل عالي الدقة 4بايت (32بت) لتخزين القيم النقدية (عملة)، حتى أربع أرقام بعد الفاصلة العشرية. وقيم البيانات المخزنة تتراوح من -214,748.3648 إلى 214,748.3647.	Smallmoney
نوع عام يخزن القيم للعديد من أنواع الحقول المخصصة الأخرى.	sql_variant
حقل خاص يخزن بشكل مؤقت نتائج الاستعلام على هيئة جدول مضغوط. وتعريف حقل (متغير) جدول معقد نوعاً ما، ويتم استخدامه عادة مصحوباً بقيود خاصة.	Table
حقل عددي صحيح دون إشارة unsigned integer field 1بايت (8بت) لمجال القيم من 0 إلى 255.	Tinyint
معرف مفرد ومميز شامل (globally unique identifier (GUID) 16بايت. تولد الدالة (الوظيفة) NEWID قيم لهذا الحقل أو المتغير.	Uniqueidentifier
بيانات ثنائية متغيرة الطول، حتى 8000بايت في الطول. إنك تحدد الطول من خلال الوسيط parameter، مثل varbinary(100). ويأخذ المتغير (الحقل) المساحة من أجل المحتوى الفعلي الحالي المخزن في الحقل. وإن الإعداد الخاص له مثل varbinary(max) يتيح مدخلات حتى 2 بليون بايت تقريباً.	Varbinary
متغير ذو طول قياسي (varchar) أو نصوص يوني كود (nvarchar) ، حتى 8000 حرف في الطول، وإنك تحدد الطول من خلال الوسيط مثل varchar(100). ويأخذ الحقل مساحة فقط لما تم تخزينه في الحقل. والإعداد الخاص له varchar(max) يسمح مدخلات حتى 2 بليون حرف تقريباً.	varchar, nvarchar
توفر التخزين لمستندات (وثائق) بيانات XML المرزمة typed و الغير مرزمة untyped، حتى 2 غيغابايت 2 GB.	Xml
عبارات "لغة معالجة البيانات" DML Statements	
تضيف العبارة INSERT سجلات (صفوف records) بيانات للجدول. يتم إضافة البيانات Data للجدول سجل واحد كل مرة. (تنوع عبارة INSERT يتيح لك إدخال عدة سجلات، ولكن هذه السجلات يجب أن تأتي من مصدر آخر لجدول موجود) لاستخدام عبارة INSERT، حدد الجدول والحقول المقصودة ومن ثم القيم المستقلة التي يتم وضعها في كل حقل. كل قيمة للبيانات تتوافق مع كل اسم عمود بيانات محدد.	
<pre>INSERT INTO LinelItems (OrderID, CustomerID, ProductID, Quantity) VALUES (10002, 'BW3', 'BEV01COF', 1)</pre>	
خذ هذه العبارة بالمقارنة مع عبارة CREATE TABLE المكتوبة سابقاً، سيعمل إجراء الإدخال هذا على إضافة سجل جديد لجدول LinelItems مع خمسة حقول (أو أعمدة) جديدة - أربع حقول معينة، زائد حقل المفتاح الرئيسي primary key الذي يتم إضافته بشكل آلي لحقل RecordID (حيث أنه تم تعليمه كمعرف IDENTITY). يعمل سكول سرفر اختيار تكامل البيانات المتنوعة بالنسبة عنك. كل حقل بيانات تعمل على إضافته يجب أن يكون من نوع البيانات الصحيح. بما أننا قمنا بتصميم حقل CustomerID على علاقة مع جدول Customer، سيفشل الإدخال إذا كان BW3 غير موجود سابقاً في جدول Customer.	
الحرفية العددية يمكن أن يتم تضمينها في عبارات سكول عند الحاجة دون أي شرط أو قيد إضافي. الحرفية النصية هي دائماً تكون محاطة بعلامة تنصيص (اقتباس) مفرد، كما تم عمله لمعرفة حقول customer و product في عبارة الإدخال INSERT هذه. وإذا أردت أن تضمن علامة تنصيص مفردة في الحرفية، قم بإدخاله مرتين:	
<pre>John O'Sullivan'</pre>	
قم بإحاطة البيانات الحرفية literal والقيم الزمنية time values بعلامات تنصيص مفردة:	
<pre>'7-Nov-2005'</pre>	
مثل قيم التاريخ والوقت هذه تقبل أي تنسيق مميز، على الرغم من أنك ستستخدم تنسيق ليس من السهل على سكول سرفر أن يسيء تفسيره misinterpret. العديد من أنواع الحقول تدعم قيمة "غير مسندة unassigned"، قيمة تشير أن الحقل لا يحتوي بيانات على الإطلاق. مثل هذه القيم تعرف "بدون قيمة null"، ويتم تخصيصها في سكول سرفر باستخدام الكلمة المحجوزة "بدون قيمة NULL". ولا يمكنك إسناد قيمة "بدون قيمة NULL" لحقل المفتاح الرئيسي، أو لأي حقل تم تعليمه بالخيار NOT NULL.	
لإزالة سجل تم إضافتها سابقاً، استخدم عبارة DELETE:	
<pre>DELETE FROM LinelItems WHERE RecordID = 92231</pre>	

تتضمن عبارة DELETE الشرط (أو التعبير WHERE حيث) (الجزء WHERE RecordID = 92231). يسمح لك الشرط WHERE من الإشارة إلى سجل أو أكثر في جدول ما وذلك بعمل مقارنة مع بيانات الحقول. ويمكن لشرط WHERE أن يتضمن كلمات محجوزة AND و OR لربط عدة شروط، وأقواس parentheses للتجميع.

```
DELETE FROM LineItems WHERE OrderID = 10001
AND ProductID = 'BRD05RYE'
```

يمكن لمثل عبارة DELETE هذه أن تحذف 0 سجل، سجل واحد أو 1000 سجل، ولذلك فإن الدقة في شرط WHERE هام جداً. لحذف جميع السجلات في الجدول قم بإخراج شرط بالكامل WHERE .

```
DELETE FROM LineItems
```

تستخدم عبارة "تحديث UPDATE" أيضاً تعبير WHERE لتعديل القيم في سجلات جدول موجود.

```
UPDATE LineItems
SET Quantity = 4
WHERE (RecordID = 92231)
```

يتم عمل الإسنادات للحقول بالتعبير "ضع SET"، ضع اسم الحقل (Quantity) على الجهة اليسارية من إشارة المساواة، والقيمة الجديدة على الجهة اليمينية من إشارة المساواة (4). ولإسناد عدة قيم في نفس الوقت، افصل كل إسناد بالفاصلة comma. وبإمكانك أيضاً تضمين صيغ formulas وحسابات calculations .

```
UPDATE LineItems SET Quantity = Quantity + 1,
ProductID = 'BEV02POP'
WHERE RecordID = 92231
```

وكما مع عبارة DELETE، فيمكن لعبارة UPDATE أن تحدث صفر سجل، سجل واحد، أو عدة سجلات بالاعتماد على أي سجلات تتطابق مع الشرط WHERE.

العبارة الأخيرة للغة معالجة البيانات DML، والأكثر استخداماً هي عبارة "اختر SELECT".

```
SELECT ProductID, Quantity FROM LineItems
WHERE RecordID = 92231
```

تقوم عبارة SELECT بعمل مسح على جدول (LineItems)، والبحث عن جميع السجلات التي تطابق المعيار المقدم (= RecordID 92231)، وتعود بأصغر جدول يحتوي فقط الحقول المشار إليها (ProductID، Quantity) للسجلات المتوافقة. والاستعلام الأكثر عمومية والذي يعود بكل الصفوف (السجلات) والحقول (الأعمدة) هو التالي: `SELECT * FROM LineItems`. يعود هذا الاستعلام بجميع السجلات من الجدول LineItems بدون أي ترتيب معين. وتعني asterisk النجمة (*) تضمين كل الحقول. "include all fields". يعود الشرط "ترتيب بواسطة ORDER BY" بنتائج مرتبة بطريقة خاصة (معينة):

```
SELECT * FROM LineItems
WHERE Quantity > 5
ORDER BY ProductID, Quantity DESC
```

يعود هذا الاستعلام بجميع السجلات التي تكون فيها قيمة حقل Quantity أكبر من 5، ويرتب النتائج أولاً بواسطة عمود ProductID (بترتيب تصاعدي ascending order) ومن ثم بواسطة حقل ذو القيمة العددية numeric quantity (بترتيب تنازلي في descending order). وذلك من خلال كتابة الكلمة (DESC).

تتيح لك دوال الإجمالي (التراكم) Aggregate functions وميزات التجميع grouping features تلخيص النتائج من مجموعة أكبر من البيانات. يوثق الاستعلام التالي الكمية المطلوبة الإجمالية total ordered quantity لكل منتج product في الجدول:

```
SELECT ProductID, SUM(Quantity) FROM LineItems
GROUP BY ProductID
```

بإمكانك استخدام joins لربط البيانات مع بعضها من جدولين مميزين أو أكثر. الاستعلام التالي يضم جدول LineItems وجدول Customer على أعمدهم CustomerID المتطابقة. وتوضح أيضاً عبارة SELECT هذه استخدام اختصارات abbreviations الجدول (السابق LI و CU) المضافة من خلال التعبير AS، وهي ليست ضرورية عادةً، ولكنها تساعد على جعل الاستعلام المعقد أكثر قابلية للقراءة:

```
SELECT LI.OrderID, CU.CustomerName, LI.ProductID
FROM LineItems AS LI INNER JOIN Customer AS CU
ON LI.CustomerID = CU.CustomerID
ORDER BY LI.OrderID, CU.CustomerName
```

يستخدم هذا الجدول "الربط الداخلي inner join" وهو واحد من خمسة أنواع للربط، وكل منها يعود بمجموعات مختلفة من السجلات بالاعتماد على العلاقة بين الجدول الأول (اليساري) والجدول الثاني (اليميني) في الربط join.

الربط الداخلي: Inner join

يعود فقط بالسجلات التي تتطابق مع بعضها في الحقول المربوطة. وهذا النوع من الربط يستخدم الكلمة المحجوزة INNER JOIN.

الربط الخارجي اليساري: Left outer join

يعود بكل سجل من الجدول اليساري فقط بالسجلات المطابقة من الجدول في الجهة اليمينية من الربط، فإذا كان سجل الجدول اليساري ليس لديه تطابق، فإنه يتصرف وكأن جميع حقول الجدول اليميني لذلك السجل تحتوي قيم "بدون قيمة NULL". هذا النوع من الربط يستخدم الكلمة المحجوزة LEFT JOIN. وأحد استخداماته يمكن أن يكون لربط جدول Product وجدول LineItems. بإمكانك إرجاع قائمة بكامل أسماء المنتجات لكل المنتجات المتاحة، بالإضافة إلى الكمية الإجمالية المطلوبة من كل منتج.

بوضع جدول المنتجات Product على الجهة اليسارية من الربط الخارجي اليساري left outer join، فإن الاستعلام سيعود بأسماء كل المنتجات، حتى ولو كان ذلك المنتج لم يتم طلبه على الإطلاق (ولا يظهر في جدول LineItems).

الربط الخارجي اليميني: Right outer join

يعمل هذا تماماً مثل الربط الخارجي اليساري، ولكن جميع سجلات الجدول اليميني يتم استعادتها، واستعادة فقط سجلات الجدول اليساري التي لديها تطابق. ويستخدم هذا النوع من الربط الكلمة المحجوزة RIGHT JOIN.

الربط الخارجي التام Full outer join

يعود بجميع السجلات من الجدولين اليساري واليميني، فيما إذا كان هناك تطابق أم لا. عندما يوجد تطابق، فإنه يتم عكس النتائج it is reflected in the results. وهذا النوع يستخدم الكلمة المحجوزة FULL JOIN.

الربط المتقاطع (المتصالب) Cross join

ويدعى أيضاً الربط الديكارتي Cartesian join (دمج مجموعتين لتوليد مجموعة واحدة مدمجة). ويعود بكل السجلات المدمجة الممكنة اليمينية واليسارية. ويستخدم هذا النوع الكلمة المحجوزة CROSS JOIN.

يركز الربط على العلاقة التي بين الجدولين. (وهذا الاستخدام لـ "العلاقة" ليس الأساس لمصطلح قاعدة البيانات العلائقية relational database) بعض الجداول توجد في علاقة "أباء- أبناء" parent-child، "فسجل واحد" أب" لديه واحد أو أكثر من السجلات الأبناء المعتمدة في جدول آخر. وهذا غالباً صحيح للطلبات orders، "ف عنوان طلبية order header" مفرد لديه عدة "بنود أسطر line items" وهذا النوع من العلاقة يعرف بواحد-إلى-العديد one-to-many، حيث أن سجل واحد يكون مرتبط بعدة سجلات في جدول آخر. والعلاقة تكون أحادية الاتجاه، والسجل الابن المعطى لا يرتبط بسجلات أبوية متعددة.

تربط العلاقة واحد-إلى-واحد one-to-one سجل مفرد في جدول واحد إلى سجل في جدول آخر. وهي بسيطة جداً، وتستخدم غالباً لتحسين القيم الموجودة في السجل الأصل من السجل التكميلي في الجدول الثاني.

في علاقة عديد - إلى - عديد many-to-many، سجل مفرد في جدول واحد يكون مصحوباً مع عدة سجلات في الجدول الثاني، وسجل مفرد في الجدول الثاني يكون أيضاً مصحوباً بعدة سجلات في الجدول الأول. ومثال حقيقي عنه، العلاقة بين المعلمين والطلاب في وضع كلية ما. فمعلم واحد لديه العديد من الطلبة في الصف (الحجرة الدراسية) ولكن كل طالب أيضاً لديه العديد من المعلمين في كل فصل دراسي semester.

يتطلب التنفيذ التطبيقي لعلاقة عديد-إلى-العديد عملياً ثلاث جداول: الجدولين المرتبطين، و جدول "وسيط بينهما go-between" والذي يربطهما مع بعض. وسأريك مثال لمثل هذا الجدول في المقطع التالي من هذا الفصل.

ما خلف أساس سكول Beyond Basic SQL

العبارات البسيطة التي جدولتها هنا تعالج موضوع إمكانيات معالجة البيانات المتاحة بشكل سطحي scratch the surface من خلال سكول. ولكن حالياً يمكن أن تكون قد لاحظت أن سكول يحاكي اللغة الانكليزية في القواعد (التركيب) على نحو رائع، وحتى أكثر مما هي عليه الفيچوال بيسك. في الحقيقة، الاسم الأصلي للغة SEQUEL والذي كان تركيب لـ "لغة الاستعلام الانكليزية الهيكلية Structured English Query Language" وبما أن عبارات سكول قد أخذت بالتعقيد أكثر، فستبدو أكثر شبيهاً لمجموعات عشوائية من الكلمات الانكليزية.

الهدف هنا هو تقديمك للتركيب القاعدي لعبارات سكول. معظم العبارات التي سنواجهها في مشروع المكتبة لن تكون أكثر تعقيداً من العبارات التي تم تضمينها في الأمثلة هنا.

استخدام قواعد البيانات في الفيچوال بيسك Using Databases in Visual Basic

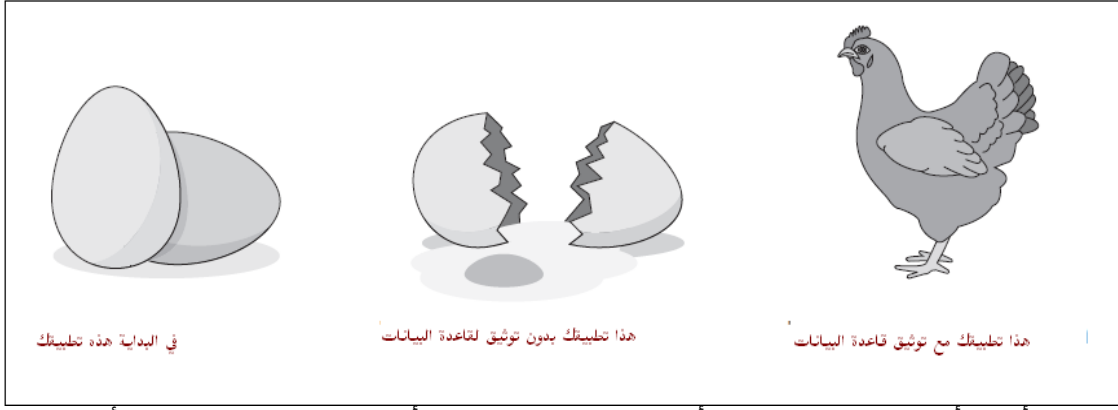
يمكن للفيچوال بيسك أن تتفاعل مع البيانات المخزنة في قاعدة بيانات بعدة طرق مختلفة:

- ✓ استخدام ADO.NET، التقنية الرئيسية للوصول للبيانات المضمنة في إطار عمل الدوت نت NET Framework. للتفاعل مع محتوى قاعدة البيانات المخزنة. وهذه هي الطريقة المستخدمة على طول مع برنامج المكتبة للتفاعل مع قاعدة بياناته. وتم شرح آدو دوت نت في الفصل العاشر، مع أمثلة عن طريقة استخدامه، وسأقدم أيضاً في ذلك الفصل كود خاص من مشروع المكتبة.
- ✓ استخدام ميزات "تحميل البيانات data binding" المتاحة في الفيچوال بيسك والفيچوال أستوديو. يؤسس التحزيم اتصال بين أداة البيانات التي على الشاشة أو كائن تمكين البيانات المشابه والمحتوى من قاعدة بيانات ما. الكود المكتوب لك بواسطة ميكروسوفت يأخذ على عاتقه عمل كل الاتصالات، فتستطيع حتى سحب وإسقاط هذه الأنواع من التفاعلات. على الرغم من أنني سأناقش تحميل البيانات في الفصل العاشر (حيث أن التحزيم يعتمد على ADO.NET)، فإنني أميل لتجنبه، لأنه يقلل مقدار تحكم الميرمج الذي يستطيع أن يبذله على معالجة بيانات المستخدم. ولن يتم استخدام تحميل البيانات في برنامج المكتبة.
- ✓ استخراج البيانات من قاعدة البيانات ضمن ملف قياسي، واستخدام ميزات معالجة ملف في الفيچوال بيسك لمعالجة البيانات. من الواضح أن هذا غير مفيد جداً، ولكنني مضطر لاستخدامه، خاصة في الأيام القديمة عندما كانت قواعد البيانات الخاصة لاتستطيع التفاعل بسهولة مع كود الفيچوال بيسك.
- ✓ كل مرة تحتاج فيها لبعض البيانات، أخبر المستخدم أنه بطريقة أو بأخرى قد تم فقدها، وبالتالي يتوجب إعادة إدخالها مباشرة. فإذا كان لديك الفضول لمعرفة كيف يبدو داخل مكتب البطالة، فهذه يمكن أن تكون فرصتك.

توثيق قاعدة البيانات Documenting the Database

المحتوى التقني الذي يشرح الجداول والحقول في قاعدة بياناتك للتطبيق تمثل معظم الجزء الهام من التوثيق المولد خلال حياة التطبيق. في الواقع، الحاجة لتوثيق جيد هو الأساس فيما يخص واحد من معتقداتي البرمجية الجوهرية: توثيق المشروع هو هام، وفي بعض الأحيان أكثر أهمية حتى من الكود المصدري source code.

إذا كنت تطور تطبيق ما يتمحور حول قاعدة بيانات تخزن محتوى لمستخدم، التوثيق الكامل والدقيق لكل جدول وحقل مستخدم في قاعدة البيانات هو واجب. وأي نقص في هذه المساحة يجب أن لا يكون ممكناً أو محتملاً، وإذا كان فإنه سيؤدي قضاياً تكامل البيانات وأبعد من التسلسل الزمني للتطور الضروري كما هو مبين في الشكل التالي:



السبب الذي يجعلني أعتقد أن توثيق قاعدة البيانات أهم من توثيق المستخدم أو المواصفات الوظيفية، هو الأثر الذي سيشتركه المستند على بيانات المستخدم. فإذا كان لديك قاعدة بيانات موثقة، فتستطيع أن تخمن ماهية المواصفات الوظيفية، وربما تأتي هذه التخمينات دقيقة جداً. فإذا كان ينقصك توثيق المستخدم user documentation، بإمكانك دائماً كتابته عندما ينتهي البرنامج. ولكن إذا كان ينقصك توثيق قاعدة البيانات database documentation، فأنت داخل عالم مؤذي. إذا لم تعمل من قبل على مشاريع قاعدة بيانات ضخمة، من المحتمل أن لا تصدقني، ولكن لو عملت على وراثه نظام قاعدة بيانات واسع ومتكامل ويحوي على الكثير من الجداول وقد شارك فيه أكثر من مطور وكل منهم وضع مجال مختلف من البيانات الذي سيسمح لكل حقل أن يحتويها، وأي من الحقول يجب أن تكون مطلوبة أم لا، فهنا سترى التضارب. فتتبع 100,000 سطر من الكود المصدري لتحديد ما يعمل كل حقل (متغير) ليست بالتسلية، وقد تأخذ هذه العملية عدة شهور لإتمامها بدقة. لذلك لا تدع هذا يحصل معك.

المشروع

لنبدأ بمجموعة المصادر التقنية لمشروع المكتبة وذلك بتصميم وتوثيق جداول قاعدة البيانات لكي يتم استخدامها بواسطة التطبيق. وتظهر مجموعة المصادر هذه في دليل تثبيت الكتاب، في الدليل الجزئي للفصل الرابع، وتحتوي على الملفات التالية: نسخة من ميكروسوفت ورد لتوثيق المشروع تقنياً وهي مقدمة في هذا المقطع مشروع هذا الفصل (كامل التوثيق موضوع هنا).

سكول.صيغة إنشاء قاعدة البيانات **Database Creation Script.sql**

صيغة قاعدة بيانات سكول سرفر SQL Server database script المستخدمة لبناء الجداول والحقول الفعلية في قاعدة البيانات. محتوى مجموعة المصدر التقني **Technical Resource Kit Content**

يحتوي هذا المقطع جدولاً بالجدول المحتواة في قاعدة بيانات المكتبة. كل جدول يتضمن وصف عام لمساعدتك في فهم هيكل قاعدة البيانات. وسوف تصادف جميع الجداول في الفصول اللاحقة، على طول مصحوبة بالكود المصدري source code، لذلك لا تغضب freak out إذا بدت بعض الجداول أو الحقول صعبة الفهم لك الآن.

جداول الامن المترابطة Security-related tables

على الرغم من أن الزبائن patrons ليسوا بحاجة لتسجيل الدخول log in للتطبيق لاستخراج البنود look up من قاعدة البيانات، فالمدرء administrators يجب أن يسجلوا الدخول قبل أن يتمكنوا من الوصول إلى الميزات المحسنة للبرنامج. فالجدول الأربعة التالية تدير إتمادات الأمن security credentials لكل مدير. يستخدم التطبيق قاعدة بيانات سكول سرفر أو Windows-based security credentials إتمادات التأمين المعتمدة على ويندوز فقط للوصول إلى قاعدة البيانات بشكل أولي، initially، وليس للحد من (تقييد) الميزات. النشاط Activity يعرف هذا الجدول ميزات التطبيق التي يمكن تأمينها باستخدام مجموعة الحقوق group rights. وهذه النشاطات يتم ربطها مع مجموعات الأمن (من جدول GroupName) لتأسيس الحقوق لمجموعة خاصة particular group.

Description	Type	Field
مفتاح رئيسي. وهذا المفتاح آلي التوليد، والقيمة المزودة تطابق داخلياً القيم المستخدمة ضمن مشروع المكتبة. وهو مطلوب.	Long	ID
Descriptive name of this activity. Required. اسم وصفي لهذا النشاط. مطلوب	Text(50)	FullName

حالياً يتم تعريف النشاطات التالية:

- 1 - Manage authors and names إدارة المؤلفين والأسماء
- 2 - Manage author and name types إدارة المؤلف وأنواع الأسماء
- 3 - Manage copy status codes إدارة أكواد الحالة للنسخ
- 4 - Manage media types إدارة أنواع الوسائط
- 5 - Manage series إدارة السلاسل
- 6 - Manage security groups إدارة مجموعات الأمن
- 7 - Manage library materials إدارة مواد المكتبة
- 8 - Manage patrons إدارة الزبائن
- 9 - Manage publishers إدارة الناشرين
- 10 - Manage system values إدارة قيم النظام

- 11 - Manage administrative users إدارة المستخدمين الإداريين
- 12 - Process and accept fees معالجة وقبول الرسوم
- 13 - Manage locations إدارة المواقع
- 14 - Check out library items تفحص بنود مخرجات المكتبة
- 15 - Check in library items تفحص بنود مدخلات المكتبة
- 16 - Access administrative features الميزات الإدارية للوصول
- 17 - Perform daily processing عمل معالجة يومية
- 18 - Run system reports تشغيل تقارير النظام
- 19 - Access patrons without patron password تمكين الزبائن من الوصول دون الحاجة لكلمة مرور الزبون
- 20 - Manage barcodes إدارة أكواد التعريف
- 21 - Manage holidays إدارة أيام العطل
- 22 - Manage patron groups إدارة مجموعات الزبائن
- 23 - View administrative patron messages عرض رسائل الزبائن الإدارية

أسماء المجموعات **GroupName**. كل سجل في هذا الجدول يعرف مجموعة أمن مفردة. فأمناء المكتبة Librarians والمدراء administrators الآخرين كل منهم ينتمي إلى مجموعة أمن مفردة.

Field	Type	Description
ID	Long - Auto	Primary key; automatically assigned. Required. مفتاح رئيسي، يتم إسناده بشكل آلي. وهو مطلوب
FullName	Text(50)	Name of this group. Required. اسم هذه المجموعة. مطلوب

نشاط مجموعة **GroupActivity**. هذا الجدول يصل السجلات في جدول النشاط **Activity** إلى السجلات في جدول اسم المجموعة **GroupName** (علاقة عديد-إلى-عديد many-to-many) لتأسيس نشاطات مجموعة أمن التي يمكن عملها.

Field	Type	Description
GroupID	Long	Primary key. The associated security group. Foreign reference to GroupName.ID. Required. مفتاح رئيسي، المصاحب لمجموعة تأمين. مفتاح ثانوي (علاقة ثانوية) لحقل معرف اسم مجموعة GroupName.ID. مطلوب
ActivityID	Long	Primary key. The activity that members of the associated security group can perform. Foreign reference to Activity.ID. Required. مفتاح رئيسي، النشاط الذي تستطيع عمله الأعضاء المصاحبة لمجموعة تأمين. مفتاح ثانوي لحقل النشاط Activity.ID. مطلوب

اسم المستخدم **UserName** يحتوي هذا الجدول السجلات الحقيقية لكل أمين مكتبة أو مدير. يتضمن كل سجل كلمة مرور المستخدم ووضع مجموعة الأمن.

Field	Type	Description
ID	Long - Auto	Primary key; automatically assigned. Required. مفتاح رئيسي، إسناد آلي. مطلوب
FullName	Text(50)	Name of this user, administrator, or librarian. Required. اسم المستخدم، المدير، أو أمين المكتبة. مطلوب
LoginID	Text(20)	User ID that gives this user access to the system. It is entered into the Library program's "login" form, along with the password, to gain access to enhanced features. Required. معرف المستخدم والذي يسمح للمستخدم من الوصول إلى النظام. ويتم إدخاله في برنامج المكتبة "تسجيل الدخول" على طول مع كلمة المرور، لتستطيع الدخول إلى الميزات المحسنة. مطلوب
Password	Text(20)	The password for this user, in an encrypted format. Optional. كلمة المرور الخاصة بهذا المستخدم، بتنسيق مشفر. اختياري
Active	Boolean	Is this user allowed to access the system? 0 for False, 1 for True. Required. هل سيسمح لهذا المستخدم من التمكن الوصول إلى النظام؟ 0 لخطأ، 1 لصواب. مطلوب

Description	Type	Field
To which security group does this user belong? Foreign reference to GroupName.ID. Required. إلى أي مجموعة أمن ينتمي هذا المستخدم؟ مفتاح ثانوي لـ GroupName.ID. مطلوب .	Long	GroupID

جدول دعم الكود Support code tables

توجد العديد من الجداول تعمل بكل بساطة على تزويد قائمة من القيم لجدول أخرى. في تطبيق ما، قائمة الجداول هذه غالباً ما تظهر كخيارات في أداة تتوسع للأسفل drop-down ("صندوق مركب combo box").
 نوع مؤلف الكود (أو البند) CodeAuthorType. في برنامج المكتبة، الكلمة author هي مصطلح عام مستخدم للمؤلفين authors، الرسامين، المحررين editors، وأي مساهم contributor مشابه آخر لبند ما في فاتورة المكتبة library's inventory. يتيح لك هذا الجدول من تعريف هذه الأدوار (الوظائف roles).

Description	Type	Field
Primary key; automatically assigned. Required. مفتاح رئيسي، إسناد آلي. مطلوب .	Long - Auto	ID
Name of this type of author or contributor. Required. اسم نوع المؤلف هذا أو المساهم. مطلوب .	Text(50)	FullName

وضع نسخ الكود CodeCopyStatus. تتضمن أكواد وضع النسخ أشياء مثل "الدوران أو معادة circulating"، "المصححة being repaired"، وأي حالة رئيسية أخرى ترغب المكتبة بوضعها. حالة تم إعادتها checked-in أو تم إخراجها أو إعارتها checked-out يتم معاملتها من خلال ميزات أخرى، كما في العلامة التي تشير فيما إذا بند ما هو بند مرجعي reference item.

Description	Type	Field
Primary key; automatically assigned. Required. مفتاح رئيسي، آلي الإسناد. مطلوب .	Long - Auto	ID
Name of this status entry. Required. اسم مدخلة الحالة. مطلوب .	Text(50)	FullName

موقع الكود (أو البند) CodeLocation. المواقع الفعلية حيث يتم تخزين بنود المكتبة. وهذا ما يمكن من فصل المواقع separate sites، أو الغرف rooms والمساحات areas ضمن الموقع المشترك common location.

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Name of this location. Required.	Text(50)	FullName
The date when Daily Processing was last done for this location. If NULL, processing has not yet been done. Optional. التاريخ، عند عمل آخر معالجة يومية لهذا الموقع. إذا كان بدون قيمة، فإن المعالجة لم يتم عملها حتى الآن. اختياري.	Date	LastProcessing

نوع وسائط الكود CodeMediaType. نوع الوسائط، مثل الكتب books، المجلات magazines، الفيديو videos، السيديات CDs.

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Name of this media type. Required.	Text(50)	FullName
Number of days for which items in this type can be checked out, before renewal. Required. عدد أيام البنود التي يمكن استخراجها للبنود التي في هذا النوع، قبل أن يتم تجديدها. مطلوب.	Integer	CheckoutDays
Number of days to add to the original checkout period for a renewal of items within this type. Required. عدد الأيام التي يجب إضافتها إلى فترة الاستخراج للبنود المعاد تجديدها ضمن هذا النوع. مطلوب.	Integer	RenewDays
Maximum number of times the item can be renewed by a patron before it must be returned. Required. العدد الأعظمي من المرات التي يمكن تجديد البند فيها من قبل الزبون قبل أن يعيده. مطلوب.	Integer	RenewTimes
Amount charged per day for an overdue item of this type. Required.	Currency	DailyFine

Field	Type	Description
		الكمية المحسوبة على كل يوم تأخير، لبند من هذا النوع. مطلوب.

مجموعة زبائن كود **CodePatronGroup**. تصنيفات المجموعات والتي يتم وضع الزبائن ضمنها. وهي ليست مجموعات أمن، ولكن هي مجموعات عامة لأهداف تقريرية reporting purposes. وتم إضافة هذا لدعم تصنيف أو تجميع الزبائن بوحدات ضمن الشركة، أو بالفئة/الرتبة class/grade ضمن مجموعة مكتبة المدرسة.

Field	Type	Description
ID	Long - Auto	Primary key; automatically assigned. Required.
FullName	Text(50)	Name of this patron group. Required.

CodeSeries. بعض البنود تظهر كجزء من سلاسل أكبر larger series أو تجمع collection. فيعرف هذا الجدول أسماء السلاسل والتجمع.

Field	Type	Description
ID	Long - Auto	Primary key; automatically assigned. Required.
FullName	Text(50)	Name of this series or collection. Required.

بنود المكتبة Library items

الجدول في هذا المقطع تدير الفواتير الحقيقية actual inventory. حيث أنه يمكن للمكتبة أن تحوز على أكثر من نسخة لبند مفرد، وتدير هذه الجداول "البند المعين بالاسم named item" وهي نسخ copies مستقلة وبشكل منفصل.

NamedItem. بند المكتبة، مثل كتاب ما، سيدي، أو مجلة. ويمثل هذا الجدول بند عام، وليس النسخة الفعلية للبند.

Field	Type	Description
ID	Long - Auto	Primary key; automatically assigned. Required.
Title	Text(150)	Title of this item. Required.
Subtitle	Text(150)	Subtitle of this item. Optional.
Description	Memo	Full description of this item. Optional.
Edition	Text(10)	Edition number for this item. Optional.
Publisher	Long	This item's publisher. Foreign reference to Publisher.ID. Optional.
Dewey	Text(20)	Dewey decimal number. Use "/" for line breaks. Optional.
LC	Text(25)	Library of Congress number. Use "/" for line breaks. Optional.
ISxN	Text(20)	ISBN, ISSN, or other standardized number of this item. Optional.
LCCN	Text(12)	Library of Congress control number. Optional.
Copyright	Integer	Year of original copyright, or of believed original copyright. Optional.
Series	Long	The series or collection in which this item appears. Foreign reference to CodeSeries.ID. Optional.
MediaType	Long	The media classification of this item. Foreign reference to CodeMediaType.ID. Required.
OutOfPrint	Boolean	Is this title out of print? 0 for False, 1 for True. Required.

ItemCopy نسخة مفردة لبند معين باسمه. النسخ المنفصلة لنفس البند ستظهر كسجلات منفصلة في هذا الجدول.

Field	Type	Description
ID	Long - Auto	Primary key; automatically assigned. Required.
ItemID	Long	The related named item record. Foreign reference to NamedItem.ID. Required.
CopyNumber	Integer	Numbered position of this item within the set of copies for a named item. Required, and unique among items with the same ItemID field value.
Description	Memo	Comments specific to this copy of the item. Optional.
Available	Boolean	Is this copy available for checkout or circulation? 0 for False, 1 for True. Required.

Description	Type	Field
Has this copy been reported missing? 0 for False, 1 for True. Required.	Boolean	Missing
Is this a reference copy? 0 for False, 1 for True. Required.	Boolean	Reference
Any comments relevant to the condition of this copy. Optional.	Text(30)	Condition
Date this copy was acquired by the library. Optional.	Date	Acquired
Value of this item, either original or replacement value. Optional.	Currency	Cost
The general status of this copy. Foreign reference to CodeCopyStatus.ID. Required.	Long	Status
Bar code found on the copy. At this time, only numeric bar codes are supported. Optional.	Text(20)	Barcode
The site or room location of this item. Foreign reference to CodeLocation.ID. Optional.	Long	Location

Publisher المنظمة التي تنشر الكتب أو بعض الأنواع الأخرى من الوسائط.

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Name of the publisher. Required.	Text(100)	FullName
URL for this publisher's web site. Optional.	Text(255)	WebSite

Author الشخص الذي يكتب writes، يحرر edits، يوضح illustrates، أو بطريقة أخرى يساهم contributes في بند كتاب أو بند وسيط. في جميع الحالات، عندما يظهر بند المؤلف author في هذا الجدول، فإنه يشير إلى أي شخص يشارك في البند.

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Last name of this author. Required.	Text(50)	LastName
First name of this author. Optional.	Text(30)	FirstName
Middle name or initial of this author. Optional.	Text(30)	MiddleName
Name suffix, such as "Jr." Optional.	Text(10)	Suffix
Year of birth. Use negative numbers for BC. Optional.	Integer	BirthYear
Year of death. Use negative numbers for BC. Optional.	Integer	DeathYear
Miscellaneous comments about this author. Optional.	Text(250)	Comments

ItemAuthor المؤلف، المحرر، وهكذا، بالنسبة لاسم بند معين. وهذا الجدول ينجز علاقة العديد-إلى-العديد many-to-many بين جدول NamedItem وجدول Author.

Description	Type	Field
Primary key. The associated named item. Foreign reference to NamedItem.ID. Required.	Long	ItemID
Primary key. The author associated with the named item. Foreign reference to Author.ID. Required.	Long	AuthorID
Relative order of this author among the authors for this named item. Authors with smaller numbers appear first. Required.	Integer	Sequence
The specific type of contribution given by this author for this named item. Foreign reference to CodeAuthorType.ID. Required.	Long	AuthorType

Keyword كلمات خاصة يمكن تطبيقها على البنود المعينة بالاسم named items لجعل البحث أكثر سهولة.

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Name of this keyword. Required.	Text(50)	FullName

ItemKeyword يصل keyword بالبند المحدد بالاسم (بالبند المسمى named item) من خلال علاقة عديد-إلى-عديد بين جدول NamedItem وجدول Keyword.

Description	Type	Field
Primary key. The associated named item. Foreign reference to	Long	ItemID

Description	Type	Field
NamedItem.ID. Required.		
Primary key. The keyword to associate with the named item. Foreign reference to Keyword.ID. Required.	Long	KeywordID
Subject عنوان المواضيع Subject headings المستخدمة لتصنيف البنود المسماة (المحددة بالاسم).		
Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Name of this subject. Required.	Text(150)	FullName
ItemSubject تصل الموضوع مع البند المسمى من خلال علاقة عديد-إلى-عديد بين جدول NamedItem وجدول Subject.		
Description	Type	Field
Primary key. The associated named item. Foreign reference to NamedItem.ID. Required.	Long	ItemID
Primary key. The subject to associate with the named item. Foreign reference to Subject.ID. Required.	Long	SubjectID

جدول المرتبطة بالزبائن Patron-related tables

الجدول في هذا المقطع تعرف سجلات الزبون الحقيقية وعلاقتها مع نسخ البند) item copies (عندما يتم تفحص مخرجات مثل هذه النسخ بواسطة الزبون).

Patron مستخدم محدد للمكتبة، وعادة للزبائن Patrons امتيازات privileges فيما يخص المخرجات (checkout).

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Last name of this patron. Required.	Text(30)	LastName
First name of this patron. Required.	Text(30)	FirstName
Date of last checkout, renewal, or return. Optional.	Date	LastActivity
Is this an active patron? 0 for False, 1 for True. Required.	Boolean	Active
Any comments associated with this patron. Optional.	Memo	Comments
Comments that are displayed to administrative users when the patron's record is accessed. Optional.	Memo	AdminMessage
Bar code found on this patron's library card. At this time, only numeric bar codes are supported. Optional.	Text(20)	Barcode
Patron's password, in an encrypted format. Required.	Text(20)	Password
Patron's email address. Optional.	Text(100)	Email
Patron's phone number. Optional.	Text(20)	Phone
Patron's street address. Optional.	Text(50)	Address
Patron's city. Optional.	Text(20)	City
Patron's state abbreviation. Optional.	Text(2)	State
Patron's postal code. Optional.	Text(10)	Postal
The group in which this patron appears. Foreign reference to CodePatronGroup.ID. Optional.	Long	PatronGroup

PatronCopy يدير هذا الجدول نسخ البنود item copies المراجعة حالياً بواسطة الزبون، أو نسخ بند تم مراجعتها سابقاً وقد يتم مراجعتها فيما بعد.

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
The associated patron. Foreign reference to Patron.ID. Required.	Long	Patron
The item copy currently or previously checked out by the patron. Foreign reference to ItemCopy.ID. Required.	Long	ItemCopy
The date when this item copy was initially checked out. Required.	Date	CheckOut
The number of times this item copy has been renewed. Set to 0 when the item copy is first checked out. Required.	Integer	Renewal

Description	Type	Field
Current due date for this item copy. Required.	Date	DueDate
The date when this item copy was returned. Optional.	Date	CheckIn
Has the item copy been returned? 0 for False, 1 for True. Required.	Boolean	Returned
Is the item copy missing and considered lost? 0 for False, 1 for True. Required.	Boolean	Missing
Total fine accumulated for this item copy. Defaults to 0.00. An administrator may reduce an accumulated fine. Required.	Currency	Fine
Total amount paid (in fees) for this item copy. Required.	Currency	Paid
When an item copy is processed for overdue fines, this field contains the last date for which processing was done. Optional.	Date	ProcessDate

PatronPayment الغرامات Fines، التسديدات (المدفوعات payments)، الطرد dismissals على سجل نسخة الزبون patron copy record. غرامات التأخير Overdue fines غير مسجلة في هذا الجدول، ولكن الغرامات الموضوعه من قبل المدير تبعاً للمسؤوليات بالنسبة للبنود المفقودة تم تسجيلها هنا .

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
The associated item checked out by the patron. Foreign reference to PatronCopy.ID. Required.	Long	PatronCopy
Date and time when this entry was recorded. Required.	Date	EntryDate
The type of payment entry. Required. The possible values are: <ul style="list-style-type: none"> • "P" - The patron made a payment. • "F" - A fine (other than a standard overdue fine) was imposed by an administrator. • "D" - A portion (or all) of the fine was dismissed. • "R" - A refund was given to the patron due to overpayment. 	Text(1) EntryType	
The amount associated with this entry. The value is always positive. Required.	Currency	Amount
A short comment about this entry. Optional.	Text(50)	Comment
The user who added this payment event. Foreign reference to UserName.ID. Optional.	Long	UserID

الجدول المتعلقة بكود التعريف Bar code-related tables

يوجد ثلاث مستويات لتعريف أو إنشاء كود تعريف (1) الصفحة التي عليها تطبع شبكة من العناوين، (2) عنوان مفرد single label على صفحة (صفحة sheet)، (3) البنود المستقلة التي تظهر على كل عنوان. الجداول الثلاث في هذا المقطع تعرف هذه المستويات الثلاث.

BarcodeSheet يصف قالب خاص بصفحة مفردة من عناوين كود التعريف .

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Name of this sheet template. Required.	Text(50)	FullName
Units used in the various measurements found in most fields in this record. Required. <ul style="list-style-type: none"> • I = Inches • C = Centimeters • P = Points • T = Twips 	Text(1)	UnitType
Width of the entire page. Required.	Number	PageWidth
Height of the entire page. Required.	Number	PageHeight
Left border, up to the edge of the printable label area. Required.	Number	MarginLeft
Right border, up to the edge of the printable label area. Required.	Number	MarginRight

Description	Type	Field
Top border, up to the edge of the printable label area. Required.	Number	MarginTop
Bottom border, up to the edge of the printable label area. Required.	Number	MarginBottom
The width of the blank area between label columns. Required.	Number	IntraColumn
The height of the blank area between label rows. Required.	Number	IntraRow
The number of label columns on this template. Required.	Integer	ColumnsCount
The number of label rows on this template. Required.	Integer	RowsCount

BarcodeLabel يصف قالب عنوان مفرد على صفحة كود تعريف. أي عدد من العناوين يمكن أن تكون على صفحة مفردة، ولكن كلها لديها نفس الشكل والشكل (التنسيق format).

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Name of this label template. Required.	Text(50)	FullName
The sheet template on which this label template appears. Foreign reference to BarcodeSheet.ID. Required.	Long	BarcodeSheet
Units used in the various measurements found in most fields in this record. Required. I = Inches • C = Centimeters • P = Points • T = Twips •	Text(1)	UnitType

BarcodeLabelItem يصف بند مفرد عندما يوجد على عنوان كود تعريف. تتضمن البنود النصوص الثابتة والمولدة، الأسطر، المستطيلات، و أكواد التعريف المولدة .

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Identifies the order in which items on the label are printed. Lower numbers are printed first. Required.	Integer	Priority
The label template on which this item appears. Foreign reference to BarcodeLabel.ID. Required.	Long	BarcodeLabel
What type of item does this record represent? Required. T = Static text • B = Barcode • N = Barcode number • L = Line • R = Rectangle •	Text(1)	ItemType
Left edge of the item relative to the left edge of the label. Measured according to the related BarcodeLabel.UnitType field. Required.	Number	PosLeft
Top edge of the item relative to the top edge of the label. Measured according to the related BarcodeLabel.UnitType field. Required.	Number	PosTop
Width of the item, or of the box in which the item is drawn. For lines, this is the x-coordinate of the end point. Measured according to the related BarcodeLabel.UnitType field. Required.	Number	PosWidth
Height of the item, or of the box in which the item is drawn. For lines, this is the y-coordinate of the end point. Measured according to the related BarcodeLabel.UnitType field. Required.	Number	PosHeight
Rotation angle, in degrees, of the box in which the item is drawn. Zero (0) equals no angle, and increasing angles proceed	Integer	Rotation

Description	Type	Field
clockwise. Ranges from 0 to 359. Only used when ItemType is T, B, N, or R. Optional.		
The name of the font used to write the text. Valid only when ItemType is T or N. Optional.	Text(50)	FontName
The size of the font used to write the text. Valid only when ItemType is T, B, or N. Optional.	Number	FontSize
The static text to display on the label. Valid only when ItemType is T. Optional.	Text(100)	StaticText
The style of the font text. May be any combination of the following four codes. <ul style="list-style-type: none"> B = Bold I = Italic U = Underline K = Strikeout Leave this field NULL to use the normal style. Valid only when ItemType is T or N. Optional.	Text(4)	FontStyle
The main color of the text, bar code, or line. When printing a rectangle, this is the border color. If NULL, black is used. A standard Windows 32-bit RGB color value. Optional.	Long	Color1
The fill color when printing a rectangle. If NULL, white is used. A standard Windows 32-bit RGB color value. Optional.	Long	Color2
The alignment of the text within the bounding box. Valid only when ItemType is T, B, or N. <ul style="list-style-type: none"> 1 = Align in top-left corner of box 2 = Align in top-center area of box 4 = Align in top-right corner of box 16 = Align in middle-left area of box 32 = Align in middle-center area of box 64 = Align in middle-right area of box 256 = Align in bottom-left corner of box 512 = Align in bottom-center area of box 1024 = Align in bottom-right corner of box 	Integer	Alignment
The number of digits in which to pad the bar code number. Set to zero (0) to ignore padding. Ranges from 0 to 20. If the bar code length is less than the specified number of digits, it is padded on the left with zeros. Only applies to ItemTypes of B and N.	Integer	PadDigits

جداول أخرى متنوعة (شتمى) Other miscellaneous tables

يوفر جدولين إضافيين دعم للميزات التي لم يتم معالجتها من خلال الجداول الأخرى. **Holiday** عندما مراجعة بند ما لزبون، فإن التاريخ الراجع يجب أن لا يتضمن العطل (أو أي يوم تكون فيه المكتبة مغلقة)، حيث أن الزبون من المحتمل أن لا تكون لديه طريقة لإعادة الكتاب في مثل هذا اليوم. يحدد (يعرف) هذا الجدول سابقاً ويرجع العطل recurring holidays.

Description	Type	Field
Primary key; automatically assigned. Required.	Long - Auto	ID
Name of this holiday. Not necessarily unique. Required.	Text(50)	FullName
The type of entry. Required. From the following list. <ul style="list-style-type: none"> A = Annual (as in "every December 25") E = Weekly (as in "every Sunday") O = One-time (as in "2/16/2004 is President's Day") 	Text(1)	EntryType
Entry-type-specific detail. Required. Differs for each entry type.	Text(10)	EntryDetail

Field	Type	Description
		Detail Value Entry Type
		Month and Day in "mm/dd" format A
		Single digit: 1=Sunday through 7=Saturday E
		Date in "yyyy/mm/dd" format O

SystemValue يخزن هذا الجدول إعدادات واسعة ونهائية شتى miscellaneous enterprise-wide settings والتي تطبق على كل شبكة حاسوب workstation. يتم تخزين الإعدادات الخاصة بشبكة حاسوب محلية Local workstation-specific settings على كل جهاز، وليس في قاعدة البيانات.

Field	Type	Description
ID	Long - Auto	Primary key; automatically assigned. Required.
ValueName	Text(50)	Name of this value. Required.
ValueData	Text(100)	Information associated with this entry. Optional.

يتم تعريف قيم النظام التالية في الوقت الحالي. واسم الكود يظهر في حقل ValueName والقيمة الموافقة تظهر في حقل ValueData.

BarcodeCode39

هل كود التعريف مخصص في تنسيق "الكود39" "code 39" أو "كود3 من 9" "code 3 of 9"؟ إذا كان كذلك، سيتم وضع نجمة قبل وبعد رقم كود التعريف bar code number قبل أن يتم طباعته على عنوان ما. استخدم القيمة 0 لخطأ False وأي قيمة غير الصفر لصواب (1-مفضلة). ويتم افتراض خطأ في حالة فقدان القيمة أو بدون قيمة missing or NULL.

BarcodeFont

اسم الخط المستخدم لطباعة أكواد التعريف. هذا الخط يجب أن يكون مثبت على أي شبكة حاسوب تعرض أو تطبع أكواد التعريف. وليس هناك حاجة لعمل مسح على أكواد التعريف.

DatabaseVersion

أي إصدار هيكل لقاعدة البيانات قيد الاستخدام حالياً؟ يتم وضعه حالياً إلى "1"، ويتم حفظه من أجل التحسينات المستقبلية.

DefaultLocation

قيمة حقل CodeLocation.ID من أجل الموقع location الذي يتم وضعه كافتراضي .

FineGrace (مهلة الغرامة)

عدد الأيام التي يمكن لبند أن يتأخر ضمنها بند ما دون أن تتسبب في غرامة without incurring a fine.

NextBarcodeItem

قيمة البداية التالية لتستخدم عند طباعة أكواد تعريف بند item bar codes .

NextBarcodeMisc

قيمة البداية التالية لتستخدم عند طباعة أكواد التعريف المختلفة (الشتى) miscellaneous bar codes .

NextBarcodePatron

قيمة البداية التالية لتستخدم عند طباعة أكواد تعريف العميل (الزبون patron bar codes) .

PatronCheckOut

تشير فيما إذا بإمكان الزبائن استخراج بنود دون الحاجة لتسجيل الدخول كمستخدمين إداريين. استخدم القيمة 0 (صفر) لتشير إلى عدم وجود امتيازات استخراج وقيمة غير الصفر للسماح لزبون من المراجعة (1-مفضلة). فإذا ما فقدت هذه القيمة أو كانت فارغة، فلن يتم السماح للزبائن من استخراج بنود دون مساعدة المدير.

SearchLimit

تشير إلى العدد الأكبر من النتائج المعادة في أي عملية بحث. فإذا ما فقدت هذه القيمة أو كانت غير صحيحة، يتم استخدام الافتراضي 250. والمجال المسموح الشامل inclusive هو بين 25 و5000.

TicketHeading

يعرض النص الذي يجب طباعته عند أعلى بطاقات مخرجات الدفع. كل الأسطر تتوسط البطاقة. بما فيه حرف الشريط العمودي (|) لتقسيم النص في عدة أسطر .

TicketFooting

يعرض النص الذي سيتم طباعته أسفل بطاقات مخرجات الدفع. كل الأسطر تتوسط البطاقة. بما فيها حرف الشريط العمودي (|) لتقسيم النص في عدة أسطر .

UseLC

يشير فيما إذا تم استدعاء أرقام تصنيف كتاب بواسطة Dewey أو مكتبة الكونغرس (في أمريكا) Library of Congress (LC). استخدم القيمة 0 (صفر) لتشير إلى Dewey، أو قيمة غير الصفر لـ LC (1-مفضلة). فإذا ما فقدت هذه القيمة أو كانت فارغة فالافتراضي هو Dewey.

إنشاء قاعدة البيانات Creating the Database

إضافة قاعدة بيانات إلى سكول سرفر هي عملية سهلة بقدر سهولة توثيقها، في الحقيقة، تتطلب كتابة أقل. وعبارات CREATE TABLE بسيطة جداً، وتبدو جميعها متشابهة إلى حد بعيد. وسأريك فقط بعضاً منها هنا. (يحتوي ملف Database CreationScript.sql على المحتوى الكامل للصيغ وقد قمت بوضعه في برامج الفصل الرابع).

التعليمات المجدولة هنا هي من أجل إدارة منصة سكول سرفر 2005 السريعة SQL Server 2005 Management Studio Express. بإمكانك إتمام كل هذه المهام باستخدام إدارة منصة سكول سرفر 2005، أو حتى أدوات أسطر الأمر command-line tools الموفرة بواسطة سكول سرفر، ولكن قد تختلف التفاصيل بالنسبة لكل خطوة. تعمل نفس عبارات CREATE TABLE مع أي أداة تختارها. إذا لم تكن قد عملت مثل هذا سابقاً. اعمل على تثبيت

سكول سرفر 2005 بطبعته السريعة SQL Server 2005 Express Edition (أو أي إصدار من قاعدة البيانات سيكون بإمكانك استخدامه). إن إدارة منصة سكول سرفر 2005 السريعة منتج مفصول عن سكول سرفر نفسه، لذلك يجب أن تعمل على تثبيتها أيضاً. معظم الجداول في مشروع المكتبة هي جداول بيانات بسيطة ذات مفتاح رئيسي مفرد. وكودها بسيط جداً. فجدول المؤلف Author هو مثال جيد.

```
CREATE TABLE Author
(
  ID bigint IDENTITY PRIMARY KEY,
  LastName varchar(50) NOT NULL,
  FirstName varchar(30) NULL,
  MiddleName varchar(30) NULL,
  Suffix varchar(10) NULL,
  BirthYear smallint NULL,
  DeathYear smallint NULL,
  Comments varchar(250) NULL
);
```

الحقول المضمنة في كل عبارة CREATE TABLE تظهر كقائمة محددة بفاصلة comma-delimited list، وجميعها (الحقول) مغلقة في أقواس. وكل حقل يتضمن إما خيار بدون قيمة NULL أو ليس بدون قيمة NOT NULL والذي يشير فيما إذا سيتم استخدام القيم بدون قيمة NULL في ذلك الحقل أم لا. خيار المفتاح الرئيسي PRIMARY KEY بشكل آلي يخصص "ليس بدون قيمة NOT NULL" بعض العبارات تنشئ جداول تربط جدولين آخرين في علاقة عديد-إلى-عديد. وأحد أمثلتها جدول GroupActivity، الذي يصل جدول GroupName مع جدول Activity.

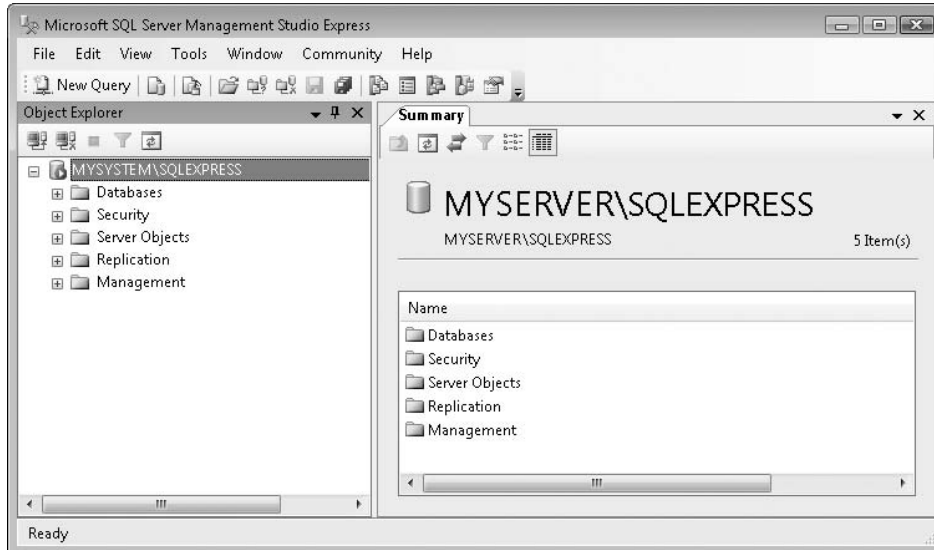
```
CREATE TABLE GroupActivity
(
  GroupID bigint NOT NULL,
  ActivityID bigint NOT NULL,
  PRIMARY KEY (GroupID, ActivityID)
);
```

لجدول المؤلف Author مفتاح رئيسي مفرد، لذلك خيار المفتاح الرئيسي يمكن أن يتم إلحاقه مباشرةً لحقل المعرف ID التابع له. بما أن جدول GroupActivity لديه حقل مفتاح رئيسي (وهو معروف في قاعدة البيانات العلائقية)، خيار المفتاح الرئيسي PRIMARY KEY يتم تخصيصه كمدخلة خاصة به تماماً، مع حقول المفاتيح المخصصة ضمن القوسين والمفصولة فيما بينها بفاصلة PRIMARY KEY (GroupID, ActivityID). سابقاً في هذا الفصل، بينت لك كيف تستطيع تأسيس علاقة لحقل في جدول آخر باستخدام قيود مرجعية (علائقية) REFERENCES constraint (كجزء من عبارة CREATE TABLE). بإمكانك أيضاً تأسيسها بعد أن يتم وضع الجداول في مكانها. (كالذي عملته في ملف صيغة سكول). إليك العبارة التي تؤسس اتصال بين جدول GroupActivity وجدول GroupName.

```
ALTER TABLE GroupActivity
ADD FOREIGN KEY (GroupID)
REFERENCES GroupName (ID);
```

حيث أنني قمت بكتابة صيغ سكول الكاملة entire SQL script لك، فإنني أطلب منك فقط معالجتها مباشرةً باستخدام إدارة منصة ميكروسوفت سكول سرفر 2005 السريعة Microsoft SQL Server 2005 Management Studio Express. (إذا كنت ستستخدم الإصدار الكامل من سكول سرفر أو بعض الأدوات الإدارية management tool الأخرى، فإن الصيغ الموفرة ستبقى تعمل، على الرغم من أن تعليمات خطوة-بخطوة ستختلف). قبل إضافة الجداول tables، نحتاج إلى إنشاء قاعدة بيانات مخصصة لمشروع المكتبة Library Project. شغل منصة إدارة ميكروسوفت سكول سرفر 2005 السريعة (شاهد الشكل التالي) لإضافة قاعدة بيانات جديدة new database لمشروع المكتبة، انقر يمين على مجلد "قاعدة البيانات Database" في مستكشف الكائن، واختر "قاعدة بيانات جديدة Database from New Database" من القائمة المختصرة. على نموذج قاعدة البيانات الجديد الذي يظهر، أدخل Library في حقل اسم قاعدة Database Name، ومن ثم انقر موافق OK.

قاعدة بيانات المكتبة مجرد غلاف قاعدة بيانات، فهي لا تحتوي أي جدول أو أي بيانات حتى الآن. لنستخدم ملف Database Creation Script.sql لإنتاج الجداول والبيانات الأولية في إدارة المنصة السريعة، اختر ملف File >> فتح Open >> أمر قائمة ملف File menu command، وأوجد ملف Database Creation Script.sql. (من الممكن أن يطلب منك تسجيل الدخول لسكول سرفر مرةً أخرى) يفتح هذا الملف مساحات لمحتوياته في لوحة جديدة ضمن إدارة المنصة السريعة Management Studio Express.

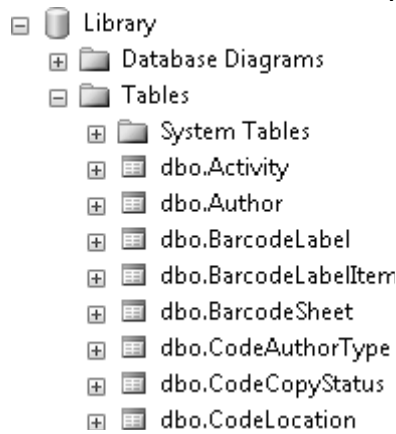


SQL Server 2005 Management Studio Express main form

كل ما بقي عليك عمله هو معالجة الصيغة script. في منطقة "شريط الأدوات" toolbar، تأكد من أن "Library" هي قاعدة البيانات المختارة (شاهد الشكل التالي). ومن ثم انقر على زر تنفيذ Execute من شريط الأدوات، أو اضغط مفتاح F5. وهي صيغة صغيرة مع القليل من العوامل not a lot going on (على الأقل من وجهة نظر سكول سرفر)، لذلك سيتم الانتهاء منها في عدة ثواني.



هذا كل شيء، أغلق لوحة الصيغة script panel. ومن ثم، ارجع لمستكشف الكائن Object Explorer، وانقر يمين على مجلد قاعدة بيانات المكتبة Library واختر تحديث Refresh من القائمة. فإذا ما وسعت بعدئذٍ تفرع قاعدة بيانات المكتبة وجدولها المتفرعة، ستري كل الجداول المنشئة بواسطة الصيغة (شاهد الشكل التالي).
والآن بعد إتمام قاعدة البيانات، حان الوقت لتبدأ البرمجة .



Partial list of database tables

سأضمن لك ملف Database Creation Script.sql كاملاً :

```
----- */ACME Library Project
* Database creation script.
* Start-to-Finish Visual Basic 2005
* Copyright (c) 2006 by Tim Patrick
/*
```

```
CREATE TABLE Activity
)
  ID          bigint          PRIMARY KEY,
  FullName   varchar(50) NOT NULL
;(
GO
```

```

INSERT INTO Activity (ID, FullName) VALUES (1, 'Manage authors and names')
INSERT INTO Activity (ID, FullName) VALUES (2, 'Manage author and name types')
INSERT INTO Activity (ID, FullName) VALUES (3, 'Manage copy status codes')
INSERT INTO Activity (ID, FullName) VALUES (4, 'Manage media types')
INSERT INTO Activity (ID, FullName) VALUES (5, 'Manage series')
INSERT INTO Activity (ID, FullName) VALUES (6, 'Manage security groups')
INSERT INTO Activity (ID, FullName) VALUES (7, 'Manage library materials')
INSERT INTO Activity (ID, FullName) VALUES (8, 'Manage patrons')
INSERT INTO Activity (ID, FullName) VALUES (9, 'Manage publishers')
INSERT INTO Activity (ID, FullName) VALUES (10, 'Manage system values')
INSERT INTO Activity (ID, FullName) VALUES (11, 'Manage administrative users')
INSERT INTO Activity (ID, FullName) VALUES (12, 'Process and accept fees')
INSERT INTO Activity (ID, FullName) VALUES (13, 'Manage locations')
INSERT INTO Activity (ID, FullName) VALUES (14, 'Check out library items')
INSERT INTO Activity (ID, FullName) VALUES (15, 'Check in library items')
INSERT INTO Activity (ID, FullName) VALUES (16, 'Access administrative features')
INSERT INTO Activity (ID, FullName) VALUES (17, 'Perform daily processing')
INSERT INTO Activity (ID, FullName) VALUES (18, 'Run system reports')
INSERT INTO Activity (ID, FullName) VALUES (19, 'Access patrons without patron password')
INSERT INTO Activity (ID, FullName) VALUES (20, 'Manage barcodes')
INSERT INTO Activity (ID, FullName) VALUES (21, 'Manage holidays')
INSERT INTO Activity (ID, FullName) VALUES (22, 'Manage patron groups')
INSERT INTO Activity (ID, FullName) VALUES (23, 'View administrative patron messages')
GO

```

```

CREATE TABLE Author
)
  ID          bigint          IDENTITY PRIMARY KEY,
  LastName    varchar(50)     NOT NULL,
  FirstName   varchar(30)     NULL,
  MiddleName  varchar(30)     NULL,
  Suffix      varchar(10)     NULL,
  BirthYear   smallint        NULL,
  DeathYear   smallint        NULL,
  Comments    varchar(250)    NULL
;(
GO

```

```

CREATE TABLE BarcodeLabel
)
  ID          bigint          IDENTITY PRIMARY KEY,
  FullName    varchar(50)     NOT NULL,
  BarcodeSheet  bigint        NOT NULL,
  UnitType    varchar(1)      NOT NULL
;(
GO

```

```

CREATE TABLE BarcodeLabelItem
)
  ID          bigint          IDENTITY PRIMARY KEY,

```



```

Priority    smallint    NOT NULL,
BarcodeLabel  bigint      NOT NULL,
ItemType     varchar(1)   NOT NULL,
PosLeft      decimal(10,4) NOT NULL,
PosTop       decimal(10,4) NOT NULL,
PosWidth     decimal(10,4) NOT NULL,
PosHeight    decimal(10,4) NOT NULL,
Rotation     smallint    NULL,
FontName     varchar(50) NULL,
FontSize     decimal(10,4) NULL,
StaticText   varchar(100) NULL,
FontStyle    varchar(4)   NULL,
Color1       bigint      NULL,
Color2       bigint      NULL,
Alignment    smallint    NULL,
PadDigits    smallint    NULL
;(
GO

CREATE TABLE BarcodeSheet
)
ID          bigint      IDENTITY PRIMARY KEY,
FullName    varchar(50) NOT NULL,
UnitType    varchar(1)   NOT NULL,
PageWidth   decimal(10,4) NOT NULL,
PageHeight  decimal(10,4) NOT NULL,
MarginLeft  decimal(10,4) NOT NULL,
MarginRight decimal(10,4) NOT NULL,
MarginTop   decimal(10,4) NOT NULL,
MarginBottom decimal(10,4) NOT NULL,
IntraColumn decimal(10,4) NOT NULL,
IntraRow    decimal(10,4) NOT NULL,
ColumnsCount smallint    NOT NULL,
RowsCount   smallint    NOT NULL
;(
GO

CREATE TABLE CodeAuthorType
)
ID          bigint      IDENTITY PRIMARY KEY,
FullName    varchar(50) NOT NULL
;(
GO

CREATE TABLE CodeCopyStatus
)
ID          bigint      IDENTITY PRIMARY KEY,
FullName    varchar(50) NOT NULL
;(
GO

CREATE TABLE CodeLocation

```

```
)
  ID          bigint      IDENTITY PRIMARY KEY,
  FullName    varchar(50) NOT NULL,
  LastProcessing datetime  NULL
;(
GO
```

```
CREATE TABLE CodeMediaType
)
  ID          bigint      IDENTITY PRIMARY KEY,
  FullName    varchar(50) NOT NULL,
  CheckoutDays smallint   NOT NULL,
  RenewDays   smallint   NOT NULL,
  RenewTimes  smallint   NOT NULL,
  DailyFine   money      NOT NULL
;(
GO
```

```
CREATE TABLE CodePatronGroup
)
  ID          bigint      IDENTITY PRIMARY KEY,
  FullName    varchar(50) NOT NULL
;(
GO
```

```
CREATE TABLE CodeSeries
)
  ID          bigint      IDENTITY PRIMARY KEY,
  FullName    varchar(50) NOT NULL
;(
GO
```

```
CREATE TABLE GroupActivity
)
  GroupID     bigint     NOT NULL,
  ActivityID  bigint     NOT NULL,
  PRIMARY KEY (GroupID, ActivityID)
;(
GO
```

```
CREATE TABLE GroupName
)
  ID          bigint      IDENTITY PRIMARY KEY,
  FullName    varchar(50) NOT NULL
;(
GO
```

```
CREATE TABLE Holiday
)
  ID          bigint      IDENTITY PRIMARY KEY,
  FullName    varchar(50) NOT NULL,
  EntryType   varchar(1)  NOT NULL,
```

```
EntryDetail varchar(10) NOT NULL
;(
GO
```

```
CREATE TABLE ItemAuthor
)
ItemID bigint NOT NULL,
AuthorID bigint NOT NULL,
Sequence smallint NOT NULL,
AuthorType bigint NOT NULL,
PRIMARY KEY (ItemID, AuthorID(
;(
GO
```

```
CREATE TABLE ItemCopy
)
ID bigint IDENTITY PRIMARY KEY,
ItemID bigint NOT NULL,
CopyNumber smallint NOT NULL,
Description varchar(max) NULL,
Available bit NOT NULL,
Missing bit NOT NULL,
Reference bit NOT NULL,
Condition varchar(30) NULL,
Acquired datetime NULL,
Cost money NULL,
Status bigint NOT NULL,
Barcode varchar(20) NULL,
Location bigint NULL
;(
GO
```

```
CREATE TABLE ItemKeyword
)
ItemID bigint NOT NULL,
KeywordID bigint NOT NULL,
PRIMARY KEY (ItemID, KeywordID(
;(
GO
```

```
CREATE TABLE ItemSubject
)
ItemID bigint NOT NULL,
SubjectID bigint NOT NULL,
PRIMARY KEY (ItemID, SubjectID(
;(
GO
```

```
CREATE TABLE Keyword
)
ID bigint IDENTITY PRIMARY KEY,
FullName varchar(50) NOT NULL
```

```

;(
GO

CREATE TABLE NamedItem
)
ID      bigint      IDENTITY PRIMARY KEY,
Title   varchar(150) NOT NULL,
Subtitle varchar(150) NULL,
Description varchar(max) NULL,
Edition  varchar(10)  NULL,
Publisher bigint      NULL,
Dewey    varchar(20) NULL,
LC       varchar(25) NULL,
ISxN     varchar(20) NULL,
LCCN     varchar(12) NULL,
Copyright smallint  NULL,
Series   bigint      NULL,
MediaType bigint      NOT NULL,
OutOfPrint bit        NOT NULL

```

```

;(
GO

CREATE TABLE Patron
)
ID      bigint      IDENTITY PRIMARY KEY,
LastName varchar(30) NOT NULL,
FirstName varchar(30) NOT NULL,
LastActivity datetime NULL,
Active   bit        NOT NULL,
Comments varchar(max) NULL,
AdminMessage varchar(max) NULL,
Barcode  varchar(20) NULL,
Password varchar(20) NOT NULL,
Email    varchar(100) NULL,
Phone    varchar(20) NULL,
Address  varchar(50) NULL,
City     varchar(20) NULL,
State    varchar(2)  NULL,
Postal   varchar(10) NULL,
PatronGroup bigint  NULL

```

```

;(
GO

CREATE TABLE PatronCopy
)
ID      bigint      IDENTITY PRIMARY KEY,
Patron  bigint      NOT NULL,
ItemCopy bigint      NOT NULL,
CheckOut datetime  NOT NULL,
Renewal smallint   NOT NULL,
DueDate datetime  NOT NULL,
CheckIn datetime  NULL,

```

```

Returned bit NOT NULL,
Missing bit NOT NULL,
Fine money NOT NULL,
Paid money NOT NULL,
ProcessDate datetime NULL
;(
GO

```

```

CREATE TABLE PatronPayment
)
ID bigint IDENTITY PRIMARY KEY,
PatronCopy bigint NOT NULL,
EntryDate datetime NOT NULL,
EntryType varchar(1) NOT NULL,
Amount money NOT NULL,
Comment varchar(50) NULL,
UserID bigint NULL
;(
GO

```

```

CREATE TABLE Publisher
)
ID bigint IDENTITY PRIMARY KEY,
FullName varchar(100) NOT NULL,
WebSite varchar(255) NULL
;(
GO

```

```

CREATE TABLE Subject
)
ID bigint IDENTITY PRIMARY KEY,
FullName varchar(150) NOT NULL
;(
GO

```

```

CREATE TABLE SystemValue
)
ID bigint IDENTITY PRIMARY KEY,
ValueName varchar(50) NOT NULL,
ValueData varchar(100) NULL
;(
GO

```

```

INSERT INTO SystemValue (ValueName, ValueData) VALUES ('BarcodeCode39', '0')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('BarcodeFont', NULL)
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('DatabaseVersion', '1')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('DefaultLocation', NULL)
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('FineGrace', '3')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('Licensee', NULL)
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('LicenseCode', NULL)
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('NextBarcodeItem', '2000001')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('NextBarcodeMisc', '1000001')

```

```

INSERT INTO SystemValue (ValueName, ValueData) VALUES ('NextBarcodePatron', '1')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('PatronCheckOut', '-1')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('SearchLimit', '250')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('TicketHeading', 'Library System')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('TicketFooting', 'Thank you')
INSERT INTO SystemValue (ValueName, ValueData) VALUES ('UseLC', '0')
GO

```

```

CREATE TABLE UserName
(
  ID          bigint      IDENTITY PRIMARY KEY,
  FullName   varchar(50) NOT NULL,
  LoginID    varchar(20) NOT NULL,
  Password   varchar(20) NULL,
  Active     bit          NOT NULL,
  GroupID    bigint      NOT NULL
)
GO

```

```

ALTER TABLE GroupActivity ADD FOREIGN KEY (GroupID) REFERENCES GroupName (ID)
ALTER TABLE GroupActivity ADD FOREIGN KEY (ActivityID) REFERENCES Activity (ID)
ALTER TABLE UserName ADD FOREIGN KEY (GroupID) REFERENCES GroupName (ID)
ALTER TABLE NamedItem ADD FOREIGN KEY (Publisher) REFERENCES Publisher (ID)
ALTER TABLE NamedItem ADD FOREIGN KEY (Series) REFERENCES CodeSeries (ID)
ALTER TABLE NamedItem ADD FOREIGN KEY (MediaType) REFERENCES CodeMediaType (ID)
ALTER TABLE ItemCopy ADD FOREIGN KEY (ItemID) REFERENCES NamedItem (ID)
ALTER TABLE ItemCopy ADD FOREIGN KEY (Status) REFERENCES CodeCopyStatus (ID)
ALTER TABLE ItemCopy ADD FOREIGN KEY (Location) REFERENCES CodeLocation (ID)
ALTER TABLE ItemAuthor ADD FOREIGN KEY (ItemID) REFERENCES NamedItem (ID)
ALTER TABLE ItemAuthor ADD FOREIGN KEY (AuthorID) REFERENCES Author (ID)
ALTER TABLE ItemAuthor ADD FOREIGN KEY (AuthorType) REFERENCES CodeAuthorType (ID)
ALTER TABLE ItemKeyword ADD FOREIGN KEY (ItemID) REFERENCES NamedItem (ID)
ALTER TABLE ItemKeyword ADD FOREIGN KEY (KeywordID) REFERENCES Keyword (ID)
ALTER TABLE ItemSubject ADD FOREIGN KEY (ItemID) REFERENCES NamedItem (ID)
ALTER TABLE ItemSubject ADD FOREIGN KEY (SubjectID) REFERENCES Subject (ID)
ALTER TABLE Patron ADD FOREIGN KEY (PatronGroup) REFERENCES CodePatronGroup (ID)
ALTER TABLE PatronCopy ADD FOREIGN KEY (Patron) REFERENCES Patron (ID)
ALTER TABLE PatronCopy ADD FOREIGN KEY (ItemCopy) REFERENCES ItemCopy (ID)
ALTER TABLE PatronPayment ADD FOREIGN KEY (PatronCopy) REFERENCES PatronCopy (ID)
ALTER TABLE PatronPayment ADD FOREIGN KEY (UserID) REFERENCES UserName (ID)
ALTER TABLE BarcodeLabel ADD FOREIGN KEY (BarcodeSheet) REFERENCES BarcodeSheet (ID)
ALTER TABLE BarcodeLabelItem ADD FOREIGN KEY (BarcodeLabel) REFERENCES BarcodeLabel (ID)
GO

```

```

----- */Create the basic security account/* .
INSERT INTO GroupName (FullName(
VALUES ('Administrators')

INSERT INTO GroupActivity (GroupID, ActivityID)
SELECT 1, ID FROM Activity

```

```
INSERT INTO UserName (FullName, LoginID, Active, GroupID(  
VALUES ('Administrator', 'admin', 1, 1'  
GO
```

المجمعات Assembly

مجمعات الدوت نت هي مجرد ملفات EXE و DLL (مكتبة الربط الديناميكي dynamic link library). بدون الحاجة لتفعيلها من قبلك، فهي تحتل مساحة تجميعية من القرص. وبما أنها لا تعمل أي شيء آخر، لذلك لننتفحص ماهي هذه المجمعات وماذا تحتوي .

ما هو المجمع What Is an Assembly

المجمع هو "وحدة نشر unit of deployment" والتي وفي معظم الحالات هي ملف فقط. فالمجمع مستودع repository معد للترجمة من قبل كود تطبيق دوت نت. فأي كود تكتبه سيتم تخزينه في النهاية في ملف EXE (إذا كان تطبيق) أو ملف DLL (من أجل مكتبات الكود code libraries أو امتدادات تطبيق). كل شيء يحتاج معرفته حول الدوت نت لتحميل وتشغيل تطبيقك يتم تخزينه في مجمع assembly. المجمعات هي إما خاصة private أو عامة public. يتم تصميم المجمعات الخاصة Private assemblies للاستخدام في تطبيق مفرد فقط. فإذا لم يكن هناك أي DLLs، فمجمع EXE هو التطبيق. تظهر المجمعات الخاصة في دليل خاص بها، دليل التنصيب installation directory للتطبيق أو المكتبة. بإمكانك تشغيل مجمعين خاصين مختلفين في نفس الوقت، ولن يتعارض bother كل منهما مع الآخر. وهذا صحيح حتى ولو كان كل مجمع يستخدم نفس تركيب فضاء الأسماء وأسماء الفئة لعناصره المكددة (المشفرة). فإذا جمعا تطبيق، كل منهما ينفذ فئة مسماة WindowsApplication1.Class1، فلن يتداخل interfere كل منهما مع الآخر عند التشغيل، فهما خاصان private، وخاص private يعني خاص private. يتم تصميم المجمعات العامة Public assemblies للاستخدام المشترك بين تطبيقات الدوت نت المتعددة .NET applications. تختلف المجمعات العامة Public assemblies عن المجمعات الخاصة private assemblies في طريقتين رئيسيتين:

- المجمعات العامة دائماً لديها اسم قوي strong name، توقيع رقمي مشفر encrypted digital signature متعلق بمجمع ما لضمان أنه يأتي من البائع المسمى named vendor أو المصدر source. (يمكن للمجمعات الخاصة أن تتضمن اسم قوي، ولكن هذا ليس بالمستوجب). يتم بناء الاسم القوي من اسم المجمع، رقم النسخة (الإصدار)، معلومات ثقافية، الكلمة أو "المفتاح عام public key"، والتوقيع الرقمي digital signature المولد من ملف مجمع يحوي الكشف manifest. يتضمن إطار عمل الدوت نت أدوات توليد اسم قوي (sn.exe) والذي يساعد في هذه المعالجة، وتتضمن فيجوال أستوديو خيارات تتيح لك من إضافة توقيع رقمي خلال عملية الترجمة compilation process. (وهي ليست تبويب معين في صفحات خاصيات المشروع project's properties). الاسم القوي لمجمع ما يجب أن (ومن الأفضل أن يكون) مفرد ومميز، إذا ما تشارك مجمعان اسم قوي مشترك، فإنهما نسخ لنفس المجمع.
- يتم تخزين المجمعات العامة في الذاكرة الانتقالية للمجمع الشامل Global Assembly Cache (GAC). على الرغم من أنك تستطيع وضع نسخة من مكوناتك المتشاركة في دليل تنصيب تطبيقك application's install directory، فستكون صحيحة التشارك فقط عندما تصل لدليل GAC. يقيم GAC في دليل مسمى مجمع assembly ضمن دليل ويندوز للكمبيوتر computer's Windows directory. (على نظامي، فهو في c:\windows\assembly) حالما يكون لمجمع الدوت نت اسم قوي مطبق، بإمكانك إضافته إلى GAC بواسطة إما سحب الملف لضمن دليل المجمع assembly أو استخدام أداة الذاكرة الانتقالية للمجمع الشامل Global Assembly Cache Tool (gacutil.exe). ولا تقلق بالنسبة لملفك الوحيد إذا لم يتجاوب مع الملفات المثبتة installed files. في النسخة المثبتة حديثاً للدوت نت .NET، يمكن أن تجد تقريباً 400 ملف موجودة مسبقاً في دليل GAC، من ضمنها كل DLLs الخاصة بمكتبة فئة إطار العمل Framework Class Libraries (FCLs). تنتج لك الدوت نت .NET. من تثبيت عدة إصدارات multiple versions من مجمع ما assembly على النظام واستخدامهم في نفس الوقت (مثل هذه المعالجة تدعى التوافقية versioning). وهذه تقدم (EXE) والمكتبات (DLL) libraries للتطبيق، ومن أجل المجمعات الخاصة والمجمعات المتشاركة في GAC. افتح مجلد مجمع GAC's assembly، وضع مستكشف المجلدات Explorer folder إلى "عرض التفاصيل Details"، ومن ثم رتب بواسطة اسم المجمع Assembly Name. إذا حركة شريط التمرير للأسفل، ستري نفس الملفات المبينة عدة مرات. يبين الشكل التالي جزء من الذاكرة الانتقالية cache. يتم جدولة نسختان من "Microsoft.VisualStudio.Windows.Forms" (من ملف Microsoft.VisualStudio.Windows.Forms.dll) واحدة مع رقم النسخة 2.0 وأخرى مع رقم النسخة 9.0. على الرغم من أنه يوجد عادة علاقة واحد -إلى واحد بين الملفات والمجمعات، فيمكن أن توجد حالات عندما يتم تركيب مجمع من عدة ملفات multiple files. على سبيل المثال، يمكن لتطبيق أن يتضمن ملف صور (رسومي graphics) خارجي في عرض المجمع assembly view الخاص به. يراقب الدوت نت هذه الملفات بحرص شديد. فإذا تم تعديل modified أي من هذه الملفات، حذف deleted، أو من ناحية أخرى تشوهت maimed، فسوف تعلم بها. فيما يخص الشرح هنا، فإن باقي الفصل سيأخذ بعين الاعتبار المجمعات ذات الملف المفرد single-file assemblies.

Name	Version	Cult	Public Key Token
Microsoft.VisualStudio.Windows.Forms	9.0.0.0	...	1bf3856ad364e35
Microsoft.VisualStudio.Windows.Forms	2.0.0.0	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	2.0.0.0	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	9.0.0.0	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	7.1.4.2	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	8.0.0.0	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	9.0.0.0	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	9.0.0.0	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	7.1.4.2	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	8.0.0.0	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	9.0.0.0	...	b03f5f7f11d50a3a
Microsoft.VisualStudio.Windows.Forms	9.0.0.0	...	b03f5f7f11d50a3a

The GAC has this duplication under control

ماذا يوجد داخل المجمع What's Inside an Assembly?

ملف EXE أو DLL للمجمع هو الملف PE "تنفيذي قابل للنقل Portable Execution" القياسي، نفس تنسيق الملف المستخدم من أجل الملفات القابلة للتنفيذ في غير الدوت نت ومكتبات الكود (مثل الكثير من أي ملف ويندوز EXE أو DLL). ما يجعل ملفات الدوت نت التنفيذية والقابلة للنقل .NET PE files. مختلفة هي كل

الأجزاء المحشوة الإضافية extra stuff الموجودة داخله بشكل عام الكلمة مجمع تشير إلى جمع الأجزاء المتنوعة مع بعضها ضمن وحدة مفردة. في مجمع ما للدوت نت، هذه "الأجزاء المتنوعة" مصممة بشكل خاص للاستخدام مع الدوت نت. يحتوي ملف دوت نت القابل للنقل NET PE على ثلاث أجزاء رئيسية:

معنون "تنفيذي قابل للنقل" A PE header

مطلوب لكل ملفات PE، هذا المقطع يحدد (يعرف) المواقع بالنسبة للمقاطع الأخرى من الملف.

مقطع كود لغة ميكروسوفت الوسيطة The MSIL code section

يتم تخزين الكود الفعلي المصاحب المجمع ككود لغة ميكروسوفت الوسيطة شبه المترجم (MSIL) Semicompiled Microsoft Intermediate Language. لسوء الحظ، شريحة chip إنتل أو AMD في كميوتريك بكل وضوح بلهأه كثيراً لمعالجة كود لغة ميكروسوفت الوسيطة بشكل مباشر، لذلك يتضمن إطار الدوت نت مترجم فوري JIT compiler (just-in-time) بمكانه تحويل لغة ميكروسوفت الوسيطة MSIL إلى كود x86 الأصلي فوراً at a moment's notice.

مقطع توصيف البيانات The Metadata section

كل التفاصيل الإضافية التي يحتاجها الدوت نت للبحث ضمنه rummage through وذلك لمعرفة ما يخص مجمعك تظهر في هذا المقطع الأساسي. بعض من هذه البنود، عندما تأخذ مع بعضها، تكون كشف المجمع assembly's manifest (نوع مستند يشرح المجمع للعالم بالكامل). في القائمة التالية من عناصر توصيف البيانات metadata elements، أشرت إلى البنود التي تظهر في الكشف manifest:

اسم المجمع The name of the assembly

(جزء من الكشف) يتم تحديد هذا الجزء على تويب التطبيق application خاصيات المشروع project's properties.

رقم الإصدار (النسخة) للمجمع The version number of the assembly

(جزء من الكشف) وهو عدد من أربع أجزاء للنسخة، كما في 1.2.3.4، ربما تسألت في يوم من الأيام عن كيفية وضع هذا العدد في مشروعك الخاص. سيتم الرد على صبرك في هذا الفصل وفي مقطع "المشروع"، عندما سوف أوضح ليس فقط طريقة ولكن اثنتين عن كيفية وضع رقم نسخة المجمع.

محتوى الاسم الفوي Strong name content

(جزء من الكشف) يتضمن المفتاح العام للنشر publisher's public key.

عدادات اللغة والثقافة Culture and language settings

(جزء من الكشف) وهو مفيد خاصة عندما تحتاج لإنشاء ملفات مصدرة خاصة باللغة.

جدولة ملف المجمع Assembly file listing

(جزء من الكشف). سترى مجمعات ملف مفرد فقط اسم ملف EXE أو DLL، ولكن بعض المجمعات يمكن أن تتضمن عدة ملفات في هذا المقطع. كل الملفات في مجمع يجب أن تظهر ضمن نفس الدليل directory، أو في دليل تابع subordinate لملف المجمع الذي يحوي الكشف manifest.

معلومات الأنواع المصدرة Exported type information

(جزء من الكشف) بعض المجمعات "تصدر" بعض من أنواعها لأن تستخدم خارج التطبيق. فتفاصيل هذه الأنواع تظهر في هذا المقطع.

لمراجع References

(جزء من الكشف) ولكن في المجمعات المتعددة الملفات، كل ملف سيحتوي قائمة خاصة به من المراجع (تتضمن وصف البيانات metadata جدولاً لجميع المجمعات الخارجية external assemblies المرجعة (المشار إليها) بواسطة تطبيقك، فيما إذا كانت خاصة private أو تظهر في GAC. تشير هذه القائمة إلى أي إصدار خاص specific version، ثقافة culture، والمنصة المستهدفة platform target للمجمع الخارجي الذي يتوقعه مجمعك.

معلومات الأنواع الداخلية Internal type information

(ليست جزء من المانifest أو الكشف) كل الأنواع المعمولة (المبرمجة) في مجمعك يتم وصفها بالكامل ضمن وصف البيانات (الميتاداتا metadata). أيضاً، يظهر هنا أي وصف بيانات إضافي additional metadata تم إضافته لأنواعك من خلال ميزة مواصفة الفيجوال بيسك Visual Basic's attribute feature. في المجمعات المتعددة الملفات، تظهر العناصر الخاصة بالمانifest-specific elements فقط في الملف الرئيسي "main" للمجمع. المانifest (الكشف) هو مجموعة جزئية من الميتاداتا ضمن مجمعك. وإني أكره أن أقول أنها الجزء الأكثر أهمية للميتاداتا، ولكن هي كذلك. فالمانifest هو تعبير عام لمجمعك، والطريقة الوحيدة التي يعلم فيها الدوت نت فيما إذا كان مجمعك شرعي (صحيح. legit).

حتى قبل ظهور الدوت نت، كانت الملفات القابلة للتنفيذ executables والمكتبية libraries تحتوي بعض "الميتاداتا"، مثل رقم النسخة (الإصدار) للملف، ولكن لم تكن هذه البيانات تستخدم لإدارة الوصول بين المكونات البرمجية، ولا حتى كانت منظمة في بطريقة عامة وقابلة للتوسيع (التمديد extensible). تشمل الميتاداتا في الدوت نت كل هذه المواصفات. إن وجود كل من لغة ميكروسوفت الوسيطة MSIL والميتاداتا metadata في كل مجمع تجعل هذا الملفات قابلة للقراءة والفهم بشكل جيد بالأدوات المناسبة With the right tools، حتى ولو كنت أبداً أي فهمها، وحتى لو استطعت، فأني شخص بإمكانه، وهذا ما يقود لمشكلة كبيرة. إن الشركات تستثمر الكثير من الوقت والنفود على الجهود التي تعمل على تطوير برمجياتها، ولا تريد من أي مبرمج متطفل أن يعكس هندسة كودها ويحصل على جميع أسرارها وخوارزمياتها السرية. وللمنع القراءة العرضية casual reading لأي من تطبيق الدوت نت. عملت ميكروسوفت وجهات أخرى third parties على تضمين المبهم (المشوش obfuscators)، وهي البرامج البرمجية التي تشفر (تدمج) محتوى أي مجمع بشكل كافي مما يجعل من الصعوبة بمكان لأي شخص من أن يفهمه، ولكن ليس بالنسبة لإطار عمل الدوت نت، سأتكلم أكثر حول التشويش obfuscation فيما بعد.

الانعكاس Reflection

إنه يبدو شيء سيء أن يتمكن الناس من الوصول لمحتوى مجمع ما، وإنه لعظيم أن يتمكن الكود في مجمع من الوصول لنفسه. تتضمن الدوت نت مواصفة تدعى reflection تتيح لك تفحص محتويات مجمع ما بشكل عام تستخدم هذه المواصفة للوصول إلى الميتاداتا metadata في مجمعك الخاص، ولكنها تعمل أيضاً مع أي مجمع متاح. تظهر معظم الميزات المتعلقة بالانعكاس في فضاء الأسماء System.Reflection namespace. من خلال الانعكاس بإمكانك استخراج تقريباً كل شيء مخزن في الميتاداتا metadata للمجمع ما، ومن ضمنها تفاصيل على كل الأنواع، أعضاءها، وحتى الوسيطات المضمنة مع أعضاء الدوال. وهذا ما يجعل التشويش obfuscation هام جداً بالنسبة للبايعين vendors، بين لغة ميكروسوفت الوسيطة المترجمة compiled MSIL والميتاداتا metadata، بإمكانك افتراضياً إعادة توليد الكود المصدري بالكامل لتطبيق ما فقط من ملفه القابل للتنفيذ. ولن يكون بتلك الصعوبة بالنسبة لشخص ما متمرس في استعادة الكثير منه ضمن الفيجوال بيسك أو C#.

المجمعات والتطبيقات Assemblies and Applications

تطبيقات الدوت نت (ملفات تنفيذية EXE) هي حالة (نسخة instance) من مجمع، ولكن يمكن لتطبيق مفرد أن يحوي على عدة مجمعات، في الحقيقة، إنه غالباً ما يكون كذلك. قمت بكتابة برنامج صغير يستخدم الانعكاس لجدولة جميع المجمعات list all assemblies actively التي تم استخدامها فعلياً actively بواسطة البرنامج نفسه، وأعطيتها الاسم الافتراضي WindowsApplication1. وعندما شغلت البرنامج على نفسه، ولد القائمة التالية:

```
mscorlib
Microsoft.VisualStudio.HostingProcess.Utilities
System.Windows.Forms
```

System
System.Drawing
Microsoft.VisualStudio.HostingProcess.Utilities.Sync
Microsoft.VisualStudio.Debugger.Runtime
vshost
System.Data
System.Deployment
System.Xml
System.Core
System.Xml.Linq
System.Data.DataSetExtensions
Microsoft.VisualBasic
WindowsApplication1
System.Runtime.Remoting

كما ترى 17 مجمع من ضمنها *WindowsApplication1* البرنامج الرئيسي main program. معظم المجمع DLLs مزودة من قبل إطار العمل-framework supplied من أجل ميكروسوفت. فيجوال بيسك Microsoft.VisualBasic المجمع هو *Microsoft.VisualBasic.dll*، ومن أجل النظام System، المجمع هو *System.dll*.

كل المجمع (ما عدا مجمع البرنامج الرئيسي) هي مشاركة المكتبات libraries من GAC. يمكن للتطبيق أيضاً من دعم مجعومات خاصة private assemblies محملة من ملفات DLL محلية.

يحمل إطار عمل الدوت نت .NET Framework. هذه المجمعات لي بشكل آلي عندما يشتغل *WindowsApplication1*، إنه يستكشف أي منها فيما إذا كان هناك حاجة إليه لكي يتم تحميله وذلك بالنظر في المنافست لـ *WindowsApplication1*. عندما حمل إطار العمل كل مجمع، فإنه يتفحص ليرى فيما إذا كانت هذه المجمعات دورها تحتاج لمجمعات إضافية ليتم تحميلها، وهكذا تطبيق بسيط وحيد أصبح أرض تفرغ للمجمعات من كافة أنحاء GAC. ولكن هذا جيد، بما أن هدف الدوت نت هو إدارة هذه المجمعات جميعها.

المجمعات وفضاء الأسماء الخاص My The My Namespace and Assemblies

إن إطار العمل مع الآلاف من فئاته، يحتوي الكثير من المنطق المشحون (المحزم packaged logic) والذي بإمكانني استخدامه في برنامجي الخاص. ولكني لا أملك تذكر المجمعات العديدة وفئاتهم الكثيرة. وقد يأخذ الكثير من الوقت لتطوف ضمن توثيق مكتبة فئات إطار العمل FCL documentation. مع الكثير من الفئات المتاحة، فإني في بعض الأحيان أرتعد shudder عندما أفكر بالجهد الذي سأبذله لإيجاد الفئة الصحيحة فقط أو الميزة التي أحتاجها لإتمام بعض مهام التطوير (البرمجة).

لحسن الحظ، لست الوحيد الذي يفكر بهذه الطريقة، فميكروسوفت تتفق معي. تاريخياً، كان مبرمجي فيجوال بيسك محميين sheltered من التعقيدات في تطوير تطبيق ويندوز. وليس هذا ما يحتاجون أن يكونوا عليه، فنحن جميعاً نعرف أن مطوري فيجوال بيسك بشكل عام مفضلين على الباقي. ولكن "كان شعار الفيجوال بيسك"التحاور على: جعل تطوير ويندوز أسرع وأسهل *Make Windows Development Fast and Easy*. واستدعاء بعض الطرق المقتصرة esoteric والراسخة ضمن أجزاء bowels النظام فضاء الأسماء System namespace لتحصل على القسم الرئيسي من البيانات وهو ليس بالسهل ولا حتى بالسرير. ولاستعادة بعض البساطة والسرور التي كانت متاحة سابقاً في الفيجوال بيسك، عملت ميكروسوفت على تقديم استدعاء فضاء الأسماء My في إصدار 2005 من اللغة. يضم استدعاء فضاء الأسماء My الكثير من الميزات المفيدة من كامل FCL، ويعمل على تنظيمها في طبقات هرمية hierarchy أكثر صغراً من أجل البساطة والسهولة في الوصول. ولقد قمت بذكر باختصار My في الفصل الأول، ولكن حالياً من الجيد أخذ نظرة أقرب لما يفعله. يبدو pretend namespace مستدعي فضاء الأسماء My مشابه كثيراً لأي فضاء أسماء آخر، مثل System، System.Reflection، و System.Windows.Forms. ولكنه في الحقيقة ليس فضاء أسماء فهو مستدعي pretend.

لسبب ما، لا تستطيع استخدام الكلمة المحجوزة Imports لإنشاء اختصار للتقوعات (التشعبات) ضمن هيكله الهرمي. على الرغم من أن بعض المقاطع من هرمية التسلسل هي ديناميكية، فهي تتغير كما يتغير مشروعك، يجداول الجدول التالي العقد nodes الرئيسية للتسلسل الهرمي.

Branch التفرع الميزات المتاحة Available features

الميزات المتاحة Available features	Branch التفرع
يوفر معلومات حول التطبيق الحالي، ومن ضمنها إعدادات الخيارات الإقليمية وطرق النشر.	My.Application
يعطي معلومات أكثر حول التطبيق ومجمعاته، ومن ضمنها الاسم name والنسخة version.	My.Application.Info
يسمح لك توليد متتبع ومخرجات تسجيل دخول لأغراض تسجيل الدخول المسجلة، ويستخدم فقط مع تطبيقات العميل	My.Application.Log
يوفر إمكانية الوصول إلى المصادر العامة المتواجدة على الكمبيوتر المحلي.	My.Computer
يشغل أصوات النظام وأصوات أخرى محددة من خلال سماعات الكمبيوتر.	My.Computer.Audio
يستخلص البيانات من حافظه النظام، وينتج لك إضافة بيانات خاصة بك لحافطة في تنوع محدد مسبقاً وتنسيقات خاصة.	My.Computer.Clipboard
يحصل على تاريخ النظام الحالي والوقت المعروف بطرق متنوعة.	My.Computer.Clock
يوفر أدوات لتفحص ومعالجة الملفات والأقراص على أنظمة الملفات filesystems المحلية أو الشبكية.	My.Computer.FileSystem
يشير إلى مجلدات ويندوز الخاصة مثل المستندات Documents، سطح المكتب Desktop، والمجلدات المؤقتة Temp.	My.Computer.FileSystem.SpecialDirectories
يوفر معلومات حول نظام التشغيل المثبت ومصادر النظام المحلية الأخرى.	My.Computer.Info
يعرض الحالة الحالية من لوحة المفاتيح ومفاتيحها.	My.Computer.Keyboard
يجعل العديد من الخصائص لفارة الكمبيوتر المحلي متاحة (ممكنة).	My.Computer.Mouse
يبلغ عن الشبكة المتاحة، ويوفر ميزات للتفاعل مع تلك الشبكة.	My.Computer.Network
يتيح لك التفاعل مع مداخل (منافذ أو شقوق) النظام التسلسلية system's serial ports.	My.Computer.Ports
يقرأ ويكتب المفاتيح keys والقيم values في المسجل registry.	My.Computer.Registry
يجلب تجمع ديناميكي لكل النماذج المعرفة في التطبيق. وهذه العدة متاحة فقط لتطبيقات نماذج ويندوز.	My.Forms
يسمح لك من توليد متتبع ومخرجات تسجيل دخول للأغراض التسجيلية المسجلة. ويستخدم مع تطبيقات ASP.NET فقط.	My.Log
هذا الكائن مشابه لمخدم الصفحات الفعال Active Server Pages القديم الكائن Request. وهو متاح فقط من أجل تطبيقات ASP.NET.	My.Request
يوفر إمكانية وصول ديناميكي لتطبيق مخصص أو مصادر محلية مخصصة ومضمنة في التطبيق.	My.Resources

هذا الكائن مشابه لمخدم الصفحات الفعالة Active Server Pages القديم الكائن Response. وهو متاح فقط من أجل تطبيقات ASP.NET	My.Response
يوفر إمكانية وصول ديناميكي لنظام إعدادات التطبيق application settings system.	My.Settings
يحدد مستخدم ويندوز Windows user الحالي، ويتضمن معلومات توثيق authentication information.	My.User
يجلب تجمع لخدمات الويب المتاحة available web services لكي تستخدم في التطبيق. وهذه العقدة غير متاحة في تطبيقات ASP.NET.	My.WebServices

ينضم فضاء الأسماء الكثير من الميزات التي ستستخدمها بانتظام، من ضمنها الوصول لرقم نسخة التطبيق، بدلاً من كتابة *System.Reflection. whatever* للحصول على رقم النسخة التابعة للمكون الرئيسي "major"، بإمكانك الآن كتابة فقط:

```
My.Application.Info.Version.Major
```

هل تحتاج لقائمة من المجمعات، ولكنك كسول جداً لأن تكتب الكلمة *Reflection*؟ طيب جرب التالي:

```
My.Application.Info.LoadedAssemblies
```

هل تحتاج لمعرفة الوقت الحالي بالنسبة للتوقيت العالمي أو في بريطانيا؟ جرب التالي:

```
My.Computer.Clock.GmtTime
```

بإمكانك الاتصال بالشبكة المحلية local area network: كما يلي:

```
My.Computer.Network.IsAvailable
```

من يشغل هذا الكمبيوتر على أية حال؟

```
My.User.Name
```

تتضمن أيضاً الفيچوال بيسك الأمر "قتل Kill" الذي يسمح لك من حذف الملفات. تزيل الطريقة My.Computer.FileSystem.Delete أيضاً الملفات، ولكنها توفر خيارات إضافية من ضمنها أنها تسمح لك من إرسال ملف إلى سلة المحذوفات Recycle Bin بدلاً من فقدانها بالكامل (للأبد).

التوجيهات (التعليمات) والمجمعات Directives and Assemblies

التوجيهات هي عبارات فيجوال بيسك، ولكنها فيما بعد، تصبح ليست كذلك. التوجيهان #Const و #If يوفران تعليمات للمترجم عن كيفية معالجة مقطع من الكود المصدري للفيجوال بيسك. (التوجيه الثالث #Region، يساعد على جلب الكود المصدري ضمن الفيجوال أستوديو بشكل مرئي، ولكن ليس له تأثير على المترجم أو على التطبيق المترجم النهائي) باستخدام التوجيهات، تستطيع إخبار المترجم على تضمين أو إخراج قطعة خاصة من الكود المصدري من المشروع النهائي. لذلك هي ليست عبارات كود مصدري للفيجوال بيسك حقيقية، ولكنها متاحة فقط في الفيجوال بيسك. ولكن لماذا تريد تضمين أو إخراج كود في تطبيق ما؟ حسناً، لعدة أسباب، على سبيل المثال، النسخ المتعددة لتطبيقك، بالاعتماد على بعض الشروط، فيمكن أن تشتري نسخة "سريعة" express ونسخة "محترفة" professional " لمنتج ما. معظم الكود يكون متشابه لكلا النسختين. وأيضاً، يمكن للنسخة الاحترافية أن تتضمن ميزات غير متاحة في النسخة السريعة، أيضاً، النسخة السريعة يمكن أن تتضمن تقديم بسيط لميزة لديها استخدام أكثر تعقيد في الطبعة الاحترافية. بعض المنتجات البرمجية تحقق هذه الحاجة باستخدام شروط فيجوال بيسك القياسية.

```
If (professionalVersion = True) Then
    ShowWhizBangFeatures()
Else
    ShowLaughableFeatures()
End If
```

إن هذا يعمل بشكل جيد، ولكن ما يزال التطبيق السريع يحتوي على الميزات المحسنة جميعها. حيث أنه فقط لا يمكن الوصول إليها من أي كود، ولما يتم تضمينه على سيدي التنصيب؟ إذا استخدمت التوجيهات، تستطيع تخفيض mark down تلك المشكلة كأنها حلت. تستخدم التوجيهات التعابير الشرطية، تشبه كثيراً الشرط professionalVersion = True في مقطع الكود السابق. ولكن يتم تعريفها بواسطة العبارة #Const، ويتم استدعاء ثابت المترجم compiler constants.

```
#Const fullVersion = True
```

تعرف هذه العبارة ثابت مترجم منطقي Boolean compiler constant. ويمكن للثابت أن يستخدم فقط مع التوجيهات، إذا حاولت استخدام fullVersion في عبارة فيجوال بيسك قياسية، فإن المترجم سيذمر complain. ولكن سيعمل بشكل جيد في توجيهه #If.

```
#If (fullVersion = True) Then
    ShowWhizBangFeatures()
#Else
    ShowLaughableFeatures()
#End If
```

إن هذا الكود يبدو مشابه كثيراً لمقطع الكود السابق، ولكن مع الإشارات # المضافة. فهما يبدوان نفس الشيء ولكنهما ليسا كذلك. مع سلاسة عبارة #If، فإن الكود التالي سيتم ترجمته في التطبيق النهائي:

```
If (professionalVersion = True) Then
    ShowWhizBangFeatures()
Else
    ShowLaughableFeatures()
End If
```

نعم، كامل مقطع الكود. ولكن مع الموجهات، ما يتم تضمينه في التطبيق المترجم يعتمد على قيمة fullVersion. فإذا كانت fullVersion صواب True فإن التالي سيحصل على الترجمة في التطبيق المترجم:

```
ShowWhizBangFeatures()
```

أما العبارات الأربع الباقية ستذهب، أي ستذهب أدراج الرياح، وكأنها لا توجد أبداً. ولكن في هذه الحالة، هذا شيء جيد. فالهدف كان الحصول على نسخة من المجمع المترجم خالٍ (مجرد) من الكود الغير مطلوب (الغير مرغوب)، وهذا ما حصل.

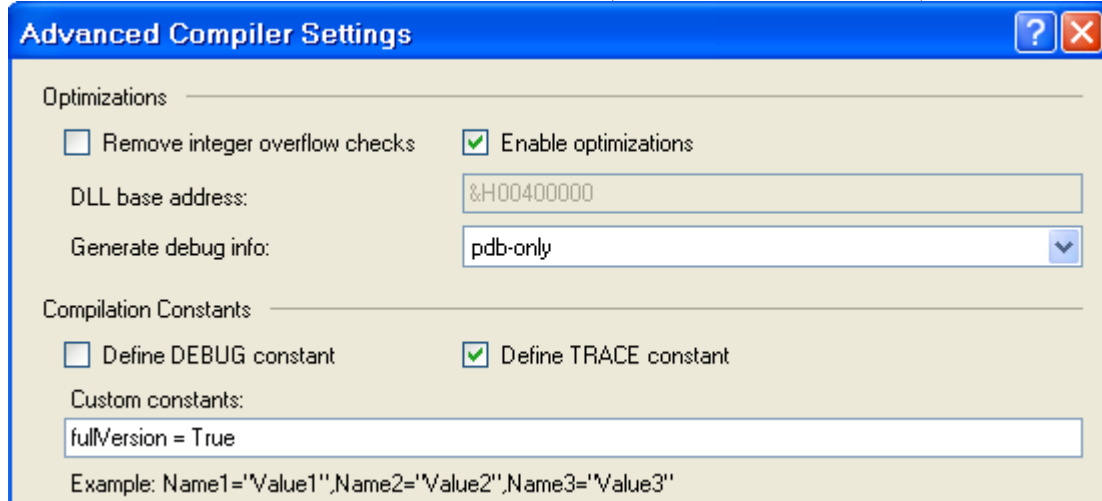
لوضع ثابت المترجم fullVersion لتوليد نسخة كاملة full version، فإنك تضمن هذا السطر عند أعلى ملف الكود المصدري والذي يتضمن مقاطع الكود #If الشرطية:

```
#Const fullVersion = True
```

عندما تكون جاهز لتوليد نسخة "سريعة"، فقط غير كل هذه الأسطر إلى نظيرها "خطأ False":

```
#Const fullVersion = False
```

شيء ما، يبدو أن تغيير هذا السطر في كل ملف كود مصدري عمل كثير، وهو كذلك. وماذا سيحدث إذا ما نسيت وضع واحد منها إلى النسخة المناسبة؟ بإمكانني إخبارك، أن هذا ليس بالجميل. لذلك توفر الفيجوال أستوديو العديد من الطرق لوضع ثوابت المترجم فوراً، وتطبيقهم لك كجزء من التطبيق. الطريقة الأكثر شيوعاً لعمل مثل هذا من خلال لوحة المترجم Compile التابعة لصفحات خصائص المشروع project properties (شاهد الشكل التالي). انقر على زر خيارات المترجم المتقدمة Advanced Compile Options، ومن ثم اعمل على إضافة ثوابت المترجم الشاملة الخاصة بك لحقل "ثوابت مخصصة Custom constants"



والآن بإضافة إما fullVersion = True أو fullVersion = False إلى هذا الحقل، تستطيع بناء نسخ مختلفة من التطبيق. يوفر مترجم الفيجوال بيسك Visual Basic compiler أيضاً ميزات تسمح لك من تثبيت صيغ ترجمة مختلفة different compile scripts لمشروعك. ولن أدخل ضمنها في هذا الكتاب، ولكن بإمكانك قراءة معلومات عنها من مستندات فيجوال أستوديو ضمن أدوات MSBuild. إذا كنت تريد هذا المستوى من الأدوات. يمكن لثوابت المترجم compiler constants أن تكون أعداد numbers أو سلاسل نصية strings بالإضافة إلى الثوابت المنطقية Booleans. بيئة تطوير الفيجوال أستوديو تعين أيضاً بعض ثوابت المترجم من أجلك. فالثوابت DEBUG و TRACE هي إما صح أو خطأ بالاعتماد على صندوق اختبار " Define DEBUG constant " و " Define TRACE constant " الذي يظهر في الشكل السابق.

مشروع Project

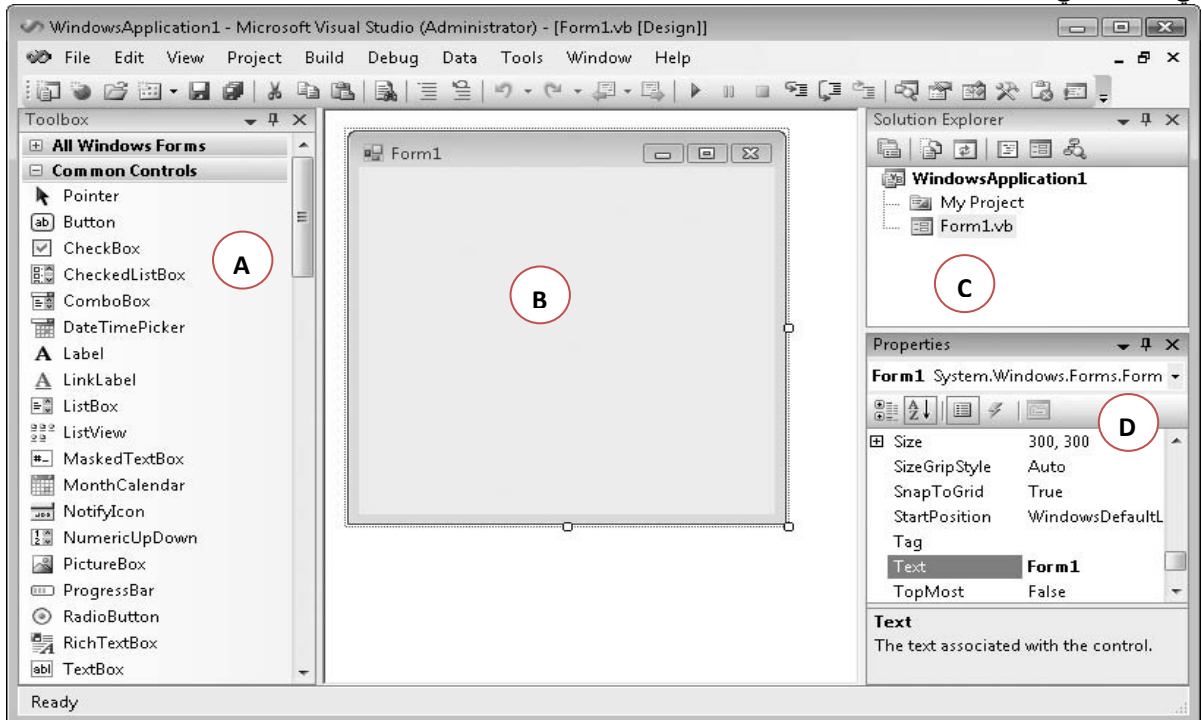
في مشروع هذا الفصل رسمياً سنتابع كتابة كود مشروع المكتبة، سنبدأ بشيء ما بسيط: بناء المعلومات الأساسية حول التطبيق والفورم التابع له، بما في ذلك رقم النسخة .version number.

هدفنا هو الوصول لفورم ساحرة تنقل conveys المعلومات الأساسية حول البرنامج.

مثل أي تطبيق فيجوال بيسك لويندوز، إنشاء هذه الفورم يتضمن خطوتين (1) إضافة الأدوات للفورم، و(2) كتابة الكود الخاص بها.

إضافة الأدوات Adding Controls

إذا كان هناك مساحة يتفوق excels فيها الفيجوال بيسك، فهي إنشاء الفورم. يمكن أن يتم إنشاء البرامج بواسطة السحب والإسقاط للأدوات الجاهزة على سطح الفورم الجاهزة. وهذا كله يتم عمله من ضمن بيئة تطوير الفيجوال أستوديو المتكاملة (IDE) Integrated Development Environment المريحة والمناسبة. كما هو مبين في الشكل التالي.



تتضمن بيئة التطوير المعروضة أربع مساحات رئيسية، والتي وسمتها بالأحرف المبيّنة في الشكل في الأعلى:

A. صندوق الأدوات toolbox:

وهو قائمة بالأدوات المضمنة وليس عرض للأدوات فقط.

B. سطح الفورم The form surface

ضع أي أداة هنا ليتم عرضها على واجهة المستخدم، والفورم " ما تراه هو ما تحصل عليه في النهاية WYSIWYG " لذلك فإنك تستطيع رؤية التصميم النهائي بالشكل الذي تصمم الفورم به.

C. مستكشف الحلول The Solution Explorer

كل الملفات المتعلقة بمشروعك تظهر هنا من أجل المشروع الحالي سترى فقط المدخلة My Project و المدخلة Form1.vb. يوجد عملياً ملفات أكثر. إذا نقرت على الزر الثاني من اليسار عند أعلى مستكشف الحلول، فإنه سيريك ملفات إضافية، ومعظم هذه الملفات يتم إدارتها بواسطة الفيچوال أستوديو من أجلك.

D. نافذة الخصائص The Properties panel

عندما تختار أداة على سطح الفورم، أو سطح الفورم نفسها، أو أي بند في مستكشف الحلول، ستظهر خصائص البند المختار في هذه المنطقة. فتستطيع تغيير إعدادات العديد من الخصائص بكتابة إعدادات جديدة. بعض الأدوات تتضمن أدوات خاصة لمساعدتك في وضع قيمة الخاصية.

سنعمل على إضافة ثمانية أدوات "عنوان labels"، وثلاث عناصر "شكل shape" و"سطر line"، ارتباطي انترنت تشعبيين web-style hyperlinks، زر button، وصورة picture لسطح الفورم.

اعمل على إعداد الفورم وذلك بضبط الخصائص التالية، انقر على سطح الفورم، ومن ثم عدل قيم هذه الخصائص باستخدام نافذة الخصائص:

الإعداد Setting	الخاصية Property
AboutProgram	(Name)
False	ControlBox
FixedDialog	FormBorderStyle
440, 311	Size
CenterScreen	StartPosition
حول مشروع المكتبة	Text

بعدها، اعمل على إضافة ثمانية أدوات "عنوان Label" لسطح الفورم من صندوق الأدوات باستخدام الأداة "Label". واستخدم القائمة التالية لوضع الخصائص بالنسبة لكل أداة عنوان.

اسم أداة العنوان الافتراضي	الخصائص التي سنعمل على تعديلها
Label1	(Name): ProgramName AutoSize: True Font/Bold: True Location: 136, 16 Text: مشروع المكتبة
Label2	(Name): ProgramVersion AutoSize: True Location: 136, 32 Text: إصدار س.ع التتقيح ص.
Label3	(Name): LicenseInfo AutoSize: False Location: 136, 48 Size: 280, 32 Text: غير مرخص
Label4	(Name): DevelopedBy AutoSize: True Location: 136, 88 Text: تمت البرمجة بواسطة:
Label5	(Name): DeveloperName AutoSize: True Location: 160, 112 Text: MHM
Label6	(Name): DeveloperBook AutoSize: True Location: 160, 128 Text: Programming Visual Basic 2008
Label7	(Name): DeveloperProject AutoSize: True Location: 160, 144 Text: In-book Project
Label8	(Name): CompanyCopyright AutoSize: True Location: 136, 208 Text: Copyright (c) 2009 by MHM.

لنضيف بعض الأسطر والمقطع الملون للفورم. تتضمن الفيچوال بيسك 6 أدوات أشكال مميزة للأسطر lines، المستطيلات rectangles، القطع الناقص ellipses والتي تستطيع تطبيقها مباشرة لسطح الفورم. أما الدوت نت فإنها لم تعد تتضمن هذه البنود، عليك إضافتها بمساعدة الكود المصدري الذي يستخدم أوامر الرسم المخصصة. ولكن نستطيع محاكاة الأسطر والمستطيلات باستخدام أداة "عنوان Label" قياسية، خالية sans من النص.

اسم أداة العنوان الافتراضي	الخصائص التي سنعمل على تعديلها
Label9	(Name): VersionDivider AutoSize: False BackColor: Black Location: 136, 80 Size: 280, 1 Text: (لا نضيف أي نص)
Label10	(Name): BackgroundSide

AutoSize: False
BackColor: White
Location: 0, 0
Size: 120, 296
Text: (لا تضيف أي نص)

(Name): BackgroundDivider Label11
AutoSize: False
BackColor: Black
Location: 120, 0
Size: 1, 296
Text: (لا تضيف أي نص)

إذا كانت أداة الأداة Label11 تحجب الصورة، انقر يمين عليها ومن القائمة المنسدلة اختر "إرسال إلى الخلفية Send To Back".
أداة "عنوان الوصلة LinkLabel" مشابهة لأكثر أدوات العنوان الأساسية Label، ولكن بإمكانك تضمين "وصلات links" في النص، مقطع قابل للنقر clickable sections والذي يكون مشابه للوصلات على صفحات الانترنت. وسنستخدمها لعرض موقع ويب وعنوان إيميل. أضف أدواتي " LinkLabel" للفرم واستخدم الإعدادات التالية لترتيب خاصيات الأدوات.

اسم أداة عنوان الوصلة الافتراضي الخاصيات المعدة

(Name): CompanyWeb LinkLabel1
AutoSize: True
LinkBehavior: HoverUnderline
Location: 160, 160
Text: <http://www.mhm.com>
(Name): CompanyEmail LinkLabel2
AutoSize: True
LinkBehavior: HoverUnderline
Location: 160, 176
Text: MHM@yahoo.com

الأداة الأخيرة التي سنضيفها هي زر والذي يسمح للمستخدم من إغلاق الفورم. أضف أداة زر Button للفرم مع الخاصيات التالية:

اسم الزر الافتراضي الخاصيات التي سيتم إعدادها
(Name): ActClose Button1
DialogResult: Cancel
Location: 344, 240
Size: 80, 24
Text: إغلاق

يمكن أن يتم تركيب الفورم بحيث أن ضغط المفتاح Esc يطلق أداة الزر على الفورم، وكان المستخدم ينقر على الزر نفسه عوضاً عن ضغط المفتاح Esc. لعمل هذا، انقر على سطح الفورم، وضع خاصيتها CancelButton إلى ActClose. لقد أجبنا هذه الخطوة حتى يتم إضافة الزر بشكل فعلي للفرم، إن الخاصية CancelButton لن تسمح أن يتم إعدادها إذا لم يكن هناك زر موجود. حسناً، والآن سنبدو الفورم مناسبة، الشيء الأخير الذي أحب أن أعمله هو وضع ترتيب التنقل، الترتيب الذي يمكن للمستخدم أن يصل به لكل حقل على الفورم عند الضغط على مفتاح "التنقل Tab" من لوحة المفاتيح. ولتحرير ترتيب التنقل، اختر سطح الفورم و اختر عرض View << الأمر "ترتيب التنقل Tab Order" من القائمة. كل أداة على الفورم التي يمكن أن تأخذ قيمة ترتيب تنقل سيكون لديها ترتيب تنقل بجانب كل منها. انقر على كل عدد أو أداة بالترتيب حتى تحصل على الترتيب الذي تريده. (شاهد الشكل التالي لترى كيف رتبنا الأدوات). أخيراً، اختر عرض View << أمر ترتيب التنقل Tab Order مرة أخرى، أو اضغط المفتاح Esc، لمغادرة عملية ترتيب التنقل.



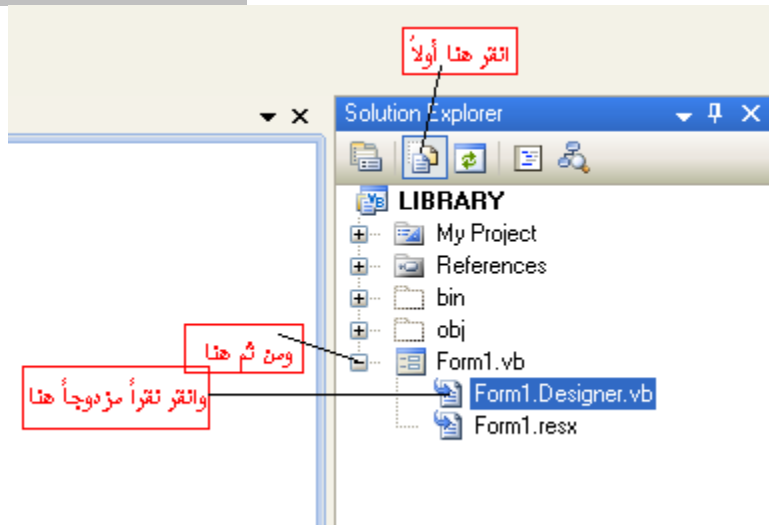
تستطيع أيضاً وضع ترتيب التنقل لكل أداة بتعديل خاصيتها TabIndex، ولكن الطريقة الأسرع التي هي التي قمنا بها للتو.

إضافة كود إلى الفورم Adding the Code to the Form

والآن، حان الوقت المناسب لإضافة بعض كود فيجوال بيسك الحقيقي. لنأخذ نظرة سريعة على مستكشف الحلول، انقر زر "أظهر جميع الملفات Show All Files" عندما تظهر جميع الملفات انقر على إشارة الزائد بجانب Form1.vb، وأخيراً، انقر مزدوج على Form1.Designer.vb (شاهد الشكل التالي).

نظراً لكونها أكثر من 200 سطر من الكود المصدري، فلن أطبعها هنا، ولكن انظر إليها بشكل عام، فعندما تعمل على سحب وإسقاط أدوات على الفورم وتعديل خصائصها، تعمل الفيچوال أستوديو على تعديل هذا الملف من أجلك. وهو جزء من فنتك للنموذج form's class (جميع النماذج هي فئات يتم اشتقاقها من System.Windows.Forms.Form). بإمكانك أن تعلم ذلك بواسطة الكلمة المحجوزة Partial عند الأعلى .

```
Partial Class AboutProgram
    Inherits System.Windows.Forms.Form
```



يحدث معظم الفعل في الإجراء InitializeComponent، عندما تنتهي من إلقاء نظرة شاملة عليه ككل أغلق مصمم الكود وارجع إلى سطح الفورم. لجعل نموذجنا نموذج حقيقي وممتع، نحتاج لعمل ثلاث أشياء:

- إظهار رقم النسخة الحقيقي للتطبيق. وسوف يتم تحديده وعرضه عندما تظهر الفورم للمرة الأولى.
- الذهاب إلى موقع الانترنت المناسب أو متلقي الايميل المناسب عند النقر على لوحة الوصلة. وهذه الأحداث يتم معاملتها في حالة الاستجابة لفعل المستخدم.
- إغلاق الفورم عندما ينقر المستخدم على زر الإغلاق. وهو أيضاً حدث مفاد من قبل المستخدم.

لنبدأ بالأسهل، إغلاق الفورم، فالأحداث هي مقاطع من الكود يتم معالجتها في حالة الاستجابة لشيء ما يحدث، وعلى الأغلب إجراء مستخدم مثل النقر بالفأرة. كل الإجراءات التي نريد أن ننجزها على هذه الفورم ستكون رد فعل (استجابة) على الحدث المطلق (المحرر). الطريقة الأسهل للوصول إلى الحدث الافتراضي للأداة هو بالنقر المزدوج عليها. جرب هذا الآن على الزر. عندما تنقر مزدوج على الزر فإن بيئة التطوير ستفتح محرر الكود (الكود المصدري)، ستضيف معالج حدث فارغ (الإجراء الفرعي ActClose_Click)

```
Public Class AboutProgram
    Private Sub ActClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ActClose.Click
```

```
End Sub
End Class
```

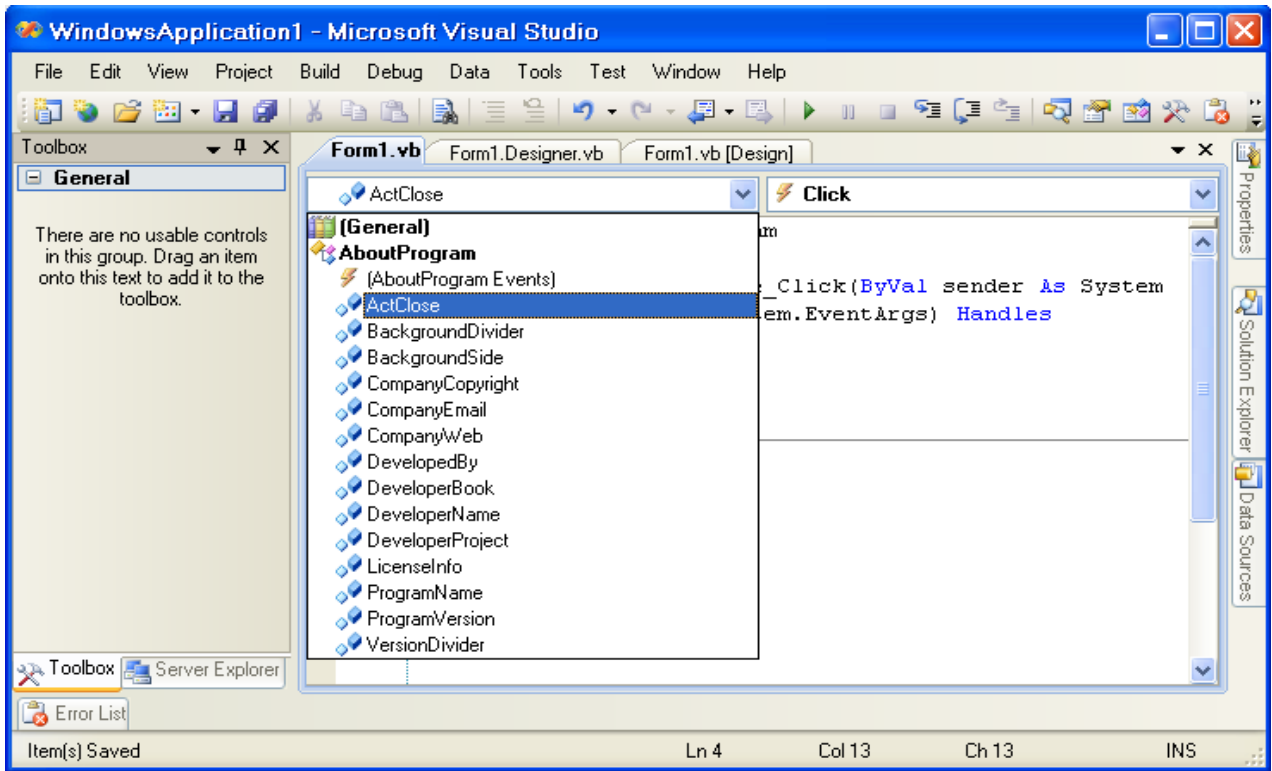
كل حدث معتمد على النماذج (وفي الحقيقة، معظم أنواع الأحداث الأخرى) في الدوت نت لديها نسبياً نفس المعاملات النسبية (1) المعامل النسبي sender والذي يشير إلى الكائن الذي أطلق هذا الحدث، و (2) المعامل النسبي e، والذي يسمح لـ sender من التزويد بأي معلومات إضافية يمكن أن تكون مفيدة في الحدث. في هذه الحالة، المعامل النسبي sender يشير إلى الزر ActClose، بما أنه الكائن الذي سيولد حدث النقر Click. وحدث نقر الزر ليس لديه أي معلومات إضافية متاحة، لذلك e هو نوع الكائن الافتراضي System.EventArgs، والذي يكون نسبياً حاجز مكان placeholder، وهو الكائن الذي منه كل أنواع المعاملات النسبية e الأكثر أهمية تستمد (تشتق منه).

اسم معالج الحدث هذا هو ActClose_Click، ولكن إذا كنت تريد تغييره إلى أي شيء تريد، فإنه لن يفقد أي شيء. ولكن عليك أن تحفظ التعبير (عالج نقر الزر intact) (Handles ActClose.Click) سليمة. وهذا هو الجزء الذي يربط معالج الحدث event handler بالحدث الفعلي. والكود اللازم لإغلاق الفورم بسيط جداً. أدخله الآن، بكتابته مباشرة:

```
Private Sub ActClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ActClose.Click
    Me.Close()
End Sub
```

تقول هذه العبارة (Me.Close())، (إني نموذج / كائن المشروع حول البرنامج، وأمر نفسي بالإغلاق). إذ شغلت البرنامج الآن (اضغط F5)، إنك تغلق الفورم بالنقر على الزر "إغلاق". بما أن نموذج المشروع هو الفورم الوحيد في التطبيق، فإن إغلاقه يؤدي بشكل آلي إلى إنهاء كامل التطبيق.

ارجع إلى البند الثاني، وصلات الانترنت. بإمكانك العودة إلى سطح الفورم والنقر المزدوج على كل لوحة وصلة لإنشاء معالج حدث لكل أداة لوحة وصلة افتراضي (في هذه الحالة، حدث LinkClicked). ولكن بإمكانك أيضاً إضافة الإجراء الفرعي لمعالج الحدث من ضمن المحرر، إما بكتابة الكود نفسه (وهذا غير مناسب) أو باستخدام قائمتي الانسدال فوق نافذة المحرر (شاهد الشكل التالي)



تظهر قائمة اسم الفئة (الكائن) في الجهة اليسارية. اختر أي مدخلة من هذه القائمة، واختر الحدث المناسب من القائمة اليمينية (قائمة اسم الطريقة Method Name). ولإضافة قالب معالج حدث لحدث `CompanyWeb's LinkClicked`، اختر أولاً `CompanyWeb` من قائمة اسم الفئة اليسارية، ومن ثم اختر من القائمة اليمينية (قائمة اسم الطريقة). وبالتالي سيظهر مقطع الكود التالي:

```
Private Sub CompanyWeb_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles CompanyWeb.LinkClicked
```

End Sub

قائمة المعاملات النسبية لهذا القالب نوعاً ما أكثر إثارة، حيث أن المعامل النسبي هو كائن من نوع: `System.Windows.Forms.LinkLabelLinkClickedEventArgs`. تسمح لك الأداة `LinkLabel` من أن يكون لديك عدة وصلات انترنت في أداة واحدة، وموزعة `interspersed` بين نصوص نظامية للمعامل النسبي خاصية "الوصلة Link" والتي تخبرك أي من الوصلات في الأداة تم النقر عليها من قبل المستخدم. بما أن لوحاتنا `labels` لها فقط وصلة واحدة، فلن نعاني في اختبارها. سنظهر فقط صفحة الويب مباشرة في أي وقت يتم النقر على الوصلة.

```
Private Sub CompanyWeb_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles CompanyWeb.LinkClicked
    ' ----- Show the company web page.
    Process.Start("http://www.mhm.com")
End Sub
```

الكائن `Process` هو جزء من فضاء الأسماء `System.Diagnostics`، وهو `Start` واحدة من أعضائه المتشاركة والتي تتيح لك من تشغيل تطبيقات ومصادر خارجية. إنك تمرر له أي عنوان انترنت URL مناسب وهو سيشغله باستخدام مستعرض المستخدم الافتراضي أو التطبيق الخاص بعنوانين الانترنت URL. لنجربه مرة أخرى مع حدث `CompanyEmail's LinkClicked`. أضف قالبه بأي طريقة تختارها ومن ثم اكتب أو أدخل الكود الذي يشغل رسالة جديدة لعنوان الايميل `email`.

```
Private Sub CompanyEmail_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles CompanyEmail.LinkClicked
    ' ----- Send email to the company.
    Process.Start("mailto:MHM@yahoo.com")
End Sub
```

الحدث الأخير الذي سنصممه هو واحد من الأحداث الأولى والتي يتم استدعاءها في دورة حياة الفورم `form`: الحدث `Load`. ويتم استدعاءه تماماً قبل أن تظهر الفورم على الشاشة. انقر مزدوج في أي مكان على سطح الفورم وهذا سينشئ قالب معالج الحدث الخاص بحدث التحميل. (بإمكانك استخدام نافذة محرر الكود ومن قائمة الفئة في الجهة اليسارية أعلى نافذة المحرر اختر الفورم ومن قائمة الطرق في أعلى اليمين اختر الطريقة `Load`).

لنضيف لهذا الحدث الكود الذي سيعرض رقم النسخة الصحيح، باستخدام `My.Application.Info.Version`، وهي نسخة (حالة) من فئة `System.Version`.

```
Private Sub AboutProgram_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    ' ----- Update the version number.
    With My.Application.Info.Version
        ProgramVersion.Text = "Version " & .Major & "." & .Minor & " Revision " & .Revision
    End With
End Sub
```

يستخدم هذا الكود العبارة `With` لتخفيض كمية الكتابة المطلوبة في عبارة الإسناد الرئيسية. داخل عبارة `With...End With`، ليس هناك حاجة لإعادة كتابة اسم الكائن الذي يظهر تماماً بعد الكلمة المحجوزة `With`، في هذه الحالة `My.Application.Info.Version`. بإمكانك الإشارة لأعضاء ذلك الكائن بكتابة النقطة فقط (.). متبوعة باسم العضو.

وضع رقم النسخة إذا شغلت البرنامج، فإنه سيعرض رقم النسخة المحددة حالياً، "1.0Revision 0" كما هو مبين في الشكل التالي:

مشروع المكتبة
version1.0revision0

سؤال، هو "أين يتم تحديد رقم النسخة هذا، وكيف استطيع تغييره؟" يتم تخزين قيم الإصدار كميئاتا (توصيف بيانات metadata) ضمن المجمع. تتضمن الفيچوال أستوديو نموذج يسمح لك تعديل الميئاتا المعلوماتي الأساسي المخزن في المجمع. للوصول للفورم، اعرض خصائص المشروع (أنقر مزدوج على My Project في مستكشف الحلول)، واختر تبويب Application، ومن ثم انقر على زر Assembly Information (شاهد الشكل التالي) يستخدم نموذج تطبيقنا رقم النسخة، والذي تم وضعه باستخدام أربع حقول نصية بجانب ملصقة إصدار المجمع. وهذه الحقول الأربع تمثل أرقام المجمع Major، Minor، Build، و Revision. بإمكانك تغييرها ومن ثم نقر موافق OK ومن ثم شغل التطبيق.

على الرغم من أن هذه النموذج مريح، فهو عبارة عن مثال آخر لكتابة بعض كود مشروعك بالفيچوال أستوديو بالنسبة عنك. كل حقل على هذا النموذج يتم حفظه في ملف الكود المصدري المضمن في مشروعك. لعرضه، تأكد من أن الزر "أظهر جميع الملفات Show All Files" مازال مختار في مستكشف الحلول Solution Explorer. مدد بند My Project باستخدام إشارة الزائد، ومن ثم انقر مزدوج على بند *AssemblyInfo.vb*. هذا الملف يعرف عدة مواصفات خاصة بالمجمع assembly-specific attributes (وسنستكشفها في الفصل 18)، إنه يتضمن المدخلات العملية التالية:

```
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices

' General information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.
' Review the values of the assembly attributes
<Assembly: AssemblyTitle("LIBRARY")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("mst")>
<Assembly: AssemblyProduct("LIBRARY")>
<Assembly: AssemblyCopyright("Copyright © mst 2009")>
<Assembly: AssemblyTrademark("")>
<Assembly: ComVisible(False)>
' The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("44d7c504-bcd1-4d64-89fe-bbf91d2e1875")>
' Version information for an assembly consists of the following four values:
'     Major Version
'     Minor Version
'     Build Number
'     Revision
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
' <Assembly: AssemblyVersion("1.0.*")>
<Assembly: AssemblyVersion("1.0.0.0")>
<Assembly: AssemblyFileVersion("1.0.0.0")>
```

كما ترى، هذا الملف تم تحديثه بواسطة القيم التي كتبها ضمن نموذج معلومات المجمع. هل تشاهد مواصفة AssemblyVersion المعرفة هنا. إذا عدلت هذه القيم، فإن التغييرات ستعكس في نموذج معلومات المجمع، وأيضاً في تطبيقك المشغل وأخيراً في المجمع المترجم. لحفظ العمل من (ملف <<File حفظ الجميع Save All))

إضافة الفورم الرئيسي Adding the Main Form

يقدر الفائدة والمواصفات الكثيرة في نموذج **AboutProgram** الذي عملناه حتى الآن فإن مثل هذا النموذج نادراً ما يكون عليه التركيز الجوهرى لتطبيق ما. في مشروع المكتبة، سيتم عرض هذه الفورم فقط عندما يتم طلبها من الفورم (النموذج) الرئيسي **Main**، لذلك لنضيف فورم رئيسي بسيط الآن. في الفيچوال أستوديو، اختر مشروع **<< Project Add Windows Form** وبندوز **Add New Item**، اختر نموذج **Windows Form** من بنود القائمة المتاحة، وسمه **MainForm.vb** قبل النقر على زر إضافة **Add**. من نافذة الخصائص خصص الخصائص التالية كما هو مبين في الجدول التالي:

الخاصية	القيمة التي سيتم إسنادها
(Name)	MainForm
FormBorderStyle	FixedSingle
MaximizeBox	False
Size	576, 459
Text	مشروع المكتبة

من صندوق الأدوات أضف أداة زر للفورم بالخصائص التالية:

الخاصية	القيمة التي سيتم إسنادها
(Name)	ActHelpAbout
Size	80, 24
Text	&حول...

الحرف الخاص & يعمل على وضع اختصار shortcut للزر. عندما تضغط على مفتاح **Alt** مع الحرف الذي يتبع "&" (في هذه الحالة حرف ح) فإن البرنامج سيتصرف وكأنك نقرت على الزر بواسطة الماوس.

انقر مزدوج على الزر وأضف الكود التالي لإجراء حدث النقر:

```
Private Sub ActHelpAbout_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ActHelpAbout.Click
    ' ----- Show the About form.
    AboutProgram.ShowDialog()
End Sub
```

هنا خصصنا مؤشر مباشر للنموذج **AboutProgram**. قبل الإصدار 2005 من الفيچوال بيسك، كان إظهار الفورم يتطلب إنشاء حالة (نسخة) من فئة الفورم قبل إظهارها:

```
(New AboutProgram).ShowDialog()
```

هذا التركيب مازال يعمل، وهي الطريقة المعتمدة إذا كنت تريد عرض عدة نسخ من نفس الفورم على الشاشة في نفس الوقت. مهما يكن التركيب **(AboutProgram.ShowDialog())** أوضح بكثير للاستخدام المفرد للنماذج، ويعكس عن قرب كيفية تقديم الفورم. عملياً، هذه العبارة تستخدم فضاء الأسماء **My**. العبارة الكاملة تبدو مشابهة للتالي:

```
My.Forms.AboutProgram.ShowDialog()
```

يسمح لك التجمع **My.Forms** من الإشارة إلى أي فورم ضمنه دون أن يكون عليك الكتابة في البداية **My.Forms**. أعضاء التجمع تمثل الحالات الافتراضية **default instances** لكل نموذج في المشروع.

هذا كل الكود الذي نحتاجه حالياً، ولكن إذا شغلت التطبيق، فإنه ما يزال يعرض فقط الفورم **AboutProgram**. هذا لأن الفورم **AboutProgram** قد تم وضعها كفورم بداية التشغيل "startup". لتبديل هذا، افتح نافذة خصائص المشروع **project's properties**، واختر تبويب **Application** تطبيق وضع حقل "فورم بداية التشغيل" **Startup form** إلى " **MainForm** " إلى

سيتم الآن عرض الفورم **AboutProgram** كفورم حوار **dialog** ("من خلال استدعاءها بالطريقة **ShowDialog**) كل فورم تتضمن الخاصية **DialogResult** والتي يتم إرجاع قيمتها بواسطة الطريقة **ShowDialog** عندما تغلق الفورم. كل زر **button** على نموذجك يمكن تركيبه لإسناد هذه الخاصية بشكل آلي ويغلق الفورم. فالزر "إغلاق" على الفورم **AboutProgram** يعمل هذا تماماً، فخاصيته **DialogResult** تم إسنادها إلى "إلغاء" **Cancel**، والتي تم إسنادها إلى خاصية الفورم **DialogResult** عندما ينقر المستخدم على زر "إغلاق". كتأثير جانبي، في أي وقت، القيمة (مالم تكن لاشيء) يتم إسنادها إلى خاصية **DialogResult** النموذج، وتغلق الفورم.

الحصيلة **upshot** من هذه الفقرة الطويلة هي أنك تستطيع الآن حذف معالج حدث النقر على زر الإغلاق، وسيبقى الزر يغلق الفورم. احذف الإجراء **ActClose_Click** من الكود المصدري للنموذج **AboutProgram**، شغل البرنامج ولاحظ ما يحصل. الزر إغلاق مازال يعمل على إغلاق النموذج **AboutProgram**، حتى بدون معالج الحدث.

بإمكانك أيضاً ترك الإجراء في مكانه، وامل على تنظيف خاصية **DialogResult** للزر "إغلاق" وأضف العبارة التالية لمعالج حدث نقر الزر:

```
Me.DialogResult = Windows.Forms.DialogResult.Cancel
```

هذه ثلاث طرق مختلفة نستطيع بها إغلاق الفورم **AboutProgram**. يوجد عدة طرق أخرى مختلفة لإتمام نفس المهمة، لذلك كن مبدعاً.

إعتماد إضافي: إضافة أيقونة Extra Credit: Adding an Icon

إضافة أيقونة مخصصة للفورم الرئيسي. اتبع الخطوات التالية:

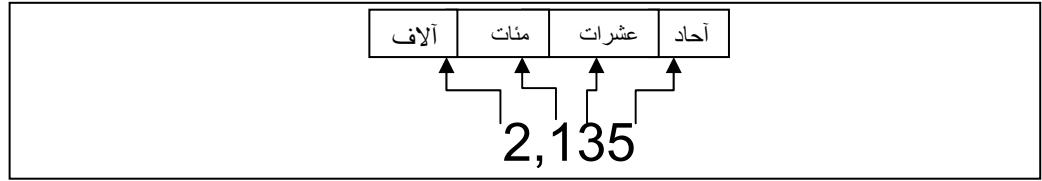
1. أعرض الفورم الرئيسي بالنقر المزدوج على البند **MainForm.vb** في مستكشف الحلول **Solution Explorer**.
2. اختر سطح الفورم
3. اختر خاصية **Icon** للفورم في نافذة الخصائص **Properties**.
4. انقر الزر "...". في هذه الخاصية، وابحث عن ملف في الدليل الفرعي للفصل الخامس، بإمكانك أيضاً استخدام أي ملف أيقونات **.ico**.

البيانات وأنواعها Data and Data Types

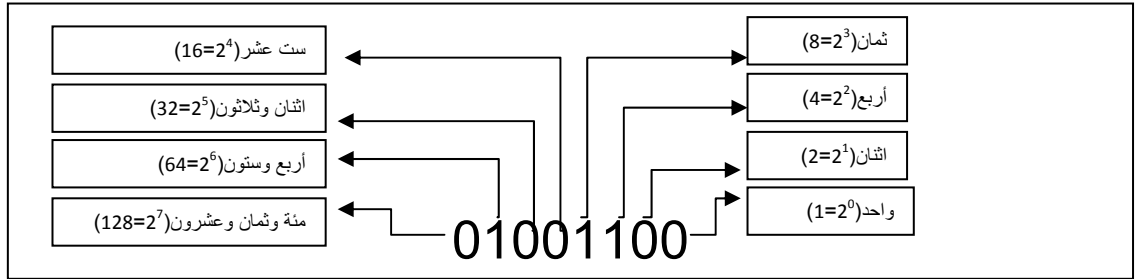
سنناقش كيف تستخدم وتعالج الفيچوال بيسك البيانات ضمن تطبيقاتك، وكيف تستطيع تسيد الأدوات التي تجعل هذه المعالجة ممكنة.

طبيعة بيانات الكمبيوتر The Nature of Computer Data

في الفصل الثاني ذكرت كيف تتحلل جميع البيانات في الكمبيوتر ضمن بتات (وحدات bits) مستقلة، النبضات الكهربائية التي تمثل إما 1 أو 0، on أو off، صواب true أو خطأ false. بما أن نظام الأرقام العشرية يتطلب أكثر من هذه القيم فقط، فإن الكمبيوتر يعمل في عالم من الثنائيات-نظام الأعداد المحدودة لرقمين 0 و 1. لحسن الحظ، من السهل جداً تمثيل الأعداد العشرية الصحيحة الأساسية باستخدام مجموعة رموز الثنائي.



لاحظ الشكل التالي الذي يبين قيم الأماكن المختلفة للأعداد المتعددة الأرقام، نفس النوع من التخطيط يمكن أن يتم استخدامه من أجل الأعداد الثنائية، فقط أسماء وقيم الموضع يتم تغييرها. من أجل الراحة، ندعو هذه المواضع بأسمائها العشرية، أو نستخدم ما يتعلق بالأساس 2. وكل هذا مبين في الشكل التالي:



موضع "8بت" 8 أرقام لعدد ثنائي.

لاستكشاف ما هو الرقم المقابل في النظام العشري، فقط اجمع الأعمدة، لنرى ذلك، يوجد واحد في الموضع الثاني والثالث، وفي الموضع السادس أيضاً، ولاشيء (صفر في باقي المواضع كما هو موضح في الرقم الثنائي 01001100 ولحساب قيمة كل موضع كما هو مبين في الشكل السابق نرفع الموضع إلى قوة (أس) الموضع بالنسبة للأساس 2) ومن ثم نضرب كل قيمة لهذا الموضع بالقيمة الموافقة في العدد الثنائي ومن ثم نجمع النتيجة لنحصل على الرقم العشري الموافق كما يلي:

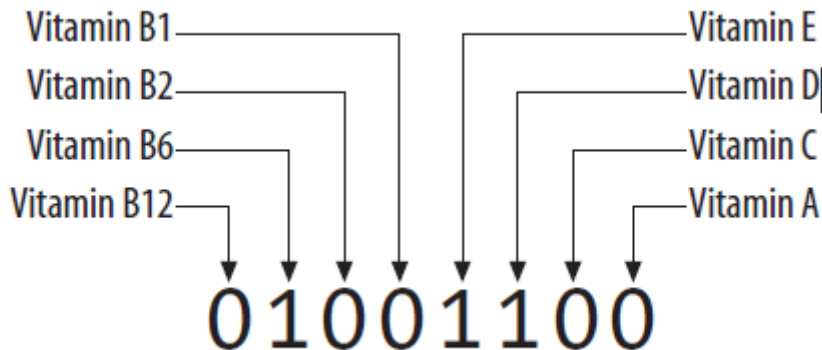
$$76 = 128 \times 0 + 64 \times 1 + 32 \times 0 + 16 \times 0 + 8 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 0$$

بما أن أي رقم ثنائي لا يمكن أن يكون أكثر من 1، فالعدد بسيط جداً. لقد بينت مثال عن 8-بت (8-رقم) ثنائي هنا والذي بدوره بإمكانه معاملة الأعداد العشرية من 0 حتى 255 ولكن إمكانك تمثيل أرقام عشرية أكبر بإضافة أرقام ثنائية أكثر. ولكن هذا مناسب من أجل القيم العددية الصحيحة، فكيف ستمثل الأعداد العشرية والكسرية؟ وماذا حول الأعداد السالبة، في أي مكان ستكون مناسبة في هذا النظام الثنائي؟ فهي ليست مجرد أعداد. فحاسبي بإمكانه معالجة البيانات النصية، مصفوفات الأعداد، الصور الرسومية، وسجلات مخصصة فكيف تم تخزين مثل تلك الأشياء في النموذج الثنائي؟ لمعالجة آلاف من النماذج البيانية، كل كمبيوتر يتضمن مجموعة صغيرة من الوحدات المتناهية في الصغر والتي تكون مناسبة بالنسبة للرياضيات، اللغة، والفن.

تنفذ الكمبيوترات "أنواع البيانات data types" لمعالجة جميع النماذج المتنوعة من البيانات لكي تكون قابلة للإدارة. يتصرف كل نوع من أنواع البيانات وكمفسر (مترجم interpreter) بين تجمع من البتات وقطعة من المعلومات التي يستخدمها لكمبيوتر بحيث يستطيع مستخدم الكمبيوتر من فهمه والانتفاع utilize منه أفضل.

أخيراً ultimately تخزن أنواع البيانات محتوياتها كبتات مستقلة من البيانات، ولكنها تختلف في كيف يتم ترجمة (تفسير) هذه البتات. تصور نوع بيانات اسمه فيتامين Vitamin والذي يحدد أي فيتامين تم تضمينه في منتج طعام. يبين الشكل التالي كيف يمكن إسناد 8-بت المستخدم سابقاً وتفسيره (ترجمته) كالفيتامينات.

مع وجود نوع بيانات، بإمكانك إسناد قيمة فيتامين لبند طعام موجود في تطبيقك. (مجرد مثال)



خذ العدد 76 الذي ناقشناه سابقاً، فهو سهل التحويل جداً إلى التمثيل الثنائي، كما في 01001100. يتضمن إطار عمل الدوت نت عدة أنواع بيانات تقوم بمثل هذا التحويل بشكل آلي، تختلف فقط بالعدد المكون من أرقام ثنائية (بتات) التي يمكنها معالجتها، في عالم الكمبيوتر، يمثل 76 أيضاً حرف من الأبجدية-الحرف الكبير L. وهذا لأن هناك نوع بيانات يؤسس قاموس بين القيم الثنائية والحروف الأبجدية (والرموز الأخرى) تستخدم برامج ويندوز ولمدة طويلة الأسكي// ASCII (الكود الأمريكي القياسي لتبادل المعلومات American Standard Code for Information Interchange) وأعدادها إلى وبالعكس كقاموس الحروف. فكيف تحول أنظمة المستندات 8-بت هذه الأعداد من 0 إلى 255 إلى كل الحروف

المتنوعة المستخدمة في اللغة الإنكليزية، ومن ضمنها علامات الترقيم punctuation والرموز (الحروف) الأخرى المتنوعة miscellaneous characters. قاموس آخر يونيكود Unicode يستخدم 16-بت من البيانات لمعاملة حوالي 65,000 حرف مختلف. تستخدم الدوت نت اليونيكود من أجل حروفها وأنواع بيانات "السلاسل النصية string". يوجد نوع بيانات آخر وهو المنطقية Boolean، والتي تستخدم بت مفرد single bit لتمثيل إما صواب True (بت ذو قيمة 1) أو خطأ False (0). القيم العددية الصحيحة السالبة Negative integers، والعشرية ذات النقطة العائمة floating-point والثابتة fixed-point، والتواريخ dates والوقت times والتي تدور خارج أنواع البيانات القاعدية على الأغلب تدار بواسطة الحاسبات وتطبيقاتها. يمكن أن يتم بناء بيانات أكثر تعقيداً من هذه الأنواع القاعدية.

البيانات في الدوت نت .NET Data in

يتم تنفيذ جميع البيانات في الدوت نت كفئات classes ضمن فضاء أسماء النظام System. أحد أنواع البيانات هذه هو System.Byte، والذي ينفذ قيمة عددية صحيحة 8-بت، تماماً كما ناقشناه سابقاً. فهو يحفظ قيم عددية صحيحة من 0 إلى 255. وهذه القيم يتم تخزينها دائماً باستخدام 8 بت من البيانات الثنائية، ولكنها تظهر بطريقة سحرية في النموذج العشري decimal form عندما تريد إحضارها. يتضمن إطار عمل الدوت نت 15 من أنواع البيانات التفسيرية الجوهرية core interpretive data types: 8 أنواع للبيانات العددية الصحيحة integers، 3 أنواع للأعداد العشرية decimal numbers، 2 نوع للبيانات الحرفية character data، نوع بيانات موحد للتواريخ والوقت dates and times، وأخيراً نوع البيانات المنطقية Boolean.

أنواع البيانات العددية الصحيحة Integer Data Types

تعتمد على عدد أنواع البيانات المتاحة (8 من بين الأنواع الأساسية الـ 15)، يمكن أن نعتقد أن معظم المبرمجين يعملون طوال اليوم مع الأعداد الصحيحة، ويمكن أن تكون محق. سواء كانت بيانات المستخدم الفعلية أو عداد حلقة أو كود مرحلي أو طرق تخزين لأنواع البيانات المعدودة (المحصية)، تظهر البيانات العددية الصحيحة في كل مكان ضمن كود الدوت نت. يعتمد مجال القيم بالنسبة لنوع البيانات العددية الصحيحة مباشرة على العدد من الأرقام الثنائية المدارة بواسطة نوع البيانات، كلما كانت الأرقام أكثر، كان المجال أكبر. وأيضاً النصف من أنواع البيانات العددية الصحيحة تخزن كلاً من القيم الموجبة والسالبة (تدعى الأعداد الصحيحة ذات إشارة "signed" integers)، بينما النصف الباقي يدعم فقط الأعداد الموجبة (بدون إشارة unsigned). تجدول القائمة التالية أنواع البيانات العددية الصحيحة المضمنة في الدوت نت، وما يصاحبها من مجالات.

نوع بيانات الدوت نت .NET data	البت Bits	النمط Style	مجال القيم Range of values
System.Byte	8	Unsigned	0 to 255
System.SByte	8	Signed	-128 to 127
System.Int16	16	Signed	-32,768 to 32,767
System.UInt16	16	Unsigned	0 to 65,535
System.Int32	32	Signed	-2,147,483,648 to 2,147,483,647
System.UInt32	32	Unsigned	0 to 4,294,967,295
System.Int64	64	Signed	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
System.UInt64	64	Unsigned	0 to 18,446,744,073,709,551,615

أنظر إلى هذه الأنواع بطريقة أخرى، وبين الجدول التالي العلاقة بين الأنواع وعدد البتات التابع لكل منها ونمط المجال:

	8-bits	16-bits	32-bits	64-bits
Signed	SByte	Int16	Int32	Int64
Unsigned	Byte	UInt16	UInt32	UInt64

أنواع البيانات العشرية Decimal Data Types

عندما جاءت الكسور fractions، لم تكن في البداية سيئة، بما أن العديد منها يمكن أن يتم تحويله إلى نموذج عددي بسيط بإدخال الفاصلة (النقطة) العشرية في العدد: 2/1 يصبح 0.5، و 4/1 يصبح أطول ولكن يبقى أصغر 0.25، و 3/1 يصبح ... 0.333333 ما الذي يحدث هنا؟ أستطيع كتابة كل هذه 3. وإلا سيصبح الكتاب 2,000 صفحة أو أكثر. أخيراً، اكتشف الناس أنه في عدة حالات، فإن كتابة كل 3 لا يستحق العناء، لذلك فإنهم توقفوا عند بعض النقاط. كما في 0.33333333. هذا ليس بالدقة الكاملة، ولكنه جيد بما فيه الكفاية، وهذا هو الحال بالنسبة للقيم العشرية المعتمدة في الكمبيوتر computer-based decimal values. يمكن أن تمتلك دقة تامة حتى نقطة معينة. بعد ذلك، يقع عليك اختيار ما هو جيد بشكل كافي بالنسبة لك. يتضمن إطار عمل الدوت نت ثلاث أنواع بيانات عشرية. اثنان منهما يقبلان دقة محددة بدلاً من المجال الكبير من القيم. والثالث لديه دقة تامة (مثالية perfect accuracy) ولكن مجاله أكثر تحديداً. يبين الجدول التالي هذه الأنواع الثلاث.

نوع بيانات الدوت نت .NET data type	الدقة Accuracy	المجال Range	وصف Description
System.Decimal	كاملة (Perfect)	محدود	يدعم تقريباً حوالي 28 أرقام موحدة على كلا جانبي النقطة العشرية. ويمكن أن يحذف بعد هذا الموضع الرقم المتاح الأخير. فدقته ضمن هذه الأرقام. وبسبب هذا، فإنه مناسب للعمل مع العملة. كلما كانت الأرقام التي لديك أكثر على يسار الفاصلة، كلما قلت لديك الأرقام المتاحة على يمين الفاصلة العشرية، والعكس بالعكس vice versa، بالنسبة للأعداد التي ليس لديها جزء عشري يكون المجال -79,228,162,514,264,337,593,543,950,335 to 79,228,162,514,264,337,593,543,950,335. (That's 29 digits)
System.Single	غير كاملة	كبير	بالنسبة للأعداد الصفرية فقط (0) على يسار الفاصلة العشرية، المجال يكون: -0.00000000000000000000000000000001 to 0.00000000000000000000000000000001
			يوفر مجال أكبر من السابق ولكن لديه بعض المشاكل في الدقة بعض الأحيان عندما تقوم بحساب معقد وأنت تعرف أن النتيجة ستكون صفر، فإن النتيجة

يمكن أن تكون 0.0000000000023. هي قريبة للصفر، ولكنها ليست صفر
بالضبط، بإمكانك استخدام
أعداد كبيرة جداً أو صغيرة جداً، للقيم السالبة، المجال هو: من $-3.402823E+38$
إلى $-1.401298E-45$. أما من أجل القيم الموجبة، فمجاله من $1.401298E-45$
إلى $3.402823E+38$
يشبه سابقه، ولكن بمجال أكبر. من أجل القيم السالبة، المجال هو: من
 $-1.79769313486231E+308$ إلى $-4.94065645841247E-324$
ومن أجل القيم الموجبة، المجال هو: من
 $1.79769313486232E+308$ إلى $4.94065645841247E-324$

System.Double غير كاملة ضم

أنواع البيانات الحرفية Character Data Types

يتضمن إطار العمل نوعين من البيانات المتعلقة بالنصوص، System.Char و System.String. نوع البيانات الحرفية Char يحفظ حرف مفرد، لا أقل ولا أكثر. ففي 16 بت، يحفظ حرف من أي من آلاف حروف اليونيكود.
ونوع بيانات السلاسل الحرفية String تسمح حتى حوالي 2 بليون حرف من حروف يونيكود، وأن يتم صفها مع بعضها البعض في مقطع نصي طويل. السلاسل الحرفية في الدوت نت ثابتة (لا يمكن تغييرها immutable)، فحالما تعمل على إنشاء سلسلة نصية ما، فلا يمكن تغييرها بأي طريقة. إذا أردت إضافة نص إلى سلسلة حرفية موجودة، فإن الدوت نت سيعمل على إنشاء وسمة (علامة) سلسلة حرفية جديدة مبنية من سلسلتين حرفيتين ثابتتين أصليتين.
على الرغم من أن الحرفية والسلسلة الحرفية أنواع بيانات مختلفة، بإمكانك بسهولة نقل البيانات بينهما، بما أنهما معتمدان على حروف اليونيكود الأساسية.

نوع بيانات الوقت والتاريخ Date and Time Data Type

يسمح لك نوع البيانات System.DateTime من تخزين إما قيم تاريخ أو وقت (أو كلاهما) كبيانات. داخلياً، إن DateTime هو عبارة عن عداد عددي صحيح بسيط يعرض تنسيق تاريخ أو وقت محول عند الحاجة. وكعدد، فإنه بحسب عدد "الدقات ticks" منذ 12:00 صباحاً 1 كانون الثاني، سنة 1 بعد الميلاد (12:00 a.m. on January 1, 1 AD). كل دقة "tick" هي تماماً 100 نانوثانية، لذلك فهو دقيق جداً. التاريخ الأعظمي المسموح به هو 31 كانون الأول، 9999، في التقويم الغريغوري (الكاثوليكي).

نوع البيانات المنطقية Boolean Data Type

يمثل نوع البيانات System.Boolean ماهية essence بيانات الكمبيوتر الحقيقية: وهو البت bit. فهو يحفظ واحد من قيمتين ممكنتين: صواب True أو خطأ False.
من المدهش Shockingly أن يستخدم نوع البيانات الذي يحتاج عملياً ما بين 2 و 4 بايت (32 بت) من مساحة (فراغ) بيانات لحفظ مسار بت مفرد من البيانات. مما أظهر أهمية القيم المنطقية في البرامج. وكطور، فإنك دائماً تختبر شروط متنوعة لترى أي منها يتحقق قبل أن تعالج مقطع من كود. كل هذه الشروط تختصر boil down أخيراً لقيم منطقية Boolean وعمليات operations. حتى أن للدوت نت طرق لنقل البيانات بسهولة بين القيم العددية الصحيحة ونوع البيانات المنطقية. في مثل هذا التحويلات، يصبح الصفر (0) خطأ وتصبح باقي جميع القيم الممكنة صواب. وعند الانتقال من المنطقي إلى ما يكافئه من القيم العددية الصحيحة فإن خطأ يصبح (0) وصواب يصبح (-1). (إذا استخدمت لغة سي شارب C#، ستجد أنها تحول صواب إلى 1 وليس -1، داخلياً في الدوت نت، يتم تحويل صواب إلى 1، ولكن لأسباب تاريخية، تستخدم الفيچوال بيسك -1. وهذا الاختلاف عادة ليس بالمشكلة ما لم تخزن القيم المنطقية كأعداد صحيحة في ملف على قرص ما وتوقع من برامج فيجوال بيسك وسي شارب C# أن تفسر interpret البيانات بشكل صحيح).

فئة الكائن The System.Object Class

لقد علمت مسبقاً أن الدوت نت بيئة تطوير كائنية التوجه (موجهة بالكائنات object-oriented development environment). كل شيء في الدوت نت - كل الكود وكل البيانات - يتم اشتقاقها (مستمدة) من فئة قاعدية واحدة: System.Object. بنفسها هذه الفئة ليس لديها الكثير من الميزات. يمكن أن تخبرك عن اسمها، نوعها، وفيما إذا حالتين (نسختين) من كائن هما في الواقع كائن واحد ونفس الكائن. غير ذلك فهي ليست مفيدة للكثير ما عدا أنه يتم استخدامها كنقطة بداية بالنسبة لجميع الفئات والأنواع. لأن جميع الفئات في الدوت نت - ومن ضمنها كل أنواع البيانات - تستمد (تشتق) من System.Object، تستطيع معاملة حالة من أي فئة (أو نوع بيانات) ككائن. وستتذكر البيانات ما هو النوع الحقيقي الذي هي عليه، لذلك إذا كان لديك System.Int32 موضوع ك System.Object، تستطيع تغييره إلى System.Int32 فيما بعد.

الأنواع ذات القيمة والأنواع المرجعية Value Types and Reference Types

ارجع إلى الفصل الأول، لقد قرأت الاختلاف بين الأنواع ذات القيمة والأنواع المرجعية. فالأنواع ذات القيمة هي خانات (حاويات buckets) تحتوي على البيانات الحقيقية، والأنواع المرجعية تحتوي تعليمات instructions عن المكان الذي تستطيع إيجاد البيانات الحقيقية. بشكل عام، تحتوي الأنواع ذات القيمة قيم بيانات صغيرة وبسيطة، بينما الأنواع المرجعية تشير إلى مقاطع بيانات معقدة وكبيرة. وهذا ليس دائماً صحيح، ولكن من أجل أغلب البيانات التي تعمل معها، فإنه سيكون صحيحاً. إن System.Object هو نوع مرجعي والذي منه يتم اشتقاق جميع الأنواع والفئات الأخرى. وهذا يتضمن جميع أنواع البيانات الأساسية، لذلك ربما تفكر أنها جميعها أنواع مرجعية أيضاً. ولكن يوجد فئة أخرى معلقة بين System.Object ومعظم أنواع بيانات الفيچوال بيسك. وهذه الفئة System.ValueType، تنفذ التعريف الأساسي والاستخدام لنوع ما ذو قيمة. يبين الجدول التالي بعض الاختلافات بين الأنواع المرجعية وذات القيمة.

الأنواع ذات القيمة: Value types

1. تشتق في النهاية من System.ValueType، والذي بدوره يشتق من System.Object.
2. أنواع البيانات الأساسية المشتقة: المنطقية Boolean، البايت Byte، الحرفي Char، التاريخ والوقت DateTime، العشري Decimal، المزدوج Double، القصير Int16، الانتغر Int32، الطويل Int64، البايت بإشارة SByte، المفرد Single، القصير بدون إشارة UInt16، الانتغر بدون إشارة UInt32، الطويل بدون إشارة UInt64.
3. توفر دعم لتراكيب الفيچوال بيسك structures.
4. يتم اشتقاق العدادات Enumerations كما يلي: System.Enum << System.ValueType << System.Object.
5. لا يمكن اشتقاق الأنواع ذات القيمة من فئات أخرى أو تراكيب ولا حتى يمكن للتراكيب أن تشتق منها.
6. لا يمكن وضع الحالات (النسخ) إلى "الشيء Nothing" (استخدام النوع "قابل لأن يكون بدون قيمة nullable" يتغلب على هذا القصور (limitation)).
7. يمكن للحالات أن تحتوي فقط بيانات من نوع محدد. على سبيل المثال، نسخ (حالات) System.Int32 يمكن أن تحتوي فقط بيانات عددية صحيحة integer بإشارة 32-bit.

8.التبديد بالكامل يتم بعمليات مجمع المخلفات.NET garbage collection..

أنواع المرجعية Reference types

- 1.في النهاية ،تشتق من System.Object.
 - 2.أنواع البيانات الأساسية المشتقة:السلاسل الحرفية String.
 - 3.توفر دعم لفئات الفيچوال بيسك "classes" Visual Basic.
 - 4.تستخدم التفاويض Delegates كمراجع(مؤشرات) لطرق الفئة class methods ،وتشتق كما يلي: System.Delegate << System.Object،أحد أنواع التفاويض "multicast delegate" يتم اشتقاقه من خلال: System.MulticastDelegate .
 - 5.يمكن أن يتم اشتقاق الأنواع المرجعية من فئات أخرى،ويمكن أن تستخدم كفئات قاعدية (يمكن أن يتم الاشتقاق منها أيضا)
 - 6.يمكن وضع الحالات(النسخ)إلى "الاشيء Nothing"
 - 7.تشير عادة الحالات(النسخ) لبيانات النوع الذي عرفها(المعرفة منه)،ولكن يمكن لحالة أن تشير أيضاً إلى النوع المشتق.على سبيل المثال،حالة من System.String يمكن أن تشير لأي بيانات تستخدم System.String كفئة قاعدية base class.
 - 8.يتم تدميرها من خلال مجمع النفايات garbage collection.
- (بالإضافة للفئات والتراكيب،تعرف الفيچوال بيسك أيضاً "الوحدات البرمجية modules".توثيق الدوت نت يحدد الوحدات البرمجية modules على أنها أنواع مرجعية،ولكنك لا تستطيع إنشاء حالات(نسخinstances) منها.)
- يمكن للأنواع ذات القيمة أن تحتوي فقط بيانات من نوعها الخاص ،ولكن الأنواع المرجعية يمكن أن تشير إلى الحالات المشتقة.وهذا هام في الدوت نت،حيث أنه تم تصميمها لتسمح لحالة System.Object من الإشارة إلى أي بيانات في تطبيق ما.نسخ(حالات) System.Object يمكن أن تشير إلى إما لبيانات لنوع ذو قيمة أو لنوع مرجعي.هذا سهل الفهم بالنسبة للأنواع المرجعية بما أن تلك الحالة(النسخة)ستشير فقط لبعض الحالات المشتقة من نفسها.ولكن إذا أسندت نوع ذو قيمة إلى System.Objec المرجعي ،فعلى الدوت نت أن يعلم(يضع علامة mark)تلك الحالة بطريقة خاصة ليشير إلى أن النوع المرجعي يحتوي نوع ذو قيمة. وهذه المعالجة(العملية) تدعى (الصندوقة boxing)وعكس العملية يطلق عليه(unboxing).على الرغم من أن الصندوقة boxingمفيدة،وفي بعض الأحيان أساسية،فإنها تأتي بأداء فعلي يزعج بشدة substantial performance hit.

أنواع بيانات الفيچوال بيسك Visual Basic Data Types

جميع أنواع البيانات المنفذة في لغة الفيچوال بيسك هي أغلفة لأنواع بيانات الدوت نت الجوهرية.فقد تم تغيير فقط بعض الأسماء للمحافظة على البساطة. يبين الجدول أنواع بيانات الفيچوال بيسك وما يكافئها في الدوت نت .

Visual Basic type	.NET type
Boolean	System.Boolean
Byte	System.Byte
Char	System.Char
Date	System.DateTime
Decimal	System.Decimal
Double	System.Double
Integer	System.Int32
Long	System.Int64
Object	System.Object
SByte	System.SByte
Short	System.Int16
Single	System.Single
String	System.String
UInteger	System.UInt32
ULong	System.UInt64
UShort	System.UInt16

كل أنواع بيانات الفيچوال بيسك قابلة للتبادل بالكامل مع ما يكافئها في الدوت نت.وأي حالة من System.Int32يمكن أن يتم معاملتها وكأنها حالة من Integer،والعكس بالعكس.

Literals

الطريقة الأسرع في تضمين قيم لنوع بيانات معين في كود الفيچوال بيسك هي استخدام المحارف (الحرفية). ولقد رأيت سابقاً في الفصل الأول من هذا الكتاب كيفية استخدام الحرفية في مثال المشروع وكما يلي:

```
MsgBox("Hello, World!")
```

هذا الاستدعاء لدالة MsgBox يتضمن محارف السلسلة الحرفية. محارف السلسلة الحرفية تظهر دائماً ضمن علامتي اقتباس مضاعفة double quotes. ومعظم المحارف العددية تظهر مع الحرف المعرف لنوع البيانات data-type-defining character في نهاية المحرف literal. ولكن يوجد تنوع آخر، يبين الجدول التالي قائمة بقيم المحارف المختلفة التي بإمكانك تضمينها في كودك.

نوع المحرف Literal type	مثال Example	الوصف Description
Boolean	True	يدعم نوع البيانات المنطقية قيمتين حرفيتين: صواب True وخطأ False.
Char	"Q"c	محارف حرفية مفردة تظهر ضمن علامتي اقتباس مضاعفة مع التذييل بالحرف c. المحارف من النوع Char ليست نفس المحارف الحرفية المفردة من نوع السلاسل الحرفية String.
Date	#11/7/2005#	تظهر المحارف التاريخية والوقت بين مجموعة من العلامات العددية. بإمكانك تضمين التاريخ، الأوقات، أو دمج منهما. يمكن لقيم التاريخ والوقت أن تكون بأي تنسيق مميز بواسطة نظام التشغيل ويندوز، على الرغم من أن الفيچوال أستوديو يمكن أن تعيد تنسيق محرف التاريخ وذلك للتوافق مع التنسيقات القياسية الخاصة بها.
Decimal	123.45D, 123.45@	القيم ذات النقطة العائمة Floating-point من النوع العشري Decimal يتم إتباعها بـ D، أو بالرمز @.
Double	123.45#، 123.45R	القيم ذات النقطة العائمة من النوع Double يتم إتباعها بالحرف الكبير R، أو بالرمز #. أيضاً، إذا استخدمت المحارف العددية مع الجزء العشري، ولكن بدون تذييل حرف لنوع البيانات، فإن ذلك المحرف سيتم إعطائه نوع البيانات من النوع المزدوج Double.
Hexadecimal	&HABCD	بإمكانك تضمين المحارف الست عشرية hexadecimal في كودك ببدء القيمة بالتعاقب الحرفي "&H"، متبوعة بالأرقام الست عشرية hex digits.
Integer	123.45%، 123.45I	القيم العددية الصحيحة من النوع Integer تكون متبوعة بالحرف الكبير I، أو بالرمز %. أيضاً، إذا استخدمت محارف عددية تقع ضمن مجال ال-Integer، ولكن دون تذييل حرف يبين نوع البيانات، فإن ذلك المحرف سيتم إعطائه النوع من نوع بيانات Integer.
Long	123.45&، 123.45L	قيم عددية صحيحة من النوع الطويل Long، تكون متبوعة بالحرف الكبير L، أو بالرمز &. أيضاً، إذا استخدمت محارف عددية تقع ضمن مجال ال-Long، ولكن دون تذييل حرف يبين نوع البيانات، فإن ذلك المحرف سيتم إعطائه النوع من نوع بيانات Long.
Octal	&O7654	بإمكانك تضمين المحارف الثمانية octal في كودك ببدء القيمة بالتسلسل الحرفي "&O"، متبوعاً بالأرقام الثمانية octal digits.
Short	123.45S	القيم الصحيحة من النوع القصير Short تكون متبوعة بالحرف الكبير S.
Single	123.45!، 123.45F	القيم ذات النقطة العائمة من النوع المفرد Single تكون متبوعة بالحرف الكبير F أو بالرمز !.
String	"A ""B"" C"	تظهر محارف السلاسل النصية String ضمن مجموعة من علامات الاقتباس المضاعفة، دون أن يتبع علامة إغلاق الاقتباس أي حرف خاص. استخدم رمزي اقتباس ضمن محرف السلسلة الحرفية إذا أردت تضمين علامة اقتباس مفردة.

الثوابت Constants

المحارف جيدة، ولكن ليست دائماً واضحة بما تعنيه. توفر الثوابت طريقة لإسناد أسماء ذات معنى للقيم الحرفية (المحارف)، ويتم معاملتها تماماً كالقيم الحرفية (المحارف)، ولكن ذات تعريف وحيد، يمكن أن يتم استخدامها مرة بعد مرة في كودك. وكل استخدام لقيمة حرفية، حتى ولو كان لديها نفس القيمة، فهي تمثل تعريف جلي وحالة من القيمة. في الفيچوال بيسك يتم تعريف الثوابت باستخدام الكلمة المحجوزة Const.

```
Const MonthsInYear As Short = 12
```

تعريف الثابت لديه الأجزاء التالية:

الاسم name

في هذه الحالة اسم الثابت هو: MonthsInYear

نوع البيانات data type

يعرف هذا المثال ثابت من النوع القصير Short. نوع البيانات دائماً تأتي بعد الكلمة المحجوزة As. إذا لم تضع الكلمة المحجوزة As، فإن نوع بيانات الثابت ستكون من نوع المحارف التي تم إسنادها له. وأنواع البيانات التالية فقط يمكن أن يتم استخدامها من أجل الثوابت: المنطقية Boolean، البايت Byte، الحرفي Char، التاريخ Date، العشري Decimal، المزدوج Double، العددي الصحيح Integer، الطويل Long، الكائن Object، البايت بإشارة SByte، القصير Short، المفرد Single، السلسلة النصية String، العددي الصحيح بدون إشارة UInteger، الطويل بدون إشارة ULong، القصير بدون إشارة UShort، أو اسم عداد ما enumeration.

المعهد initializer

المعهد المسند هنا هو 12. حالما يتم إسنادها، فإن هذه القيمة لا يمكن أن يتم تعديلها بينما يكون كودك في وضع التشغيل. والثوابت دائماً أنواع ذات قيمة، وليست من النوع المرجعي. والممهديات عادة تكون محارف بسيطة، ولكن بإمكانك دائماً تضمين حسابات بسيطة:

```
Const Seven As Integer = 3 + 4
```

مستوى الوصول access level

يمثل تعريف الثابت تنسيق مثالي MonthsInYear لتعريف ثابت مضمن ضمن كود إجراء. بإمكانك أيضاً تعريف ثوابت خارج إطار الإجراءات، ولكن يبقى ضمن فئة أو نوع آخر. عندما تعمل هذا، عادة تعمل على إضافة الكلمة المحجوزة لمحدد الوصول access modifier تماماً قبل الكلمة المحجوزة Const. وهذه الكلمة المحجوزة تشير إلى الكيفية التي بها سيستخدم معظم الكود هذا الثابت. وسأشرح محددات الوصول access modifier فيما بعد، ولكن الثوابت المعرفة ضمن إجراء يمكن استخدامها فقط ضمن ذلك الإجراء. حالما تعرف ثابت، بإمكانك استخدامه في أي مكان تود استخدام ما يكافئه من المحارف.

```
Const GreatGreeting As String = "Hello, World!"
```

```
...Later...
```

```
MsgBox(GreatGreeting)
```

العدادات Enumerations

العدادات واحدة من أنواع الدوت نت الجوهرية، تتيح لك تجميع مسميات، وقيم عددية صحيحة متعلقة بهذه المسميات كمجموعة. حالما تربطهما ببعضهما، يمكن للعداد أن يتم استخدامه مثل أي نوع بيانات آخر، بإمكانك إنشاء متغيرات بحيث تكون حالات خاصة من عداد. العدادات هي تشييد متعدد الأسطر، يعرف السطر الأول الاسم ونوع البيانات الملقاة تحت العداد. كل عضو عداد يظهر في سطر منفصل، يتم الإنهاء بسطر End Enum الذي في النهاية يغلق العداد.

Enum CarType As Integer

```
Sedan = 1
StationWagon = 2
Truck = 3
SUV = 4
Other = 5
```

End Enum

يتضمن السطر الأول التصريح عن العداد بالكلمة المحجوزة Enum، واسم العداد (CarType)، ونوع البيانات المضمن (Integer). والتعبير ك As لنوع البيانات هو اختياري، فإذا لم تضعه، فإن النوع الافتراضي للعداد هو العدد الصحيح Integer. فإذا زودت بنوع البيانات، فيجب أن يكون واحد من الأنواع التالية: البايت Byte، العددي الصحيح Integer، الطويل Long، البايت بإشارة SByte، القصير Short، العددي الصحيح بدون إشارة UInteger، الطويل بدون إشارة ULong، أو القصير بدون إشارة UShort.

كل عضو من العداد (السطر من 2 إلى 6) يجب أن يتضمن على الأقل اسم عضو (مثل Sedan). بإمكانك بشكل اختياري إسناد قيمة عددية لبعض أو لجميع الأعضاء، كما فعلنا بالمثال. فإذا ما نقص عضو إسناد ما، فيتم وضعه بزيادة واحد عن العضو السابق. أما إذا لم يتم إسناد أي عضو لقيمة، فيتم إسناد الأول لـ 0، والثالي لـ 1، وهكذا. حالما يتم التعريف، تتصرف أعضاء العداد بكثير من الشبه للثوابت العددية الصحيحة، بإمكانك استخدامها في أي مكان تريد استخدام عادةً الثوابت أو المحارف. عندما تشير لأعضاء عداد في كودك، ضمن كلاً من اسم العداد واسم العضو.

CarType.Sedan

العبارة Enum لا يمكن استخدامها ضمن طريقة method أو إجراء procedure. بدلاً من ذلك، إنك تعرف عداد كعضو من نوع (فئة) class، تركيب structure، أو وحدة برمجية module)، أو كنوع خاص قائم بذاته، تماماً مثل الفئة. يتضمن إطار عمل الدوت نت العديد من العدادات المفيدة المعرفة مسبقاً معدة intended للاستخدام مع ميزات features إطار العمل. على سبيل المثال، العداد System.DayOfWeek يتضمن أعضاء لكل أيام الأسبوع.

Variables المتغيرات

المحارف مفيدة وجيدة، والثوابت والعدادات أفضل، ولكن ولا واحد منهم يمكن أن يتم تبديله حالما يبدأ البرنامج. وهذا ما يميل لجعل تطبيقك جامد وغير مرن. المتغيرات هي حاويات مسماة من أجل البيانات، مثل الثوابت تماماً، ولكن يمكن أن يتم تعديل محتوياتها أثناء تشغيل التطبيق. وأيضاً، يمكن أن تحتوي على أنواع ذات قيمة وأنواع مرجعية. إليك التركيب الأساسي لتعريف متغير جديد:

Dim customerName As String

الكلمة المحجوزة Dim أصلاً من الكلمة *dimension* - تعرف متغير جديد، في هذه الحالة، سمي المتغير customerName مع نوع البيانات من نوع السلسلة الحرفية String. هذه الحاوية المسماة جاهزة لحفظ أي قيمة نصية، أسند له محارف نصية string literals، أو متغيرات نصية أخرى string variables، أو قيم معادة من دالات functions تولد سلاسل حرفية strings. وبما أنه من النوع المرجعي، فيمكن أن يتم وضعه أيضاً إلى "الشيء Nothing" (قيمة خاصة بالفجوال بيسك وهي كلمة محجوزة تعني "هذا النوع المرجعي فارغ، فارغ بكل معنى الكلمة")

```
customerName = Nothing ' لاشيء، Nothing
customerName = "محمد" ' Literal حرفي
customerName = GetCustomerName(customerID) ' نتيجة دالة، Function result
```

جميع المتغيرات تحتوي قيمتها الافتراضية حتى تضع لها شيء ما آخر. من أجل الأنواع المرجعية والأنواع القابلة لوضع بدون قيمة nullable، القيمة الافتراضية هي "الشيء Nothing"، أما من أجل القيم العددية الافتراضي هو 0، المنطقية Booleans الافتراضي هو خطأ False، بإمكانك تضمين إسناد أولي كجزء من عبارة Dim لإعادة قيادة override الإسناد الافتراضي.

```
Dim countdownSeconds As Short = 60
Dim processingDate As Date = Today
Dim customerName As String = GetCustomerName(customerID)
```

السطر الأخير يبين نوع مرجعي-سلسلة حرفية-تم إسناد لها نتيجة سلسلة حرفية لدالة. بإمكانك أيضاً إسناد حالة موسومة بـ new لحالة مرجعية إلى متغير من النوع المرجعي. وبالتالي فهو جديد، فهو يستخدم الكلمة المحجوزة الخاصة New، والتي تقول "إنني أعمل على إنشاء حالة جديدة من نوع بيانات خاص." يوجد العديد من التنوعات المختلفة ولكن كلها تعطي نفس النتيجة.

```
' ----- One-line variation.
Dim someEmployee As New Employee
' ----- Another one-line variation.
Dim someEmployee As Employee = New Employee
' ----- Two-line variation.
Dim someEmployee As Employee
someEmployee = New Employee
```

تذكر أن الأنواع المرجعية هي عمليات buckets تحتوي على أدلة directions لموقع البيانات الحقيقي، فعندما يبتثق متغير مرجعي للوجود وللمرة الأولى، فإنه يحتوي على لاشيء Nothing. وهذا كل شيء، فالمعلبة bucket لا تحتوي تعليمات instructions على الإطلاق بما أنه لا يوجد بيانات مرتبطة مخزنة في أي مكان. وعندما تسند حالة جديدة new instance لمتغير من النوع المرجعي، فإن تلك الحالة يتم تخزينها في مكان ما في الذاكرة memory، وتعليمات عن موقع تلك البيانات يتم إفراغها dumped ضمن المعلبة. في مقطع الكود السابق، كل استخدام للكلمة المحجوزة New يعمل على إنشاء حالة (نسخة) بيانات جديدة في مكان ما في الذاكرة. وموقع هذه البيانات يتم إسناده فيما بعد إلى المتغير someEmployee.

تتضمن العديد من الفئات واحد أو أكثر من المشيدات constructors، الإجراءات الممهدة initialization routines التي تثبت القيم الأولية initial values لحالة. بإمكانك استدعاء مشيد خاص من خلال العبارة New. يتضمن نوع بيانات السلسلة النصية string مشيدات constructors تتيح لك بناء سلسلة نصية أولية. واحد من هذه المشيدات الخاصة يتيح لك إنشاء نص يحتوي على عدة نسخ من حروف خاصة. العبارة التالية تعمل على إسناد سلسلة حرفية من 25 نجمة للمتغير lotsOfStars.

```
Dim lotsOfStars As New String("c", 25)
```

سيتم مناقشة المشيدات بالتفصيل في الفصل 8 إن شاء الله.

يمكن لعبارة Dim أن تظهر في أي مكان ضمن إجراء، ولكن بالنسبة للتقليد فإنها تظهر تماماً عند بداية الإجراء، قبل أي عبارة منطقية أخرى.

```
Sub MyProcedure ()
    Dim myVariable As Integer
    ' الكود الإضافي يتم وضعه هنا...
End Sub
```

كما مع الثوابت، يمكن أن يتم تعريف المتغيرات إما ضمن إجراء، أو خارج إجراء ولكن ضمن نوع ما. (المتغيرات والثوابت المصرح عنها خارج إجراء تعرف بالحقول *fields*. والمتغيرات والثوابت المصرح عنها ضمن إجراء تعرف أيضاً بالمتغيرات المحلية *local variables* والثوابت المحلية *local constants*، على الترتيب.) يتم استبدال الكلمة المحجوزة Dim بواسطة واحد من محددات الوصول *access modifiers* فيما يخص الحقول *:fields*

خاص Private

المتغيرات الخاصة Private variables يمكن أن يتم استخدامها بواسطة أي عضو أو إجراء ضمن النوع، ولكن ليس في أي مكان آخر. إذا عملت على اشتقاق فئة جديدة new class من فئة قاعدية base class والتي تتضمن متغير لنوع خاص private، فإن الكود في تلك الفئة المشتقة لن يكون له إمكانية الوصول على الإطلاق إلى ذلك المتغير الخاص، حتى أنه لن يعلم أبداً أنه موجود.

صديق Friend

المتغيرات الصديقة هي خاصة بمجمع ما assembly. يمكن أن يتم استخدامها بواسطة أي كود في النوع المرتبطة به، ولكن أيضاً بواسطة أي كود في أي مكان في نفس المجمع. حالياً هذه هي المتغيرات الصديقة.

العام Public

المتغيرات العامة تكون متاحة في أي مكان. فهي ممكنة لكتابة تطبيق ما أو مكون يعرض أنواعه لكود أبعد من نفسه. وأي شيء يتم تعليمه كعام يمكن أن يتم عرضه في هذه الطريقة.

المحمي Protected

المتغيرات المحمية تشابه كثيراً المتغيرات الخاصة، ولكن الكود في الفئات المشتقة يمكن أيضاً أن تصل إليها. بإمكانك استخدام الكلمة المحجوزة Protected فقط في تعريف الفئة class، فلن تعمل في التراكيب structure والوحدات البرمجية module.

الصديق المحمي Protected Friend

تجمع المتغيرات الصديقة والمحمية كل ميزات الصديق والمحمي. يمكن أن يتم استخدامها فقط في الفئات. يمكن لفئة مفردة أو نوع أن يحتوي على الحقول fields والمتغيرات المحلية local variables والثوابت constants.

```
Class mclass
    ' ----- Here's a field.
    Private InternalUseOnly As Boolean
    Sub MyProcedure ()
        ' ----- Here's a local variable.
        Dim myVariable As Integer
    End Sub
End Class
```

توجد تنوعات تركيبية أخرى لعبارة Dim، بعض منها سأناقشه فيما بعد في هذا الفصل والفصول الأخرى.

العمر والمدى Scope and Lifetime

عندما تعرف متغير ضمن إجراء procedure، فإن له مدى على مستوى الإجراء *procedure-level scope*. وهذا يعني أنك تستطيع استخدام المتغير في أي مكان ضمن الإجراء. سيكون لإجرائك على الأرجح "عبارات مقطعية block statements" وهذه العبارات، مثل For...Next و If...Then والتي تتطلب أكثر من سطر وحيد من الكود المصدر لاكتمالها. إذا أضفت عبارة Dim بين سطري بداية ونهاية واحد من هذه العبارات، فإن ذلك المتغير المصرح عنه سيكون لديه فقط مدى على مستوى المقطع *block-level scope*. وسيكون متاح فقط ضمن ذلك المقطع من الإجراء.

```
For counter = 1 To 10
    Dim processResult As Integer
    ' ----- More code here.
Next counter
MsgBox (processResult) ' هذا السطر من الكود سيفشل
```

يصرح هذا الكود عن processResult ضمن مقطع For...Next. لذلك، فهو متاح فقط للاستخدام داخل ذلك المقطع، وأي محاولة لاستخدام خارج إطار مقطع For فسينتج عنه خطأ مباشر immediate error.

إن عمر *lifetime* المتغير على مستوى الإجراء *procedure-level variable* يبدأ عندما يدخل الكود للمرة الأولى ذلك الإجراء، وينتهي عندما يخرج الكود الإجراء. وهذا صحيح بالنسبة لكل من المتغيرات على مستوى المقطع وأيضاً على مستوى الإجراء. وهذا يعني أنك إذا أسندت لمتغير على مستوى مقطع قيمة ما قبل أن يوجد المقطع، فإنه (المتغير) سيبقى لديه تلك القيمة إذا ما عدت ودخلت ذلك المقطع خلال الاستدعاء لنفس الإجراء.

بالنسبة للحقول *fields* (المتغيرات على مستوى الفئة *class-level variables*)، يعتمد المدى *scope* على مستوى الوصول المستخدم عند التصريح عن المتغير. وعمر الحقل يبدأ عندما يتم إنشاء نسخة (حالة) من الفئة في الكود. وينتهي عندما يتم تدمير الحالة أو تخرج تماماً من الاستخدام.

تقاليد تسمية المتغيرات والثوابت Variable and Constant Naming Conventions

لا تعليق.

استنتاج النوع المحلي Local Type Inference

المعاملات operators

يتضمن الفيچوال بيسك تشكيلة من المعاملات التي تتيح لك معالجة قيم متغيرتك. لقد رأيت سابقاً معاملاً الإسناد *assignment operators* (=)، والذي يتيح لك إسناد قيمة مباشرة لمتغير. معظم المعاملات الأخرى تتيح لك بناء تعابير تضم عدة قيم أصلية بطرق منهجية (قانونية) للإسناد النهائي لمتغير. خذ العبارة التالية:

squareArea = length * width

تتضمن هذه العبارة معاملين: الإسناد assignment والضرب multiplication. معامل الضرب يوحد قيمتين (الطول والعرض) باستخدام الضرب. ومعامل الإسناد يخزن الناتج في المتغير squareArea. بدون المعاملات، ستكون تحت ضغط كبير لحساب المساحة area أو أية صيغة أخرى قد تكون معقدة complex formula.

يوجد نوعين من المعاملات غير الإسنادية non-assignment operators: أحادي unary وثنائي binary. معاملات الأحادية تعمل فقط مع قيمة وحيدة، أو مع عامل Operands. معاملات الثنائية تتطلب عاملين، ولكن النتيجة في معالجة وحيدة. العوامل Operands تتضمن المحارف (المحرفية literals)، الثوابت constants، المتغيرات variables، وقيم الدالة لمعادلة function return values. يبين الجدول التالي معاملات مع تفاصيل الاستخدام.

Operator	الوصف Description
+	(الإضافة Addition). يعمل على إضافة عاملين مع بعضهما، لإنتاج مجموع sum. بعض المبرمجين يستخدم هذا المعامل لإتمام توحيد سلسلة حرفية string concatenation، ولكن من الأفضل ضم السلاسل الحرفية باستخدام معامل آخر (&) والمصمم خصيصاً لذلك الهدف. التركيب: عامل1 operand1 + عامل2 operand2. مثال: 2+3.
+	الموجب الأحادي Unary plus. يضمن عودة عامل بإشارته الحالية، إما موجب أو سالب. حيث أن معاملات بشكل آلي تحفظ إشارتها، وهذا المعامل عادة زائد (فائض redundant) وقد يأتي مساعد عندما نناقش "إعادة قيادة معاملات operator overloading". التركيب: + عامل operand. مثال: +5.
-	الطرح. يطرح عامل (الثاني) من آخر (الأول) ويعود بالفرق. التركيب: operand1 - operand2. مثال: 10-4.
-	الناقص الأحادي Unary negation. يعكس إشارة عامله. عندما يتم استخدامه مع الأعداد المحرفية، فإنه ينتج قيمة سالبة. وعندما يستخدم مع متغير يحتوي قيمة سالبة فإنه يعطي نتيجة موجبة. التركيب: -operand2. مثال: -34.
*	الضرب يضرب عاملين مع بعضهما، ويعود بالناتج. التركيب: operand1 * operand2. مثال: 3 * 8.
/	القسمة. يقسم عامل (الأول) على آخر (الثاني)، ويعود بحاصل القسمة. إذا كان معامل الثاني يحتوي 0، والتقسيم على صفر ينتج عنه خطأ. (عند العمل مع قيم المفرد Single والمزدوج Double، فإن التقسيم على صفر يعود بدليل "لانهاية" أو "ليس بعدد". التركيب: operand1 / operand2. مثال: 9 / 3.
\	القسمة الصحيحة Integer division. تقسم عامل (الأول) على عامل آخر (الثاني) وتعود بحاصل القسمة، أولاً truncating تحذف أي جزء عشري من تلك النتيجة. وكذلك إذا كان معامل الثاني يحتوي صفر، والقسمة على صفر يحدث خطأ (راجع قائمة التنبيهات مع المعامل /). التركيب: operand1 \ operand2. مثال: 9 \ 4.
Mod	الباقي Modulo. يقسم عامل (الأول) على آخر (الثاني)، ويعود بالباقي كقيمة صحيحة. إذا كان معامل الثاني يحتوي على صفر، فالتقسيم على الصفر يحدث عنه خطأ (راجع قائمة التنبيهات مع المعامل /). التركيب: operand1 Mod operand2. مثال: 10 Mod 3.
^	الأس (القوة Exponentiation). يرفع عامل (الأول) إلى قوة الآخر (الثاني). التركيب: operand1 ^ operand2. مثال: 2 ^ 8.
&	ضم النصوص (السلاسل الحرفية String concatenation). يضم عاملين لبعضهما، ويعود بنتيجة سلسلة حرفية موحدة. كلا عاملين يتم تحويلهما إلى ما يكافئهما من السلسلة الحرفية String قبل أن يتم ضمهما مع بعضهما. التركيب: operand1 & operand2. مثال: "O" & "K".
And	عملية "و"، حرف العطف. Conjunction. ينجز توصيل (جمع) منطقي أو ضم بتات bitwise على عاملين، ويرجع النتيجة. من أجل العمليات المنطقية (البولين Boolean) ستكون النتيجة صواب فقط إذا ما قيم عاملين لصواب. من أجل العمليات البتات (العديدية الصحيحة)، كل بت (وحدة bit) خاص في النتيجة سيتم وضعها إلى 1 فقط إذا كان البتين (الوحدتين) المتتابعين في كلا عاملين هما واحد. التركيب: operand1 And operand2. مثال: isOfficer And isGentleman.
Or	الفصل Disjunction. ينجز فصل منطقي أو وحدوي (بتات) على عاملين، والعودة بنتيجة من أجل العمليات المنطقية (البولينية)، ستكون النتيجة صح إذا ما تم تقييم أي من عاملين لصواب. من أجل عمليات البتات (العديدية الصحيحة)، كل بت خاص في النتيجة سيتم وضعه لواحد إذا كان البت الموافق في عامل 1. التركيب: operand1 Or operand2. مثال: enjoyMountains Or enjoySea.
AndAlso	توحيد قاصر الدارة Short-circuited conjunction. هذا المعامل مكافئ للنسخة المنطقية من المعامل And، ولكن إذا تم تقييم عامل الأول إلى خطأ، فإن معامل الثاني لن يتم تقييمه على الإطلاق. وهذا المعامل لا يدعم عمليات البتات bitwise. التركيب: operand1 AndAlso operand2. مثال: isOfficer AndAlso isGentleman.
OrElse	الفاصل قاصر الدارة Short-circuited disjunction. هذا المعامل مكافئ للنسخة المنطقية للمعامل Or، ولكن إذا تم تقييم عامل الأول إلى صح، فإن معامل الثاني لن يتم تقييمه على الإطلاق. وهذا المعامل لا يدعم عمليات البت bitwise (العديدية الصحيحة). التركيب: operand1 OrElse operand2. مثال: enjoyMountains OrElse enjoySea.
Not	النفي Negation. ينجز نفي منطقي أو بتات على عامل مفرد. من أجل العمليات المنطقية (البولين) ستكون النتيجة صواب إذا تم تقييم عامل إلى خطأ، وخطأ إذا ما تم تقييم عامل إلى صح. ومن أجل عمليات البتات (العديدية الصحيحة)، كل بت خاص في النتيجة سيتم وضعه إلى 1 إذا ما كان بت معامل مطابق هو 0، ويتم وضعه إلى 0 إذا ما كان بت معامل هو 1. التركيب: Not operand1. مثال: Not readyToSend.
Xor	استبعاد Exclusion. ينجز استثناء "أو (استثناء) exclusive or" على عاملين، ويعود بنتيجة من أجل العمليات المنطقية (البولين)، ستكون النتيجة صح فقط إذا كان للعوامل قيم منطقية مختلفة (صواب أو خطأ True or False). من أجل عمليات البتات (العديدية الصحيحة)، كل بت خاص في النتيجة سيتم وضعه إلى 1 فقط إذا البتات المطابقة في معاملات مختلفة. التركيب: operand1 Xor operand2. مثال: chickenDish Xor beefDish.
<<	الرفع اليساري Shift left. المعامل رفع يساري Shift Left يرفع البتات للعامل الأول إلى اليسار بعدد المواضع المخصصة في معامل الثاني، ويعود بنتيجة. البتات المسحوبة للنهاية اليسرى للنتيجة يتم فقدانها lost، والبتات المضافة إلى النهاية اليمنى هي دائماً 0. وهذا المعامل يعمل بشكل أفضل إذا كان معامل الأول قيمة عديدية صحيحة بدون إشارة. التركيب: operand1 << operand2. مثال: H25 << 3.
>>	الرفع اليميني Shift right. يرفع معامل الرفع اليميني Shift Right البتات للعامل الأول إلى اليمين بعدد المواضع المخصصة في العامل الثاني، ويعود بنتيجة. البتات المسحوبة للنهاية اليمينية للنتيجة يتم فقدانها (ضايغها)، والبتات المضافة إلى النهاية اليسرى دائماً نفس البت الأصلي في الموضع اليساري على الأغلب. وهذا المعامل يعمل أفضل إذا كان عامل الأول قيمة عديدية صحيحة بدون إشارة. التركيب: operand1 >> operand2. مثال: H25 >> 2.

= المساواة (equals) المقارنة (comparison). يقارن عاملين ويعود صواب إذا كانا متساويين في القيمة.

التركيب: operand1 = operand2. مثال: expectedAmount = actualAmount.

<> لا يساوي (Not equals). يقارن عاملين ويعود بصواب True إذا كانا غير متساويين في القيمة.

التركيب: operand1 <> operand2. مثال: startValue <> endValue.

< أقل من (Less than). يقارن عاملين ويعود بصواب إذا كان معامل الأول أقل في القيمة من الثاني. عند مقارنة قيم لسلسلة حرفية، فالراجع

يكون صواب إذا ظهر معامل الأول أولاً عند ترتيب سلسلتين حرفيتين.

التركيب: operand1 < operand2. مثال: raiseRate < inflationRate.

> أكثر (Greater than). يقارن عاملين ويعود صح إذا كان الأول أكبر من الثاني في القيمة. وعند مقارنة قيم لسلسلة حرفية، العائد

يكون صواب إذا ظهر معامل الأول أخيراً عند ترتيب سلسلتين حرفيتين.

التركيب: operand1 > operand2. مثال: raiseRate > inflationRate.

<= أقل من أو يساوي (Less than or equal to). يقارن عاملين ويعود بصواب إذا كان الأول أقل أو يساوي بالقيمة معامل الثاني.

التركيب: operand1 <= operand2. مثال: raiseRate <= inflationRate.

>= أكبر من أو يساوي (Greater than or equal to). يقارن عاملين ويعود بصواب إذا كان الأول أكبر من الثاني أو يساويه في القيمة.

التركيب: operand1 >= operand2. مثال: raiseRate >= inflationRate.

Like نموذج (عينة) مقارنة (Pattern comparison). يقارن عامل أول إلى نموذج مخصص في الطرف (عامل) الثاني، ويعود بصواب إذا

كان هناك تطابق. يدعم طرف (عامل) العينة (النموذج Pattern) بعض كروت التعريف wildcard (المحددة القيمة مسبقاً) القاعدية وخيارات

مختارة، وتم شرحها كاملاً في المستندات المزودة بالفيجوال أستوديو. يتضمن الدوت نت ميزة تدعى التعبيرات النظامية

regular expressions والتي توفر الكثير وأهم النماذج الشاملة مطابقة (ملائمة) الحل solution.

التركيب: operand1 Like operand2. مثال: governmentID Like ssnPattern.

Is مقارنة نوع (Type comparison). يقارن عامل الأول بكائن آخر object، نوع بيانات data type، أو "لا شيء" Nothing، ويعود بصواب

إذا كان هناك تطابق. وسأوثق هذا المعامل بالتفصيل أكثر فيما بعد في النص. وفي الفصل الثامن.

التركيب: operand1 Is operand2. مثال: someVariable Is Nothing.

IsNot مقارنة نفي النوع (Negated type comparison). وهذا المعامل يقطع (يقصر) دائرة استخدام المعاملين Is و Not مع بعضهما. التعبيران

التاليان متكافئان: first IsNot second ؛ Not (first Is second)

التركيب: operand1 IsNot operand2. مثال: something IsNot somethingElse.

TypeOf نسخة (حالة) مقارنة (Instance comparison). يعود بنوع البيانات لقيمة أو متغير. نوع كل فئة أو نوع بيانات في الدوت نت يتم

تنفيذها (تنفيذه) ككائن، بالاعتماد على System.Type. المعامل TypeOf يمكن أن يتم استخدامه مع المعامل Is.

التركيب: TypeOf operand1 Is typeOperand. مثال: TypeOf someVariable Is Integer.

AddressOf استعادة التفويض (Delegate retrieval). يعود بتفويض (مشروح في الفصل الثامن) والتي تمثل حالة خاصة من طريقة إجراء.

التركيب: AddressOf method1: AddressOf one.SomeMethod. مثال: AddressOf one.SomeMethod.

GetType النوع المسترجع (Type retrieval). يعود بنوع البيانات لقيمة أو متغير. تماماً مثل المعامل TypeOf، مهما يكن، يعمل GetType مثل دالة (وظيفة)، ولا يحتاج

إلى أن يتم استخدامه مع المعامل Is.

التركيب: GetType(operand1). GetType(one). مثال: GetType(operand1).

تستخدم المعاملات غير الإسنادية Non-assignment operators عاملاتها operands للحصول على نتيجة، ولكن لا تسبب تبديل عوامل نفسها بأي

طريقة. معاملة الإسناد يعمل على تحديث العامل الذي يظهر في جهته اليسارية. بالإضافة إلى عامل الإسناد القياسي، تتضمن الفيجوال بيسك عدة معاملات تجمع (تضم)

معامل الإسناد مع بعض المعاملات الثنائية. يبين الجدول التالي قائمة بمعاملات الإسناد هذه.

المعامل Operator معتمد على Based on

= معاملة الإسناد القياسي.

+ (إضافة addition)

- (طرح subtraction)

* (ضرب multiplication)

/ (قسمة division)

\ (قسمة عددي صحيح integer division)

^ (القوة الأس) exponentiation

& (سلسلة) (تتابع حرفي) concatenation.

<< (الرفع اليساري shift left)

>> (الرفع اليميني shift right)

معاملات الإسناد هذه هي مجرد اختصارات للمعاملات المجسدة الكاملة. على سبيل المثال، لإضافة 1 إلى متغير رقمي، بإمكانك استخدام واحد من هاتين الطريقتين:

```
totalSoFar = totalSoFar + 1
```

```
totalSoFar += 1
```

المتغيرات الستاتيكية Static Variables

عادة ينتهي عمر المتغير على مستوى الإجراء المحلي عندما ينتهي الإجراء. ولكن في بعض الأحيان ومن المحتمل أنك تريد من متغير أن يحفظ قيمته بين كل استدعاء ضمن الإجراء. وبعض الأحيان أيضاً ممكن أن تحتاج إلى مليون دولار ولكن لا تستطيع الحصول عليها دائماً. ولكن بإمكانك الحصول على متغيرات تحفظ قيمها إذا أردت. وهي تدعى المتغيرات الستاتيكية. للتصريح عن متغير ستاتيكي static variables، استخدم الكلمة المحجوزة Static في مكان الكلمة المحجوزة Dim.

```
Static keepingTrack As Integer = 0
```

إن إسناد المتغير إلى صفر يتم عمله مرة واحدة فقط. وعندما تعمل على إنشاء حالة (نسخة) من نوع يحتوي على هذه العبارة، فيما بعد، فإنه يحفظ القيمة التي تم إسنادها له من قبل مهما تكن حتى يتم تدمير الحالة (النسخة). يمكن أن يتم إنشاء المتغيرات الستاتيكية فقط ضمن الإجراءات.

المصفوفات Arrays

تعمل التطبيقات البرمجية غالباً مع مجموعات من البيانات المترابطة، وليس فقط مع قيم البيانات المعزولة. تتضمن الفيچوال بيسك طريقتين رئيسيتين للعمل مع مثل هذه المجموعات من البيانات: التجمعات collections (سنبافقشها فيما بعد) والمصفوفات arrays. تعمل المصفوفة على إسناد مواقع عديدة إلى كل بند مضمن (محتوى) في المجموعة، تبدأ بـ 0 وتنتهي بأقل 1 من عدد البنود المضمنة (المحتواة). فمصفوفة من خمس بنود لديها عناصر مرقمة من 0 إلى 4. وكمثال، تصور أنك ترمج تطبيق يحاكي حديقة حيوانات. يمكن أن تضمن مصفوفة مسماة الحيوانات animals والتي تتضمن اسم كل حيوان في حديقتك:

. حيوان #0: أكل النمل Aardvark

. حيوان #1: القرد Baboon

. حيوان #2: الشمبانزي Chimpanzee

. حيوان #3: الحمار Donkey

. وهلم وجر .

تعرف الفيچوال بيسك عناصر المصفوفة array elements بالعدد الموضوع ضمن أقواس parenthesized number بعد اسم المصفوفة. فمن أجل حيواناتنا، يعمل إسناد بسيط على وضع اسم نصي لكل حيوان في عنصر المصفوفة.

```
animal(0) = "Aardvark"
animal(1) = "Baboon"
animal(2) = "Chimpanzee"
animal(3) = "Donkey"
```

استخدام كل عنصر مصفوفة سهل جداً:

```
MsgBox("The first animal is: " & animal(0))
```

فكل عنصر من المصفوفة لا يختلف كثيراً عن المتغير القائم بذاته. في الحقيقة، بإمكانك فقط اعتبار مجموعة من الحيوانات في كود المثال لأن تكون متغيرات واضحة: متغير مسمى animal(0)، والآخر مسمى animal(1)، وهكذا. ولكن هي أفضل من المتغيرات العادية لأنك تستطيع معالجتهم جميعاً كمجموعة. على سبيل المثال، تستطيع تفحص كل عنصر باستخدام الحلقة For...Next. خذ مصفوفة عديدة صحيحة مسماة "كل بند item": مع عناصر مرقمة من 0 إلى 2. يضيف مقطع الكود التالي البنود المستقلة للمصفوفة وكأنهم متغيرات مختلفة .

```
Dim totalAmount As Integer
totalAmount = eachItem(0) + eachItem(1) + eachItem(2)
```

ولكن بما أن البنود في مصفوفة عديدة، تستطيع استخدام الحلقة For...Next لعمل مسح لكل العناصر، بالدور (واحد واحد)

```
Dim totalAmount As Integer = 0
For counter As Integer = 0 To 2
    ' ----- Keep a running total of the items.
    totalAmount += eachItem(counter)
Next counter
```

قبل أن تسند قيم إلى عناصر المصفوفة، أو استخلاص هذه العناصر، عليك التصريح وتحجيم المصفوفة التي تريد. تعمل عبارة Dim على إنشاء مصفوفة كما تعمل مع المتغيرات العادية، والعبارة ReDim تعمل إعادة تحجيم المصفوفة بعد أن تكون موجودة مسبقاً.

```
Dim animal(0 To 25) As String ' 26-element array
Dim moreAnimals() As String ' An undefined String array
ReDim moreAnimals(0 To 25) ' Now it has elements
```

عادة، ستعمل العبارة ReDim على مسح wipe أي بيانات موجودة ومخزنة في كل عنصر لمصفوفة. إضافة الكلمة المحجوزة Preserve تعمل على صيانة retains جميع البيانات.

```
ReDim Preserve moreAnimals(0 To 30) ' Keeps elements 0 to 25
```

كل عنصر في المصفوفة كائن مستقل والذي يمكن أن يتم إسناد بيانات له عند الحاجة. في هذا المثال، كل عنصر هو سلسلة حرفية String، ولكن تستطيع استخدام نوع ذو قيمة أو نوع مرجعي ترغب به في التصريح عن المصفوفة. إذا أنشئت مصفوفة من عناصر كائنية، بإمكانك دمج ومطابقة البيانات في المصفوفة، فالعنصر 0 لا يحتاج أن يحتوي نوع بيانات كما هو الحال في العنصر 1.

المصفوفة نفسها هي أيضاً كائن مستقل -حالة (نسخة) فئة والتي تدير مجموعتها من العناصر المحتواة. إذا أردت تخصيص كامل المصفوفة، وليس فقط واحد من عناصرها (وكان هناك وقت عند الحاجة لعمل هذا)، استخدم اسمها دون أي قوس أو قيم موضوعة بترتيب معين (مرتبة).

المصفوفات المتعددة الأبعاد Multidimensional Arrays

تدعم مصفوفات الفيچوال بيسك أكثر من بعد واحد one dimension (أو "رتبة rank"). تشير الأبعاد dimensions إلى عدد المجالات المستقلة المدعومة بالمصفوفة. فمصفوفة وحيدة البعد one-dimensional array، مثل مصفوفة الحيوانات animal المذكورة سابقاً، تتضمن مجالاً وحيداً. أما المصفوفة ذات بعدين تتضمن مجالين مفصولين بفاصلة comma-delimited ranges، تشكل شبكة منظمة من العناصر، مع مجالات مفصلة من أجل الصفوف والأعمدة.

```
Dim ticTacToeBoard(0 To 2, 0 To 2) As Char ' 3 x 3 board
```

يمكن أن يكون لمصفوفة حتى 60 بعد مختلف، على الرغم من أنه توجد طرق أفضل لتنظيم البيانات من تقسيمها ضمن عدة أبعاد.

حدود المصفوفة Array Boundaries

الحد الأقل لبعد أي مصفوفة هو 0، وكما أشرت بالشرط 0 To x عند تعريف أو إعادة تحجيم المصفوفة redimensioning. بإمكانك عملياً إزالة الجزء من العبارة "0 To"، وتعمل على تضمين فقط الحد الأعلى upper bound .

```
' ----- These two lines are equivalent.
Dim animal(0 To 25) As String
Dim animal(25) As String
```

هاتان العبارتان كلاهما يعملان على إنشاء مصفوفة بـ 26 عنصر. مرقمة من 0 إلى 25. يوجد العديد من الحالات الخاصة حيث يتم السماح للحدود الأدنى غير صفر، كالعامل مع المصفوفات الأقدم المولدة عن تقنية برمجة المكونات COM-generated arrays. ولكن تركيب التصريح القياسي

في الفيچوال بيسك لايسمح لك بإنشاء مصفوفات مع حدود دنيا غير الصفر(أي دائماً الحد الأدنى هو الصفر nonzero lower bounds). لتحديد الحد الأدنى والأعلى الحالي لبعده مصفوفة، استخدم الدالة LBound والدالة UBound.

```
MsgBox("The board is " & (UBound(ticTacToeBoard, 1) + 1) &
" by " & (UBound(ticTacToeBoard, 2) + 1))
```

إذا كانت مصفوفتك تحتوي فقط على بعد واحد، فليس عليك أن تخبر عن الحد الأدنى أو الحد الأعلى لأي بعد تريد أن تتحقق منه.

```
MsgBox("The upper element is numbered " & UBound(animals))
```

تتضمن كل مصفوفة أيضاً الطريقة GetLowerBound والطريقة GetUpperBound واللذان تعودان بنفس النتيجة التي يعود بها كل من LBound و UBound على الترتيب. مهما يكن رقم البعد الذي تمرره إلى الطريقة GetLowerBound والطريقة GetUpperBound يبدأ من الصفر. بينما القيم بالنسبة ل LBound و UBound تبدأ العد من 1.

```
MsgBox("The board is " & (ticTacToeBoard.GetUpperBound(0) + 1) &
" by " & (ticTacToeBoard.GetUpperBound(1) + 1))
```

إسناد قيم أولية(تمهيد) للمصفوفات

حالما تصرح عن عناصر مصفوفتك، بإمكانك تخزين واستخراج العناصر عندما تحتاج لذلك. ومن الممكن أيضاً تخزين العناصر في مصفوفتك حالما تصرح عنها. وتظهر قائمة العناصر الجديدة في مجموعة ضمن حاصرتين(قوسين مديبين curly braces)

```
Dim squares( ) As Integer = {0, 1, 4, 9, 16, 25}
```

عليك عدم وضع مواصفتي الحد الأدنى والأعلى lower and upper bound specifications عند إنشاء مصفوفة بهذه الطريقة. فمصفوفة "المربعات squares" المبينة هنا سيكون لها عناصر مرقمة من 0 إلى 5.

الأنواع "بدون قيمة" Nullable Types

متغيرات الأنواع ذات القيمة تعمل بشكل جيد جداً، تحفظ قيم بياناتها مدة عمرها. والأنواع المرجعية تعمل أيضاً بشكل جيد، ولكن يمكن ملئها بلاشيء Nothing. وهذا الاختلاف قد أصبح شوكية في جانب الأنواع ذات القيمة. ولكن مع الإصدار فيجوال بيسك 2008، أصبحت الأنواع ذات القيمة الآن من الممكن إسناد لها لاشيء Nothing. وهذا النوع الجديد "بدون قيمة nullable" أساسي عندما تريد أن تكون لديك حالة غير محددة من أجل نوع ذو قيمة قياسي(وهو مفيد خاصة عند العمل مع حقول قواعد البيانات). خذ هذه الفئة والتي تدير معلومات موظف:

```
Public Class Employee
    Public Name As String
    Public HireDate As Date
    Public FireDate As Date
    Public Sub New(ByVal employeeName As String, ByVal dateHired As Date)
        Me.Name = employeeName
        Me.HireDate = dateHired
    End Sub
End Class
```

تعمل هذه الفئة بشكل جيد، ما عدا "تاريخ الطرد FireDate" فهو ليس صحيح بشكل افتراضي، سيتم وضع تاريخ الطرد FireDate إلى 1 كانون الثاني، سنة 1 بعد الميلاد، 12:00. وتستطيع استخدام ذلك التاريخ وكأنه تاريخ "عدم الطرد". ولكن ماذا سيحدث إذا ما طردت شركتك أحداً تماماً في تلك اللحظة، هل سيكون قد مر قرنين؟ لحل مثل هذه القضية، تتيح لك الأنواع "بدون قيمة nullable" من إسناد واستخراج لاشيء من متغيرات للنوع ذو القيمة. هذا الفيغاميون يعني enriched الأنواع ذات القيمة ويتم التصريح عنه باستخدام تركيب علامة الاستفهام الخاصة.

أي من هاتين العبارتين سيعمل بشكل جيد؟

```
Public FireDate As Date?
Public FireDate2? As Date
```

كل من هاتين العبارتين سيعمل ولكني أفضل العبارة الأولى، بحيث يتم إضافة علامة الاستفهام إلى لنوع البيانات. وحالما يتم التصريح. يمكن للنوع ذو القيمة أن يأخذ إما بيانات قياسية أو لاشيء.

```
FireDate = Nothing
FireDate = #7/18/2008#
If (FireDate Is Nothing) Then...
```

يوجد تركيب خاص عند تعريف أنواع ذات قيمة خاصة بك كـ "بدون قيمة nullable"، ولكن بما أنها تستخدم مواصفة الفيجوال بيسك "generics"، لذا سانتظر حتى الفصل 16 لأقدمها لك.

دوال الفيجوال بيسك المشتركة Common Visual Basic Functions

في هذا المقطع الأخير سأعمل على تضمين قوائم بالدوال الجاهزة في الفيجوال بيسك، والعديد منها ستستخدمه بانتظام في تطبيقاتك. وأيضاً ما تم جدولته هنا بعض من أعضاء مكتبة فئة إطار العمل Framework Class Library (FCL) والتي تكرر مواصفات كانت جزء من لغة الفيجوال بيسك قبل الدوت نت، ولكن تم نقلها إلى إطار العمل من أجل إمكانية وصول عامة.

دوال التحويل Conversion Functions

تتيح لك دوال التحويل من تحويل واحد من نوع بيانات الفيجوال بيسك إلى نوع آخر. وليس لك الحرية الكاملة من أجل الكل، لذلك لا تذهب بعيداً وتعمل على تحويل النص "مرحباً" إلى قيمة عددية صحيحة وتتوقع منها أن تعمل. ولكن تحويل الأعداد من نوع عددي إلى آخر، أو تحويل الأعداد بين الأنواع العددية والسلاسل الحرفية يعمل بشكل عام جيداً. كل هذه العبارات (ماعدا CType) لديها نفس التركيب:

```
dest = CXxxx(source)
```

حيث source هو القيمة التي يجب تحويلها بواسطة CXxxx. ليس عليك إسناد النتيجة إلى متغير ما dest، تستطيع استخدام النتيجة في أي مكان ترغب به كما هو الحال مع قيم المتغيرات والقيم الحرفية، يبين الجدول التالي دوال التحويل.

الدالة Function	الوصف Description
CBool	يحول قيمة ما إلى منطقي Boolean
CByte	يحول قيمة ما إلى بايت Byte
CChar	يحول قيمة ما إلى حرف Char. إذا كانت القيمة المصدرية هي من نوع سلسلة حرفية string، سيتم تحويل فقط الحرف الأول.
CDate	يحول قيمة ما إلى تاريخ Date. إذا كانت القيمة المصدرية هي سلسلة حرفية فيجب أن تكون في تنسيق وقت أو تاريخ صحيح.
CDBl	يحول قيمة ما إلى مزدوج Double.

CDec	يحول قيمة ما إلى عشري Decimal.
CInt	يحول قيمة ما إلى عددي صحيح Integer.
CLng	يحول قيمة ما إلى طويل Long.
CObj	يحول قيمة ما إلى كائن Object. وهذا مفيد عند تريد تخزين نوع قيمة ما كحالة كائن Object.
CSByte	يحول قيمة ما إلى بايت بإشارة SByte.
CShort	يحول قيمة ما إلى قصير Short.
CSng	يحول قيمة ما إلى مفرد Single.
CType	يحول قيمة ما إلى أي نوع محدد، فئة، أو واجهة، إما في تطبيقك أو في مكتبة فئة إطار العمل FCL. والتركيب هو: CType(sourceData, newType) حيث هو newType نوع بيانات، على سبيل المثال:

```
Dim x As String
x = CType(5, String)
```

تحويل العدد الصحيح 5 إلى سلسلة حرفية. وكما مع دالات التحويل الأخرى، لا تستطيع تحويل البيانات من نوع إلى آخر إذا كانت الأنواع متضاربة، أو لو لم يوجد تحويل متاح بحيث يعرف كيف يعمل على إنتاج النوع المستهدف من النوع المصدر.

CUInt	يحول قيمة ما إلى عددي صحيح بدون إشارة UInteger.
CULng	يحول قيمة ما إلى طويل بدون إشارة ULong.
CUShort	يحول قيمة ما إلى قصير بدون إشارة UShort.

الدوال الخاصة بالتاريخ Date-Related Functions

تتضمن الفيچوال بيسك العديد من الدوال المصممة لإدارة قيم التاريخ والوقت. يجدر الجدول التالي هذه الدوال. معظم هذه الدوال تقبل معامل نسبي arguments مصدري واحد أو أكثر، وتعمل على إرجاع إما تاريخ Date، سلسلة حرفية String أو نتيجة عددية numeric result.

Function	الوصف Description
DateAdd	يضيف أو يطرح قيمة تاريخ أو وقت إلى تاريخ البدء. مثلاً، تستطيع إضافة 12 دقيقة، أو طرح ثلاث سنوات من تاريخ معطى.
DateDiff	تعود بالفارق بين قيمتي وقت أو تاريخ. تستطيع تخصيص الفترة، مثل شهر، أو ثانية.
DatePart	تعود بمكون واحد لتاريخ أو وقت. مثل ساعة أو تعود بسنة.
DateSerial	تعود بتاريخ Date مبني من قيم شهر أو يوم أو سنة.
DateString	تعود بالتاريخ الحالي كسلسلة حرفية. تستطيع أيضاً وضع التاريخ على كمبيوتر محلي باستخدام هذه الكلمة المحجوزة.
DateValue	تعود بحصة تاريخية من قيمة وقت وتاريخ موحد (مدمج)، ويتم طرح (إخراج) الحصة (قسم) الوقت (لا يتم تضمينه).
Day	تعود باليوم من قيمة تاريخ معطى.
FormatDateTime	تعمل على تنسيق تاريخ معطى أو وقت كسلسلة حرفية. باستخدام تنسيقات مجموعة صغيرة أو محددة مسبقاً، لقد تم تضمين هذه الكلمة المحجوزة بالتوافق مع كود صيغة الفيچوال بيسك المصغرة الأقدم VBScript code.
Hour	تعود بالساعة من قيمة وقت معطى.
IsDate	تشير فيما إذا التاريخ المزودة به هذه الدالة هو تاريخ صحيح valid date.
Minute	تعود بالدقيقة من قيمة وقت معطى.
Month	تعود بالشهر من قيمة تاريخ معطى.
MonthName	تعود باسم الشهر من قيمة شهرية عددية، من 1 إلى 12.
Now	تعود بالتاريخ والوقت الحالي. مكافئة لـ TimeOfDay.
Second	تعود بالثواني من قيمة وقت معطى.
TimeOfDay	تعود بالتاريخ والوقت الحالي. مكافئة لـ Now.
Timer	تعود بعدد الثواني التي مرت منذ منتصف ليل اليوم الحالي. ويتم إعادة وضع هذه الدالة إلى 0 كل منتصف ليلة.
TimeSerial	تعود بتاريخ Date مبني من قيم ساعات خاصة أو دقائق أو ثواني.
TimeString	تعود بالوقت الحالي كسلسلة حرفية. تستطيع أيضاً وضع الوقت الذي على كمبيوتر محلي باستخدام هذه الكلمة المحجوزة.
TimeValue	تعود بحصة portion الوقت من قيمة تاريخ ووقت مدمج، وحصة التاريخ يتم طردها (إزالتها discarded).
Today	تعود بالتاريخ الحالي.
Weekday	تعود بعدد صحيح يشير إلى اليوم من الأسبوع.
WeekdayName	تعود باسم اليوم من الأسبوع بالنسبة ليوم عددي صحيح من أسبوع.
Year	تعود بالسنة من قيمة تاريخ معطى.

المتغيرات التي تم إنشائها كـ System.DateTime (أو تاريخ من فيچوال بيسك) كلها تتضمن العديد من الخصائص والطرق والتي توفر مواصفات مشابهة للدوال الجدولة في الجدول السابق، على سبيل المثال، الخاصية "ثانية" تعود بعدد الثواني.

```
Dim meetingTime As Date
meetingTime = #11/7/2005 8:00:03 AM#
MsgBox (meetingTime.Second) ' Displays '3'
MsgBox (Second (meetingTime)) ' Also displays '3'
```

تستطيع إما استخدام دوال الفيچوال بيسك الجوهرية (القاعدية intrinsic) أو ما يكافئها من طرق وخصائص System.DateTime في كودك. وكلا التقنيتان توفران نفس النتيجة.

الدوال العددية Numeric Functions

تتضمن الفيچوال بيسك الكثير من الميزات للعمل على نحو مدهش مع الأعداد، بالإضافة إلى المعاملات التي تعالج البيانات، يجدر الجدول التالي العديد من الدوال الخاصة بالأعداد.

Function	الوصف Description
Fix	تبتز القسم (الحصة) العشرية من عدد ما، وتعود فقط بالحصة الصحيحة. مشابهة للدالة Int.

FormatCurrency	تنسق عدد معطى كقيمة عملة currency. باستخدام تنسيقات لمجموعة صغيرة معرفة مسبقاً. وهذه الخاصية تم تضمينها من أجل التوافق العائد لكود صيغة فيجوال بيسك المصغرة VBScript الأقدم.
FormatNumber	تنسق عدد معطى كعدد عام. باستخدام تنسيقات لمجموعة صغيرة معرفة مسبقاً. وهذه الخاصية تم تضمينها من أجل التوافق العائد لكود صيغة فيجوال بيسك المصغرة الأقدم.
FormatPercent	تنسق عدد معطى كنسبة مئوية. باستخدام تنسيقات لمجموعة صغيرة معرفة مسبقاً. وهذه الخاصية تم تضمينها من أجل التوافق العائد لكود صيغة فيجوال بيسك المصغرة الأقدم.
Int	تعود بعدد صحيح والذي يكون أقل من أو يساوي القيمة المزودة له. مشابهة للدالة Fix.
IsNumeric	تشير فيما إذا البيانات الموفرة لهذه الدالة هي عدد محقق valid number.
Oct	تنسق عدد ما كثمانى octal. وتعود بالتمثيل النصي له string representation.
Val	تستخرج العدد المحقق الأول first valid number من نص string وترجع به.

يتضمن إطار عمل الدوت نت الفئة System.Math والتي تحتوي العديد من أعضاء الدوال المتعلقة بالرياضيات. بعض منها، مثل Round، Sin و LOG، تم تنفيذها كدوال جوهرية في فيجوال بيسك 6، ولكن تم نقلها من اللغة إلى الفئة Math في الدوت نت. تتضمن فيجوال بيسك أيضاً العديد من الدوال المستخدمة الحسابات المالية financial والإحصائية (أو المحاسبية accounting). وكانت هذه الدوال مضمنة في فيجوال بيسك 6. وبما أنها ليست على علاقة بالمشروع الذي ناقشه في هذا الكتاب، لذلك سأعمل فقط على جدولت أسماؤها هنا: DDB, FV, IPmt, IRR, MIRR, NPer, NPV, Pmt, PPmt, PV, Rate, SLN, and SYD.

دوال السلسلة الحرفية String Functions

معالجة السلاسل الحرفية هو جزء جوهري من برمجة ويندوز. ميزة XML الجديدة المضمنة مع الدوت نت هي إجراءات روتينية string-manipulation routines لمعالجة النصوص بشكل رائع، على الرغم من التعقيدات المخفية hidden from view التي معها. تتضمن فيجوال بيسك العديد من الدوال المصممة لمعالجة السلاسل الحرفية strings والحروف characters. وهي مجدولة في الجدول التالي وكما مع معظم الدوال الأخرى، هذه الدوال تعود بنص جديد أو قيمة، تترك النص الأصلي أو القيم المصدرية سليمة intact. والاستثناء الوحيد (الأعز) هو عبارة Mid، والتي تعدل القيمة المصدرية لمتغير source variable's value.

Function	Description
Asc, AscW	تعود بالقيمة العددية أسكي ASCII أو يونيكود Unicode الحرف ما.
Chr, ChrW	لعدد معطى، فإن هذه الدوال تعود بما يطابق هذا العدد من حروف الأسكي ASCII أو اليونيكود Unicode.
Filter	تعود بمصفوفة والتي هي مجموعة جزئية من مصفوفة مصدرية، ولكن تضمن فقط العناصر التي تطابق نموذج ما.
Format	تنسق قيم عددية، أو تاريخ أو وقت باستخدام أكواد تنسيق خاصة أو معرفة مسبقاً.
GetChar	تستخرج حرف وحيد من سلسلة حرفية أكبر.
InStr	تعود بموضع مجموعة جزئية ضمن سلسلة حرفية أكبر.
InStrRev	تعود بموضع مجموعة جزئية ضمن سلسلة حرفية أكبر، البحث من نهاية السلسلة الحرفية حتى البداية.
Join	تعود بسلسلة حرفية string مبنية من سلسلة concatenation مصفوفة من السلاسل الحرفية array of strings.
LCase	تحول سلسلة حرفية لما يكافئها ولكن بحالة الأحرف الصغيرة lowercase equivalent.
Left	تعود بالحصة اليسارية leftmost من سلسلة حرفية.
Len	تعود بطول سلسلة حرفية.
LSet	تعمل بجانب اليسار ضمن سلسلة حرفية أكبر بالمساحة (الفاغ).
LTrim	تزيل المسافات من بداية السلسلة الحرفية.
Mid	تستخرج مجموعة نصية جزئية من وسط middle سلسلة حرفية أكبر.
Mid	عبارة تعدل مجال من الأحرف في سلسلة حرفية موجودة بمحتوى جديد. وهذه ليست دالة، ولكن عبارة فيجوال بيسك خاصة. وتركيبها يختلف كثيراً عن معظم ميزات فيجوال بيسك.
Replace	تبدل حالات من مجموعة نصية جزئية بمجموعة جزئية أخرى، والكل ضمن سلسلة حرفية أكبر.
Right	تعود بالحصة اليمينية rightmost لسلسلة حرفية.
RSet	تعمل بجانب اليمين ضمن سلسلة حرفية أكبر بالمسافات (الفاغات spaces).
RTrim	تزيل المسافات من نهاية سلسلة حرفية.
Space	تولد سلسلة حرفية تحوي عدد محدد من محارف الفراغ space characters. مشابهة للدالة StrDup.
Split	تقسم نص إلى مصفوفة من مجموعات نصية فرعية بالاعتماد على محدد معين delimiter.
Str	تحول عدد ما إلى تمثيله النصي.
StrComp	تقارن سلسلتين نصيتين، وتعود بعدد صحيح يشير إلى أمر ترتيبهما sort order.
StrConv	تحول نص ما إلى تنسيق جديد بالاعتماد على كود التحويل. بعض التحويلات تتضمن تغيير حالة المحتوى.
StrDup	تولد نص يحتوي عدد مخصص من حرف معين. مشابهة للدالة Space. ولكن تعمل مع أي حرف.
StrReverse	تعكس (تقلب) الحروف في سلسلة حرفية.
Trim	تزيل الفراغات من بداية ونهاية سلسلة حرفية.
UCase	تحول سلسلة حرفية لما يكافئها ولكن بحالة الأحرف الكبيرة uppercase equivalent.

المتغيرات المنشئة ك System.String (أو سلاسل حرفية للفيجوال بيسك). كل منها يتضمن العديد من الخصائص والطرق والتي توفر ميزات مشابهة للدوال المجدولة في الجدول السابق، على سبيل المثال، تعود الخاصية Length بعدد الأحرف في سلسلة حرفية.

```
Dim simpleString As String = "abcde"
MsgBox (simpleString.Length) ' Displays '5'
MsgBox (Len (simpleString)) ' Also displays '5'
```

بإمكانك استخدام إما دوال فيجوال بيسك القاعدية Visual Basic functions intrinsic أو ما يكافئها من طرق وخصائص System.String في كودك. كل تقنية توفر نفس النتيجة. على الرغم من أن تفاصيل التركيب والخيارات يمكن أن تتنوع (تختلف vary).

دوال أخرى Other Functions

تتضمن فيجوال بيسك العديد من الدوال التي ترفض أن تكون مقحمة squeezed في أي من التصنيفات السابقة. يجدر الجدول التالي قائمة بهذه الدوال.

الوصف Description	الدالة Function
تحويل قيمة ما من نوع بيانات إلى آخر. ويجب تحديد نهاية وبداية نوع البيانات. مشابهة للدوال CType و TryCast.	DirectCast
تعود بتمثيل نصي لكود خطأ. وتعمل فقط مع أكواد خطأ النظام التي كانت متاحة سابقاً في الفيجوال بيسك 6، وهذه الأكواد ماتزال متاحة في الدوت نت.	ErrorToString
تشير فيما إذا البيانات المزودة لها هي مصفوفة محققة (valid array صحيحة).	IsArray
تشير فيما إذا البيانات المزودة لها هي قيمة قاعدة بيانات "بدون قيمة NULL".	IsDBNull
تشير فيما إذا البيانات المزودة بها حالة خطأ error condition.	IsError
تشير فيما إذا البيانات المزودة بها غير محددة، أو تم وضعها للشيء Nothing.	IsNothing
تشير فيما إذا البيانات المزودة بها هي نوع مرجعي أو نوع ذو قيمة.	IsReference
تعود بكود اللون من مجموعة صغير من الألوان المعرفة مسبقاً.	QBColor
تعود بكود اللون المبنى من المكونات المستقلة للأحمر، الأخضر، والأزرق.	RGB
اسم نوع بيانات الفيجوال بيسك المعطى، تعود هذه الخاصية بالاسم المكافئ لنوع بيانات الدوت نت.	SystemTypeName
تحويل قيمة من نوع بيانات لآخر. ويجب تحديد نوع بيانات البداية والنهاية. مشابهة للدوال CType و DirectCast.	TryCast
تعود باسم نوع البيانات الذي يلخص نوع بيانات المحتوى الموفر. والنص المعاد هو ملخص مجمل generalized summary، وليس بالضرورة اسم نوع البيانات "صواب true".	TypeName
تعود بكود يشير إلى نوع البيانات العام للمحتوى المزود به.	VarType

مشروع Project

سنستخدم في هذا الفصل ميزات كل من الدوال ونوع البيانات التي درسناها، كل الكود في هذا الفصل سيظهر في وحدة برمجية للفيجوال بيسك مسماة "عامية General" وسيتم تخزينها باسم General.vb.

لقد عملت على إضافة ملف مع وحدته البرمجية وتعريف الوحدة البرمجية يظهر كما يلي (افتح قالب مشروع ويندوز واعمل على إضافة وحدة برمجية له من قائمة مشروع PROJECT << إضافة وحدة برمجية module وسمها General.vb) وأضف في بداية تعريف الوحدة البرمجية الكلمة المحجوزة Friend.

```
Friend Module general
```

```
End Module
```

كما ذكرت كل الكود الذي سنضيفه في هذا الفصل سيظهر ضمن هذين السطرين. تذكر أن الوحدات البرمجية تشبه كثيراً الفئات والتراكيب، ولكن لا تستطيع إنشاء حالة منها، فكل أعضائها تتشارك مع جميع أجزاء كودك المصدري. وهذا ما يتيح لها الاستخدام في أي مكان في التطبيق. لاحتياج لعمل أي شيء خاص لجعلها متاحة لكامل البرنامج. إلا وضع إمكانية الوصول لكل عضو عند الحاجة. في البداية سنعمل على إضافة بعض الثوابت العامة المستخدمة على طول البرنامج. أضف الكود التالي بين السطرين السابقين كما يلي:

```
Friend Module general
' ----- Public constants.
Public Const ProgramTitle As String = "The Library Project"
Public Const NotAuthorizedMessage As String =
    "You are not authorized to perform this task."
Public Const UseDBVersion As Integer = 1
' ----- Constants for the MatchingImages image list.
Public Const MatchPresent As Integer = 0
Public Const MatchNone As Integer = 1
Public Enum LookupMethods As Integer
    ByTitle = 1
    ByAuthor = 2
    BySubject = 3
    ByKeywordAny = 4
    ByKeywordAll = 5
    ByPublisher = 6
    BySeries = 7
    BySeriesID = 8
    ByBarcode = 9
    ByDatabaseID = 10
    ByAuthorID = 11
    BySubjectID = 12
    ByPublisherID = 13
End Enum
' ----- Security values.
Public Enum LibrarySecurity As Integer
    ManageAuthors = 1
    ManageAuthorTypes = 2
    ManageCopyStatus = 3
    ManageMediaTypes = 4
    ManageSeries = 5
    ManageGroups = 6
    ManageItems = 7
    ManagePatrons = 8
    ManagePublishers = 9
    ManageValues = 10
    ManageUsers = 11
    ProcessFees = 12
End Enum
```



```

ManageLocations = 13
CheckOutItems = 14
CheckInItems = 15
AdminFeatures = 16
DailyProcessing = 17
RunReports = 18
PatronOverride = 19
ManageBarcodeTemplates = 20
ManageHolidays = 21
ManagePatronGroups = 22
ViewAdminPatronMessages = 23

```

```
End Enum
```

```
Public Const MaxLibrarySecurity As LibrarySecurity = LibrarySecurity.ViewAdminPatronMessages
```

```
End Module
```

هذه الثوابت و العدادات توضح نفسها ، سيتم استخدام UseDBVersion لضمان أن التطبيق يطابق قاعدة البيانات المستخدمة عند إتاحة عدة إصدارات منها.

الثوابت MatchPresent و MatchNone سيتم استخدامها لعمليات بحث بنود المكتبة. يعرف العدادان الأكواد التي تعين نوع بحث بنود المكتبة لإنجاز (طرق بحث LookupMethods)، و أكواد الأمن تستخدم لحصر الميزات التي سيكون مدير ما administrator قادر على عملها في التطبيق (LibrarySecurity)

حان الوقت لإضافة بعض الطرق. الطريقة الأولى CenterText، ، تمرکز سطر من النص ضمن عرض معين. للمثال، إذا كان لديك نص " Hello, World " (12 حرف في الطول) وأردت وضعه في مركز سطر يمكن أن يكون بطول 40 حرف، ستحتاج إلى إضافة 14 فراغ لبداية السطر (محددة بطرح 12 من 40، ومن ثم تقسيم النتيجة على 2). يستخدم الإجراء زوج من دوال الفيچوال البيسك الخاصة بالسلاسل الحرفية (مثل Trim، Left و Len) لمعالجة واختبار البيانات، ومعامل القسمة الصحيحة \ للمساعدة على حساب عدد الفراغات التي يجب إدخالها.

```
Public Function CenterText(ByVal origText As String, ByVal textWidth As Integer) As String
```

```
'جعل قطعة من النص في مركز السطر ذو اتساع معين، وإذا كان النص طويل يتم بتره
```

```
Dim resultText As String
```

```
resultText = Trim(origText)
```

```
If (Len(resultText) >= textWidth) Then
```

```
'البتّر عند الحاجة
```

```
Return Trim(Left(origText, textWidth))
```

```
Else
```

```
'إضافة فراغات في البداية والنهاية
```

```
Return Space((textWidth - Len(origText)) \ 2) & resultText
```

```
End If
```

```
End Function
```

```
Public Function LeftAndRightText(ByVal origLeft As String, ByVal origRight As String,
```

```
ByVal textWidth As Integer) As String
```

```
'بناء نص بالاتساع المحدد ولديه نص مضبوط على يمين ويسار السطر
```

```
Dim leftText As String
```

```
Dim rightText As String
```

```
leftText = Trim(origLeft)
```

```
rightText = Trim(origRight)
```

```
If (Len(leftText & rightText) >= textWidth) Then
```

```
Return Trim(Left(leftText & rightText, textWidth))
```

```
Else
```

```
Return leftText & Space(textWidth - Len(leftText & rightText)) & rightText
```

```
End If
```

```
End Function
```

تبدأ الدالة CenterText بعمل نسخة من النص الأصلي (origText)، تزيل أي فراغات إضافية بالدالة Trim. ومن ثم تختبر تلك النتيجة لرؤية فيما إذا ستتناسب على السطر. فإذا لم تكن كذلك، فإنها تقطع chops off حروف التذيل والتي تكون زائدة (غير متناسبة)، وتعود بتلك النتيجة. من أجل النصوص التي تتناسب والسطر textWidth اتساع الحروف، تضيف الدالة عدد مناسب من الفراغات لبداية النص، وتعود بالنتيجة. يضيف الكود السابق دالة مسماة LeftAndRightText. تعمل تماماً مثل الدالة CenterText، ولكنها تضع سلسلتين نصيتين مختلفتين عند أقصى النهايات اليسرى واليمنى لسطر النص.

لنتقل الآن إلى الجزء الثالث من الكود، يضيف هذا الجزء إجراء مسمى DigitsOnly. يعمل على بناء نص جديد مركب من الأرقام فقط الموجودة في النص المصدري، origText. وهو يعمل هذا باستدعاء الدالة IsNumeric لكل حرف في origText، واحد واحد. فكل رقم موجود يتم سلسلته (إلحاقه بالسلسلة) إلى نهاية destText.

```
Public Function DigitsOnly(ByVal origText As String) As String
```

```
'إعادة أرقام موجودة في نص فقط
```

```
Dim destText As String
```

```
Dim counter As Integer
```

```
'اختبار كل حرف
```

```
destText = ""
```

```
For counter = 1 To Len(origText)
```

```
If (IsNumeric(Mid(origText, counter, 1))) Then
```

```
destText &= Mid(origText, counter, 1)
```

```
Next counter
```

```
Return destText
```

```
End Function
```

الدالتين الأخيرتين، CountSubStr و GetSubStr، تحسبان وتستخرجان نصوص فرعية substrings من نصوص أكبر بالاعتماد على محدد delimiter. تتضمن الفيچوال بيسك دالتين، Mid و GetChar، تستخرجان أيضاً نصوص فرعية من نصوص أكبر، ولكنهما يعتمدان على موضع النص الفرعي. تتفحص الدالتين CountSubStr و GetSubStr النصوص الفرعية أولاً باستخدام المحدد delimiter لتقسيم النص الأكبر في أجزاء.

```
Public Function CountSubStr(ByVal mainText As String, ByVal subText As String) As Integer
    ' العودة بإحصاء عدد المرات التي تم مصادفة وجود جزء نصي في نص رئيسي
    Dim totalTimes As Integer
    Dim startPos As Integer
    Dim foundPos As Integer
    totalTimes = 0
    startPos = 1
    ' مواصلة البحث حتى لم نعد نجد النص الجزئي أبداً
    Do
        ' البحث عن النص الجزئي
        foundPos = InStr(startPos, mainText, subText)
        If (foundPos = 0) Then Exit Do
        totalTimes += 1
        ' الانتقال لما بعد إيجاد النص
        startPos = foundPos + Len(subText)
    Loop
    ' إعادة الإحصاء
    Return totalTimes
End Function

Public Function GetSubStr(ByVal origString As String, ByVal delim As String, ByVal whichField As Integer) As String
    ' استخراج نص محدد بمحدد معين من نص أكبر
    Dim stringParts() As String
    ' معالجة بعض الأخطاء.
    If (whichField < 0) Then Return ""
    If (Len(origString) < 1) Then Return ""
    If (Len(delim) = 0) Then Return ""
    ' تقسيم النص الأصلي إلى أجزاء محددة بمحدد
    stringParts = Split(origString, delim)
    ' لنرى هل الجزء الذي نريده موجود، فإذا كان كذلك لنرجع به
    If (whichField > UBound(stringParts) + 1) Then Return "" Else Return stringParts(whichField - 1)
End Function
```

تحسب الدالة CountSubStr عدد مرات ظهور نص فرعي في نص أكبر. وتستخدم دالة الفيچوال بيسك InStr لإيجاد موضع النص الجزئي (subText) في نص أكبر (mainText). وتحافظ على عمل هذا حتى الوصول إلى نهاية mainText، والمحافظة على تشغيل الحساب (totalTimes) لعدد التطابقات. ولتكون على دراية أكثر، قمت باستخدام مقارنة مختلفة لتنفيذ الدالة GetSubStr. وهذه الدالة تعود بمقطع محدد من نص ما. على سبيل المثال، العبارة التالية تحصل على comma-delimited portion الجزء المحدد بالفاصلة الثالث للنص bigString.

```
bigString = "abc,def,ghi,jkl,mno"
MsgBox(GetSubStr(bigString, ",", 3)) ' Displays: ghi
```

استخدمت دالة الفيچوال بيسك Split لتقسيم النص الأصلي (origString) لعدة مصفوفات أصغر من النصوص (stringParts)، باستخدام delim كنقطة تقسيم. ومن ثم عملت على إرجاع رقم العنصر whichField من النتيجة. بما أن whichField يبدأ بـ 1 والمصفوفة تبدأ بـ 0، فيجب علي ضبط الموضع للعودة بالعنصر الصحيح.

نماذج ويندوز Windows Forms

تقنية الدوت نت المعروفة بنماذج ويندوز *Windows Forms* تتضمن كل الفئات والميزات المطلوبة لتطوير تطبيقات "سطح مكتب" قياسية لميكروسوفت ويندوز. ولقد أصبحت الآن واحد من أنواع التطبيقات المتاحة على طول مع تطبيقات الكونسول، تطبيقات الانترنت، والتطبيقات الخدمية... الخ.

الرسائل ومضخة الرسائل Messages and the Message Pump

عندما تتفاعل مع ويندوز فمن السهل عليك (كإنسان) أن تلاحظ النماذج المختلفة والأدوات التي على الشاشة. فإمكانك التفاعل مع هذه النواذ من خلال لوحة المفاتيح والماوس، يحتفظ ويندوز بقائمة النواذ المعروضة على الشاشة، كيف تتداخل ويغطي بعضها الآخر، ولأي تطبيق تنتمي كل نافذة. (بعض التطبيقات تكون مجزأة إلى عدة مسارات threads وجميعها تعمل في نفس الوقت. في مثل هذه البرامج يحتفظ ويندوز بمسار جميع النواذ على قاعدة كل مسار، وليس على قاعدة كل تطبيق فقط). يحدث كل إجراء مدخل من قبل المستخدم (مثل نقرات الماوس وضغطات المفاتيح) في طابور رسائل النظام *system message queue* بواسطة مشغل (أو معرف driver) الجهاز ذو الصلة. فعندما تنقر على الشاشة بواسطة الماوس يستخرج ويندوز رسالة النظام من هذا الطابور، محدداً أين نقرت، ومحاوياً استكشاف أي نافذة حدث نقر الفارة عليها، ومن ثم إعلام تطبيق ويندوز حول نقر الفارة بإضافة رسالة إلى طابور رسائل التطبيق. ويعمل نفس الشيء من أجل إدخال لوحة المفاتيح والإجراءات الأخرى التي ومن المحتمل يحتاج ويندوز أن يعرف شيء ما حولها. لعمل وظيفة ضمن بيئة ويندوز، يتضمن تطبيقك (أو مسار معين ضمن تطبيقك) مضخة الرسائل *message pump*، وهي مقطع من الكود يراقب طابور الرسائل. تتضمن كل رسالة قادمة (أو داخلية) رقم المعرف ID للنافذة المقصودة. يستخرج الكود الرسالة من الطابور، ويمررها إلى إجراء النافذة (ويدعى أيضاً *WndProc*) للنافذة المالكة من أجل المعالجة النهائية. في لغة C تبدو مضخة الرسائل هذه مشابهة للتالي:

```
while (!done)
{
/* ---- Extract and examine the next message. */
MSG msg;
if (GetMessage(&msg, NULL, 0, 0))
{
/* ---- WM_QUIT means it's time to exit the program. */
if (msg.message == WM_QUIT)
done = true;
/* ---- Send the message to the right window. */
TranslateMessage(&msg);
DispatchMessage(&msg);
}
}
```

بالتالي كل تطبيق (عملياً، كل مسار ضمن تطبيق) لديه مضخة رسائل واحدة وإجراءات متعددة لكل نافذة. توجد مضخة الرسائل لتمرير أو إرسال الرسائل إلى إجراء النافذة الصحيح.

إجراءات النافذة Window Procedures

بما أن مضخة الرسائل ترسل الرسائل إلى إجراءات النافذة المستقلة، يوجه روتين إجراء النافذة *WndProc* المعالجة إلى مقاطع الكود المستقلة أو الإجراءات المعتمدة على نوع الرسالة القادمة، إليك منطق عام (شبه كود) يوضح تركيب إجراء نافذة نموذجي:

```
If (message type is a mouse click)
Do mouse-click related code
Else If (message type is a key press)
Do key-press related code
Else If (message type is a window resize)
Do window-resizing-related code
Else...
```

(يستخدم شبه الكود عبارات *If* المتتالية، ولكن عملياً يستخدم إجراء نافذة بشكل أكثر شيوعاً عبارة "اختار حالة" *Select Case* للمعالجة الرسالة القادمة). لذلك، يشبه إجراء النافذة آلة البيع، إذا ضغط الزبون زر الكولا، تقوم بمعالجة ما يعيد بعلبة كولا. وإذا ضغط الزبون على زر العلكة، تقوم بمعالجة ما يعيد علكة. وإذا ضغط الزبون على زر إعادة العملة المعدنية، تحتفظ بالنقود. من أجل كل نوع رسالة (على الأقل بالنسبة لتلك التي يريد البرنامج معالجتها)، تتم معالجة بعض الكود عندما تصل الرسالة. فإذا عدنا إلى الفيچوال بيسك 1.0، جميع التطبيقات المولدة تتضمن مضخة رسالة *message pump* وإجراءات *WndProc* من أجل كل نافذة، وجميعه مخفي. المهمة الرئيسية لإجراءات *WndProc* كان استدعاء الكود في معالجات حدث الفيچوال بيسك.

النواذ في الدوت نت .NET Windows in

خذها من شخص اعتاد كتابة تطبيقات ويندوز في لغة سي C: كتابة مضخات الرسائل *message pumps* وإجراءات النافذة *window procedures* ليس بتلك السهولة. عملت ميكروسوفت على محاولة تقنيق بعض الملل من خلال تقنيات متنوعة، من ضمنها فاحص الرسالة *Message Crackers* وفئات إطار العمل *MFC*. وفي النهاية نجحت فيچوال بيسك في دفن التعقيد تحت النظام المنطقي القريب من المبرمج *programmer-friendly logical system*.

يستخدم إطار عمل الدوت نت *.NET Framework*. نظام مشابه تماماً للتنفيذات الأقدم في الفيچوال بيسك، باحتواء إجراء النافذة *WndProc* استدعاء معالجات حدث خاصة مكتوبة من قبلك. فهو يحزم جميع الفعالية والبساطة في تقنية تدعى نماذج ويندوز *Windows Forms*. جميع فئاتها تظهر في فضاء الأسماء *System.Windows.Forms*. العديد من هذه الفئات تنفذ أنواع معينة من النواذ، مثل النافذة الرئيسية الاعتيادية، الأزرار، صناديق النصوص، قوائم الصندوق المركب المنسدلة، وهكذا.

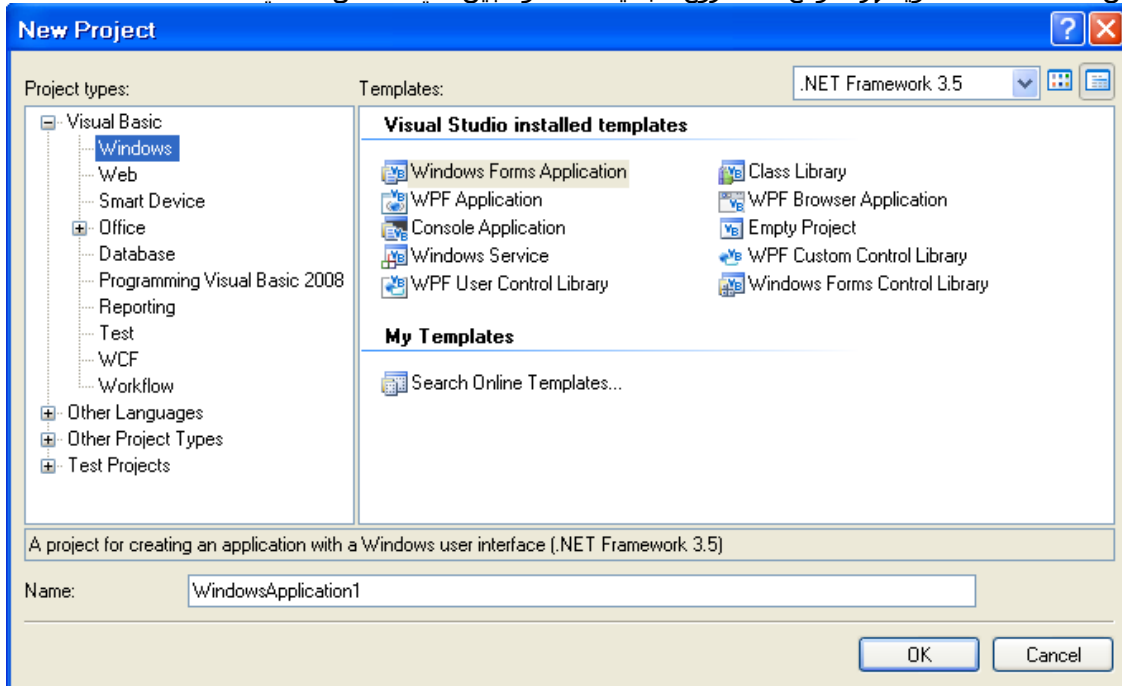
إذا كنت تريد، ما يزال بإمكانك الوصول إلى مضخة الرسائل message pump وروتينات إجراء النافذة WndProc المتنوعة. تتضمن كل فئة خاصة بنافذة طريقة WndProc تستطيع إعادة قيادتها وصنعها بنفسك. فمضخة الرسائل موجودة في الطريقة System.Windows.Forms.Application.Run. تستطيع أن تأمر أي من هذه المكونات والتحكم بكل شيء بنفسك، ولكنك ستكتشف أن معالج نماذج ويندوز ممتنع جداً، وستعمل بجد لتنسى حتى ما هي مضخة الرسائل message pump أو ما تعني.

النماذج والأدوات Forms and Controls

في الدوت نت وكما في إصدارات فيجوال بيسك الأقدم، النوافذ تكون منضدة ضمن النماذج والأدوات. ولكنها ماتزال نوافذ، مبنية من نفس المكونات الجوهرية (القاعدية)، إذا كنت لا تصدقني، راجع فئات النماذج المتنوعة والأدوات، ففي الدوت نت كل من النماذج والأدوات تشتق من الفئة المشتركة System.Windows.Forms.Control، والتي تلخص التخصص الوظيفي الجوهرية للنوافذ (نافذة window). بعض الأدوات المزودة في الدوت نت (وأيضاً مع فيجوال بيسك الأقدم) لا تنفذ عملياً على عناصر نافذة الشاشة، وهذه الأدوات مثل أداة "المؤقت Timer" فهو لا يتضمن على واجهة ملموسة، ولكنها توفر برمجة حسية مشابهة للأدوات المرئية.

تصميم تطبيقات نماذج ويندوز Designing Windows Forms Applications

إنشاء تطبيق نماذج ويندوز Windows Forms application في الفيجوال بيسك سهل، لنجربه، ابداً فيجوال أستوديو واختر مشروع جديد New Project من قائمة ملف File ويظهر نموذج المشروع الجديد كما هو مبين في الشكل التالي.



اختر نوع المشروع "Project type ويندوز Windows"، ومن ثم من نافذة القوالب Templates اختر قالب تطبيق نماذج ويندوز Windows Forms Application. امنح المشروع الاسم الذي تريد في حقل الاسم Name، ومن ثم انقر موافق OK. سيكون للمشروع الجديد فورم (نموذج Form1) وحيد جاهز لتستخدمه. عند هذه النقطة، تكون الفيجوال أستوديو قد أضافت حوالي 250 سطر من الكود المصدري لتطبيقك. إذا نقرت على زر إظهار جميع الملفات Show All Files في لوحة مستكشف الحلول Solution Explorer (تم شرحها سابقاً في الفصل الأول) وافتح الملفات المتنوعة في المشروع، بإمكانك رؤية الكود بنفسك. أهم ما في الكود في الملف Form1.Designer.vb، تم تحديثه قليلاً هنا:

```
Partial Class Form1
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode() >
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

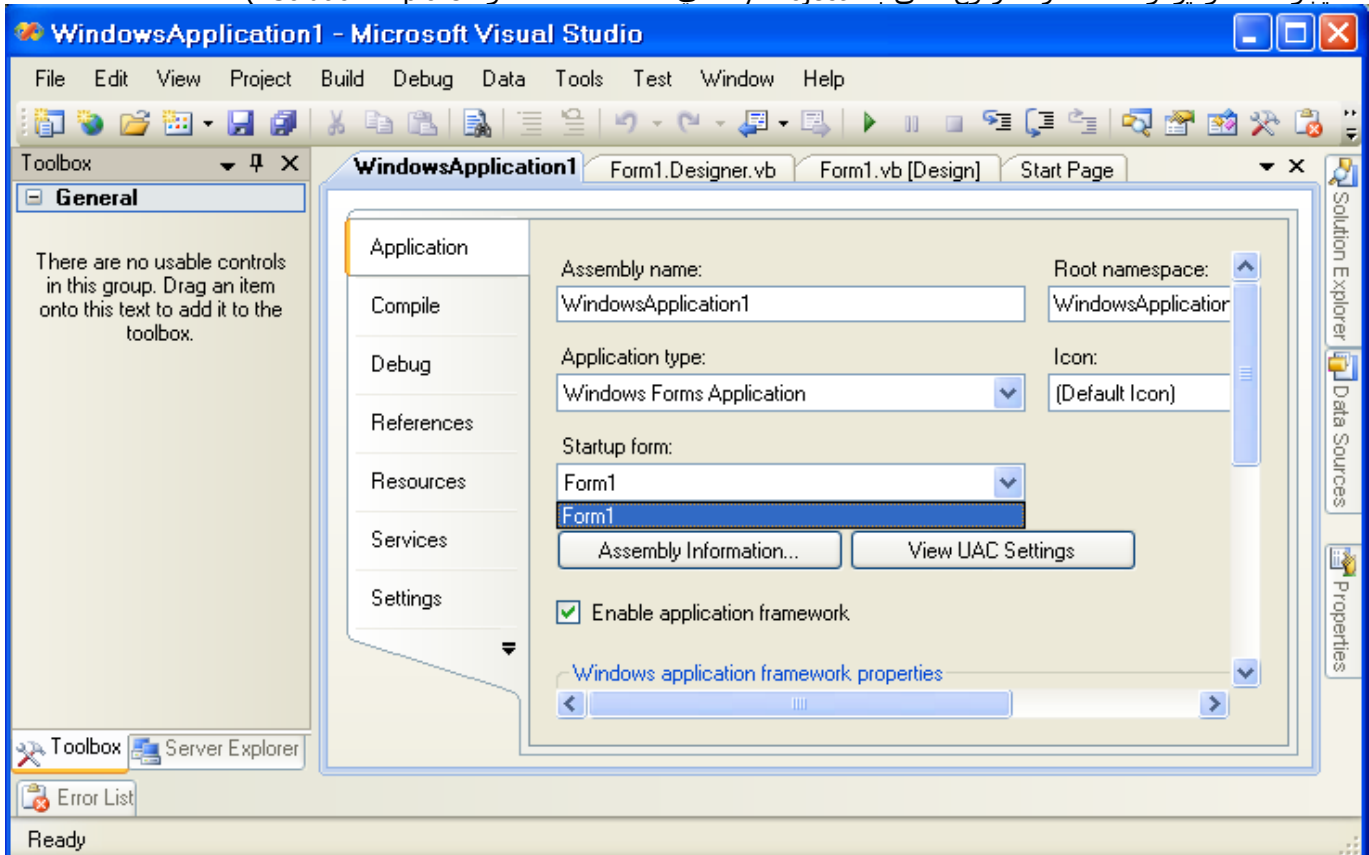
    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
```

```
'Do not modify it using the code editor.
<System.Diagnostics.DebuggerStepThrough()>
Private Sub InitializeComponent()
    components = New System.ComponentModel.Container()
    Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
    Me.Text = "Form1"
End Sub
```

End Class

كل الكود الذي ينفذ سلوك النموذج يظهر في فئة النموذج (الفرم Form class) في فضاء الأسماء System.Windows.Forms. فورم المشروع الأولية، Form1، ترث من الفئة القاعدية base form، وتستقبل كل التخصص الوظيفي للفرم والإعدادات الافتراضية. وأي تغييرات وقت التصميم يتم عملها لواجهة المستخدم Form1، مثل إضافة أدوات أبناء، يتم إضافتها إلى إجراء "التمهيد للمكونات InitializeComponent" بشكل آلي عندما تستخدم الفيچوال أستوديو. راجع هذا الروتين بشكل دوري periodically لرؤية كيف يتغير هذا الإجراء.

معظم البرامج سيكون لديها عدة فورمات، افترض أن بإمكان الدوت نت اختيار واحد من النماذج بشكل عشوائي لعرض متى يشتغل البرنامج للمرة الأولى. وهذا سيكون لهو وغير قابل للتنبؤ به unpredictable. إنك تشير إلى نموذج بداية التشغيل starting من خلال خصائص المشروع project's properties، بواسطة الحقل "نموذج الانطلاق Startup form" على تويب التطبيق Application (شاهد الشكل التالي) (تظهر نافذة خصائص المشروع project's properties عندما تختار مشروع Project << أمر القائمة خصائص Properties في الفيچوال أستوديو، أو عند النقر المزدوج على بند My Project في مستكشف الحلول Solution Explorer).



عندما يبدأ تطبيق الدوت نت، يستدعي إطار العمل الطريقة المسماة Main في مكان ما ضمن كودك. إنك تشير أي نموذج رئيسي يتم استخدامه من خلال حقل "Startup form". إن هذا الحقل يتضمن قائمة بكل النماذج، ما عليك هو اختيار ما تريد منها. ولكن أنتظر، إنك لم تضيف الطريقة "رئيسية Main" إلى أي فورم؟ لا مشكلة، إن الفيچوال بيسك ستكتب روتين (إجراء Main routine) رئيسي بسيط دون تدخل منك والذي سيعرض النموذج المشار إليه. هذا الروتين يتم إضافته وقت الترجمة، فيعمل فقط المعالجة الأقل المطلوبة لعرض الفورم.

إذا أردت إضافة روتين رئيسي مخصص لفورمك، أو إلى بعض الفئات الأخرى غير النماذج في تطبيقك، فهذا ليس بالمشكلة، إذا كنت لم تختار (لم تضع علامة صح) على الحقل "Enable application framework" المبين في الشكل السابق، على نفس تويب الخصائص المبين (لاحظ أنني قمت باختياره (وضعت إشارة صح فيه)) القائمة "نموذج الانطلاق Startup form" تتغير لتضمين أي فئة في تطبيقك فيها روتين رئيسي مناسب (موافق compatible). ولكن تطبيق إطار العمل يمكن الكثير من التخصص الوظيفي المقدر باعتدال cool functionality. وهذا كله بدون تدخل منك. عطل هذا الخيار فقط عندما تحتاج تحكّم دقيق على عمر التطبيق منذ البداية (مبكراً).

استخدام طرق Main مخصصة Using Custom Main Methods

إذا قررت كتابة روتين "رئيسي Main" خاص بك في فئة غير الفورم، فإنك تريد في النهاية عرض نموذج تطبيقك الرئيسي. بالعودة إلى الفيچوال بيسك 6، فعندما تريد عرض نموذج ما، فإنك تستدعي الطريقة "أظهر Show":

Form1.Show

هذا التركيب البسيط اختفى عندما ظهر الإصدار الأول للدوت نت من الفيجوال بيسك في 2002، ولكن عاد مع الإصدار 2005. قل إنك تريد بداية تطبيقك من الإجراء Sub Main في وحدة برمجية module مفصولة عن الفورم الرئيسي main form. أولاً، تحتاج لإضافة وحدة برمجية جديدة module إلى المشروع. اختر مشروع Project << إضافة وحدة برمجية Add Module. عدل كود الوحدة البرمجية الجديدة بحيث يبدو مشابهة لمقطع الكود التالي:

```
Module Module1
    Public Sub Main()
        Form1.Show()
    End Sub
End Module
```

في نافذة خصائص المشروع، لا تضع علامة صح لحقل " Enable application framework "، واختر إما Module1 أو Sub Main من قائمة Startup form. وهذا بسيط جداً. إذا شغلت البرنامج الآن، سيعمل البرنامج وينغلق بسرعة. في الحقيقة، ستظهر Form1 فقط لحظة مفتوحة قبل وجود البرنامج. لماذا لم تثبت ال Form1؟ يتواجد البرنامج مباشرة لأن مضخة الرسائل المزعجة، أو بدقة أكثر، العوز لمضخة الرسائل message pump. كل نافذة (أو فورم أو أداة) لديها إجراء "إجراء ويندوز WndProc"، ولكن يوجد فقط مضخة رسائل واحدة مميزة message pump لكل تطبيق أو مسار تنفيذ thread. في هذا البرنامج البسيط، Form1 لديها إجراء "إجراء ويندوز WndProc" الخاص بها، ولكنه لا يتحكم بمضخة الرسائل بنفسه. عليك على وجه التخصيص إخبار البرنامج ببدء تشغيل مضخة الرسائل message pump. بما أن مضخة الرسائل القياسية standard message pump لتطبيق نماذج ويندوز تظهر في الطريقة System.Windows.Forms.Application.Run، عدل كود Sub Main ليتضمنها مما يمكن المضخة ويحفظ الفورم 1 معروضة حتى تغلقها:

```
Module Module1
    Public Sub Main()
        System.Windows.Forms.Application.Run(Form1)
    End Sub
End Module
```

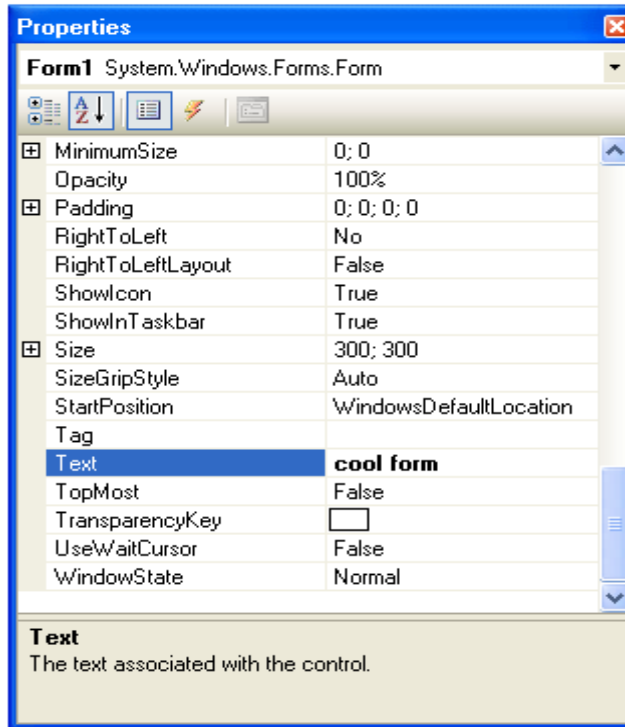
بإمكانك إضافة جميع الأنواع للكود الممهد لإجراء Sub Main، وإظهار النموذج الرئيسي main form فقط عندما يكون كودك جاهز للتفاعل مع المستخدم.

إذا أردت إضافة نموذج جديد لتطبيقك، استخدم مشروع Project << إضافة نموذج ويندوز Add Windows Form.

العمل مع النماذج Working with Forms

في الدوت نت كل النماذج هي فئات بسيطة، تنوع عن الفئة System.Windows.Forms.Form. كل مرة تعمل على إنشاء نموذج جديد، فإنك تنشئ فئة مشتقة معتمدة على فئة النموذج المشتركة common Form class. ويتم تحميل فئتك الجديدة مع التخصص الوظيفي، وهي تتضمن الحقول fields، الطرق methods، الأخطاء bugs، الخصائص properties، والأحداث events التي تتركب فئة الفورم. تأخذ الفيجوال أستوديو هذه العناصر وتجلبهم بطريقة تجعل منها سهلة البرمجة بالنسبة للفورم من خلال كل الكود المصدري source code ومن خلال السحب والإسقاط drag-and-drop على واجهة المستخدم لمصمم نماذج الفيجوال أستوديو Visual Studio Forms Designer.

عندما تضيف نموذج form للمرة الأولى لتطبيقك، استخدم لوحة الخصائص Properties (شاهد الشكل التالي) لتعديل الفورم لما تحب. وهذه اللوحة تظهر الخصائص المبدئية للبنود المختارة حالياً ضمن بيئة الفيجوال أستوديو. فهي تتضمن مدخلة مفصولة لإعداد كل خاصية، معظمها يمكن أن يتم تحديثه من خلال مدخلة النص text البسيطة. على سبيل المثال، بإمكانك تعديل العنوان المعروض في أعلى الفورم بتعديل المحتوى لخاصية النص Text من Form1 إلى Cool Form.



إليك بعض الخصائص التي تظهر في نافذة الخصائص فيما يخص الفورم.

الخاصية (Property)	الوصف (Description)
(Name)	هذه هي اسم الفورم، أو بدقة أكثر اسم الفئة التي تمثل هذه الفورم. يتم تسميتها بشكل افتراضي FormX لتستخدم في الكود حيث X هو عدد ما وقد يكون من المتطلب تغيير هذا الاسم لشيء ما أكثر دلالة.
AcceptButton	تشير لأي زر تم وضعه على الفورم سابقاً وسوف يطلق عند الضغط على مفتاح Enter.
AutoScroll	إذا وضعت هذا الحقل إلى صواب، فإن النموذج يضيف بشكل آلي أشرطة تمرير والتي تتحرك حول محتوى الفورم إذا ما تم تحجيم الفورم لتصبح صغيرة جداً لإظهار كل شيء.
BackColor	لون خلفية النموذج.
BackgroundImage	استخدم هذه الخاصية مع الخاصية BackgroundImageLayout لوضع صورة على خلفية الفورم.
CancelButton	تشبه الخاصية AcceptButton، ولكن لإسناد زر يتم إطلاقه عند الضغط على المفتاح Esc.
ContextMenuStrip	تسمح لك من إنشاء قائمة اختصار مخصصة تظهر عندما ينقر المستخدم يمين على خلفية الفورم. تشير ContextMenuStrip إلى أدوات مفصلة تعمل على إضافتها للنموذج.
ControlBox	إظهار ورؤية صندوق التحكم في الزاوية اليسارية العلوية للفورم من خلال إعداد هذه الخاصية.
Cursor	تشير إلى تخطيط مؤشر الفأرة.
FormBorderStyle	هذه الخاصية تشير إلى نوع الفورم المعروضة. الافتراضي قابلة للتحجيم Sizable.
Icon	تضع صورة معروضة في الزاوية العلوية اليسارية لحدود الفورم.
IsMdiContainer	تمكنك دعم "واجهة متعددة مستندات multiple document interface" على هذه الفورم. وهذا ما يتيح للنموذج الرئيسي أن يحتوي عدة نماذج أبناء. يمكن أن تعرض الفيچوال أستوديو نفسها والنماذج ونافذة الكود المصدري في تخطيط بيئة متعددة المستندات MDI style.
KeyPreview	إذا وضعت هذه الخاصية لصواب، فإن أحداث الفورم KeyDown و KeyPress ستحصل على معالجة أي مفتاح تم إدخاله من قبل المستخدم، حتى ولو كانت هذه المفاتيح القصد منها التحكم بالمحتوى على الفورم. وهذا مفيد عند الحاجة لالتقاط المفاتيح المقدمة لكامل الفورم، مثل استخدام المفتاح F1 لإطلاق مساعدة عبر الإنترنت.
Location	موقع الفورم على الشاشة وتؤثر أيضاً الخاصية StartPosition في موقع الفورم.
MainMenuStrip	تحدد أداة "عروة القائمة MenuStrip" لاستخدام القائمة الرئيسية للفورم. يتم إضافة أداة عروة القائمة MenuStrip المشار إليها بشكل منفصل للنموذج.
MaximizeBox	ظهور أو عدم ظهور صندوق التكبير في الزاوية اليسارية العليا.
MinimizeBox	ظهور أو عدم ظهور صندوق التصغير في الزاوية اليسارية العليا.
MinimumSize	إمكانية تحجيم الفورم. (الحد الأصغر لتحجيم الفورم).
Opacity	الشفافية، تسمح بتخصيص مستوى الشفافية للون معين.
ShowInTaskbar	تخصص فيما إذا هذه الفورم يجب أن تظهر كبنء في شريط مهام النظام.
Size	تشير إلى الحجم الحالي للفورم من خلال الخصائص الفرعية العرض Width والارتفاع Height.
StartPosition	كيفية وضع الفورم على الشاشة عندما تظهر للمرة الأولى على الشاشة.

بإمكانك وضع أي نوع بيانات تريدها في هذه الخاصية، فهي موجودة لكي تستخدمها.

لقد جدولت أقل من نصف الخصائص هنا، فمن الواضح أن لديك الكثير من التحكم على الفورم وكيفية إظهارها للمستخدم. ما يهم فعلياً هو أن العديد من هذه الخصائص غير مقصورة على النماذج. بعض من هذه الخصائص تأتي من فئة `System.Windows.Forms.Control` المشتركة، وتظهر أيضاً في كل الأدوات الأخرى التي تستخدم نفس الفئة القاعدية `base class`. وتتضمن هذه الخصائص مثل `Location`، لون الخلفية `BackColor`، والنص `Text`، على الرغم من أن النص المعروض في عنوان `caption` الفورم والنص `Text` المعروض على زر الأمر يختلفان بشكل ملحوظ في إيجادهم، فإن الاستخدام في الكود متطابق.

```
Form1.Text = "This is a form caption."
Button1.Text = "This is a button caption."
```

على الرغم من أنك تستطيع وضع كل الخصائص من نافذة الخصائص، بإمكانك أيضاً تحديث وعرضها من خلال الكود. في الحقيقة، إذا ما عدلت أي من الخصائص من خلال نافذة الخصائص، فتكون قد حدثتهم خلال الكود المصدري، بما أن الفيجوال أستوديو يعمل على تحديث كودك تماماً دون تدخل منك. جرب هذا، ضع أي من الخصائص في لوحة الخصائص للفورم، ومن ثم اعرض الإجراء "التمهيد للمكونات `InitializeComponent`" في ملف `Form1.Designer.vb`. تستطيع مثلاً وضع الخاصية `TopMost` إلى صواب، وستجد العبارة الجديدة التالية قرب النهاية السفلية للإجراء.

```
Me.TopMost = True
```

بإمكانك عمل هذا من جانبك لتخصيص الخصائص من خلال الكود فما عليك سوى وضع الكائن متبوع بنقطة ومن الخاصية التي تريد كما فعل الفيجوال أستوديو مع الخاصية `TopMost`.

```
Me.Text = "The Library Project"
```

بإمكانك أيضاً استخراج قيمة الخاصية من خلال أسماءها كما يلي:

```
MsgBox ("عنوان هذا الفورم: " & Me.Text)
```

بإمكانك الوصول لطرق الفورم المتنوعة بنفس الطريقة. على سبيل المثال، الطريقة لإغلاق التي تغلق الفورم:

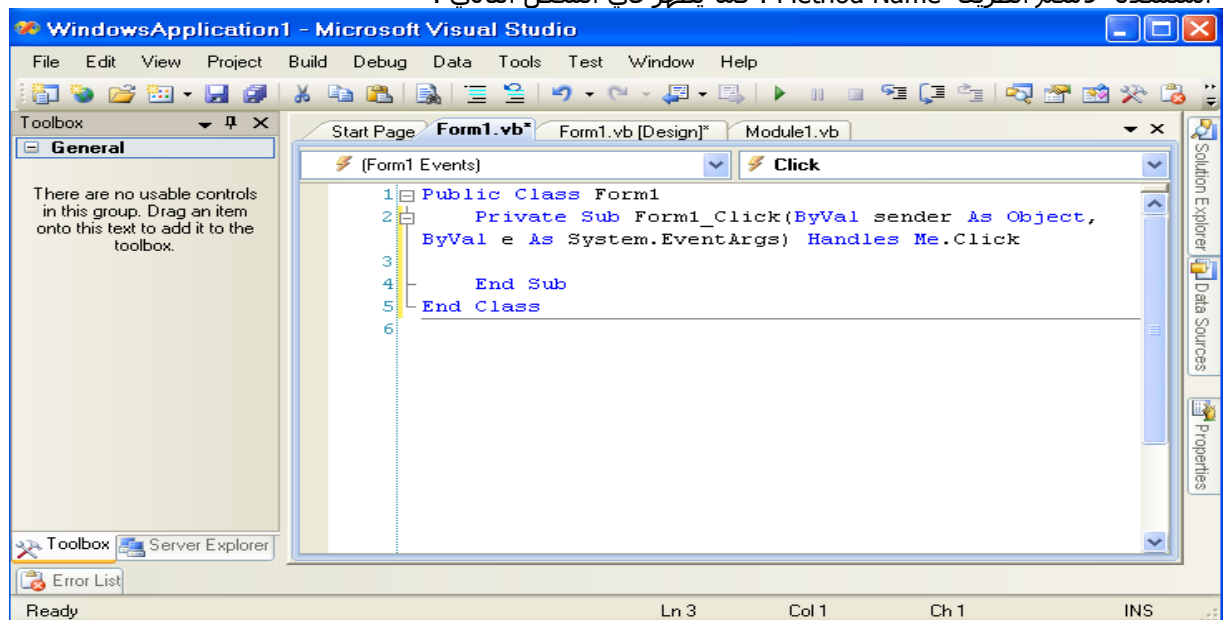
```
Me.Close()
```

بالطبع هذه العبارة تحتاج أن تظهر ضمن إجراء ما محقق مثل معالج حدث ما `event handler`، لنضيف بعض الكود لحدث نقر `Click event` على الفورم لذلك عندما ينقر المستخدم على الفورم فإن الكود الجديد سيعمل على تبديل عنوان الفورم بحيث يذكرنا ما هو هذا العنوان، وإغلاق الفورم يؤدي إلى الخروج من البرنامج. للوصول `Access` إلى الكود المصدري اختر `Form1.vb` في مستكشف الحلول `Solution Explorer` ومن ثم انقر على زر عرض الكود `View Code` عند أعلى مستكشف الحلول. يظهر مقطع كود الفورم الافتراضي كما يلي.

```
Public Class Form1
```

```
End Class
```

هذه هي الحصة المتنقلة لفئة الفورم `Form1`، الجزء الذي تظهره الفيجوال استديو للعامّة، وليس الأجزاء المخفية الأكثر أهمية (الجزء في `Form1.Designer.vb`). ولكن سنكون قادرين على جعل هذا المقطع هام. أضف حدث نقر `Click event` لسطح النموذج `form's surface` باختيار (أحداث الفورم `Form1 Events`) من قائمة اسم الفئة `Class Name` (أعلى ويسار محرر نص الكود)، ومن ثم اختر نقر من القائمة المنسدلة لاسم الطريقة `Method Name`، كما يظهر في الشكل التالي:



عدل معالج الحدث بحيث يعرض الكود المبين هنا:

```
Private Sub Form1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Click
    Me.Text = "The Library Project"
    MsgBox ("The form's caption is: " & Me.Text)
```



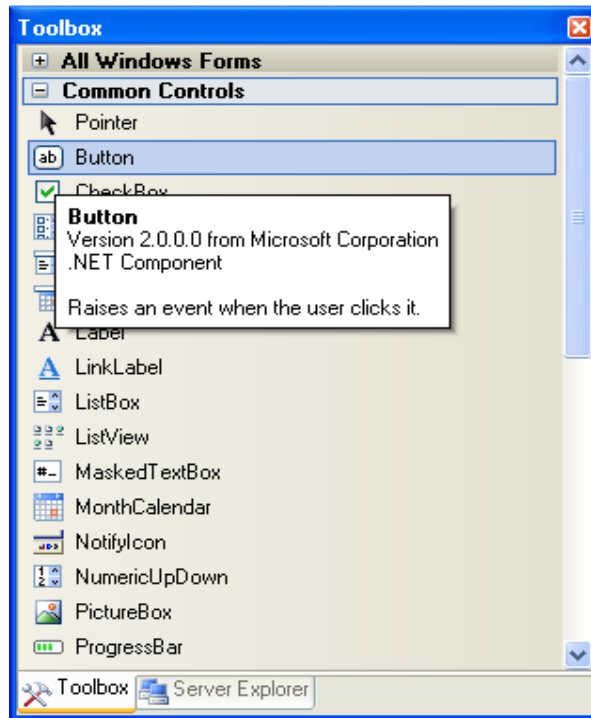
```
Me.Close()
```

```
End Sub
```

إذا شغلت التطبيق ونقرت على سطح الفورم، سيظهر صندوق رسالة مع عنوان الفورم قبل الخروج من التطبيق لاحظ الشكل التالي. **إضافة أدوات Adding Controls** تتضمن الفيچوال بيسك والدوت نت الكثير من أدوات نماذج ويندوز، وحتى يمكن عمل أدوات واستخدامها كأدوات العادية الموفرة في صندوق الأدوات. بإمكانك بناء أدواتك الخاصة، إما باشتقاقها من فئات أدوات موجودة أو تنفيذهم بالكامل من الصفر.



صندوق أدوات Toolbox الفيچوال أستوديو يتضمن جميع الأدوات الأساسية التي تحتاجها لبناء تطبيقات برمجية عالية النوعية -high quality أو حتى منخفضة النوعية وبإحاف pathetic low-quality. للوصول إلى صندوق الأدوات-الجزء الذي يظهر في الشكل التالي-من خلال عرض << View صندوق الأدوات Toolbox .



يوجد أربع طرق لإضافة أداة لفورم:

1. النقر المزدوج على الأداة في صندوق الأدوات. فتظهر نسخة من الأداة على سطح الفورم في موقعها الافتراضي مع جميع الإعدادات الافتراضية.
2. سحب وإسقاط الأداة من صندوق الأدوات Toolbox إلى الفورم form .

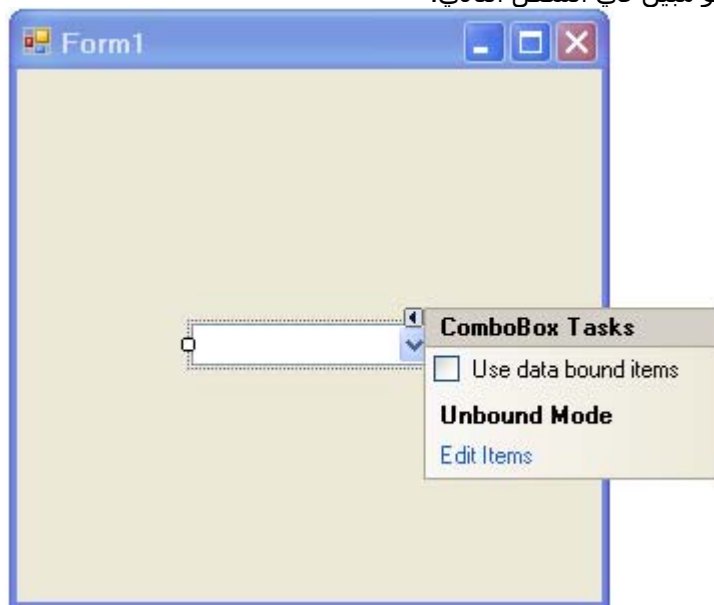
3. انقر على الأداة في صندوق الأدوات، ومن ثم استخدم الماوس لسحب المنطقة المستطيلة على الفورم حيث ستظهر الأداة. بعض الأدوات، مثل أداة الصندوق المركب ComboBox، لديها حدود بالنسبة للعرض والارتفاع، فليس من الضروري أن تحجم نفسها عند تنوي.

4. أضف أداة للفورم باستخدام الكود المصدري للفيجوال أستوديو. كما تضيف أدوات للفورم في الفيجوال أستوديو، فهي تكتب الكود المصدري لك. لا يوجد سبب لعدم إضافة هذا الكود بنفسك، على الرغم من أنه يوجد تحذير في الملف *Form1.Designer.vb* يخبرك بعدم تحديث هذا الملف، بإمكانك تعديل الإجراء *InitializeComponents* بشكل يدوي إذا ما طبقت بدقة نمط الكود المولد بواسطة الفيجوال أستوديو. بإمكانك أيضاً إضافة أدوات في مناطق أخرى من كودك، مثل حدث تحميل الفورم. إضافة أدوات بشكل آلي يقع فيما وراء مجال هذا الكتاب، ولكن تابع وحرب هذا.

بعض الأدوات ليس لديها واجهة مستخدم حقيقية موجودة في التطبيق المشغل. هذه الأدوات، عند إضافتها للفورم، تظهر في لوحة أسفل سطح الفورم. وتستطيع الاستمرار في التفاعل معها تماماً مثل الأدوات المعتمدة على الفورم.

حالما تظهر الأداة على الفورم، استخدم الماوس لنقل الأداة أو إعادة تحجيمها باستخدام المراسي (الأنكر *anchors*) التي تظهر عند يتم اختيار الأداة. العديد من الأدوات تكون محدودة في خيار التحجيم. أداة الصندوق المركب ComboBox، على سبيل المثال، يمكن فقط إعادة تحجيمها أفقياً، وحجمها الشاقولي يتم تحديده بواسطة الأشياء مثل الخط *font* المستخدم في الأداة. بينما تسمح لك أدوات أخرى إعادة تحجيمها. ولكن فقط في بعض الأحيان. أداة الرقعة (للصاق *Label*) يمكن تحجيمها يدوياً عندما تكون خاصيتها *AutoSize* قد تم وضعها إلى خطأ *False* فقط.

تتضمن بعض الأدوات زر سهم صغير، ويظهر على الأغلب قرب الزاوية العلوية اليمينية من الأداة. وهو عبارة عن بطاقة (اقتباس *Smart Tag* سريع)، مشابه لميزة الاقتباس السريع المضمنة في ميكروسوفت أوفيس. يوفر النقر على الاقتباس السريع الوصول السريع إلى الميزات المصاحبة للأداة. كما هو مبين في الشكل التالي.

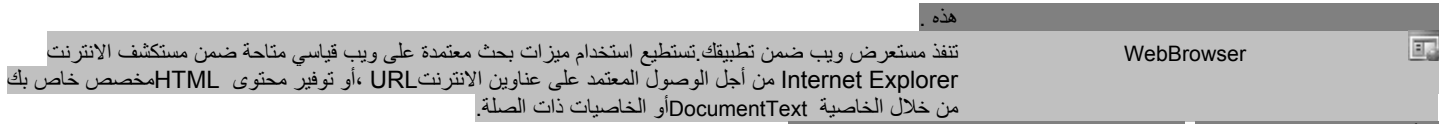


تبين القائمة التالية بعض الأدوات الشائعة الأكثر استخداماً، وكلها مضمنة بشكل افتراضي مع تطبيق نماذج ويندوز Windows Forms application وبشكل افتراضي في صندوق الأدوات Toolbox. إذا ما أنشئت تطبيق نماذج انترنت *web-based applications* في الفيجوال أستوديو-المستخدم لتصميم تطبيقات معتمدة على الويب باستخدام ASP.NET - فإن الأدوات المتاحة ستختلف عن هذه القائمة.

الأيقونة Icon	الأداة Control	الوصف Description
	BackgroundWorker	تتضمن الدوت نت دعم للتطبيقات ذات مسارات التنفيذ المتعددة <i>multithreaded</i> . يتيح لك الأداة <i>BackgroundWorker</i> البدء بمهمة الخلفية <i>background task</i> تماماً من الوضعية المريحة لنموذجك. وهي مفيدة بشكل خاص عندما ترغب في تحديث فورم بالاعتماد على عرض العناصر المتفاعلة مع مسارات تنفيذ "worker" أخرى. إنك تستأنف مهمة عمل جديدة من خلال طريقة الأداة <i>RunWorkerAsync</i> ، وتنجز العمل الحقيقي في حدثها <i>DoWork</i> .
	Button	زر دفعي قياسي. إن حدث النقر <i>Click</i> هو الميزة الأكثر استخداماً من الناحية البرمجية. على الرغم من أنك تستطيع استخدام الخاصية <i>DialogResult</i> لإطلاق إجراء حوار خاص.
	CheckBox	تنفذ هذه الأداة طريقتين (فعال <i>on</i> ، غير فعال <i>off</i>) أو ثلاث طرق (فعال <i>on</i> ، غير فعال <i>off</i> ، آخر <i>other</i>) لحقل الاختيار "اختبار" <i>checked</i> . تشير الخاصية <i>ThreeState</i> إلى العدد الكلي من الخيارات. استخدم الخاصية المنطقية "مختبر <i>Checked</i> " من أجل <i>checkboxes</i> صناديق الاختيار ذات الطريقتين، أو الخاصية <i>CheckState</i> من أجل صناديق الاختيار ذات ثلاث طرق.
	CheckedListBox	تدمج هذه الأداة أفضل ما في كل من أداة صندوق القائمة <i>ListBox</i> وصندوق الاختبار <i>CheckBox</i> ، تمنحك قائمة حيث يكون كل بند فيها يمكن أن يتم اختياره بطريقتين أو ثلاث. الطريقة <i>GetItemChecked</i> والطريقة <i>GetItemCheckState</i> (ونظيرهما "وضع <i>Set</i> ") تتوفران واحد من عدة طرق لتفحص حالة بنود في القائمة. كن على دراية أن أداة مشابهة مسماة <i>CheckBoxList</i> وهي من أجل الاستخدام في تطبيقات ASP.NET فقط.
	ColorDialog	تعرض نموذج ويندوز قياسي مستخدم لاختيار لون من قبل المستخدم. عرض حوار اللون باستخدام طريقة الأداة

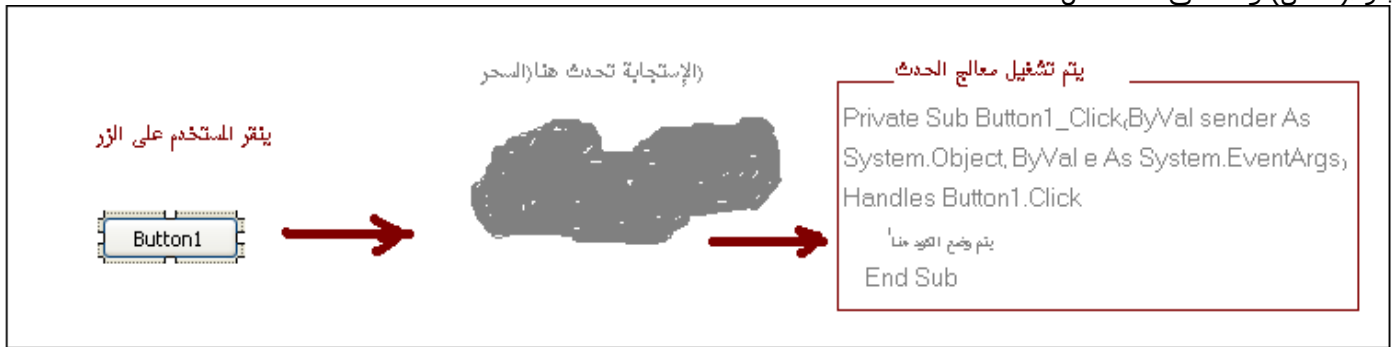
ComboBox	،وتحصل على النتيجة بواسطة الخاصية Color. ShowDialog تنفذ هذه الأداة أداة صندوق مركب ComboBox منسدل للأسفل لنافذة قياسية، وفي معظم أنماطها المتنوعة يمكن للقائمة أن تتضمن أي كائن ترغب به، فهي غير محصورة فقط بالسلاسل الحرفية strings. وتستطيع أيضاً توفير كود "ownerdraw" مخصص يسمح لك رسم كل بند قائمة بنفسك.
ContextMenuStrip	هذه الأداة تتيح لك تصميم اختصار أو قائمة "سياق context" ،ليتم عرضها عندما ينقر المستخدم يمين على الفورم أو أداة تختارها أنت. ويتم تصميمها واستخدامها في الكثير من الشبه كما هو الحال لأداة MenuStrip.
DataGridView	تنفذ هذه الأداة جدول قياسي مثل الشبكة المستخدمة لعرض أو تحديث بيانات في خلايا مستقلة. ويتم تحميلها مع عرض خيارات كثيرة. يمكن للبيانات المعروضة أن ترتبط بمصدر بيانات خارجي، أو بإمكانك تركيبها على الطائر النمط "virtual data" ،يتيح لك أيضاً تحميل بيانات عند الحاجة فقط.
DateTimePicker	تتيح هذه الأداة للمستخدم إدخال تاريخ، وقت أو كلاهما، من خلال إما مدخلة النص الأساسية أو الأدوات المعتمدة على الماوس. على الرغم من أنه نموذج غير حر كحقل النص البسيط، فهو يعمل على إجبار اختيار لتاريخ ما أو وقت ما. بإمكانك وضع النهايات العظمى والصغرى على اختيار المستخدم. توفر الأداة MonthCalendar واجهة بديلة لاختيار تاريخ مخصص.
DomainUpDown	من خلال هذه الأداة، يختار المستخدم نموذج من بين قائمة من الخيارات التي تعرفها، الخيارات التي لها ترتيب ملزم خاص. استخدم هذه الأداة كبديل عن أداة ComboBox أو أداة TrackBar عندما يكون هناك مسوغ warranted.
FolderBrowserDialog	تعرض نموذج ويندوز القياسي المستخدم من أجل اختيار دليل أو مجلد من قبل المستخدم. تعرض صندوق حوار الاختيار باستخدام طريقة الأداة ShowDialog، والحصول على النتيجة باستخدام الخاصية SelectedPath.
FontDialog	تعرض نموذج ويندوز قياسي يتم استخدامه من أجل اختيار الخط من قبل المستخدم. وتعرض صندوق حوار الاختيار باستخدام الطريقة ShowDialog، والحصول على النتيجة بواسطة الخاصية Font. وهناك خاصيات أخرى تمكن الوصول إلى مكونات الخط المختار.
GroupBox	توفر طريقة بسيطة لتجميع الأدوات بشكل مرئي على الفورم. الأدوات التابعة يتم رسمها أو لصقها مباشرة على هذه الأداة. للوصول إلى نفس التخصص الوظيفي بدون حدود مرئية أو عنوان استخدم الأداة Panel.
HelpProvider	تتيح لك الإشارة إلى تفاصيل المساعدة عبر الشبكة من أجل أدوات أخرى على الفورم. عند استخدامها فإنها تضيف العديد من شبه الخاصيات "المساعدة Help" لكل أداة فورم أخرى من خلالها تستطيع تزويد تفاصيل سياق المساعدة عند تنفيذها بشكل مناسب، سيتم عرض محتوى المساعدة عبر الشبكة عندما يضغط المستخدم على المفتاح F1 في سياق الأداة الفعالة.
HScrollBar	تنفذ هذه الأداة شريط التمرير الأفقي، بحيث تسمح للمستخدم من التمرير خلال قطاع العرض أو قائمة من الاختيارات. من أجل التنفيذ الشاقولي لهذه الأداة، استخدم الأداة VScrollBar. تتضمن العديد من الأدوات الأخرى نسخة خاصة بها لأشرطة التمرير هذه.
ImageList	تغلف هذه الأداة مجموعة من الصور الصغيرة أو الأيقونات من أجل الاستخدام من قبل أدوات أخرى تدعم قائمة صور. يتم استخدام قوائم الصور بشكل شائع من قبل أداة عرض القائمة ListView، شريط الأدوات Toolbar، وأداة عرض الشجرة TreeView.
Label	تعرض هذه الأداة نص ثابت على الفورم. وتستطيع رسم خطوط مستطيلات وما إلى ذلك.
LinkLabel	تنفذ هذه الأداة كما في الأداة السابقة ولكنها تتضمن وصلة أو أكثر links ضمن محتوى النص. وهي مشابهة لنص الوصلات القياسي في محتوى مستعرض الويب، تستدعي معالج حدثها LinkClicked عندما ينقر المستخدم على أي من الوصلات المضمنة.
ListBox	تنفذ هذه الأداة أداة صندوق نص قياسي، عارضة قائمة من البنود بحيث يستطيع المستخدم اختيار ما يشاء. يمكن للبنود المضمنة أن تكون أي كائن ترغب، فهي ليست محدودة لنصوص فقط. وتستطيع توفير كود حامل رسم ownerdraw يسمح لك رسم كل بند قائمة بنفسك.
ListView	تحضر هذه الأداة مجموعة من البنود مع خاصيات عرض اختيارية. تستطيع إضافة بيانات خاصة بعمود من أجل عرض التفاصيل details. تظهر البنود في الأداة كمجموعة من كائنات الفئة ListViewItem.
MaskedTextBox	تتوخ عن صندوق النص القياسي يساعد المستخدم على إدخال عدد منسق أو بيانات نصية بعرض قالب أو قطاع، على سبيل المثال تستطيع إجبار المستخدم على إدخال رقم الهاتف في هيئة "xxxxxx-xxx" باستخدام القناع العددي مع تضمين رموز الشحطة أو الواصلة القصيرة.
MenuStrip	تتيح لك هذه الأداة تصميم قوائم الفورم القياسية، والتي يتم عرضها على طول أعلى الفورم. يتم تنفيذ القوائم ضمن شريط القائمة من خلال حالات (نسخ) الفئة ToolStripMenuItem. وشريط القائمة هو شريط أدوات يشبه في تنفيذه قائمة ويندوز القياسية. تستطيع إضافة أنواع أخرى من الأدوات للقائمة، متضمنة شريط أدوات خاص بأدوات الصندوق المركب. قوائم حساسة للسياق، يتم عرضها عندما ينقر المستخدم على الفورم أو أداة معينة، ويتم تنفيذها من خلال الأداة ContextMenuStrip.
MonthCalendar	تعرض مجموعة جزئية من التقويم، مركزة العرض على العرض الخاص بالشهر. يمكن أن يتم عرض أكثر من شهر في نفس الوقت. في تركيب شاقولي، أفقي، أو في تركيب شبكة. توفر الأداة DateTimePicker واجهة بديلة من أجل الاختيار الخاص بالتاريخ.
NotifyIcon	تتيح لك وضع أيقونة في منطقة "صينية النظام system tray" لشريط مهام ويندوز، وربط رسائل مهمة إلى المستخدم من خلال هذه الواجهة بما أن هذه الأداة ليس لديها واجهة مستخدم خاصة بالفورم، من الممكن استخدامها بدون أن يكون لديها عرض نموذج قياسي.
NumericUpDown	تسمح للمستخدم من اختيار قيم عديدة باستخدام طريقة مقطع أعلى/أسفل القابل للتمرير. استخدم هذه الأداة كبديل عن الأدوات ToolStrip، TextBox، TrackBar، أو VScrollBar عندما يكون هناك مسوغ.
OpenFileDialog	تعرض نموذج ويندوز القياسي المستخدم لفتح ملف مختار من قبل المستخدم. يستطيع المستخدم اختيار واحد أو أكثر من الملفات الموجودة من أنظمة الملفات المحلية أو البعيدة. تعرض صندوق حوار الاختيار باستخدام طريقة الأداة ShowDialog، تحصل على النتيجة بواسطة الخاصية FileName أو الخاصية FileNames، توفر الطريقة OpenFileDialog طريقة سريعة لفتح ملف مختار.

تعرض نموذج ويندوز قياسي يتم استخدامه لترتيب صفحة الطباعة بواسطة المستخدم. تعرض صندوق حوار الاختيار بواسطة طريقة الأداة ShowDialog، تحصل على النتيجة بواسطة الخاصية PageSettings والخاصية PrinterSettings.	PageSetupDialog
هذه الأداة عبارة عن أداة تجميع على الفورم. الأدوات التابعة يتم رسمها أو لصقها على هذه الأداة، للوصول إلى نفس التخصص الوظيفي مع حدود ظاهرة وعرض عنوان للمستخدم استخدم الأداة GroupBox.	Panel
تعرض هذه الأداة صورة بهيئات متنوعة.	PictureBox
تعرض نموذج ويندوز قياسي يتم استخدامه لاختيار خاصية الطباعة وطباعة مستند من قبل المستخدم. تعرض صندوق حوار الاختيار باستخدام طريقة الأداة ShowDialog. يتم استخدام هذه الأداة بالعلاقة مع حالة من الفئة PrintDocument، والتي يتم إنشائها من خلال الكود أو بواسطة الأداة PrintDocument.	PrintDialog
يتم استخدام هذه الأداة كجزء من عمليات معاينة الطباعة والطباعة. فهي تضيف دثار حول تنفيذ الطباعة الخاصة بك، موفرة طريقة ثابتة لاختيار وطباعة صفحات مستند.	PrintDocument
توفر هذه الأداة واجهة قياسية من أجل معاينة الطباعة، منفذة جميع العناصر لحوار معاينة الطباعة بالكامل. عندما تستخدمها مع الفئة أو الأداة PrintDocument، فهي تعرض بدقة على الشاشة ما سيظهر على صفحة الطباعة النهائية في الحقيقة، ليس من الضروري أن يعلم كودك فيما إذا سيرسل طابعته إلى الطابعة أو عرض معاينة الطباعة.	PrintPreviewDialog
يوفر تغذية راجعة رسومية للمستخدم من أجل مجال إتمام مهمة. عادة، يكون المجال بين 0 و 100%، ولكن تستطيع توفير مجال مخصص. تشير الخاصية Value للإعدادات الحالية بين حدود المجال الأدنى Minimum والأعلى Maximum.	ProgressBar
تسمح هذه الأداة للمستخدم من تحرير أعضاء معينة لحالة فئة مرافقة بشكل ظاهر أو تصويري. ولوحة خصيات بيئة الفيجوال أستوديو هي حالة عن هذه الأداة. تستخدم هذه الأداة بشكل كبير المواصفات المعقدة على فئة للتحكم بالعرض وتحرير ميزات الخاصيات.	PropertyGrid
تنفذ هذه الأداة زر اختيار تبديل radio ويندوز قياسي. على الرغم من عرض نقطة دائرية أكثر شيوعاً، يمكن أن تظهر الأداة أيضاً كزر مفصلي toggle button بوضع الخاصية Appearance بشكل مناسب. تشير الخاصية Checked إلى القيمة الحالية للأداة. جميع أدوات زر التبديل RadioButton التي تظهر ضمن نفس "سياق المجموعة" group context " تتصرف بأسلوب خاص مشترك، استخدم الأدوات Panel و GroupBox لإنشاء سياقات لمجموعة معينة.	RadioButton
تسمح لك هذه الأداة من تصميم وعرض تقارير محزمة ومربوطة إلى تجمعات أو مصدر بيانات أو دوت نت ADO.NET. وتعمل أيضاً مع خدمات تقارير سكول سرفر SQL Server المولدة للتقارير. استخدام هذه الأداة لتصميم تقرير سيضيف ملف rdcl إلى تطبيقك والذي يحتوي تصميم تقرير حقيقي.	ReportViewer
تعرض هذه الأداة نموذج ويندوز قياسي يتم استخدامه لاختيار حفظ save ملف من قبل المستخدم. يمكن للمستخدم من أن يختار ملف جديد أو موجود من أنظمة الملفات المحلية أو البعيدة. وتطلب هذه الأداة من المستخدم بشكل اختياري من معاينة الملفات الموجودة. عرض صندوق حوار الاختيار باستخدام الطريقة ShowDialog، والحصول على النتيجة بواسطة الخاصية FileName. توفر الطريقة OpenFileDialog طريقة سريعة لفتح الملف المختار.	SaveFileDialog
تضيف هذه الأداة شريط تقسيم (أو تقطيع split bar) تستطيع بواسطته تقسيم فورمك إلى عدة قطاعات قابلة للتحميل، كل منها يحتوي أداة لوحة Panel. استخدم الخاصية Orientation لتبديل اتجاه التقسيم. سيئاتر الترتيب الذي تضيف فيه أدوات SplitContainer يقابلية استخدام التقسيمات، أوصى بالتدريب على هذه الأداة.	SplitContainer
تعرض هذه الأداة شريط الحالة status bar، عادة على طول الحافة الدنيا من فورمك، من خلالها تستطيع عرض حالة، ومعلومات حساسة بالسياق أخرى للمستخدم. يمكن للشريط من أن يحتوي عدة أدوات أشرطة تقدم ProgressBar، لوحة شريط حالة StatusStripPanel، وأداة عنوان شريط أدوات ToolStripLabel.	StatusStrip
تتيح لك تقسيم الأدوات التي على فورمك إلى عدة قطاعات مجزولة. كل اسم محدد (مسمى) لديه أداة صفحة جدول TabPage، والتي تعمل بشكل مشابه كثيراً إلى. اضع أو الصق أدوات تابعة مباشرة إلى كل أداة TabPage.	TabControl
تنفذ هذه الأداة صندوق نص ويندوز قياسي، في نمط وحيد السطر singleline ومتعدد الأسطر multiline. محتوى الجسم الرئيسي هو مجموعة على طول خاصية النص Text. تسمح لك الخاصيات PasswordChar و UseSystemPasswordChar من تقنيح المدخلات عندما يعمل المستخدم على كتابة كلمة المرور.	TextBox
تطلق هذه الأداة حدث التوقيت بالفترة interval التي تحددها أنت. قياس الفترة، بالملي ثانية milliseconds، يتم وضعها من خلال الخاصية Interval. إذا تم وضع الخاصية Enabled إلى صواب True، سيتم استدعاء معالج الحدث Tick كل مرة (يقابل أو يلاقي) الفترة. على الرغم من أنك تستطيع وضع فترة صغير كملي ثانية واحدة، فمن غير المحبب أن تعمل هذا التحقق بشكل متكرر مع أجهزة الكمبيوتر اليومية.	Timer
تنفذ هذه الأداة شريط أدوات toolbar تظهر عليه أدوات أخرى. وتأتي مع مجموعة أدوات مرافقة وفئات توفر معالجة وميزات تفاعل المستخدم متقدمة.	ToolStrip
توفر طريقة مريحة لإضافة أدوات شريط قائمة MenuStrip، شريط حالة StatusStrip، وشريط أدوات ToolStrip لحواف الفورم.	ToolStripContainer
تتيح لك هذه الأداة تحديد "تلميح أداة tool tip" من أجل الأدوات الأخرى على الفورم. عند استخدامها، فهي تضيف شبه خاصية "تلميح أداة ToolTip" لكل أدوات الفورم الأخرى، من خلالها تستطيع تزويد نص تلميح الأداة المرافق. عندما تحوم الفارة فوق أداة يظهر نص التلميح المرافق، وذلك من خلال ظهور نافذة نص صغيرة بشكل مؤقت فوق الأداة لتوفير معلومات مفيدة للمستخدم.	ToolTip
تسمح للمستخدم من عمل اختيار من بين عدد صغير من القيم المرتبة ذات الصلة. وهو موازن (أو مثيل أو نظير) للعالم الحقيقي لأداة التحكم بحجم الصوت على الراديو. استخدم هذه الأداة كبدل لأداة HScrollBar و NumericUpDown أو VScrollBar عندما يكون هناك مسوغ.	TrackBar
تجلب هذه الأداة مجموعة بنود في ترتيب هرمي. وهي مشابهة تماماً إلى قسم "شجرة الأدلة directory tree" مستكشف ملفات ويندوز أو فيستا. كل بند في الشجرة هو عقدة node يمكن أن يكون لديها صفر عقدة أو عقد أبناء أكثر.	TreeView
تنفذ هذه الأداة شريط تمرير شاقولي، مع السماح للمستخدم من التمرير شاقولياً على القطاع المعروض أو قائمة اختيارات من أجل التنفيذ الأفقي لهذه الأداة، استخدم الأداة HScrollBar، تتضمن العديد من الأدوات الأخرى نسخة خاصة بها لأشرطة التمرير	VScrollBar



الأحداث و التفويض Events and Delegates

كل نموذج وأداة في الدوت نت تحتوي على إجراء ويندوز WndProc الخاص بها، وكما أنها تعالج الرسالة الواردة من مضخة الرسائل، وترجم هذه الرسائل إلى أحداث events. إن الأحداث Events هي تقنية دوت نت القياسية التي تتحكم-وجميع الفئات الأخرى- باستخدام كل من الفورم والأدوات. فعندما تُضمن فورم أو أداة في تطبيقك، تستطيع عرض واحد، أو بعض، أو كل هذه الأحداث، وكتابة الكود الذي يستجيب بشكل مناسب. كل الكود المخصص الذي تكتبه لكل حدث يظهر في معالج الحدث event handler. ولكن ما الذي يحدث فعلياً بين إصبع المستخدم على الماوس والمنطق في معالج حدث مخصص؟. يبين الشكل التالي بشكل بياني ما يحدث عملياً بين الإجراء (الفعل) والمنطق المخصص.



من الواضح، أنه ما يزال بعض الغموض mystery محيط بمعالج الحدث. الأدوات Controls - وكل الفئات classes - تحدد أي الأحداث ستجعل منها متاحة. فمن أجل الأدوات يتم إتاحة العديد من الأحداث بالتوازي مع إجراءات المستخدم الأولية: النقر Click، الماوس للأسفل MouseDown، ضغط مفتاح KeyPress، والحجم تغيير SizeChanged. ولكن يوجد أيضاً العديد من الأحداث يمكن أن يتم إطلاقها فقط من خلال التعديلات على الأداة خلال الكود المصدري: TabIndexChanged (عندما يتم تغيير ترتيب مفتاح التنقل tab-key للأدوات)، BackgroundImageChanged و CursorChanged هي فقط ثلاث من الأحداث التي لا يمكن للمستخدم أن يؤثر عليها بشكل مباشر. العديد من الأحداث الأخيرة ترتبط بالتغيرات على مستوى النظام system-level changes، مثل الحدث SystemColorsChanged، والذي يتم إطلاقه عندما يعدل المستخدم جدول الألوان الواسعة للنظام system-wide color scheme من خلال لوحة التحكم control panel.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
```

End Sub

يستقبل معالج الحدث هذا معاملين نسبيين arguments من حدث تم إطلاقه: System.Object (حالة المرسل sender) و System.EventArgs (حالة e)، معالجات أحداث أخرى يمكن أن تستخدم معاملات نسبية مختلفة قليلاً، لذلك كيف تعرف ما تستخدم؟ إن أي حدث معرف ضمن فئة أداة يجب أن يشير أيضاً إلى عدد ونوع المعاملات النسبية التي يرسلها إلى معالج الحدث. تتضمن الفيچوال بيسك العبارة Event التي تعرف الأحداث. مع أن وعلى الأرجح أنه تم كتابة الزر في السي شارب C#، إليك ما يبدو عليه تعريف حدث لحدث نقر الزر في الفيچوال بيسك:

```
Public Event Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

هذا التعريف يبدو مشابه كثيراً لمعالج الحدث السابق (Private Sub)، وهو كذلك. تؤسس العبارة Event عقد (اتفاقية) تمرير وسيط parameter-passing contract بين الأداة والكود الذي يريد أن يستقبل إشعارات (بيانات) الحدث event notifications. في هذه الحالة، حدث النقر بعد promises أن يرسل معاملين نسبيين إلى معالج الحدث. الأول هو المرسل sender، وهو مرجع (دليل reference) للكائن الذي يشير إليه الحدث. من أجل أدوات الزر Button controls، هذا الوسيط يستقبل مؤشر لنسخة (حالة) الزر نفسه. المعامل النسبي الثاني، e، يوفر طريقة لتمرير كائن كامل المعلومات الإضافية. إن الفئة System.EventArgs ليس لديها الكثير من المعلومات، ولكن بعض الأحداث تستخدم تنوع ما في المعامل النسبي الثاني الذي يستخدم System.EventArgs كفئة قاعدية له. هذا يكشف أن المعاملات النسبية المستخدمة لحدث النقر هي مشتركة إلى حد ما بين الأدوات والأحداث المختلفة. فبعوضاً عن إعادة كتابة قائمة المعاملات النسبية في كل عبارة حدث، يمكن لمصمم الأداة أن يعرف تفويض delegate: نوع ما للدوت نت يعمل على تعريف قائمة معاملات نسبية، أما من أجل الدالات، قيمة راجعة.

```
Public Delegate Sub StandardEventDelegate(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

يمكن لعبارة الحدث فيما بعد استخدام تفويض معرف كاختصار لقائمة الوسيطات الكاملة المكتوبة.

```
Public Event Click As StandardEventDelegate
```

سواء كان الحدث يستخدم تفويض أو قائمة المعاملات النسبية الكاملة، فله سيطرة تامة firm grasp على ما يحتاج أن يرسله من بيانات إلى أي معالج حدث مستقبل. وإنه يرسل هذه المعاملات النسبية باستخدام عبارة الفيچوال بيسك RaiseEvent. دعنا نفتفي أثر هذه المعالجة من أجل أداة الزر. عندما ينقر المستخدم على الزر، تعمل مضخة الرسائل message pump على إيجاد طريقة لترسل رسالة إلى إجراء WndProc الخاص بأداة الزر. وإن ذلك الزر يتفحص الرسالة، ويرى أنها نقر بالماوس ويقرر أن يخبر معالج الحدث event handlers حولها. ومن ثم، من ضمن كود WndProc، يعمل على إطلاق الحدث.

```
RaiseEvent Click(Me, New System.EventArgs)
```

تشير كلمة الفيچوال بيسك المحجوزة Me إلى حالة (نسخة) أداة الزر نفسها. لا يحتوي المعامل النسبي e لأداة الزر أبعد من الحقل الافتراضية المضمنة في نسخة (حالة) System.EventArgs، لذلك فإن WndProc فقط يرسل حالة (نسخة) فارغة جديدة. الأدوات ذات حدث مع معاملات نسبية أخرى سيكون قد أنشئ حالة أولاً، يملأها بالبيانات المتعلقة، ويمرر تلك النسخة إلى معالج الحدث. إذا ما أطلق حدث في تطبيق، ولم يكن هناك معالج حدث يسمعه، هل يعمل صوت؟ من المحتمل لا. لا يوجد متطلبات لأن يكون لحدث ما معالجات فعالة مستقبلية (منصتة). ولكن عندما نريد أن نهتم بحدث، كيف نفعل هذا؟ الطريقة القياسية لعمل هذا في تطبيق نماذج ويندوز هناك خطوات معالجة الأولى، يحتاج مستخدم الأداة (فئة فورمك your form class) أن يعلن عن أداة، بمعنى "أريد أن أعرض أحداثك (أي أن الفورم ستعرض أحداث الأداة الموضوعة على سطحها)" ومن ثم توصل معالجات الأحداث بأحداث خاصة. سابقاً في هذا الفصل، رأينا أن إضافة أداة إلى واجهة المستخدم يحدث عملياً على كتابة كود مصدري source code في الملف Form1.designer.vb. إليك الكود الذي يتم إضافته من أجل أداة زر مسماة Button1 (مع أرقام الأسطر).

```
01 Partial Class Form1
02 Inherits System.Windows.Forms.Form
03
04 Friend WithEvents Button1 As System.Windows.Forms.Button
05
06 Private Sub InitializeComponent ( )
07 Me.Button1 = New System.Windows.Forms.Button
08
09 Me.Button1.Location = New System.Drawing.Point(48, 16)
10 Me.Button1.Name = "Button1"
11 Me.Button1.Size = New System.Drawing.Size(75, 23)
12 Me.Button1.TabIndex = 0
13 Me.Button1.Text = "Button1"
14 Me.Button1.UseVisualStyleBackColor = True
15
16 Me.Controls.Add(Me.Button1)
17 End Sub
18 End Class
```

الكود في الطريقة InitializeComponent يعمل على إنشاء حالة لأداة الزر (السطر 7)، يعمل على تعديل خصائصه للحصول على المظهر الذي نريد (السطر 9 إلى السطر 14)، يلحقه بالفورم من خلال السطر 16. ولكن يوجد سطر واحد إضافي يعمل على تعريف حقيقة الزر كنوع لمتغير مرجعي reference type variable (السطر 4):

```
Friend WithEvents Button1 As System.Windows.Forms.Button
```

سنحدث فيما بعد عن الحقول على مستوى الفئة class-level fields، والزر Button1 هو حقل نموذجي على مستوى الفئة. ولكن الكلمة المحجوزة WithEvents المضمنة في العبارة هي ما تتيح للأداة معرفة أن أحداً يريد عرض إشعارات الحدث. الآن، في أي وقت ينطلق حدث الزر، فإنه يعرف أنه يمكن للفورم Form1 أن تحتوي على معالجات حدث ترافق وتهتم (تستقبل watching and listening). الجزء الثاني للخطوة: تتضمن عملية الوصل بين الحدث والمعالج event-to-handler connection process (الربط الحقيقي للمعالج). لننظر مرةً أخرى على تعريف معالج حدث النقر Click event handler لنسخة الزر Button1.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
```

```
End Sub
```

فهي مجرد طريقة فئة عادية، ولكن مع الشرط Handles المعلق في نهاية التعريف. هذا التعبير هو ما يربط معالج الحدث event handler مع حدث Button1.Click نفسه. تستطيع إتباع الكلمة المحجوزة Handles بعدة أسماء أحداث.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click, Button2.Click, Button3.Click
```

```
End Sub
```

الآن، سيستمع معالج حدث مفرد لنقر عدة أزرار من عدة أدوات مختلفة. بإمكانك حتى مزج الأحداث المعروضة، فيمكن لمعالج حدث واحد أن يستمع لأحداث مسماة مختلفة.

```
Private Sub ManyEvents(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.MouseDown, Button2.MouseUp
End Sub
```

نوع آخر هو أن يكون لمعالجات حدث متعددة عرض حدث مفرد، على الرغم من أن على الفيچوال بيسك أن تقرر استدعاء أي معالج أولاً

```
Private Sub FirstHandler(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
End Sub
```

```
Private Sub SecondHandler(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
End Sub
```

يوجد طريقة أخرى لوصول أحداث events بمعالجات أحداث event handlers والتي لاتتضمن سواء الكلمة المحجوزة WithEvents أو الكلمة المحجوزة Handles، وهذا ما يتيح لك السيطرة على ترتيب المعالجة لعدة معالجات حدث. حالما يكون لديك حالة لفئة تعرض حدث event-exposing class، فإنك تربط معالج إلى واحد من أحداثه باستخدام العبارة AddHandler. العبارة التالية تربط حدث نقر الزر Button1 بمعالج حدث مسمى MyHandler. يجب أن يكون للطريقة MyHandler قائمة معاملات نسبية محققة من أجل تعريف الحدث.

```
AddHandler Button1.Click, AddressOf MyHandler
```

تزيل العبارة المخصصة RemoveHandler اتصال معالج حدث بحدث ما.

سنمضي الكثير من الوقت في كتابة معالجات الأحداث. ولكنني سأدخل أيضاً ضمن جميع التفاصيل بحيث تستطيع أخذ فوائد جمة عن هذه التقنية. فلا تسمح لك الفيچوال بيسك من عرض الأحداث الخاصة بالأدوات فقط ولكن أيضاً تتيح لك تصميم أحداث جديدة في فئاتك الخاصة. بإمكانك استخدام الكلمات المحجوزة Delegate أو Event، RaiseEvent، Handles، WithEvents، AddHandler، RemoveHandler، من أجل أحداثك المخصصة، والتي يتم إطلاقها بواسطة شروط ما تختارها أنت. فإذا كان لديك فئة تمثل موظف employee، بإمكانك جعلها تطلق حدث "الطرد Fired" عندما يفقد الموظف عمله. بإضافة أحداث خاصة، فإنك تجعل من الممكن لكود خاص من أن يتم وصله بمنطق فئتك class logic، أو حتى ولو لم يكن لمبرمج ما صلاحية الدخول للكود المصدري التابع لفئتك.

جعل النماذج مفيدة Making Forms Useful

إن أي نموذج تعمل على تعريفه ككائن بدء لمشروعك يظهر بشكل آلي عندما يبدأ البرنامج. أما بقية النماذج الأخرى فتحتاج لعرضها بشكل يدوي، باستخدام إما الطريقة "إظهار Show" أو الطريقة "إظهار حوار ShowDialog" لتلك الفورم. مثلاً، إذا كان لديك فورم تدعى Form2 تستطيع عرضها باستخدام الطريقة ShowDialog:

```
Form2.ShowDialog()
```

تعرض الطريقة Show نموذج غير نمطي *modeless form*. النماذج غير النمطية يمكن الوصول إليها بشكل مستقل عن جميع الفورمات الأخرى في التطبيق المشغل. ويمكن لكل الفورمات الغير نمطية أن يتم تفعيلها في أي وقت بالنقر عليها فقط، والفورم التي تنقر عليها ستأتي إلى الواجهة أمام جميع الفورمات الأخرى وستستقبل تركيز الإدخال input focus. يمكن لبرنامج أن يكون له واحد، أو اثنان أو حتى الكثير من النماذج الغير نمطية تفتح في نفس الوقت، ويمكن للمستخدم أن يتنقل بينها بكل حرية.

النماذج النمطية *Modal forms* تتحكم بجميع الإدخالات في التطبيق، طالما أنها تظهر على الشاشة. والنماذج النمطية تدعى عادةً "الحوارات dialogs"، فعلى المستخدم أن يكمل النموذج النمطي ويغلقه قبل أن يتمكن من الوصول لفتح أي نموذج آخر في التطبيق. فنافذة صندوق الرسالة التي تظهر عندما تستخدم الدالة MsgBox هي نافذة حوار نمطية. تعرض الطريقة ShowDialog نماذج نمطية، وتتيح لك العودة بقيمة من تلك الفورم. والقيمة المعادة هي أعضاء العداد DialogResult. System.Windows.Forms.DialogResult. تعرض نموذج نمطي استخدم طريقة الفورم ShowDialog وبشكل اختياري التقط قيمتها المعادة.

```
Dim theResult As DialogResult
theResult = Form2.ShowDialog()
```

الحوارات النمطية Modal dialogs مفيدة لتحديث سجل ما يتطلب نقر على زر موافق OK عند اكتمال التغييرات. لنقول أنك كنت تكتب تطبيق يعرض قائمة كتب ألكسندر دامس. من المحتمل أن يتضمن نموذجين (1) النموذج الرئيسي الذي يعرض قائمة الكتب، (2) النموذج الثانوي (الابن) والذي يتيح لك كتابة اسم كتاب مفرد. ألن يكون من العظيم إذا استطعت العودة باسم الكتاب (أو ربما، بالرقم المعرف ID لسجل ما خاص بالكتاب كما تم تخزينه في قاعدة البيانات) بدلاً من قيمة نتيجة الحوار DialogResult. إذا كانت الطريقة ShowDialog، طريقة عامة public method لفئة فورم مضمنة، تستطيع العودة بنتيجة كود، من المحتمل أننا نستطيع إضافة طرق عامة أخرى للنموذج تعود بنتيجة كود بحيث يكون لديها معنى حقيقي، في الواقع، نستطيع ذلك. خذ الفورم الابن (المسمى BookEntry) مع حقل البيانات المدخلة (عنوان الكتاب BookTitle)، والأزرار OK (ActOK) و Cancel (ActCancel)، كما هو مبين في الشكل التالي.

عندما ندعمه بالكود التالي، هذا الفورم البسيط يعود بما تم كتابته في الحقل عندما ينقر المستخدم على زر موافق (في البداية يرفض القيم الفارغة)، أو يعود بنص فارغ عند النقر على إلغاء Cancel.

```
Public Class BookEntry
    Public Function EditTitle() As String
        ' ----- Show the form, and return what the user enters.
        If (Me.ShowDialog() = DialogResult.OK) Then
            Return BookTitle.Text.Trim
        Else
            Return ""
        End If
    End Function
    Private Sub ActCancel_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        ActCancel.Click
        ' ----- Return a blank title for "Cancel."
        Me.DialogResult = DialogResult.Cancel
        ' ----- Continue with EditTitle( )
    End Sub
End Class
```

```

End Sub
Private Sub ActOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ActOK.Click
' ----- Only accept valid titles.
If (Len(BookTitle.Text.Trim) = 0) Then
MsgBox("Please supply a valid title.")
Else
Me.DialogResult = DialogResult.OK
' ----- Continue with EditTitle( )
End If
End Sub
End Class

```

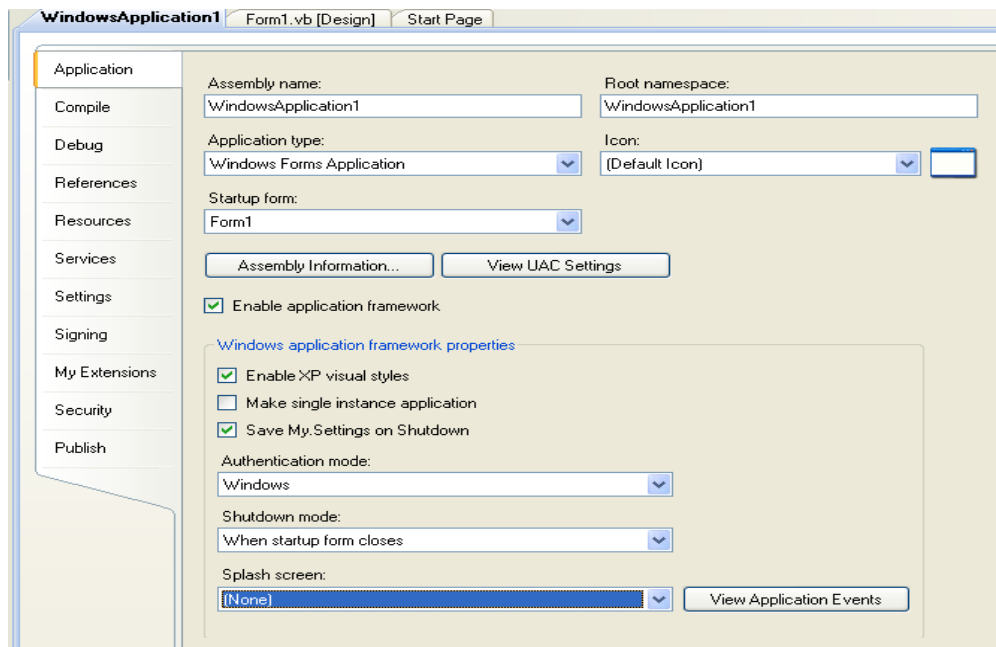
لاستخدام هذه الفورم، تستدعي الفورم الرئيسي الطريقة EditTitle، والتي تعود بعنوان الكتاب المدخل من قبل المستخدم.

```
Dim newTitle As String = BookEntry.EditTitle()
```

يعرض الروتين EditTitle الفورم بشكل نمطي form modally من خلال الطريقة ShowDialog، وينتظر هناك فقط حتى يغلق المستخدم الفورم. إغلاق الفورم يتم من خلال الزر موافق أو الزر إلغاء، وإعداد خاصية DialogResult للفورم لديه تأثير جانبي على إغلاق الفورم. حالما يتم إغلاق الفورم، يتم إرجاع تنفيذ إلى EditTitle، والذي يعمل تفحص سريع قبل العودة بالقيمة النهائية. وبالتالي نحصل عليها: واجهة عامة جديدة من أجل قيمة النماذج المعادة الأكثر أهمية. سنستخدم هذه الطريقة كثيراً في تطبيق مشروع المكتبة.

مشروع Project

كود مشروع هذا الفصل ينفذ الفورم الرئيسي الأساسي Main لمشروع المكتبة، بالإضافة إلى الفورم "Splash" التي تظهر عندما يشتغل المشروع للمرة الأولى. تعرف ميكروسوفت أن هذه كانت حاجة مشتركة، متضمنة دعم لكل من الفورمات الرئيسية والمنتشرة في نظام إطار عمل تطبيق الفيجوال بيسك. بشكل افتراضي، هذا النظام يتم تمكينه من خلال صفحات خاصيات المشروع (شاهد الشكل التالي).



تشير حقول "فورم الانطلاق Startup form" والنافذة "المنتشرة Splash screen" إلى الفورم الرئيسي main ومنتشرة splash على الترتيب. فهذا سريع، وسهل، وهو مجرد شيء معد للاستخدام، لذلك لندخل في العمل. سيكون الآن وقت عظيم لتحميل المشروع البادئ للفصل السابع.

تركيب الشاشة المنتشرة Configuring the Splash Screen

لقد عملت على إضافة ملف فورم جديد للمشروع، مسمى *Splash.vb* (أي أن الفورم نفسه مسمى Splash)، متضمناً بعض العناصر المعروضة. تحرى عن الصورة على الفورم. يتم جلبها من خلال أداة "صندوق الصورة PictureBox"، ولكن تم تخزينها في التطبيق كمصدر *resource*، يتم إلحاق مجموعة من السلاسل الحرفية *strings* والصور *images* لكودك المصدري. يتضمن مجلد المصادر *Resources* في مستكشف الحلول ملف هذه الصور. ويتم وصله ضمن صندوق الصورة من خلال خاصية *Image* للأداة. ومن المؤكد أنها تجعل الفورم ذات مظهر جيد. عملك يكون ربط هذه الفورم ضمن تتابع بدء *startup* التطبيق.

اذهب إلى خاصيات المشروع (المعروض في الشكل في الأعلى). ضع حقل "Splash screen" إلى *Splash*. وهذا له تأثير جانبي على إعداد *My.Application.SplashScreen* لفورم *Splash*. والآن شغل البرنامج. سترى أن النافذة المنتشرة تظهر حوالي 1/100 من الثواني، وبسرعة يتم استبدالها بواسطة الفورم الرئيسي.

كما رأيت تعديل حقل "Splash screen" يؤدي إلي ظهور النافذة المنتشرة splash screen بإيجاز. ولكن سيحفظها التطبيق ظاهرة حتى يكتمل عقد التحضير الكافي للفورم الرئيسي. بما أننا لم نعمل أي تحضير، فإن التطبيق يعرض الفورم الرئيسي حالاً.

أخيراً، سنضيف حزمة كود بدء خاص بقاعدة البيانات bunch of database-related startup code والذي سيأخذ وقت أكثر. ولكن حالياً، سنعمل على تزييفه fake. في وحدة model إطار العمل للتطبيق، أي كود تريد معالجته عندما يبدأ البرنامج الظهور للمرة الأولى يتم في حدث البدء (Startup) وهذا الحدث هو واحد من التجميعات الصغيرة لأحداث مضمنة مع التسلسل الهرمي My. يظهر الكود المصدري لهذه الأحداث في ملف ApplicationEvents.vb file، الملف الذي تضيفه الفيچوال بيسك بشكل آلي لمشروعك عند الحاجة. استخدم زر "عرض أحداث التطبيق View Application Events" في صفحات خصائص المشروع project properties ومن تبويب تطبيق Application لفتح ملف الكود المصدري.

```
Namespace My
    Partial Friend Class MyApplication
```

```
End Class
```

```
End Namespace
```

لنفرض أن المتطلبات الأولية لمشروع المكتبة تأخذ حوالي ثلاث ثواني. تتضمن الدوت نت الطريقة Sleep والتي تؤجل الكود لعدد معين من الملي ثانية.

```
Namespace My
    Partial Friend Class MyApplication
```

```
Private Sub MyApplication_Startup(ByVal sender As Object, ByVal e As
Microsoft.VisualBasic.ApplicationServices.StartupEventArgs) Handles Me.Startup
    ' ----- Take a three-second nap.
    System.Threading.Thread.Sleep(3000)
End Sub
End Class
```

```
End Namespace
```

شغل البرنامج الآن، وسترى أن النافذة المنتشرة تبقى حوالي ثلاث ثواني (3000ملي ثانية). بالتالي توجد فرصة جيدة لأن يأخذ كل من قاعدة البيانات وكود بدء التطبيق وقت أقل من ثلاث ثواني، أعني، نحن نتكلم عن خادم سكول SQL هنا، من المفترض أن يتوهج بسرعة blazing fast.

لذلك من الواضح أننا مانزال بحاجة لتأجيل انتقال النافذة المنتشرة. إن الكائن My.Application يحدث لأن يتضمن الخاصية الأفضل التي نحتاجها لنؤكد على التأجيل. تشير الخاصية MinimumSplashScreenDisplayTime إلى العدد الأقل من الملي ثانية الذي يجب أن يتم عرض الشاشة المنتشرة. القسم السيئ هو أن عليك إسنادها في مكان غريب حقاً، على الأقل عند المقارنة إلى كيفية برمجة الفيچوال بيسك التي تعلمناها حتى الآن.

احذف كل الكود الذي عملنا على إضافته حتى الآن، أي احذف الطريقة MyApplication_Startup بالكامل. ومن ثم ضع الكود التالي مكانه.

```
Namespace My
    Partial Friend Class MyApplication
        Protected Overrides Function OnInitialize (ByVal commandLineArgs As System.Collections.
ObjectModel.ReadOnlyCollection(Of String)) As Boolean
            ' ----- Display the splash form for at least 3 seconds.
            My.Application.MinimumSplashScreenDisplayTime = 3000
            Return MyBase.OnInitialize(commandLineArgs)
        End Function
    End Class
End Namespace
```

يحتوي مقطع الكود على الكثير من الأشياء التي لم نتكلم عنها حتى الآن، ولن أتكلم عنها حتى عدة فصول لاحقة. يكفي القول أن الطريقة OnInitialize هي واحدة من أول الأشياء التي تحدث في عمر البرنامج، وأن هذا هو المكان الذي يتم فيه إسناد MinimumSplashScreenDisplayTime.

الشيء الأخير الذي عليك عمله للشاشة المنتشرة splash screen هو تضمين بعض الكود الذي يعرض رقم النسخة version number. لقد عملنا هذا سابقاً من أجل نموذج في الفصل السابق، لذلك سنعمل هنا على إضافة كود مشابه لمعالج حدث تحميل Load event فورم Splash. وسنحدث رسالة حقوق الطباعة copyright، أيضاً، أفتح الكود المصدري لفورم Splash وأضف الكود التالي:

```
Public Class Splash
    Private Sub Splash_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        ' تحديث رقم الاصدار
        With My.Application.Info.Version
            ProgramVersion.Text = "Version " & .Major & "." &
.Minor & " Revision " & .Revision
        End With
        ProgramCopyright.Text = My.Application.Info.Copyright
    End Sub
End Class
```

الآن شغل البرنامج ولاحظ كيف تنتظر فترة محددة ومن ثم تظهر الفورم الرئيسية.

تركيب الفورم الرئيسية Configuring the Main Form

على الرغم من أننا صممنا الفورم الرئيسي في فصل سابق، فهو متناثر نسبياً، وتتضمن فقط زر واحد هو "حول About". مشروع هذا الفصل يضيف كل عناصر واجهة المستخدم المستخدم للفورم. في الحقيقة، لقد عملت على إضافة الأدوات لسطح تلك الفورم من أجلك (شاهد الشكل التالي). ولكن تستطيع إضافة بعض معالجات الحدث التي تجعل الفورم تعرض بعض حيوتها pizzazz.



كل كود الحدث العام من أجل الفورم يظهر كما يلي.

```
Private Sub ActLibraryItem_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles ActLibraryItem.LinkClicked
    ' ----- Library Item mode.
    TaskLibraryItem()
End Sub
Private Sub PicLibraryItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PicLibraryItem.Click
    ' ----- Library Item mode.
    TaskLibraryItem()
End Sub
Private Sub ActPatronRecord_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles ActPatronRecord.LinkClicked
    ' ----- Patron Record mode.
    TaskPatronRecord()
End Sub
Private Sub PicPatronRecord_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PicPatronRecord.Click
    ' ----- Patron Record mode.
    TaskPatronRecord()
End Sub
Private Sub ActHelp_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles ActHelp.LinkClicked
    ' ----- Help mode.
    TaskHelp()
End Sub
Private Sub PicHelp_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PicHelp.Click
    ' ----- Help mode.
    TaskHelp()
End Sub
Private Sub ActCheckOut_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles ActCheckOut.LinkClicked
    ' ----- Check Out mode.
    TaskCheckOut()
End Sub
Private Sub PicCheckOut_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PicCheckOut.Click
    ' ----- Check Out mode.
    TaskCheckOut()
End Sub
Private Sub ActCheckIn_LinkClicked(ByVal sender As System.Object, ByVal e As System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles ActCheckIn.LinkClicked
```

```

' ----- Check In mode.
TaskCheckIn()
End Sub
Private Sub PicCheckIn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles PicCheckIn.Click
' ----- Check In mode.
TaskCheckIn()
End Sub
Private Sub ActAdmin_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles ActAdmin.LinkClicked
' ----- Administration mode.
TaskAdmin()
End Sub
Private Sub PicAdmin_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
PicAdmin.Click
' ----- Administration mode.
TaskAdmin()
End Sub
Private Sub ActProcess_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles ActProcess.LinkClicked
' ----- Daily Processing mode.
TaskProcess()
End Sub
Private Sub PicProcess_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles PicProcess.Click
' ----- Daily Processing mode.
TaskProcess()
End Sub
Private Sub ActReports_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles ActReports.LinkClicked
' ----- Print Reports mode.
TaskReports()
End Sub
Private Sub PicReports_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles PicReports.Click
' ----- Print Reports mode.
TaskReports()
End Sub
Private Sub TaskLibraryItem()
' ----- Library item task.

' ----- Update the display.
AllPanelsInvisible()
PanelLibraryItem.Visible = True
ActLibraryItem.BackColor = SystemColors.Control
LabelSelected.Location = New System.Drawing.Point(LabelSelected.Left, PicLibraryItem.Top)
Me.AcceptButton = ActSearch

' ----- Get ready for a new search.
On Error Resume Next
SearchText.Focus()
End Sub
Private Sub TaskPatronRecord()
' ----- Patron Record mode.

' ----- Update the display.
AllPanelsInvisible()
PanelPatronRecord.Visible = True
ActPatronRecord.BackColor = SystemColors.Control
LabelSelected.Location = New System.Drawing.Point(LabelSelected.Left, PicPatronRecord.Top)
Me.AcceptButton = ActAccessPatron
End Sub
Private Sub TaskHelp()
' ----- Help mode.

' ----- Update the display.

```

```

AllPanelsInvisible()
PanelHelp.Visible = True
ActHelp.BackColor = SystemColors.Control
LabelSelected.Location = New System.Drawing.Point(LabelSelected.Left, PicHelp.Top)
Me.AcceptButton = Nothing
End Sub
Private Sub TaskCheckOut()
' ----- Check Out mode.

' ----- Update the display.
AllPanelsInvisible()
PanelCheckOut.Visible = True
ActCheckOut.BackColor = SystemColors.Control
LabelSelected.Location = New System.Drawing.Point(LabelSelected.Left, PicCheckOut.Top)
Me.AcceptButton = ActDoCheckOut
End Sub
Private Sub TaskCheckIn()
' ----- Check In mode.

' ----- Update the display.
AllPanelsInvisible()
PanelCheckIn.Visible = True
ActCheckIn.BackColor = SystemColors.Control
LabelSelected.Location = New System.Drawing.Point(LabelSelected.Left, PicCheckIn.Top)
Me.AcceptButton = ActDoCheckIn

' ----- Reset the input fields.
CheckInDate.Value = Today
CheckInDate.MaxDate = Today
CheckInDate.MinDate = DateAdd(DateInterval.Day, -6, Today)
CheckInDay.Text = "Today"
CheckInDay.BackColor = SystemColors.Control
CheckInDay.ForeColor = SystemColors.ControlText
CheckInBarcode.Text = ""
LabelCheckedIn.Visible = False
CheckedInDetail.Visible = False
CheckedInDetail.Text = "No information available."
CheckInBarcode.Focus()
End Sub
Private Sub TaskAdmin()
' ----- Administration mode.

' ----- Update the display.
AllPanelsInvisible()
PanelAdmin.Visible = True
ActAdmin.BackColor = SystemColors.Control
LabelSelected.Location = New System.Drawing.Point(LabelSelected.Left, PicAdmin.Top)
Me.AcceptButton = Nothing
End Sub
Private Sub TaskProcess()
' ----- Daily Processing mode.

' ----- Update the display.
AllPanelsInvisible()
PanelProcess.Visible = True
ActProcess.BackColor = SystemColors.Control
LabelSelected.Location = New System.Drawing.Point(LabelSelected.Left, PicProcess.Top)
Me.AcceptButton = ActDoProcess
End Sub
Private Sub TaskReports()
' ----- Print Reports mode.

' ----- Update the display.
AllPanelsInvisible()
PanelReports.Visible = True
ActReports.BackColor = SystemColors.Control

```

```

LabelSelected.Location = New System.Drawing.Point(LabelSelected.Left, PicReports.Top)
Me.AcceptButton = ActDoReports
End Sub
Private Sub MainForm_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
' ----- Prepare the form.

' ----- Adjust the various panel positions.
PanelLibraryItem.Location = New Point(TaskListBackground.Width, WelcomeBackground.Height)
PanelPatronRecord.Location = PanelLibraryItem.Location
PanelHelp.Location = PanelLibraryItem.Location
PanelCheckOut.Location = PanelLibraryItem.Location
PanelCheckIn.Location = PanelLibraryItem.Location
PanelAdmin.Location = PanelLibraryItem.Location
PanelProcess.Location = PanelLibraryItem.Location
PanelReports.Location = PanelLibraryItem.Location
PanelLibraryItem.Visible = True
ActSearchLimits.PerformClick()
End Sub
Private Sub AllPanelsInvisible()
' ----- Hide all work panels.
PanelLibraryItem.Visible = False
PanelPatronRecord.Visible = False
PanelHelp.Visible = False
PanelCheckOut.Visible = False
PanelCheckIn.Visible = False
PanelAdmin.Visible = False
PanelProcess.Visible = False
PanelReports.Visible = False

' ----- Correctly color all action labels.
ActLibraryItem.BackColor = Color.White
ActPatronRecord.BackColor = Color.White
ActHelp.BackColor = Color.White
ActCheckOut.BackColor = Color.White
ActCheckIn.BackColor = Color.White
ActAdmin.BackColor = Color.White
ActProcess.BackColor = Color.White
ActReports.BackColor = Color.White
End Sub
Private Sub SearchText_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles
SearchText.Enter
' ----- Highlight the entire text.
SearchText.SelectAll()
End Sub
Private Sub ActSearchLimits_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ActSearchLimits.Click
' ----- Toggle the display of the media type and location fields.
If (ActSearchLimits.Top = LabelMoreLimitsTop.Top) Then
' ----- Hide the two fields.
LabelSearchMediaType.Visible = False
SearchMediaType.Visible = False
LabelSearchLocation.Visible = False
SearchLocation.Visible = False
ActSearchClear.Location = New System.Drawing.Point(ActSearchClear.Left,
SearchMediaType.Top)
ActSearchLimits.Location = New System.Drawing.Point(ActSearchLimits.Left,
SearchMediaType.Top)
ActSearch.Location = New System.Drawing.Point(ActSearch.Left, SearchMediaType.Top)
ActSearchLimits.Text = "Mor&e >>"
Else
' ----- Show the two fields.
ActSearchClear.Location = New System.Drawing.Point(ActSearchClear.Left,
LabelMoreLimitsTop.Top)
ActSearchLimits.Location = New System.Drawing.Point(ActSearchLimits.Left,
LabelMoreLimitsTop.Top)

```

```

        ActSearch.Location = New System.Drawing.Point(ActSearch.Left, LabelMoreLimitsTop.Top)
        LabelSearchMediaType.Visible = True
        SearchMediaType.Visible = True
        LabelSearchLocation.Visible = True
        SearchLocation.Visible = True
        ActSearchLimits.Text = "<< L&ess"
    End If
End Sub
Private Sub CheckInBarcode_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles
CheckInBarcode.Enter
    ' ----- Highlight the entire text.
    CheckInBarcode.SelectAll()
End Sub
Private Sub CheckOutBarcode_Enter(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CheckOutBarcode.Enter
    ' ----- Highlight the entire text.
    CheckOutBarcode.SelectAll()
End Sub
Private Sub MainForm_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles MyBase.KeyDown
    ' ----- The keys F2 through F9 access the different sections of the form.
    Select Case (e.KeyCode)
        Case Keys.F2
            TaskLibraryItem()
            e.Handled = True
        Case Keys.F3
            TaskPatronRecord()
            e.Handled = True
        Case Keys.F4
            ' ----- Allow form to handle Alt+F4.
            If (e.Alt = True) Then
                Me.Close()
            Else
                TaskHelp()
            End If
            e.Handled = True
        Case Keys.F5
            TaskCheckOut()
            e.Handled = True
        Case Keys.F6
            TaskCheckIn()
            e.Handled = True
        Case Keys.F7
            TaskAdmin()
            e.Handled = True
        Case Keys.F8
            TaskProcess()
            e.Handled = True
        Case Keys.F9
            TaskReports()
            e.Handled = True
    End Select
End Sub

```

معظم هذا الكود يوجد لينقل أشياء تخص العرض على سبيل المثال، يمكن للمستخدم أن يصل للميزات المختلفة للفورم بالنقر على الأيقونات أو لصاقات النصوص ذات الصلة على طول الجهة اليسارية من الفورم. كل أيقونة ولصاقة تطلق واحد من سبع روتينات (إجراءات) والتي توجد لترتيب الأساس. الأيقونة اليسارية الأكبر PicLibraryItem ، تستدعي الروتين المشترك TaskLibraryItem عندما يتم النقر عليها.

الإجراء TaskLibraryItem يضبط اللوحات المتنوعة والحقول على العرض بحيث يرى المستخدم هذه الحقول المطلوبة للبحث في بنود المكتبة.

الإجراء AllPanelsInvisible يعمل أيضاً بعض التعديلات على الشاشة.

أحب تضمين النص الموجود في حقل صندوق نص TextBox مختار عندما يصبح أداة فعالة. كل أداة نص text تتضمن الطريقة SelectAll والتي تحقق هذا العمل. سوف نستدعي تلك الطريقة من خلال كل حدث دخول Enter لأداة صندوق نص TextBox ، الحدث الذي يحدث عندما تستقبل أداة تركيز إدخال لوحة المفاتيح keyboard. راجع الكود السابق (Private Sub SearchText_Enter).

إن استخدام الماوس من أجل الوصول إلى الميزات المختلفة للفورم شيء جيد، ولكن للتعامل مع كيبورد (لوحة مفاتيح) المستخدم يجب أن نضيف الكود الذي يدعم ميزات الوصول باستخدام المفاتيح من F2 إلى F9. (راجع الكود السابق (Private Sub MainForm_KeyDown)).

فكل ضرب مفتاح keystroke يأتي إلى معالج حدث KeyDown، وإن عبارة Select Case تختبره. وعندما يطابق حالة مدخلة، فإن الكود ضمن مقطع الحالة Case يتم تنفيذه. ضغط المفتاح F2 يحرر الكود في مقطع الحالة Keys.F2. والمفاتيح هي واحدة من العديد من العدادات الجاهزة والتي تستطيع استخدامها في تطبيقات الدوت نت. لاحظ الكود الخاص بالنسبة للمفتاح F4. فهو يسمح للضم Alt-F4 من الخروج من التطبيق، والذي هو ضم مفتاحي قياسي بالنسبة لبرامج الويندوز الموجودة.

عادة، جميع ضربات المفاتيح keystrokes تذهب إلى الأداة الفعالة، وليس للفورم. لتمكين معالج الحدث MainForm.KeyDown، يجب أن يتم وضع خاصية الفورم KeyPreview إلى صح True. ضع هذه الخاصية وذلك بالرجوع إلى مصمم الفورم وسندوق الخصائص.

جعل البرنامج نسخة مفردة (حالة مفردة). Making the Program Single-Instance.

مشروع المكتبة يتم تصميمه للاستخدام فقط ضمن مكتبة صغيرة، فهو سيعمل فقط على شبكات عمل workstations قليلة في نفس الوقت، ويمكن أن يصل حتى 10 على الأكثر. وليس هناك حاجة لتشغيل أكثر من نسخة واحدة على شبكة مفردة، بما أن كل نسخة تتضمن جميع ميزات التطبيق الممكنة. واحدة من الميزات الجامدة المضمنة مع الفيجوال بيسك هي إمكانية إنشاء "تطبيق ذو حالة مفردة single-instance application"، والتي تجبر على إدارة تشغيل واحد في نفس الوقت على كل محطة (شبكة عمل). على الرغم من أنه كان بإمكانك إنشاء مثل هذه التطبيقات من قبل، ولكن الآن متاحة من خلال نقرة ماوس وحيدة.

لجعل مشروع المكتبة تطبيق وحيد الحالة single-instance application، اعرض صفحات خاصيات المشروع the project properties، ومنه لوحة التطبيق Application، ومن ثم اختر الحقل "اجعل التطبيق حالة مفردة Make single instance application". عندما يحاول المستخدم تشغيل نسخة ثانية، فإن الدوت نت سترفض تنفيذ المطلوب بدلاً من ذلك، ستطلق حدث التطبيق StartupNextInstance. وأية معالجة خاصة ترغب في إتقانها على التشغيل الثاني للحالة سيتم عملها في هذا المعالج. مثل معالج حدث الانطلاق Startup، يظهر معالج الحدث StartupNextInstance في الملف ApplicationEvents.vb.

من أجل مشروع المكتبة، الشيء الوحيد الحقيقي الذي نريد عمله حقاً عندما يحاول المستخدم تشغيل حالة ثانية هو التأكيد على أن التطبيق معروض من قبل وهو مركزي. بحيث يستطيع المستخدم عرضه بسهولة. افتح الملف ApplicationEvents.vb وأضف معالج الحدث StartupNextInstance.

```
Private Sub MyApplication_StartupNextInstance(ByVal sender As Object, ByVal e As Microsoft.VisualBasic.ApplicationServices.StartupNextInstanceEventArgs) Handles Me.StartupNextInstance
```

```
    ' ----- Force the main form to the front.
    My.Application.MainForm.Activate()
End Sub
```

Classes and Inheritance

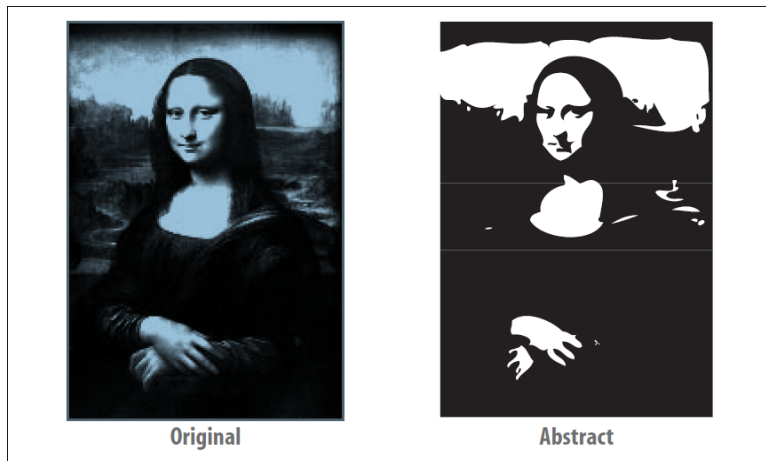
(OOP) basic foundation (object-oriented programming (OOP)) System.Object

Object-Oriented Programming Concepts

bits CPU
 MS-DOS CPU . bytes
 EXE () . file
 ordinary procedural languages assembly language
 OOP . noun noun
object object-oriented programming
 () . noun noun
 real-world objects DVD
) " "
 ()
 . *polymorphism* *inheritance* *encapsulation* *abstraction* :
 () " *object* "
 . *implementation* *Class*
)
 ()

The Object

Abstraction



software object

- .1
- .2
- .3
- .4
- .5

calculations processing rules . *properties*
 . methods () properties
 . *members* of the object ()

Encapsulation

encapsulated

" *interfaces* "

mouse display keyboard ()
 . power button microphone jack speakers ports 1394 USB power connector
blackboxes
 USB printer driver
 inexplicable

Inheritance

. *base class* original class
derived

override

() *animals*

gestation *mammal*
 name *avian mammal* *avian-derived class*
 () *animal avian () animal leg count species*
animal generic instance . animal mammal

private

Public

protected

).
 assembly *friend* *friend*
 (. *Public*

()

Polymorphism

" type " flag field : flag C . non-OOP language
 mammal-specific fields " mammal " -
) " many forms " *Polymorphism*
 generic *animal* *mammal* (overridden
 ()
 derived instance . base instance
animal *avian* *mammal* *animal*
subtyping *polymorphism*
 () Overloading " Overloading "
 ()
 (()
)
 ()

Interfaces and Implementation

() ()
 ()
 () *interface* ()
 ()
implementation
implementation
 ()
 (*instance*) object
 distinct instance

OOP in Visual Basic and .NET

Classes

structures *classes*
 : Class
 Class []
 End Class
 ()
) Namespace (assembly top-level namespace
 ()
 Namespace GoodGuys
 Class Superhero
 End Class
 End Namespace
 :
 WindowsApplication1
 WindowsApplication1.GoodGuys.Superhero

```

Partial Class Superhero
    Protected Private Public :
        Protected Friend Friend
        .NET Framework Class Libraries (FCLs)
        " Class "
    End Class Class
    
```

Class Members

```

Class Superhero
    members
    sprinkle
    Partial
    End Class
    
```

Variable fields

top-level members

access modifiers

```

Class Superhero
    Public Name As String
    Protected TrueIdentity As String
    End Class
    
```

Constant fields

```

.LINQ
properties members
local Const
literals
Private Const BaseStrengthFactor As Integer = 1
    
```

Enumerations

```

Private Enum GeneralSuperPower
    Flight
    Strength
    Speed
    VisionRelated
    HearingRelated
    WaterRelated
    TemperatureRelated
    ToolsAndGadgets
    GreatCostume
    End Enum
    
```

Sub methods

```

Sub logic code functions subs
    properties
    Public Sub DemonstrateMainPower(ByVal strengthFactor As Integer)
        ' ----- Logic code appears here.
    End Sub
    
```

()
data

DemonstrateMainPower

strengthFactor

.arguments

.Exit Sub

Return

Function methods

. return value

sub

Function

As

```
Public Function GetSecretIdentity(ByVal secretPassword As String) As String
    If (secretPassword = "Krypton") Then
        ' ----- I created a class field named
        ' TrueIdentity earlier.
        Return TrueIdentity
    Else
        GetSecretIdentity = "FORGET IT BAD GUY"
    End If
End Function
```

return value assignment

assignment-to-function-name style

Exit Function

Properties

read-

read-write

. methods

fields

Properties

:" Set " "Get " accessors

write-only

only

```
Public WriteOnly Property SecretIdentity() As String
    Set(ByVal value As String)
        TrueIdentity = value
    End Set
End Property
```

Delegates

() method values arguments

```
Public Delegate Sub GenericPowerCall(ByVal strengthFactor As Integer)
```

Events

syntax

(actions)

method

(signature

```
' ----- Non-delegate definition.
Public Event PerformPower(ByVal strengthFactor As Integer)
' ----- Delegate definition.
Public Event PerformPower As GenericPowerCall
```

Declares

DLL

external

DLL

Declare

```
Public Declare Function TalkToBadGuy Lib "evil.dll" (ByVal message As String) As String
```

built-

.DLL

(in

.evil.dll

" unmanaged "

Interfaces

. class templates

abstract classes

Interfaces

Nested types

" child "

()

" parent "

()

Mhm76

```
Class Superhero
  Class Superpower
  End Class
End Class
```

. don't go overboard

variety

Shared Class Members

()

variable

shared member

Shared

property

Shared

function
instance

sub

field

```
Class ClassWithSharedValue
  Public Shared TheSharedValue As Integer
End Class
'...later, in some other code...
ClassWithSharedValue.TheSharedValue = 10
```

instances

" shared "

instance members

Overloaded Members and Optional Arguments

Overloads

method

```
Class House
  Public Overloads Sub PaintHouse()
    ' ----- Use the same color(s) as before.
  End Sub
  Public Overloads Sub PaintHouse(ByVal baseColor As Color)
    ' ----- Paint the house a solid color.
  End Sub
  Public Overloads Sub PaintHouse(ByVal baseColor As Color, ByVal trimColor As Color)
    ' ----- Paint using a main and a trim color.
  End Sub
  Public Overloads Sub PaintHouse(ByVal baseColor As Color, ByVal coats As Integer)
    ' ----- Possibly paint with many coats, of paint
    ' that is, not of fabric.
  End Sub
End Class
```

PaintHouse
signature

: coats

```
Public Overloads Sub PaintHouse(ByVal whichColor As Color)
Public Overloads Sub PaintHouse(ByVal baseColor As Color, ByVal coats As Integer)
  .coats optional argument
Public Overloads Sub PaintHouse(ByVal baseColor As Color, Optional ByVal coats As Integer
= 1)
End Sub
```

Optional

coats

default value

. = 1

1

coats

Inheritance

Mhm76

base (ancestor-descendant) -()
 .Inherits derived class class

```
Class Animal
' ----- Animal class members go here.
End Class
Class Mammal
    Inherits Animal
' ----- All members of Animal are automatically
' part of Mammal. Add additional Mammal
' features here.
End Class
```

Inherits
 Inherits
 Partial Inherits
 (1⊗)
 overrides
 (2) Overridable

```
Class Animal
    Public Overridable Sub Speak()
        MsgBox("Grrrr.")
    End Sub
End Class
Class Canine
    Inherits Animal
    Public Overrides Sub Speak()
        MsgBox("Bark.")
    End Sub
End Class
```

.Speak
 attribute
 Overridable
 Animal
 " Canine "
 . NotOverridable
 nonoverridable

```
Class Canine
    Inherits Animal
    Public NotOverridable Overrides Sub Speak()
        MsgBox("Bark.")
    End Sub
End Class
```

MustOverride

```
Class Animal
    Public MustOverride Sub DefenseTactic()
End Class
```

End DefenseTactic). MustOverride
 . deficiency .(Sub

DefenseTactic Animal

. MustInherit

" MustOverride "

```
Public MustInherit Class Animal
    Public MustOverride Sub DefenseTactic()
End Class
```

Animal-derived class (reference type) Animal ()

```
Dim zooMember As Animal
    Dim monkey As New Simian ' Simian is derived from Animal
zooMember = monkey
```

```
NotInheritable Class Animal
    End Class
```

NotInheritable
 Overrides
 NotInheritable
 noninheritable
 MustInherit
 Inherits
 NotOverridable
 .Shadows
 String
 function
 override
 Integer return code
 " " " "
 PeanutButter
 (class
 nested enumeration constant)
 .(grabs)
 PeanutButter

```
Class Food
    Public Sub PeanutButter()
    End Sub
End Class
Class Snack
    Inherits Food
    Public Shadows PeanutButter As String
    ' Hey, it's not even a "Sub"
End Class
```

Creating Instances of Classes

creating class : deriving classes : designing classes : instances New instances

```
Dim myPet As Animal = New Animal
    ' ----- Or...
Dim myPet As New Animal
    ' ----- Or...
Dim myPet As Animal
    myPet = New Animal
```

```
myPet.Name = "Fido"
```

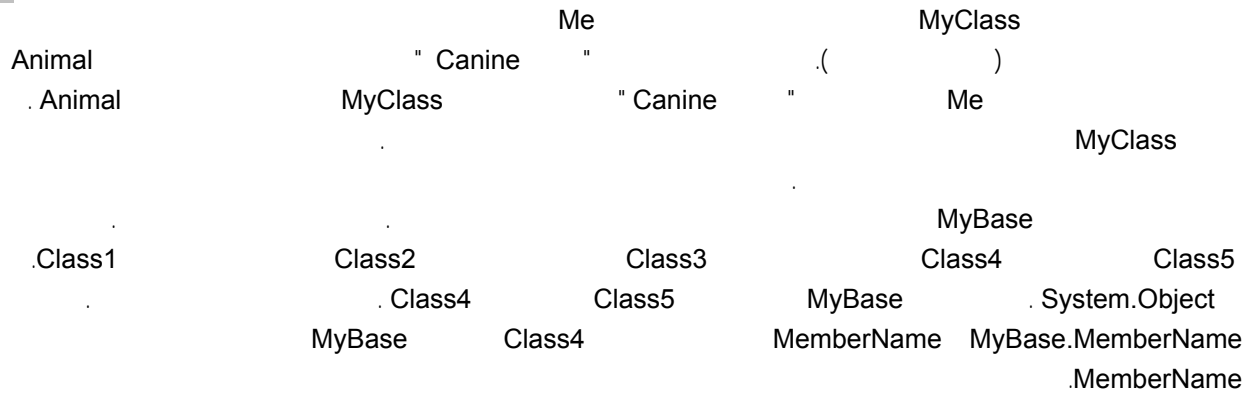
```
Dim myPet As Animal
    Dim myDog As Canine
    myDog = New Canine
    myDog.Name = "Fido"
    myPet = myDog ' Since Canine derives from Animal
    MsgBox(myPet.Name) ' Displays "Fido"
```

narrowing (On) Option Strict) CType .() conversions

```
myDog = CType(myPet, Canine)
```

(qualification) () .Me

```
Class Animal
    Public Name As String
    Public Sub DisplayName()
        ' ----- Either of these lines will work.
        MsgBox(Name)
        MsgBox(Me.Name)
    End Sub
End Class
```



```
Class Animal
    Public Overridable Sub ObtainLicense()
        ' ----- Perform Animal-specific licensing code.
    End Sub
End Class
Class Canine
    Inherits Animal
    Public Overrides Sub ObtainLicense()
        ' ----- Perform Canine-specific licensing code, then...
        MyBase.ObtainLicense() ' Calls code from Animal class
    End Sub
End Class
```

Constructors and Destructors

constructor : *destructor*

garbage collection : dictated

implicit : explicit

instance : memory space : reserving : minimal constructor-level activities : variable field

default constructor

```
Class Animal
    Public Name As String
    Public Sub New()
        ' ----- Every animal must have some name.
        Name = "John Doe of the Jungle"
    End Sub
End Class
```

String . Name Animal constructor reference types

" Nothing " " New "

argument signature

```
Class Animal
    Public Name As String
```


Mhm76

```
Public Sub New()
    ' ----- Every animal must have some name.
    Name = "John Doe of the Jungle"
End Sub
Public Sub New(ByVal startingName As String)
    ' ----- Use the caller-supplied name.
    Name = startingName
End Sub
Public Sub New(ByVal startingCode As Integer)
    ' ----- Build a name from a numeric code.
    Name = "Animal Number " & CStr(startingCode)
End Sub
End Class
```

```
MsgBox((New Animal).Name)
' Displays "John Doe of the Jungle"
MsgBox((New Animal("Fido")).Name)
' Displays "Fido"
MsgBox((New Animal(5)).Name)
' Displays "Animal Number 5"
```

System.Object

```
Class Canine
    Inherits Animal
    Public Sub New()
        MyBase.New() ' Calls Animal.New( )
        ' ----- Now add other code.
    End Sub
End Class
```

Killing

```
myDog = Nothing
```

```
myDog = New Canine
myDog.Name = "Fido"
myDog = New Canine ' Sorry Fido, you're gone
```

override

Finalize

System.Object

Protected

Finalize

```
Class Animal
    Protected Overrides Sub Finalize()
        ' ----- Cleanup code goes here. Be sure to call the
        ' base class's Finalize method.
        MyBase.Finalize()
    End Sub
End Class
```

garbage collection

()

Finalize

()

()

30

cleanup

(

)

(

).IDisposable

Mhm76

```

Class Animal
    Implements IDisposable
    Protected Overrides Sub Finalize()
        ' ----- Cleanup code goes here. Be sure to call the
        ' base class's Finalize method.
        MyBase.Finalize()
    End Sub
    Public Overloads Sub Dispose() Implements IDisposable.Dispose
        ' ----- Put cleanup code here. Also make these calls.
        MyBase.Dispose() ' Only if base class is disposable.
        System.GC.SuppressFinalize(Me)
    End Sub
End Class

```

" Finalize " garbage collector SuppressFinalize
 Dispose
 . Using .()

```

Using myPet As New Animal
    ' ----- Code here uses myPet.
End Using
' ----- At this point, myPet is destroyed, and Dispose is
' called automatically by the End Using statement.

```

Interfaces

MustInherit MustOverride
 MustOverride
 abstract definitions MustOverride Interfaces .interface
 MustOverride .implementation
 () agreement contract .(class
 (By convention) . interface

```

Interface IBuilding
    Function FloorArea() As Double
    Sub AlterExterior()
End Interface

```

.access modifiers
 structures classes interfaces events properties
 (Inherits) interfaces interfaces
 Implements

```

Class House
    Implements IBuilding
    Public Function FloorArea() As Double Implements IBuilding.FloorArea
        ' ----- Add implementation here.
    End Function
    Public Sub PaintHouse() Implements IBuilding.AlterExterior
        ' ----- Add implementation here.
    End Sub
End Class

```

)
 AlterExterior FloorArea .(
 .PaintHouse

```

Dim someHouse As New House
    Dim someBuilding As IBuilding
    someBuilding = someHouse
    someBuilding.AlterExterior() ' Calls someHouse.PaintHouse( )

```

```
Class House
    Implements IBuilding, IDisposable
```

```
Public Sub PaintHouse() Implements IBuilding.AlterExterior, IContractor.DoWork
End Sub
```

generic

logic

Superhero House Animal ()

IDisposable

Modules and Structures

modules

structures

Shared

" Modules

"

()

```
Friend Module GenericDataAndCode
    ' ----- Application-global constant.
    Public Const AllDigits As String = "0123456789"
    ' ----- Application-global function.
    Public Function GetEmbeddedDigits(ByVal sourceString As String) As String
    End Function
End Module
```

(qualification)

(Private

modules

Structures

(System.Object

) System.ValueType

syntax

Integer

constructor

()

destructors

Partial Methods

code generators

partial Form class

2008

Partial methods

()

optional

implemented half

(2) unimplemented half

(1⊗)

Partial

```
Partial Private Sub ImplementIfYouDare()
End Sub
```

. Private functions sub

. ByRef

ByVal

" Partial " ()

implemented half

.jaws

```
Private Sub ImplementIfYouDare()
    MsgBox("I did it, so there.")
End Sub
```

auto-generated side

Animal

```
Partial Class Animal
    Public Sub Move()
        ' ----- Interesting movement code, then...
        MoveSideEffects()
    End Sub
    Partial Private Sub MoveSideEffects()
    End Sub
End Class
```

side effects

moves

```
Partial Class Animal
    Public Sub Move()
        ' ----- Interesting movement code, then...
    End Sub
End Class
```

Move

MoveSideEffects

Related Issues()

MsgBox

" MsgBox

MsgBox

carryover

" MsgBox

" Microsoft.VisualBasic

. MsgBoxResult

```
Public Function MsgBox(ByVal Prompt As Object, Optional ByVal Buttons As MsgBoxStyle =
MsgBoxStyle.OkOnly, Optional ByVal Title As Object = Nothing) As MsgBoxResult
```

icons

buttons

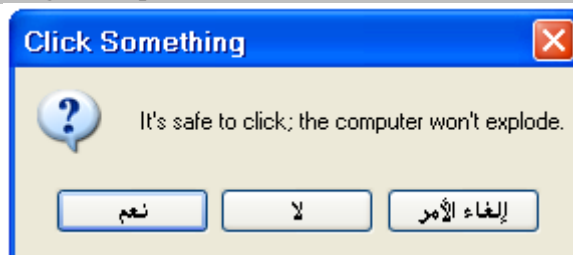
Buttons

string

Prompt

Title

```
Dim result As MsgBoxResult = MsgBox("It's safe to click; the computer won't explode.",
MsgBoxStyle.YesNoCancel Or MsgBoxStyle.Question, "Click Something")
```



Microsoft.VisualBasic

MsgBox

.MessageBox.Show

C#

MsgBox

Using DoEvents

```

        defers
        "(( ) )
        .patient
        ( Refresh ) ( )
        ( ) " " ( )
        .DoEvents
        My DoEvents (.screen update )" paint "
        My.Application.DoEvents()
        DoEvents
    
```

ParamArray Arguments (ParamArray)

optional arguments

.ParamArray

parameter array argument

ParamArray

```

Public Function CalculateAverage(ByVal ParamArray sourceData() As Decimal) As Decimal
    ' ----- Calculate the average for a set of numbers.
    Dim singleValue As Decimal
    Dim runningTotal As Decimal = 0@
    If (sourceData.GetLength(0) = 0) Then
        Return 0@
    Else
        For Each singleValue In sourceData
            runningTotal += singleValue
        Next singleValue
        Return runningTotal / sourceData.GetLength(0)
    End If
End Function
    
```

.decimal

CalculateAverage

```
MsgBox(CalculateAverage(1, 2, 3, 4, 5)) ' Displays: 3
```

ComboBox

ListBox

(1⊗)

(2)

Supporting List and Combo Boxes

List :

ComboBox

ListBox

6

```

( -32 ) ItemData ( )
ItemData
    
```

```

cboMonth.AddItem("January")
cboMonth.ItemData(cboMonth.NewIndex) = 1
cboMonth.AddItem("February")
cboMonth.ItemData(cboMonth.NewIndex) = 2
    
```

```

...
cboMonth.AddItem("December")
cboMonth.ItemData(cboMonth.NewIndex) = 12
ID

```

```

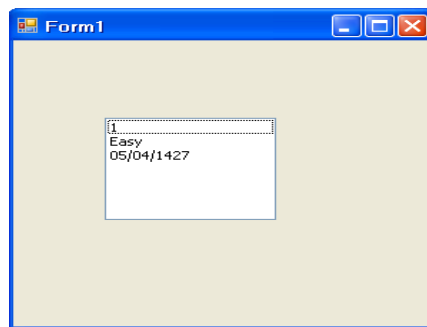
nMonth = cboMonth.ItemData(cboMonth.ListIndex)
ListBox ItemData List
: .Items : .ComboBox
System.Object ( )
( ) .collection
.list
" ToString "
ListBox
override System.Object ToString
" DisplayMember "
ListBox

```

```

Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        ListBox1.Items.Add(1)
        ListBox1.Items.Add("Easy")
        ListBox1.Items.Add(#5/3/2006#)
    End Sub
End Class

```



```

identifier ValueMember ListBox ItemData
.ValueMember Items
Items
ComboBox ListBox
ID
Add <<Project Add ListItemData.vb Class

```

```

Public Class ListItemData
End Class

```

```

DisplayMember
initialization ToString

```

```

Public Class ListItemData
    Public ItemText As String
    Public ItemData As Integer

    Public Sub New(ByVal displayText As String, ByVal itemID As Integer)

```

Mhm76

```

' ----- Initialize the record.
ItemText = displayText
ItemData = itemID
End Sub

Public Overrides Function ToString() As String
' ----- Display the basic item text.
Return ItemText
End Function

Public Overrides Function Equals(ByVal obj As Object) As Boolean
' ----- Allow IndexOf() and Contains() searches by ItemData.
If (TypeOf obj Is Integer) Then
Return CBool(CInt(obj) = ItemData)
Else
Return MyBase.Equals(obj)
End If
End Function

End Class
    
```

```

ListBox
ListBox1.Items.Add(New ListItemData("Item Text", 25))
( )
Equals
IndexOf
Items
True
Equals
ListBox
ListBox
ListItemData
Integer value
    
```

```
Dim itemPosition As Integer = SomeListBox.Items.IndexOf(5)
```

Editing Code Tables

ComboBox

(Code)

collections

()

CodeCopyStatus

Description	Type	Field
Primary key; automatically assigned. Required. مفتاح رئيسي، ألي الإسناد. مطلوب .	Long - Auto	ID
Name of this status entry. Required. اسم مدخلة الحالة. مطلوب .	Text(50)	FullName

ID

derived versions

()

)" detail

" (

)" summary

" :

"

.(

." detail

"

."summary

The Generic Detail Form

BaseCodeForm.vb

(New Windows Form

<< Project)

Setting	Property
BaseCodeForm	(Name)
False	ControlBox
FixedDialog	FormBorderStyle
False	ShowInTaskbar
406, 173	Size
CenterScreen	StartPosition
Code Form	Text

.(Code << View)

.MustInherit

instantiation

```
Public MustInherit Class BaseCodeForm
```

End Class

MustOverride

```
Public Overridable Function AddRecord() As Integer
    ' ----- Prompt to add a new record. Return the ID
    '         when added, or -1 if cancelled.
    Return -1
End Function
Public Overridable Function DeleteRecord(ByVal recordID As Integer) As Boolean
    ' ----- Prompt the user to delete a record.
    '         Return True on delete.
    Return False
End Function
Public Overridable Function EditRecord(ByVal recordID As Integer) As Integer
    ' ----- Prompt the user to edit the record. Return the
    '         record's ID if saved, or -1 on cancel.
    Return -1
End Function
```

ستأخذ فئة "التفاصيل" التفصيلية و"مسؤولية" أداة صندوق القائمة ListBox على فورم "الملخص" summary "ببنودها. نحتاج لطريقتين من أجل معالجة هذا: واحدة تعمل على إضافة جميع البنود، وأخرى تحديث بند وحيد. ستكون الفئة المشتقة ضرورية لعدم هذه الميزات.

```
' ----- Fill a ListBox control with existing records.
Public MustOverride Sub FillListWithRecords( ByRef destList As ListBox, ByRef exceededMatches
As Boolean)
    ' ----- Return the formatted name of a single record.
Public MustOverride Function FormatRecordName(ByVal recordID As Integer) As String
    ' ----- Return a description of this editor.
Public MustOverride Function GetEditDescription() As String
    ' ----- Return the title-bar text for this editor.
Public MustOverride Function GetEditTitle() As String
```

على الرغم من أن معظم الجداول ستوفر قائمة قصيرة بالأكواد المرتبة أبجدياً، بعض الجداول ستضمن عدد كبير (ربما الآلاف) من الأكواد. ستدعم فورم "الملخص" summary " طريقة بحث، لإيجاد كود موجود بسرعة. بما أن النماذج المشتقة الخاصة ستستخدم هذه الميزة، فلن نضمن MustOverride.

```
Public Overridable Sub SearchForRecord(ByRef destList As ListBox, ByRef exceededMatches As Boolean)
    ' ----- Prompt the user to search for a record.
    Return
End Sub
```

أخيراً، ستشير فورم التفاصيل إلى آياً من الميزات المتاحة التي يمكن استخدامها من فورم "الملخص". وستستدعي نموذج الملخص كل من الدوال التالية ومن ثم تمكين أو لا تمكين ميزات حسب الحاجة.

```
Public Overridable Function CanUserAdd() As Boolean
    ' ----- Check the security of current user to see
    '         if adding is allowed.
    Return False
End Function
Public Overridable Function CanUserEdit() As Boolean
    ' ----- Check the security of the user to see
    '         if editing is allowed.
    Return False
End Function
Public Overridable Function CanUserDelete() As Boolean
    ' ----- Check the security of the user to see
    '         if deleting is allowed.
    Return False
End Function
Public Overridable Function UsesSearch() As Boolean
    ' ----- Does this editor support searching?
    Return False
End Function
```

هذا كل ما تحتاجه فورم التفاصيل الشاملة generic detail form. فيما بعد في هذا الكتاب، سنعمل على إنشاء إصدارات مشتقة لكل جدول كود code tables.

نموذج "الملخص" الشاملة. The Generic Summary Form.

()
ListEditRecords.vb

()

ListBox

():



```
Public Class ListEditRecordsvb
    Private DetailEditor As Library.BaseCodeForm

    Public Sub ManageRecords(ByRef UseDetail As Library.BaseCodeForm)
        ' ----- Set up the form for use with this code set.
        Dim exceededMatches As Boolean

        DetailEditor = UseDetail
        RecordsTitle.Text = DetailEditor.GetEditTitle()
        RecordsInfo.Text = DetailEditor.GetEditDescription()
        Me.Text = DetailEditor.GetEditTitle()
        ActAdd.Visible = DetailEditor.CanUserAdd()
        ActEdit.Visible = DetailEditor.CanUserEdit()
        ActDelete.Visible = DetailEditor.CanUserDelete()
        ActLookup.Visible = DetailEditor.UsesSearch()
        DetailEditor.FillListWithRecords(RecordsList, exceededMatches)
        RefreshItemCount(exceededMatches)
        Me.ShowDialog()
    End Sub
    Private Sub RefreshButtons()
        ' ----- Update the buttons as needed.
        ActEdit.Enabled = CBool(RecordsList.SelectedIndex <> -1)
        ActDelete.Enabled = CBool(RecordsList.SelectedIndex <> -1)
    End Sub
    Private Sub RefreshItemCount(ByVal exceededMatches As Boolean)
        If (exceededMatches) Then
            ResultCount.Text = "Results limited to first " & _
                RecordsList.Items.Count & " items"
        ElseIf (RecordsList.Items.Count = 1) Then
            ResultCount.Text = "1 record displayed"
        Else
            ResultCount.Text = RecordsList.Items.Count & " records displayed"
        End If
    End Sub
    Private Sub ActClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        ActClose.Click
    End Sub
End Class
```

Mhm76

```

' ----- Close the form.
DetailEditor.Close()
DetailEditor = Nothing
Me.Close()
End Sub
Private Sub ActAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ActAdd.Click
' ----- Let the user add a record.
Dim newID As Integer
Dim newPosition As Integer

' ----- Prompt the user.
newID = DetailEditor.AddRecord()
If (newID = -1) Then Return

' ----- Add this record to the list.
newPosition = RecordsList.Items.Add((New Library.ListItemData( _
    DetailEditor.FormatRecordName(newID), newID)))
RecordsList.SelectedIndex = newPosition
RefreshButtons()
RefreshItemCount(False)
End Sub
Private Sub ActEdit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ActEdit.Click
' ----- Let the user edit a record.
Dim newID As Integer
Dim listItem As Library.ListItemData

' ----- Retrieve the item to edit.
If (RecordsList.SelectedIndex = -1) Then Return
listItem = CType(RecordsList.SelectedItem, Library.ListItemData)

' ----- Prompt the user.
newID = DetailEditor.EditRecord(listItem.ItemData)
If (newID = -1) Then Return

' ----- Modify the text of this item.
listItem.ItemText = DetailEditor.FormatRecordName(newID)
RecordsList.Items(RecordsList.SelectedIndex) = listItem
RefreshItemCount(False)
End Sub
Private Sub ActDelete_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ActDelete.Click
' ----- Let the user edit a record.
Dim listItem As Library.ListItemData

' ----- Retrieve the item to edit.
If (RecordsList.SelectedIndex = -1) Then Return
listItem = CType(RecordsList.SelectedItem, Library.ListItemData)

' ----- Prompt the user.
If (DetailEditor.DeleteRecord(listItem.ItemData) = False) Then Return

' ----- Remove this item from the display.
RecordsList.Items.RemoveAt(RecordsList.SelectedIndex)
RefreshButtons()
RefreshItemCount(False)
End Sub
Private Sub ActLookup_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ActLookup.Click
' ----- Prompt for user limitation of the displayed items.
Dim exceededMatches As Boolean

DetailEditor.SearchForRecord(RecordsList, exceededMatches)
RefreshButtons()
RefreshItemCount(exceededMatches)
End Sub
Private Sub RecordsList_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
RecordsList.Click

```

```

' ----- Enable controls as needed.
RefreshButtons()
End Sub
Private Sub RecordsList_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles RecordsList.DoubleClick
' ----- Same as the edit button.
If (ActEdit.Visible = True) Then ActEdit.PerformClick()
End Sub
Private Sub RecordsList_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles RecordsList.SelectedIndexChanged
' ----- Enable controls as needed.
RefreshButtons()
End Sub
End Class

(
)
Private DetailEditor As Library.BaseCodeForm
. ManageRecords .BaseCodeForm
Public Sub ManageRecords(ByRef UseDetail As Library.BaseCodeForm)
. BaseCodeForm ManageRecords
" summary "
" ActAdd " " Visible " Boolean CanUserAdd " detail "
. ListBox " " FillListWithRecords
" " Me.ShowDialog
.DetailEditor " " " "
.AddRecord " "
Private Sub ActAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ActAdd.Click
' ----- Let the user add a record.
Dim newID As Integer
Dim newPosition As Integer
' ----- Prompt the user.
newID = DetailEditor.AddRecord()
If (newID = -1) Then Return
' ----- Add this record to the list.
newPosition = RecordsList.Items.Add(New Library.ListItemData(
DetailEditor.FormatRecordName(newID), newID))
RecordsList.SelectedIndex = newPosition
RefreshButtons()
RefreshItemCount(False)
End Sub
(
)

```

البرمجة الوظيفية Functional Programming

في هذا الفصل سنغطي موضوعي برمجة رئيسين في الفيجوال بيسك: تعابير الوسيط (Lambda) ومعالجة الأخطاء *error handling*. وكلاهما غامض، الأول بسبب استخدامه الحرف الإغريقي في اسمه والآخر لأنه بالإضافة لكونه بالإغريقي فهو صعب بالنسبة لكل المبرمجين يجب أن يتم عمل تعابير الوسيط لأمدا بمبدأ شامل للبرمجة الوظيفية *functional programming*، الفكرة هي أن كل مهمة معالجة يمكن أن يتم التعبير عنها كدالة (وظيفة) وتلك الدالة يمكن أن يتم تمريرها طوعاً أو كرهاً *willy-nilly* ضمن الكود المصدري.

الفيجوال بيسك ليست لغة برمجة وظيفية (برمجة بالدوال) حقيقية، ولكن تقديم تعابير الوسيط لأمدا *lambda* في الفيجوال بيسك 2008 يجلب للغة بعض هذه الطرق الوظيفية والوسائل.

تعابير الوسيط لأمدا Lambda Expressions

يتم تسمية تعابير الوسيط لأمدا لحسابات *lambda calculus* (أو λ -calculus)، نظام رياضي تم تصميمه في 1930s بواسطة ألونزو كارك *Alonzo Church*، على الرغم من أن عمله كان في البداية نظرياً، فإنه قاد إلى ميزات وتراكيب تقييد معظم لغات البرمجة اليوم. على وجه التخصيص، توفر حسابات لأمدا الأساس المنطقي *rational* للدوال الفيجوال بيسك *functions*، والمعاملات *arguments*، والقيم المعادة *return values* التي درسناها سابقاً. لذلك ما الفائدة من إضافة ميزة جديدة للفيجوال بيسك وتسميتها *lambda* بما أن أشياء الوسيط لأمدا موجودة في اللغة؟ سؤال عظيم، ولكن لاجواب. تتيح لك تعابير لأمدا من تعريف كائن ما يحتوي على دالة كاملة. على الرغم من أن هذا نوعاً ما شيء جديد في الفيجوال بيسك. فميزة مشابهة قد كانت موجودة في لغة البيسك *BASIC* ولفترة طويلة. وقد وجدت صيغة يدوية قديمة من لغة البرمجة الأقدم التي استخدمتها *BASIC PLUS*، فكانت توفر العبارة *DEF*، التي تتيح لك تعريف دوال بسيطة. إليك بعض مثال الكود من تلك والتي تطبع قائمة من خمس تربيعات

```
100: DEF (SQR(X) = X * X)
110: For I = 1 To 5
120:   Print(I, SQR(I))
130: Next I
140: End
```

يظهر تعريف الوظيفة (الدالة) في السطر 100، تعود بمربع أي معامل نسبي ممر لها. وتستخدمه في النصف الثاني في السطر 120، مولدة المخرجات التالية:

```
1 1
2 4
3 9
4 16
5 25
```

تعمل تعبيرات لأمدا في الفيجوال بيسك بطريقة مشابهة، تتيح لك تعريف متغير كدالة بسيطة. إليك مكافئ الفيجوال بيسك من أجل الكود السابق:

```
Dim sqr As Func(Of Integer, Integer) = Function(x As Integer) x * x
For i As Integer = 1 To 5
MsgBox(i & vbTab & sqr(i))
Next i
```

تعبير لأمدا الفعلي في السطر الثاني:

```
Function(x As Integer) x * x
```

تبدأ تعابير لأمدا بالكلمة المحجوزة *Function*، متبوعة بقائمة المعاملات النسبية الممررة داخلها *passed-in* ضمن أقواس. بعد ذلك يأتي تعريف الدالة نفسها، يستخدم تعبير ما *x * x* المعاملات الممررة داخلياً لتوليد نتيجة ما نهائية. في هذه الحالة النتيجة هي قيمة *x* مضروبة بنفسها. شيء ما لن تراه في تعبير لأمدا وهو عبارة *Return*. عوضاً عنه، تبدو القيمة الراجعة واقعة خارج نطاق التعبير بطبيعة الحال. وهذا لما تحتاج نوع ما من متغير لحفظ التعريف *definition* والراجع *return* والنتيجة *result* في تركيب مشابه لدالة.

```
Dim sqr As Func(Of Integer, Integer)
```

يتم تعريف متغيرات تعبير لأمدا باستخدام الكلمة المحجوزة *Func*. تطابق قائمة نوع بيانات المعامل النسبي قائمة المعاملات النسبية لتعبير لأمدا الفعلي، ولكن مع نوع بيانات إضافي ملقى في النهاية *Integer* والذي يمثل نوع بيانات القيمة العائدة. إليك تعبير لأمدا الذي يتفحص فيما إذا يعمل معامل نسبي عددي صحيح *Integer* على إرجاع أو عدم إرجاع نتيجة منطقية *Boolean*.

```
Public Sub TestNumber()
Dim IsEven As Func(Of Integer, Boolean) = Function(x As Integer) (x Mod 2) = 0
MsgBox("Is 5 Even? " & IsEven(5))
End Sub
```

يعرض هذا الكود رسالة تقول "هل يقبل 5 القسمة على 2؟ خطأ" خلف المشهد، تولد الفيجوال بيسك دالة فعلية، وتربطها بالمتغير الذي يستخدم التفويض (التفويض كما تذكر، هو أسلوب لتعين طريقة ما وبشكل عام من خلال متغير متمايز) الكود التالي هو حقيقة ما يولده المترجم من أجل عينة الكود السابق.

```
Private Function HiddenFunction1(ByVal x As Integer) As Boolean
Return (x Mod 2) = 0
End Function
Private Delegate Function HiddenDelegate1(ByVal x As Integer) As Boolean
Public Sub TestNumber()
Dim IsEven As HiddenDelegate1 = AddressOf HiddenFunction1
MsgBox("Is 5 Even? " & IsEven(5))
End Sub
```

في هذا الكود، تعبير لأمدا والمتغير *IsEven* المتعلق به قد تم تبديلهما بدالة حقيقية (*HiddenFunction1*) وتفويض وسيط (*HiddenDelegate1*). على الرغم من أن الوسائط لأمدا جديدة في الفيجوال بيسك 2008، فإن هذا النوع من التخصص الوظيفي المكافئ قد أصبح ممكناً منذ نسخة الفيجوال بيسك الأولى للدوت نت. توفر تعابير لأمدا أبسط تركيب عندما تعمل الدالة المرجعية التفويض *delegate-referenced function* على إرجاع النتيجة من التعبير فقط.

لقد تم إضافة تعابير لامدا للفيجوال بيسك 2008 بشكل رئيسي لدعم التخصص الوظيفي الجديد لينكو (الفصل 17). وهي مفيدة بشكل خاص عندما تحتاج لتوفير تعبير ما كقاعدة (قانون) معالجة كود آخر، وخاصة الكود المكتوب بواسطة المشارك الثالث (عامل أجنبي). وفي تطبيقاتنا الخاصة، المشارك الثالث (العامل الأجنبي) هو ميكروسوفت.

تضمين وسيطات لامدا Implying Lambdas

تعابير لامدا جيدة، ولكن من الواضح أن تخصص وظيفي مكافئ كان موجود سابقاً في اللغة. وبنفسها تعابير لامدا هي فقط تبسيط لتركيب تفويض- دالة function-delegate غير مرتب (فوضوي). ولكن عندما تضم تعابير لامدا مع ميزات استنتاج النوع *type inference* فسوف تحصل على شيء ما أفضل. ما تحصل عليه بالأنواع المستنتجة هو تعابير لامدا. وهذا ليس اسماً رومانسياً، ولكنه أداة عظيمة جديدة لنقول أنك أردت كتابة تعبير لامدا يعمل على ضرب عددين ببعضهما.

```
Dim mult As Func(Of Integer, Integer, Integer) = Function(x As Integer, y As Integer) x * y
MsgBox(mult(5, 6)) ' Displays 30
```

إن هذا نقل لقالب كبير من الكود: إنني أخبر الفيجوال بيسك كل شيء، وهي تطيعني بدون تردد. ولكن توجد أيضاً ترجمة أكثر حرية laissez faire للكود الذي يجلب استنتاج النوع ضمن التشغيل.

```
Dim mult = Function(x As Integer, y As Integer) x * y
```

هذا كود أقل بكثير. لقد تعبت من كتابة Integer مرة بعد مرة. يعمل الكود لأن الفيجوال بيسك نظر إلى ما هو مسند لـ mult وبشكل مناسب عين (حدد) نوع بياناته القوية. وفي هذه الحالة، mult هو من نوع Function(Integer, Integer) As Integer (شاهد الشكل التالي). وحتى أنه وبشكل مناسب ضمن النوع الراجع (المعاد).

```
Dim mult = Function(x As Integer, y As Integer) x * y
Dim mult As <Function(Integer, Integer) As Integer>
```

يفترض أنك قد وضعت "خيار الاستنتاج Option Infer" لـ "فعال On" في كودك المصدري، أو من خلال صفحات خصائص المشروع Project properties (وهو الافتراضي). نستطيع اختصار تعريف mult حتى لأكثر من ذلك.

```
Dim mult = Function(x, y) x * y
```

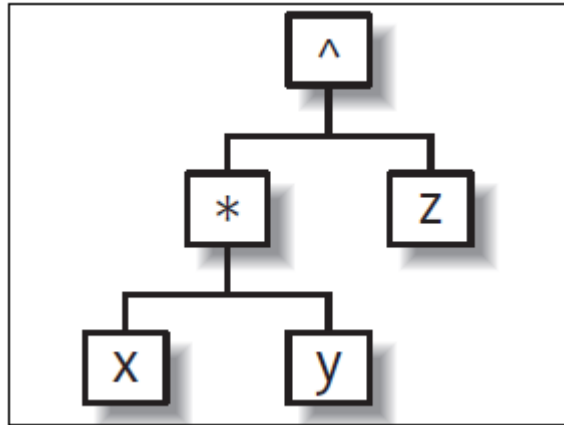
في هذا السطر سيستنتج الفيجوال بيسك نفس الدالة، ولكنه سيستخدم نوع بيانات الكائن Object على طول محل العددي الصحيح Integer. أيضاً، إذا كان لديك "خيار التدقيق Option Strict" وقد تم وضعه إلى "فعال On" (والذي يجب عليك)، فهذا السطر لن يتم ترجمته حتى تضيف شرط As المناسب.

أشجار التعبير Expression Trees

داخلياً، يعمل الفيجوال بيسك على تغيير تعبير لامدا إلى "شجرة تعبير" أي تركيب هرمي يجعل العوامل operands مصاحبة لمعاملاتها operators. خذ هذا التعبير الشبه معقد الذي يرفع تعبير مضروب إلى قوة.

```
Dim calculateIt = Function(x, y, z) (x * y) ^ z
```

يولد الفيجوال بيسك شجرة تعبير للمتغير calculateIt والذي يبدو كما في الشكل التالي.



عندما يحين وقت استخدام تعبير لامدا، يدور الفيجوال بيسك على الشجرة، يحسب القيم من المستويات الأدنى إلى الأعلى. ويتم تخزين شجرات التعبير هذه ككائنات معتمدة على فئات في فضاء أسماء System.Linq.Expressions. إذا كنت لا تحب كتابة تعابير لامدا، بإمكانك بناء شجرات تعبيرك الخاصة باستخدام هذه الكائنات. ومهما يكن، لن أخوض بهذا الموضوع.

الوسيطات لامدا المعقدة Complex Lambdas

على الرغم من أن تعابير لامدا لا يمكن أن تحتوي عبارات فيجوال بيسك مثل الحلقات For...Next، ما يزال بإمكانك بناء بعض الحسابات المعقدة نوعاً ما باستخدام معاملات قياسية. الاستدعاءات الخارجية للدوال يمكن أيضاً أن تظهر في الوسيطات لامدا. في عينة الكود التالي يُوَجَل mult عمله لدالة MultiplyIt.

```
Private Sub DoSomeMultiplication()
    Dim mult = Function(x As Integer, y As Integer) MultiplyIt(x, y) + 10
    MsgBox(mult(5, 6)) ' Displays 40
End Sub
Public Function MultiplyIt(ByVal x As Integer, ByVal y As Integer) As Integer
    Return x * y
End Function
```

هذا بسيط جداً. ولكن تحصل أشياء أكثر أهمية عندما يكون لديك تعابير لامدا تعود بتعابير لامدا أخرى. تم اختراع حسابات لامدا بشكل جزئي لرؤية كيف يمكن أن يتم تقسيم دالة معقدة في عدة دوال أكثر بساطة. وحتى القيم الحرفية (المحرفية) يمكن أن يتم تعريفها كوسيطات لامدا. إليك تعبير لامدا يعود دائماً بالقيمة 3.

```
Dim three = Function() 3
```

لقد رأيت منذ حين تعابير لامدا التي تقبل أكثر من معامل نسبي:

```
Dim mult1 = Function(x As Integer, y As Integer) x * y
```

في حسابات لامدا، يمكن أن يتم تقسيم هذا إلى دوال أبسط، حيث كل منها يتضمن فقط معامل نسبي وحيد.

```
Dim mult2 = Function(x As Integer) Function(y As Integer) x * y
```

نوع بيانات mult2 ليس نفس نوع بيانات mult1، ولكن كلاهما يولد نفس الجواب من نفس قيم x و y. عندما تستخدم mult1، فإنه يحسب الناتج لـ x و y ويعود به. وعندما تستخدم mult2، فإنه يشغل أولاً الجزء Function(x As Integer)، والذي يعود بـ لامدا آخر محسوب بتمرير قيمة x في تعريفه. إذا مررت فيه "5" كقيمة لـ x، فإن لامدا

```
Function(y As Integer) 5 * y
```

المعادة هي: و لامدا هذا يتم حسابه، والناتج من 5 و y يتم إعادته. واستدعاء mult2 في الكود أيضاً مختلف قليلاً. فلا تمرر في المعاملين النسبيين في نفس الوقت. بدل ذلك، تمرر في المعامل النسبي لأجل x، ومن ثم تمرر y لـ لامدا الأولى المعاد به.

```
MsgBox(mult2(5)(6))
```

عندما تشغيل الجزء mult2(5)، يتم تبديله بـ لامدا المعاد أولاً. ومن ثم لامدا المعاد أولاً يتم معالجته باستخدام (6) كعامله النسبي. أليس هذا بالبسيط؟. الجزء الهام الذي يجب تذكره هو أنه من الممكن بناء تعابير لامدا معقدة صعوداً من تعابير لامدا الأساسية (القاعدية). ستستخدم الفيجوال بيسك هذه الحقيقة عند توليد كود من أجل التعابير المتعلقة بلينكو LINQ. وستحدث عنها في الفصل 17، وحتى ذلك الحين، ستدير الفيجوال بيسك الكثير من تعابير لـ لامدا التي تركز على لينكو LINQ من أجلك خلف المشهد (غير مرئية).

رفع (ترقية) المتغير Variable Lifting

على الرغم من أنك تستطيع تمرير معاملات نسبية ضمن تعبير لـ لامدا، من المحتمل أيضاً أن تستخدم متغيرات أخرى ضمن مجال تعبير لـ لامدا.

```
Private Sub NameMyChild()
    Dim nameLogic = GetChildNamingLogic()
    MsgBox(nameLogic("John")) ' Displays: Johnson
End Sub
Private Function GetChildNamingLogic() As Func(Of String, String)
    Dim nameSuffix As String = "son"
    Dim newLogic = Function(baseName As String) baseName & nameSuffix
    Return newLogic
End Function
```

تعود الدالة GetChildNamingLogic بتعبير لـ لامدا. وتعبير لـ لامدا ذلك يتم استخدامه في الطريقة NameMyChild بتمرير John كعامل نسبي لـ لامدا. وهو يعمل، ولكن السؤال كيف. المشكلة هي nameSuffix، المستخدم في منطق تعبير لـ لامدا، وهو متغير محلي ضمن الطريقة GetChildNamingLogic. جميع المتغيرات المحلية يتم تدميرها عندما تخرج الطريقة. في الوقت الذي يتم فيه استدعاء الدالة MsgBox، سيكون nameSuffix قد انتهى منذ وقت. ولكن ما يزال الكود يعمل وكأن nameSuffix مستمر. لجعل هذا الكود يعمل، تستخدم الفيجوال بيسك ميزة جديدة تدعى رفع المتغير variable lifting. رؤية nameSuffix سيستم الوصول له من خارج نطاق (مدى) GetChildNamingLogic. تعيد الفيجوال بيسك كتابة كودك المصدري، تعدل متغير محلي إلى متغير له مدى أوسع wider scope. في النسخة الجديدة من الكود المصدري، تضيف الفيجوال بيسك فئة الإغلاق (النهاية closure class): الفئة المولدة بشكل ديناميكي والتي تحتوي على كل من تعبير لـ لامدا والمتغيرات المحلية المستخدمة بواسطة التعبير. عندما تضمهما مع بعضهما البعض، فأى كود يتمكن من الوصول لتعبير لـ لامدا ستكون له أيضاً إمكانية الوصول للمتغير المحلي.

```
Private Sub NameMyChild()
    Dim nameLogic = GetChildNamingLogic()
    MsgBox(nameLogic("John")) ' Displays: Johnson
End Sub
Public Class GeneratedClosureClass
    Public nameSuffix As String = "son"
    Public newLogic As Func(Of String, String) = Function(baseName As String) baseName & Me.nameSuffix
End Class
Private Function GetChildNamingLogic() As Func(Of String, String)
    Dim localClosure As New GeneratedClosureClass
    localClosure.nameSuffix = "son"
    Return localClosure.newLogic
End Function
```

الكود الحقيقي المولد بواسطة الفيجوال بيسك هو أكثر تعقيداً من هذا، وسيضمن كل كود التحويل تفويض-دالة الذي كتبت حوله سابقاً. ولكن هذه هي الفكرة الأساسية. فئات الإغلاق ورفع (ترقية) المتغير هي ميزات أساسية لتعابير لـ لامدا بما أنك لا تعرف أين مكان تعابير لـ لامدا على الإطلاق.

بوادئ الكائنات Object Initializers

لبدء، إن خاصيات الكائن object properties غير مداراة بواسطة المشيدات، تحتاج لإسناد هذه الخاصيات بشكل منفصل بعد إنشاء حالة فئة مباشرة.

```
Dim newHire As New Employee
    newHire.Name = "John Doe"
    newHire.HireDate = #2/27/2008#
    newHire.Salary = 50000@
```

توفر العبارة With تركيب أفضل نوعاً ما.

```
Dim newHire As New Employee
    With newHire
        .Name = "John Doe"
        .HireDate = #2/27/2008#
```

```
.Salary = 50000@
```

```
End With
```

تم في الفيجوال بيسك 2008 تضمين بناء syntax جديد يتيح لك دمج التصريح (بالكلمة المحجوزة New) وإسناد الأعضاء يتضمن البناء تنوع جديد لعبارة With.

```
Dim newHire As New Employee With {.Name = "John Doe", .HireDate = #2/27/2008#, .Salary = 50000@}
```

معالجة الأخطاء في الفيجوال بيسك Error Handling in Visual Basic

تصحيح ومعالجة الأخطاء هما اثنان من معظم نشاطات البرمجة الأساسية التي تقوم بها في أي وقت. حتى في التطبيق الخالي التصحيح، يوجد دائماً سبب للاعتقاد بأن المستخدم سيفقد أشياء غاية في الأهمية، عملك أن تكون حارس على بيانات المستخدم عند إدارتها بواسطة المستخدم، ولحفظها آمنة، حتى من تهون المستخدم نفسه، وحتى من قبل كودك المصدري نفسه. إن أكثر من 50% من الكود يكون محجوز لمعالجة الأخطاء، البيانات السيئة، استثناءات النظام، والإخفاقات، بشكل مركزي، كل هذا الكود الإضافي يبطل التطبيق ويضيف الكثير من التكاليف غير المباشرة لما تم تسميته "البloatware".

جوهر الأخطاء في الفيجوال بيسك The Nature of Errors in Visual Basic

سوف نتعامل مع ثلاث تصنيفات رئيسية للأخطاء في تطبيقات الفيجوال بيسك:

أخطاء وقت الترجمة Compile-time errors

بعض الأخطاء هي واضحة (صريحة) جداً بحيث أن الفيجوال بيسك سوف ترفض ترجمة تطبيقك بشكل عام، بعض الأخطاء تنتمي لقضايا بنائية بسيطة والتي يمكن أن يتم تصحيحها بعدة ضربات مفاتيح. ولكن تستطيع أيضاً تمكين ميزات في برنامجك التي تزيد عدد الأخطاء المميزة من قبل المترجم. على سبيل المثال، إذا وضعت خيار التدقيق Option Strict لـ "فعال" On في تطبيقك أو ملف الكود المصدري، فإن تحويلات التضييق ستولد أخطاء وقت الترجمة.

```
' ----- افرض : Option Strict On
Dim bigData As Long = 5&
Dim smallData As Integer
' ----- The next line will not compile.
smallData = bigData
```

تتضمن الفيجوال أستوديو 2008 الميزات التي تساعدك في إيجاد وحل أخطاء وقت الترجمة. مثل هذه الأخطاء تم تعليمها بـ "بخط أزرق متعرج blue squiggle". بعض الأخطاء تطلب أيضاً من الفيجوال بيسك عرض خيارات تصحيح من خلال نافذة الانسداد (السياق) كما هو مبين في الشكل.

```
1 Option Strict On
2 Public Class Form1
3     Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
4
5         Dim bigData As Long = 5&
6         Dim smallData As Integer
7         ' ----- The next line will not compile.
8         smallData = bigData
9     End Sub
10 End Class
11
```

Option Strict On disallows implicit conversions from 'Long' to 'Integer'.

Replace 'bigData' with 'CInt(bigData)'.

```
Dim smallData As Integer
' ----- The next line will not compile
smallData = CInt(bigData)bigData
End Sub
End Class
```

Expand All Previews

أخطاء وقت التنفيذ Runtime errors

تحدث أخطاء وقت التنفيذ عندما يسبب جمع (تركيب) من بيانات وكود شرط غير صحيح فيما يظهر من كود على أنه صحيح (محقق). مثل هذه الأخطاء تحدث باستمرار عندما يدخل المستخدم بيانات غير صحيحة في التطبيق، ولكن كودك الخاص يمكن أن يولد أخطاء وقت التنفيذ. خذ المقطع التالي من الكود:

```
Public Function GetNumber() As Integer
' ----- Prompt the user for a number.
' Return zero if the user clicks Cancel.
Dim useAmount As String
' ----- InputBox returns a string with whatever
' the user types in.
useAmount = InputBox("Enter number.")
If (IsNumeric(useAmount) = True) Then
' ----- Convert to an integer and return it.
Return CInt(useAmount)
Else
' ----- Invalid data. Return zero.
Return 0
End If
End Function
```

يبدو هذا الكود معقول لحد ما، وفي معظم الحالات، هو كذلك. فهو يطلب من المستخدم عدد ما، ويحول أعداد صحيحة لتتسابق عددي صحيح، ويعود بالنتيجة. الدالة IsNumeric (تستأصل) أي مدخلات غير عددية غير صحيحة. استدعاء لهذه الدالة سيعود بالحقيقة بأعداد صحيحة محققة للقيم العددية المدخلة، و 0 للمدخلات غير الصحيحة.

ولكن ماذا إذا ما تم إدخال قيمة مثل "342304923940234" وبما أنها قيمة عددية محققة، فسيتم تمريرها للدالة IsNumeric ولكن بما أنها تتجاوز حجم نوع بيانات العددي الصحيح، فإنها ستولد خطأ وقت التنفيذ كما هو مبين في الشكل التالي.



بدون كود إضافي لمعالجة الأخطاء أو يتفحص حدود البيانات المحققة (الصحيحة)، فإن الروتين GetNumber سيولد خطأ وقت التنفيذ هذا، ومن ثم يؤدي إلى إجهاض كامل البرنامج.

الأخطاء المنطقية Logic errors

الأخطاء المنطقية هي الثالثة، وهي أكثر أنواع الأخطاء انتشاراً. المتسبب بها هو أنت، المبرمج، لا تستطيع لوم المستخدم على هذه الأخطاء. من قضايا سياق المعالجة إلى الحسابات غير الصحيحة، الأخطاء المنطقية هي لوح تطوير البرمجيات وللحصول على نتيجة تحتاج إلى وقت أكثر للتصحيح من كلا النوعين السابقين مع بعضهما. تم تخصيص الكثير للأخطاء المنطقية في هذا الكتاب. تستطيع أن تجبر العديد من الأخطاء المنطقية خارج نطاق كودك بإضافة اختبارات كافية للبيانات الغير صحيحة، وبشكل مناسب اختبار تطبيقك تحت شروط وظروف متنوعة.

لن يكون لديك الكثير من الصعوبة في التعامل مع أخطاء وقت الترجمة compile-time errors. فهم عام لمبادئ البرمجة في الدوت نت والفيجوال بيسك، واستخدام نظامي للأدوات المضمنة مع الفيجوال أستوديو 2008، سيساعدك في سرعة إيجاد والتخلص من هذه الأخطاء compile-time errors.

القضية الأكبر هي: ماذا تعمل بخصوص أخطاء وقت التنفيذ؟ حتى ولو تفحصت كل البيانات الممكنة وشروط المصادر الخارجية، من المحتمل أن تمنع جميع أخطاء وقت التنفيذ. لن تعلم أبداً متى ستفشل عملية الاتصال بالانترنت، أو يخطئ المستخدم في كبل الطابعة، أو خدش على دي في دي DVD سيغطي بيانات فاسدة. في أي وقت تتعامل مع المصادر الموجودة خارج كودك المصدري، فستكون لديك فرصة أكبر لحدوث أخطاء وقت التنفيذ.

يبين الشكل السابق ما تعمل الفيجوال بيسك عندما تصادف خطأ وقت التنفيذ: فهي تعرض للمستخدم حوار خطأ شامل، وتوفر فرصة لتجاهل الخطأ (من المحتمل فساد أي من البيانات الغير محفوظة) أو الخروج من البرنامج مباشرة (فقدان كامل للبيانات الغير محفوظة).

على الرغم من أن هذين الفعلين يتركان للمستخدم الكثير من السحر، ولكن لا يعطيان طابع الثقة للمستهلك في مهارتك بكتابة الكود. إن المستخدم سيلومك لأي أخطاء تنتج بواسطة تطبيقك، وحتى لو كانت المشكلة الحقيقية قد تم إزالتها تماماً من كودك.

لحسن الحظ، تتضمن الفيجوال بيسك ثلاث أدوات لمساعدتك في التعامل بالكامل مع أخطاء وقت التنفيذ، إذا حدثت وعندما تحدث هذه الأخطاء. هذه الميزات الثلاث للفيجوال بيسك: معالجة الأخطاء غير التركيبية unstructured error handling، معالجة الأخطاء التركيبية structured error handling، ومعالجة الأخطاء الغير معالجة unhandled error handling. يمكن أن يتم استخدام كل من هذه الثلاث في تطبيق الفيجوال بيسك لمنع بيانات المستخدم والمستخدم نفسه- من أخطاء غير مراده.

معالجة الأخطاء الغير تركيبية Unstructured Error Handling

لقد أصبحت معالجة الأخطاء الغير تركيبية جزء من الفيجوال البيسك منذ الظهور الأول في عام 1990. وهو سهل الاستخدام، يلتقط كل الأخطاء الممكنة في مقطع من كود، ويمكن تمكينه أو تعطيله حسب الحاجة بشكل افتراضي، لا تتضمن إجراءات الطرق والخصائص أي معالجة للأخطاء على الإطلاق، لذلك عليك إضافة كود معالج الخطأ-التركيبية والغير تركيبية لكل إجراء حيثما تحس أن هناك حاجة له.

الفكرة التي تقع خلف معالجة الأخطاء الغير تركيبية هي إلى حد ما أساسية. ببساطة تضيف سطر في كودك يقول "إذا حدث أي خطأ بأي حال، اقفز بشكل مؤقت إلى هذا المقطع الآخر من إجرائي حيث لدي كود خاص للتعامل معه. هذا" المقطع الآخر " يدعى "معالج الخطأ error handler".

```
Public Sub ErrorProneRoutine ()
    ' ----- Any code you put here before enabling the
    ' error handler should be pretty resistant to
    ' runtime errors.
    ' ----- Turn on the error handler.
On Error GoTo ErrorHandler
    ' ----- More code here with the risk of runtime errors.
    ' When all logic is complete, exit the routine.
Return
ErrorHandler:
    ' ----- When an error occurs, the code temporarily jumps
    ' down here, where you can deal with it. When you're
    ' finished, call this statement:
Resume
    ' ----- which will jump back to the code that caused
    ' the error. The "Resume" statement has a few
    ' variations available. If you don't want to go
    ' back to main code, but just want to get out of
    ' this routine as quickly as possible, call:
Return
End Sub
```


تتمتع عبارة On Error أو تعطل معالجة الخطأ في الإجراء. عندما يحدث خطأ ما، تضع الفيجوال بيسك تفاصيل ذلك الخطأ في كائن خطأ شامل global Err object. وهذا الكائن يخزن نص وصفي للخطأ (الكود العددي للخطأ (إذا كان ممكناً)) أو تفاصيل مساعدة عبر الإنترنت فيما يتعلق بالموضوع، وقيم أخطاء نوعية. وسأجدول التفاصيل بعد قليل.

تستطيع تضمين العديد من عبارات On Error في كودك كما تريد. وكل واحدة يمكن أن توجه كود منحرف إلى رقعة (عنوان) مختلف. يمكن أن يكون لديك معالج خطأ وحيد لأخطاء الشبكة، وواحد خاص بأخطاء الملفات، وواحد لأخطاء الحساب، وهكذا. أو من المحتمل أن يكون لديك معالج خطأ كبير والذي يستخدم عبارات If...Then...Else لتفحص شرط الخطأ المخزن في كائن الخطأ Err الشامل global Err object.

```
ErrorHandler:
If (Err.Number = 5) Then
' ----- Handle error-code-5 issues here.
```

تستطيع إيجاد أرقام الأخطاء من أجل الأخطاء المشتركة في مستندات الفيجوال أستوديو، ولكنها معتمدة على أعداد مكتوبة بدقة وغير قابلة للتغيير تجعل معالجة الأخطاء الغير تركيبية أقل انتشاراً في هذه الأيام مما كانت عليه قبل الدوت نت. لايهم دائماً ما الذي سبب الخطأ. أحياناً، إذا عملت على تمكين معالجة الخطأ حيث كان علي أن لا أمكنه فهذه ليست نهاية العالم، فيما لو وصل الإجراء للنهائية في مسألة تحرير الخطأ، ببساطة أقوم بالتبليغ عن تفاصيل الخطأ للمستخدم، وأتجاوز سطر التوجيه.

```
Public Sub DoSomeWork ()
On Error GoTo ErrorHandler
' ----- Logic code goes here.
Return
```

```
ErrorHandler:
MsgBox ("An error occurred in 'DoTheWork':" & Err.Description)
Resume Next
End Sub
```

يبلغ مقطع الكود هذا عن الخطأ، ومن ثم يستخدم العبارة Resume Next (تنوع للعبارة القياسية Resume) للعودة بسطر الكود مباشرة للسطر الذي يلي السطر الذي سبب الخطأ. يوجد خيار آخر يستخدم (تابع "من عنوان آخر" Resume some_other_label) والتي تعود بالتحكم لمنطقة معينة من الكود من خلال مناطق نوعية مسماة.

تعطيل معالجة الخطأ Disabling Error Handling

استخدام العبارة On Error GoTo يمكن من معالجة خطأ معين. على الرغم من أنك تستطيع استخدام عبارة On Error GoTo ثانية لإعادة توجيه الأخطاء لمعالج خطأ آخر في إجرائك، الحد الأعلى الذي يبقى خلاله معالج خطأ واحد ساري المفعول هو أي وقت فحالمًا تفعل (تمكن) معالج خطأ ما، فإنه يبقى ساري المفعول حتى نهاية الإجراء، أو تعيد توجيه الأخطاء لمعالج آخر، أو تخصص تبديل إلى غير فعال turn off لمعالجة الخطأ في الروتين (الإجراء). لياخذ هذا الأخير دوره، أصدر العبارة التالية:

```
On Error GoTo 0
```

تجاهل الأخطاء Ignoring Errors

لايعمل معالج خطأ أي شيء خاص. خذ مقطع معالجة الخطأ التالي:

```
ErrorHandler:
Resume Next
```

عندما يحدث خطأ ما، فهذا المعالج يرجع التحكم مباشرة للسطر الذي يلي السطر الذي سبب الخطأ تماماً. تتضمن الفيجوال بيسك اختصار لهذا الفعل.

```
On Error Resume Next
```

بإصدار العبارة. فإن جميع الأخطاء ستقطن في كائن الخطأ (كما يتم عمله مع جميع الأخطاء، لأمشكلة في كيفية معالجتها) ومن ثم يتجاوز السطر الذي سبب الخطأ. ولم يتم إعلام المستخدم عن الخطأ، وسيستمر في استخدام التطبيق مع تجاهل الأخطاء.

معالجة الأخطاء التركيبية Structured Error Handling

كانت معالجة الأخطاء الغير تركيبية مجرد طريقة لمعالجة الأخطاء المتاحة في الفيجوال بيسك قبل الدوت نت. وعلى الرغم من ذلك فقد كانت سهلة الاستخدام، وهي لا تفي بالمتطلبات المفردة بخصوص إعلان الإصدار 2002 للفيجوال بيسك دوت نت أنه نظام برمجة كائنية التوجه. وهكذا، عملت ميكروسوفت على إضافة معالجة الأخطاء التركيبية structured error handling للغة، وهي طريقة تستخدم كائنات قياسية للاتصال بالأخطاء، وكود معالجة الخطأ الذي يكون أكثر تكاملاً بإحكام مع الكود الذي يعرضه.

تستخدم هذه الصيغة من معالجة الأخطاء عبارة Try...Catch...Finally المتعددة الأسطر لالتقاط ومعالجة الأخطاء.

```
Try
' ----- Add error-prone code here.
Catch ex As Exception
' ----- Error-handling code here.
Finally
' ----- Cleanup code goes here.
End Try
```

الشرط (التعبير) حاول The Try Clause

تم تصميم عبارة Try لعرض مقطع أصغر من الكود. وتستطيع أيضاً وضع جميع الكود المصدري لإجراء ما ضمن مقطع Try، وهو أكثر ما يكون مشهور بوضع ضمن ذلك المقطع فقط العبارات التي من المحتمل أن تسبب أخطاء.

```
Try
```

```
My.Computer.FileSystem.RenameFile(existingFile, newName)
```

```
Catch ex As Exception
```

العبارات الآمنة يمكن أن تبقى خارج نطاق الحصة Try...End Try للعبارة Try...End Try. ما يشكل عبارة برمجية آمنة بالضبط هو الموضوع الكثير المناقشة. ولكن نوعين من العبارات هي بشكل عام غير آمنة (1) تلك العبارات التي تتفاعل مع أنظمة خارجية، مثل ملفات قرص، أو شبكة أو مصادر الأجهزة المركبة على الكمبيوتر، أو حتى المقاطع الكبيرة من الذاكرة، و(2) تلك العبارات التي يمكن أن تؤدي بمتغير ما أو تعبير لأن يتجاوز الحدود المصممة لنوع بيانات ما لذلك المتغير أو التعبير.

التعبير (الشرط) التقط The Catch Clause

يعرف الشرط Catch معالج خطأ. وكما مع معالجة الخطأ الغير تركيبية، تستطيع تضمين معالج خطأ شامل في عبارة Try، أو تستطيع تضمين معالجات متعددة لأنواع مختلفة من الأخطاء. وكل معالج يتضمن الكلمة المحجوزة Catch الخاصة به.

```
Catch ex As ErrorClass
```

المعرف ex يوفر اسم متغير ما لتفعيل كائن الخطأ الذي تستطيع استخدامه ضمن مقطع Catch. تستطيع منحه أي اسم تريده، وقد يتفاوت من شرط لأخر، ولكن ليس عليه أن يكون كذلك.

تطابق ErrorClass فئة الاستثناء exception: فئة خاصة مصممة بشكل خاص لنقل معلومات الخطأ. فئة الاستثناء الأكثر شمولية هي System.Exception، ويوجد فئات استثناء أخرى أكثر خصوصية تشق من الفئة System.Exception. بما أن العبارة Try...End Try تنفذ "معالجة موجهة بالكائنات object-oriented error processing" فإن جميع الأخطاء يجب أن يتم تخزينها ككائنات. يتضمن إطار عمل الدوت نت العديد من فئات الاستثناء المسبقة التعريف تم اشتقاقها سابقاً من الفئة System.Exception والتي تستطيع استخدامها في تطبيقك. على سبيل المثال، System.DivideByZeroException تلتقط أي خطأ ينشأ (بوضوح) عن تقسيم عدد ما على صفر 0.

```
Try
```

```
result = firstNumber / secondNumber
Catch ex As System.DivideByZeroException
    MsgBox("Divide by zero error.")
Catch ex As System.OverflowException
    MsgBox("Divide resulting in an overflow.")
Catch ex As System.Exception
    MsgBox("Some other error occurred.")
End Try
```

عندما يحدث خطأ ما، يختبر كودك الاستثناء مقابل كل تعبير Catch حتى يجد الفئة المطابقة. يتم فحص التعابير Catch بالترتيب من الأعلى إلى الأسفل، لذلك تأكد من وضع الأكثر عمومية في الأخير، وإذا وضعت في البداية System.Exception، فإن ولا تعبير من تعابير Catch الأخرى في مقطع Try سيتم إطلاقه أبداً لأن كل استثناء يتطابق مع System.Exception. إن عدد شروط Catch التي تضمنها، أو أي استثناءات exceptions تعرضها، يعود إليك. إذا تركت جميع تعابير Catch فارغة، فإنها ستصرف نوعاً ما مثل العبارة On Error Resume Next، وإذا ما حدث خطأ ما، فإن جميع العبارات المتبقية في مقطع Try سيتم تجاوزها. وسيتابع التنفيذ بالمقطع Finally، ومن ثم مع الكود الذي يلي كامل عبارة Try.

الشرط (التعبير) أخيراً The Finally Clause

يمثل الشرط Finally جزء من مقطع Try الذي يقول "اعمل هذا أو مت". إذا ما حدث خطأ ما في عبارة Try، فإن الكود في مقطع Finally سيتم تنفيذه دائماً بعد أن يتم اكتمال عبارة Catch المناسبة (ذات الصلة). إذا لم يحدث خطأ، فإن المقطع Finally سيبقى قيد المعالجة قبل مغادرة العبارة Try. إذا أصدرت العبارة Return في مكان ما في عبارة Try، كذلك سيبقى المقطع Finally قيد المعالجة قبل مغادرة الروتين (الإجراء). إذا استخدمت العبارة Exit Try للخروج من مقطع Try مبكراً، فإن المقطع Finally كذلك سيبقى قيد المعالجة. إذا، بينما يكون مقطع Try قيد المعالجة، فإن مقطع Finally سيبقى أيضاً قيد المعالجة. الشروط Finally هي اختيارية، لذلك فإنك تضمن فقط شرط Finally واحد عند الحاجة له. الوقت الوحيد الذي تحتاج فيه لشرط Finally هو عندما تزيل كل شروط Catch في عبارة Try.

الأخطاء الغير معالجة Unhandled Errors

لقد بينت لك سابقاً في هذا الفصل كيف يمكن للأخطاء الغير معالجة للبيانات الفاسدة أن تقود لانهايار التطبيقات، أو مسارات حلزونية. جميع المبرمجين الجيدين يفهمون أهمية كود معالجة الأخطاء، ويقومون بعمل جهد إضافي لتضمين إما كود معالجة الأخطاء التركيبية أو الغير تركيبية. من المحتمل أن تفكر أن إجراء ما لا يعمل شيء قد يؤدي إلى توليد أخطاء. وسأتركه بدون كود معالجة الخطأ وبالتالي أوفر بعض الوقت في كتابة الكود. هذا ما يخطر ببالك، وبوضوح وبدون أي إنذار: يظهر خطأ ما، وينهار!

عادة، جميع الأخطاء الغير معالجة تظهر تراكم الاستدعاء، تبحث عن الإجراء الذي يتضمن كود معالجة الخطأ. على سبيل المثال، خذ الكود التالي:

```
Private Sub Level1()
    On Error GoTo ErrorHandler
    Level2()
    Return
ErrorHandler:
    MsgBox("Error Handled.")
    Resume Next
End Sub
Private Sub Level2()
    Level3()
End Sub
Private Sub Level3()
    ' ----- The Err.Raise method forces an
    ' unstructured-style error.
    Err.Raise(1)
```

عندما يحدث خطأ في المستوى 13، يبحث التطبيق عن معالج الخطأ الفعال في الإجراء، ولكن لا يجد شيء. لذلك، وبشكل مباشر يخرج من المستوى 13 ويعود إلى المستوى 12، حيث يبحث مرةً أخرى عن معالج الخطأ الفعال. مثل هذا البحث، بكل أسى، سيكون عديم الفائدة ويديمي القلب، يغادر الكود المستوى 12 و ينتقل عائداً إلى المستوى 11، ويتابع بحثه على معالج حدث معقول. وهذه المرة سيجد واحد. تتفجر المعالجة مباشرة إلى مقطع معالج الخطأ ErrorHandler وتنفذ الكود في ذلك المقطع. إذا كان المستوى 11 ليس فيه معالج خطأ، ولا يوجد كود يشير إلى التراكم المتضمن على معالج خطأ، فإن المستخدم سيرى نافذة رسالة الخطأ البائسة (راجع الشكل السابق)، متبوعة بموت البرنامج المخيبة Disappointment.

لحسن الحظ، تعمل الفيجوال بيسك على دعم معالج خطأ "التقط الكل" catchall "والذي يصطاد مثل هذه الاستثناءات الغير مداراة ويتيح لك فعل شيء ما بخصوصها. وهذه الميزة تعمل فقط إذا كان لديك حقل "تمكين إطار عمل التطبيق" Enable application framework "وقد تم اختياره في تبويب التطبيق Application لصفحات خاصيات المشروع project properties. للوصول لقالب كود معالج الأخطاء الشامل، انقر زر "عرض أحداث التطبيق View Application Events" على نفس تبويب خاصيات المشروع. اختر ("أحداث تطبيقي MyApplication Events") من اسم الفئة Class Name للقائمة المنسدلة فوق نافذة الكود المصدرية، ومن ثم اختر UnhandledException من قائمة اسم الطريقة Method Name. يظهر الإجراء التالي في نافذة الكود:

```
Private Sub MyApplication_UnhandledException(ByVal sender As Object, ByVal e As
Microsoft.VisualBasic.ApplicationServices.UnhandledExceptionEventArgs) Handles Me.UnhandledException
End Sub
```

أضف كود معالجة الخطأ الشامل في هذا الروتين. يتضمن المعامل النسبي e عضو الاستثناء والذي يوفر الوصول لتفاصيل الخطأ بواسطة كائن System.Exception. إن عضو e.ExitApplication هو خاصية منطقية Boolean والتي تستطيع تعديلها إما للاستمرار أو للخروج من التطبيق. بشكل افتراضي، يتم وضعها إلى صواب، ولذلك عدلها إذا كنت تريد الحفاظ على البرنامج مشغول. حتى عندما يبقى البرنامج مشغول، فإنك ستفقد مسار الحدث النشط (الفعال) الذي أطلق الخطأ. إذا ظهر الخطأ (أتى) من النقر على زر ما من قبل المستخدم، فإن حدث نقر الزر بالكامل، وكل طرق استدعاءه، سيتم هجرها abandoned مباشرةً وسينتظر البرنامج لإدخال جديد من قبل المستخدم.

إدارة الأخطاء Managing Errors

توجد أشياء أخرى تدور حول الأخطاء يجب أن تتعرف عليها في برامج الفيجوال بيسك.

توليد الأخطاء Generating Errors

صدق أو لا تصدق، توجد حالات من المحتمل أنك تريد فيها توليد أخطاء وقت التنفيذ في كودك. في الحقيقة، العديد من أخطاء وقت التنفيذ التي تصادفها في كودك تحدث بسبب أن ميكروسوفت كتبت كود في مكتبات إطار العمل Framework Class Libraries (FCLs) التي تنتج أخطاء بشكل خاص. وهذا تصميمي. لنقول أن لديك خاصية لفئة والتي تقبل فقط قيم النسب المئوية من 0 إلى 100، ولكن كنوع بيانات عديدة صحيحة.

```
Private StoredPercent As Integer
Public Property InEffectPercent() As Integer
Get
Return StoredPercent
End Get
Set(ByVal value As Integer)
StoredPercent = value
End Set
End Property
```

قواعدياً لا يوجد شيء خطأ في هذا الكود، ولكنه لن يمنع أي شخص من وضع قيمة تخزين مئوية إما 847 أو -847 - وكلاهما خارج نطاق المجال المرغوب. تستطيع إضافة عبارة الشرط If لمحدد accessor للوصول Set لرفض البيانات المرفوضة، ولكن الخاصيات لا توفر طريقة للعودة بكود الحالات الفاشلة. والطريقة الوحيدة لإعلام الكود المستدعي عن المشكلة هي توليد استثناء.

```
Set(ByVal value As Integer)
If (value < 0) Or (value > 100) Then
Throw New ArgumentOutOfRangeException("value", value, "The allowed range is
from 0 to 100.")
Else
StoredPercent = value
End If
End Set
```

والآن، محاولة وضع قيمة للخاصية InEffectPercent خارج المجال 0-to-100 سيولد خطأ، خطأ يمكن أن يتم التقاطه بواسطة معالج الخطأ On Error أو Try...Catch. تقبل العبارة Throw كائن System.Exception أو كائن مشتق منه كعامل نسبي لها، وترسل كائن الاستثناء ذلك إلى ركام الاستدعاء في البحث عن معالج خطأ ما.

ما يشابه عبارة Throw هي الطريقة Err.Raise. وهي تتيح لك توليد أخطاء باستخدام عدد معتمد على نظام الأخطاء والمعروفة أكثر بالنسبة للفيجوال بيسك 6 والبيانات الأقدم. إني أوصي باستخدام العبارة Throw، حتى ولو كنت توظف معالجة الأخطاء الغير تركيبية في مكان آخر في كودك.

دمج طرق معالجة الأخطاء Mixing Error-Handling Methods

إنك حر في مزج طرق معالجة الأخطاء التركيبية والغير تركيبية بصورة عامة في تطبيقك، ولكن يمكن أن يستخدم إجراء أو طريقة وحيدة طريقة واحدة فقط. ويمكن أن لا تستخدم كل من On Error و Try...Catch...Finally في نفس الروتين. يمكن لروتين يستخدم On Error أن يستدعي روتين آخر يستخدم Try...Catch...Finally بدون أية مشكلة.

الآن يمكن أنك تفكر بنفسك "أحياناً أستطيع بكل سهولة أن أحدد الوقت الذي أريد استخدام معالجة الأخطاء الغير تركيبية، وأحدد الأوقات الأخرى عندما أريد باختيار مقارنة تركيبية على الأغلب." وكل هذا يبدو معقولاً جداً، ولكن دعني أحذرك سلفاً أنه توجد معالجة أخطاء مترتبة بحيث تجعلك سخريه لعقود إذا ما استخدمت عبارة

On Error في كودك. لمثل هؤلاء المبرمجين ما هو أساسي عندهم هو التوجه الكائني المجرد، وأي كود يستخدم طرق غير كائنية لتحقيق ما يمكن تحقيقه من خلال محاكاة البرمجة الكائنية التوجه OOP يجب أن يدمر. رفض عبارة On Error لمثل هذا هو من الغباء بمكان. فكما تذكر من فصول سابقة، أن كل شيء في تطبيق الدوت نت موجه بالكائنات، وبما أن جميع الكود يظهر في سياق كائن ما، فإذا كنت تستخدم معالجة الأخطاء الغير تركيبية، ماتزال تستطيع الحصول على كائن استثناء مناسب من خلال الطريقة (Err.GetException())، لذلك ليست الكائنات هي القضية الحقيقية. تقرير متى يتم استخدام معالجة الأخطاء التركيبية أو الغير تركيبية لا تختلف عن ما تم تقريره للاستخدام في السي شارب #C أو الفيچوال بيسك لكتابة تطبيقاتك. من أجل معظم التطبيقات، المفاضلة في غير محلها. فيمكن أن يكون لدى لغة بعض الميزات الخاصة esoteric بها والتي يمكن أن توجهك في اتجاه (مثل المعاملات النسبية الاختيارية للطرق في الفيچوال بيسك) ولكن 99.9 من الميزات المتبقية تكون متشابهة إلى حد ما. ونفس الشيء صحيح بالنسبة لطرق معالجة الخطأ. وفي بعض الأحيان هناك ميزة سهلة وأفضل من أخرى. على سبيل المثال، خذ الكود التالي والذي يستدعي ثلاث طرق، ولا واحدة منها تتضمن معالج خطأ خاص بها.

```
On Error Resume Next
RefreshPart1 ()
RefreshPart2 ()
RefreshPart3 ()
```

من الواضح، أني لا أبالي فيما إذا كان الخطأ يحدث في واحد من الإجراءات أو لا. إذا ما سبب خطأ ما خروج مبكر من RefreshPart1، الروتين التالي، RefreshPart2، سيبقى قابل للاستدعاء. وهكذا. غالباً ما احتاج لكود تفحص الخطأ أكثر إتقاناً من هذا. ولكن في كود قليل التأثير، فإن هذا سيكون كافياً. لإتمام نفس الشيء باستخدام معالجة الخطأ التركيبية سيكون هناك تضمين أكثر بقليل.

```
Try
RefreshPart1 ()
Catch
End Try
Try
RefreshPart2 ()
Catch
End Try
Try
RefreshPart3 ()
Catch
End Try
```

ذاك الكثير من الكود الزائد بالنسبة لنفس التخصص الوظيفي. إذا كنت من الكارهين لعبارة On Error. بكل الوسائل استخدم المقطع الثاني من الكود. ولكن إذا كنت مبرمج أكثر عقلانية، استخدم كل طريقة إذا ما كانت مناسبة في تصميم كوك.

فئة System.Exception نظام الاستثناء The System.Exception Class

إن الفئة System.Exception هي الفئة القاعدية بالنسبة لجميع الاستثناءات التركيبية structured exceptions. عندما يحدث خطأ ما، تستطيع تفحص أعضائها لتحديد طبيعة الخطأ بدقة. وإنك تستخدم أيضاً هذه الفئة (أو واحدة من فئاتها المشتقة) لبناء استثناء خاص بك في انتظار in anticipation استخدام عبارة Throw. . يجدر الجدول التالي أعضاء الكائن System.Exception.

الوصف Description	عضو الكائن Object member
خاصية توفر إمكانية الوصول لتجميع أزواج قيم المفاتيح، وكل منها يوفر معلومات إضافية خاصة بالاستثناء.	Data
خاصية تعرف موقع مساعدة على الشبكة يزود بمعلومات متعلقة بهذا الاستثناء.	HelpLink
خاصية، إذا كان استثناء ما هو تأثير جانبي عن خطأ آخر، فإن الخطأ الأصلي يظهر هنا.	InnerException
خاصية، وصف نصي للخطأ.	Message
خاصية تعرف اسم الاستثناء أو الكائن الذي سبب ذلك الخطأ.	Source
خاصية تعود بنص يوثق بالكامل مسار المقدار الحالي، قائمة بكامل استدعاءات الإجراءات النشيطة التي قادة لعبارات تتسبب بالخطأ.	StackTrace
خاصية تعرف اسم الطريقة التي أطلقت الخطأ.	TargetSite

الفئات المشتقة من System.Exception يمكن أن تتضمن خاصيات إضافية توفر تفاصيل إضافية لنوع خطأ نوعي.

كائن Err الخطأ The Err Object

يوفر الكائن Err إمكانية الوصول لأغلب الأخطاء الحديثة من خلال أعضائه المتنوعة. أي وقت يحدث خطأ ما، توثق الفيچوال بيسك تفاصيل الخطأ في أعضاء هذا الكائن. على الأغلب يتم الوصول له (الكائن الخطأ) ضمن معالج خطأ غير تركيبية للإشارة أو عرض تفاصيل الخطأ. يبين الجدول التالي أعضاء كائن الخطأ Err.

الوصف Description	عضو الكائن Object member
طريقة تنظف كل الخاصيات في الكائن، تضع هذه الخاصيات لقيمها الافتراضية عادة، إنك تستخدم الكائن فقط لتحديد التفاصيل للخطأ الذي تم إطلاقه. ولكن تستطيع أيضاً استخدامه لإدخال خطأ ضمن تفاصيل خطأ خاصة بك. لاحظ شرح الطريقة Raise فيما بعد هنا.	Clear
عنوان رقم السطر الأقرب لمكان حدوث الخطأ. في تطبيقات الفيچوال بيسك الحديثة، عناوين الأسطر المرقمة على الأغلب لا تستخدم لذلك بشكل عام هذا الحقل هو 0.	Errl property
وصف نصي للخطأ.	Description property
الموضع الذي ضمنه ملف المساعدة على الشبكة فيما يتعلق بالخطأ. إذا كانت هذه الخاصية وخاصية HelpFile تم وضعها، فإن المستخدم يستطيع الوصول لملف معلومات المساعدة ذات الصلة بالموضوع على الشبكة.	HelpContext property
خاصية، القيمة العددية المعادة من معظم الاستدعاء الحديث لـ مكتبة الربط الديناميكي التابعة للدوت نت الأولية، فيما إذا كان خطأ أم لا.	LastDLLError
الكود العددي للخطأ الفعال.	Number property

استخدم هذه الطريقة لتوليد خطأ وقت التنفيذ. على الرغم من أن هذه الطريقة تعمل على تضمين بعض المعاملات النسبية لوضع خاصيات أخرى في الكائن Err، بإمكانك أيضاً وضع الخاصيات بنفسك قبل استدعاء الطريقة Raise. أي خاصيات تعمل على إعدادها سيتم حفظها في الكائن ليتم اختبارها بواسطة كود معالج الخطأ الذي يستقبل الخطأ. اسم التطبيق، أو الفئة، أو الكائن الذي يولد الخطأ الفعالي.

الكائن تصحيح The Debug Object

الفيجوال بيسك6 (والأقدم) تتضمن أداة مساعدة والتي ستخرج بسرعة معلومات تصحيح من البرنامج، وعرض مثل هذه المخرجات في "النافذة المباشرة Immediate Window" لبيئة تطوير الفيجوال بيسك.

```
Debug.Print ("Reached point G in code")
```

إصدار الدوت نت للفيجوال بيسك يحسن كائن التصحيح Debug مع ميزات أكثر، وتغيير بسيط في البناء. تم تبديل الطريقة Print بـ WriteLine، الطريقة Write تخرج نص بدون لائحة عودة المشيرة carriage return في النهاية.

```
Debug.WriteLine ("Reached point G in code")
```

كل شيء تخرجه باستخدام الطريقة WriteLine (أو ما يشابهها) يذهب إلى سلسلة "المصغيات listeners" (الملحقة (المرفقة) بكائن التصحيح Debug. تستطيع إضافة منصاتك الخاصة، متضمنة المخرجات لملف العمل. ولكن كائن التصحيح Debug يتم استخدامه فعلياً فقط عند تصحيح برنامجك. حالما تترجم الإصدار الأخير لبرنامجك، فلا تعمل أي من الميزات المتعلقة بالتصحيح بعد ذلك، بشكل تصميمي.

إذا كنت ترغب بتسجيل حالة البيانات من التطبيقات المحررة، خذ استخدام الكائن My.Application.Log (أو My.Log في برامج ASP.NET). فهو مشابه للكائن Debug، يرسل الكائن مخرجاته لأي عدد من المنصات المسجلة registered listeners. بشكل افتراضي، جميع المخرجات تذهب لمخرجات تصحيح قياسية (تماماً مثل كائن التصحيح Debug) وإلى ملف التسجيل logfile الذي يتم إنشاؤه بشكل خاص لمجمع تطبيقك application's assembly.

ميزات أخطاء أخرى للفيجوال بيسك Other Visual Basic Error Features

تتضمن لغة الفيجوال بيسك القليل من العبارات الخاصة بالأخطاء والميزات الأخرى والتي يمكن أن تجد أنها مفيدة:

الدالة (الخطأ إلى نص) ErrorToString function

ترجع هذه الطريقة برسالة الخطأ المرافقة لعدد خطأ النظام العددي numeric system error، على سبيل المثال، (10) ErrorToString تعود بـ "يتم إصلاح هذه المصفوفة أو إغلاقها بشكل مؤقت This array is fixed or temporarily locked" وهي مفيدة فقط مع أكواد الأخطاء الغير تركيبية الأقدم.

الدالة (هل هو خطأ) IsError function

عندما تزود معامل نسبي كائنني لهذه الخاصية، فإنها تعود بصواب إذا كان الكائن هو كائن System.Exception (أو مشتق منه)

مشروع Project

سيكون كود مشروع هذا الفصل مختصر. وسيظهر كود معالجة الخطأ Error-handling على مدى كامل التطبيق، ولكن سنضيفه قليلاً قليلاً كلما تقدمنا في صناعة المشروع. أما الآن، دعنا نركز على إجراءات أو روتينات معالجة الخطأ المركزي والذي سيشغل بعض الفعل الأساسي عندما يحدث خطأ ما في أي مكان في البرنامج. أما بالنسبة لتعابير لاما، سنؤجل مثل هذا الكود حتى فصل متأخر.

معالج الخطأ العام General Error Handler

بقدر أهمية ودقة الحاجة إلى معالجة الخطأ، فإن تطبيق العمل النموذجي لن يصادف تنوع كبير في أنواع الخطأ. وتطبيقات مثل مشروع المكتبة غير محصنة بشكل رئيسي بالنسبة لثلاثة أنواع من الأخطاء (1) أخطاء إدخال البيانات، (2) الأخطاء التي تحدث عند قراءة البيانات من أو كتابة البيانات إلى جدول قاعدة بيانات، (3) الأخطاء المرتبطة بالطباعة. من المؤكد أنه توجد أخطاء عديدة لتجاوز الحد أو أخطاء أخرى مرتبطة بالبيانات التي قيد الاستخدام، ولكنها على الأغلب تتفاعل مع المصادر الخارجية، مثل قاعدة البيانات والتي هي محور اهتمامنا.

بسبب محدودية أنواع الأخطاء التي تحدث في التطبيق، فمن الممكن كتابة روتين شامل يخبر المستخدم عن الخطأ بأسلوب ثابت (مستقر). كلما حدث خطأ وقت التنفيذ، سنستدعي هذا الروتين المركزي، لننتج للمستخدم معرفة ما يجري. يمكن لمقطع الكود عندما يحدث الخطأ أن يقرر فيما إذا سيجري أي فعل مكافئ خاص، أو أن يستمر وكان ليس هناك خطأ قد حدث.

في المشروع، افتح ملف الفئة General.vb، وأضف الكود التالي كطريقة جديدة للوحدة البرمجية General.

```
Public Sub GeneralError(ByVal routineName As String, ByVal theError As System.Exception)
    ' ----- Report an error to the user.
    On Error Resume Next
    MsgBox("حدث الخطأ التالي عند الموقع" & routineName & ":" & vbCrLf & vbCrLf & theError.Message,
    MsgBoxStyle.OkOnly Or MsgBoxStyle.Exclamation, ProgramTitle)
    My.Application.Log.WriteException(theError)
End Sub
```

إليك كيف يعمل. عند تصادف خطأ في روتين ما، يستدعي معالج الخطأ النافذ المفعول الطريقة المركزية GeneralError.

```
Public Sub SomeRoutine ()
    On Error GoTo ErrorHandler
    ' ----- Lots of code here.
    Return
ErrorHandler:
    GeneralError("SomeRoutine", Err.GetException())
    Resume Next
End Sub
```

تستطيع استخدامه مع الأخطاء التركيبية كما يلي:

```
Try
    ' ----- Troubling code here.
Catch ex As System.Exception
```

Mhm76

```
GeneralError("SomeRoutine", ex)
End Try
```

الهدف من الطريقة العامة GeneralError بسيط: ينقل للمستخدم أن هناك خطأ قد حدث، ومن ثم ينتقل للأمام. تستطيع تحسين الروتين ببعض الميزات الإضافية. تسجيل الخطأ إلى ملف خارجي (أو إلى أي منصة مسجل log listener) يمكن أن يساعدك فيما بعد إذا ما احتجت إلى تفحص الأخطاء الناتجة عن التطبيق. كما في السطر التالي المضاف إلى الطريقة GeneralError.

```
My.Application.Log.WriteException(theError)
```

بالطبع، إذا ما حدث خطأ بينما تتم الكتابة إلى السجل، فذلك سيكون مشكلة كبيرة، لذلك قمنا بإضافة سطر آخر لبداية الطريقة GeneralError.

```
On Error Resume Next
```

التقاط الأخطاء الغير معالجة. Unhandled Error Capture.

كما ذكرت سابقاً، إنها فكرة جيدة أن تضمن معالج خطأ شامل في كودك، في حالة إذا ما تغلب خطأ ما على دفاعاتك. لتضمن هذا الكود، أعرض جميع الملفات في مستكشف الحلول باستخدام زر "إظهار جميع الملفات Show All Files"، افتح الملف *ApplicationEvents.vb*، وأضف الكود التالي إلى الفئة *MyApplication*.

```
Private Sub MyApplication_UnhandledException(ByVal sender As Object, ByVal e As
Microsoft.VisualBasic.ApplicationServices.UnhandledExceptionEventArgs) Handles Me.UnhandledException
' ----- Record the error, and keep running.
e.ExitApplication = False
GeneralError("Unhandled Exception", e.Exception)
End Sub
```

بما أن لدينا سابقاً الروتين GeneralError لتسجيل أخطاءنا، من الممكن أن نستفيد منها هنا.

الأدو نت ADO.NET

إذا كنت مطور ويندوز جديد، فإنك لم تتعرف بعد على الاختصارات التالية، بعض الأحيان متعارضة، وأحيان أخرى أدوات تتفاعل قاعدة بيانات متممة:

ODBC—Open Database Connectivity. فتح قاعدة البيانات في الحالة المتصلة.

ISAM—Indexed Sequential Access Method. طريقة الوصول التسلسلية المفهرسة.

DAO. كائنات الوصول للبيانات. Data Access Objects.

Remote Data Objects. كائنات البيانات البعيدة.

OLE . ربط وتضمين الكائنات لقاعدة البيانات Object Linking and Embedding for Databases

ADO. تفعيل س لكائنات البيانات. ActiveX Data Objects

عندما تنظر إلى هذه القائمة، من المحتمل أن تفكر (هذا عظيم، يوجد الكثير من الخيارات للاختيار منها). فستكون أحق بهذا التفكير. فهذه القائمة ليست عظيمة، إنها مخيفة. تصور، ولدقيقة فقط، فنحن لم نتحدث حول واجهات interfaces قاعدة البيانات، ولكن من ناحية أخرى، هي قضايا أكثر عملية.

عندما قدمت ميكروسوفت تقنية كائن قاعدة البيانات الجديدة ضمن المزيج، عملت على تبعها سريعاً بطفرة إعادة برمجة لـ "الميراث" الأقدم لتطبيقات للفيجوال بيسك والتي تعود لتقنية قاعدة البيانات الأحدث. وهذا لم يكن ممكناً دائماً، إلى هنا، من الواضح أن أدو نت ، مكتبة قاعدة بيانات ميكروسوفت للدوت نت، مختلف.

إذا كنت على اطلاع على تقنية ADO، تجهز لتتأسف. فإن أدو نت ليس بالخليفة الطبيعي لأدو ADO. إنه بالكامل تقنية جديدة ليس لها علاقة بأدو، وعلى الرغم من ذلك فهي تتشارك بعض التقنية مع أدو والأدوات الأخرى الأقدم، وتعمل أدو نت هذا لتلعب بعقلك.

ما هو الأدو الدوت نت؟ What Is ADO.NET?

إن أدو دوت نت مجموعة من الفئات، المضمنة مع إطار عمل الدوت نت، وتمثل الطريقة الرئيسية التي بواسطتها تتفاعل تطبيقات الدوت نت مع قواعد البيانات العلائقية وأنظمة إدارة وفتح البيانات الخاصة الأخرى ولكنها ليست للتفاعل فقط، في الحقيقة، إن أدو دوت نت قاعدة بيانات علائقية جزئية في الذاكرة قائمة بذاتها. تستطيع إنشاء الجداول والعلاقات (الروابط) من خلال كائنات أدو دوت نت، إضافة، وإزالة أي سجلات، استعلامات الجداول بالاعتماد على معيار "الاختيار" `SELECT` وعمل مهمات أخرى بسيطة والتي هي نموذجية بالنسبة لأنظمة قواعد البيانات العلائقية القائمة بذاتها.

جميع الفئات المضمنة مع أدو دوت نت تظهر في فضاء الأسماء System.Data، بعض فضاءات الأسماء التابعة (الثانوية subordinate) توفر فئات مشتقة مجهزة باتجاه منصات قواعد بيانات خاصة (نوعية)، على سبيل المثال، يستهدف فضاء الأسماء System.Data.SqlClient قواعد بيانات سكول SQL Server databases، System.Data.OracleClient يركز على أنظمة إدارة قواعد بيانات أوراكل Oracle RDBMS systems، ويمكن لموفرات قواعد بيانات أخرى تطوير تنفيذات انسيابية لفئات أدو نت المتنوعة للاستخدام مع أنظمتها الخاصة، وتزود بها كفضاءات أسماء منفصلة.

تنفذ ADO.NET تجربة بيانات غير متصلة *disconnected data experience*. في برمجة قواعد البيانات التقليدية، وخاصة تطبيقات سطح المكتب desktop applications، الاتصال بين تطبيق ما وقاعدة بياناته تم إصلاحه ولأجل طويل. عندما يبدأ التطبيق، يبدأ الاتصال. وعندما يخرج التطبيق بعد عدة ساعات، فإن الاتصال سيتم إنهاءه أخيراً. ولكن في عالم مواقع الانترنت الضخمة والقابلة للتوسع، المحافظة على قاعدة البيانات في وضع الاتصال لساعات طويلة دون انقطاع تكون في بعض الأحيان تذبذب (إسراف) وغالباً يكون مستحيل.

تشجعك أدو دوت نت على فتح اتصالات طويلة بشكل كافي للحصول على البيانات التي تفي *fulfills* باحتياجاتك المباشرة. حالما تكون البيانات لديك، فإنك تسقط الاتصال حتى تحتاج في الوقت أحر استخراج، أو إدخال، أو تحديث محتوى قاعدة البيانات. إذا أصدرت عبارة سكول التالية:

```
SELECT * FROM Customer WHERE BalanceDue > 0
```

لديك خيار (1) اختبار جميع السجلات حالياً وبسرعة وبطريقة بسيطة، أو (2) تحميل البيانات إلى جدول في الذاكرة مثل الكائن، وإغلاق الاتصال، والعمل مع السجلات المحملة وكأنها السجلات الأصلية. إذا استخدمت الطريقة الأولى، تستطيع تمضية وقت جيد في التنقل بين السجلات، وتأخذ عدة دقائق في معالجة كل منها. ولكن أدو دوت نت يمتعض (يرفض *frowns upon*) هذا النوع من التصرف الأناني. الهدف هو الدخول وأخذ البيانات بالسرعة الممكنة بسبب الطبيعة الغير متصلة *disconnected nature* للدوت نت، فبعض التقنيات الشائعة في تطبيقات قواعد البيانات تحتاج التغيير. على سبيل المثال، البحث طويل الأمد في سجلات قاعدة البيانات خلال تعديل المستخدم ("التزامن السيئ *pessimistic concurrency*") صعب التحقيق في البيئة الغير متصلة للدوت نت. سيكون عليك استخدام طرق أخرى، مثل التعاملات (المداولات) أو ميزات الإجراء المخزن الصغيرة، لإنجاز نفس الهدف.

نظرة شاملة عن الدوت نت Overview of ADO.NET

يقسم الدوت نت عالمه إلى نصفي كرة: الموفرات *providers* ومجموعة البيانات *data set*. توفر الموفرات إمكانية الوصول إلى محتوى ما، مثل قاعدة بيانات أوراكل. مجموعة البيانات تعمل على تحضير وجلب المحتوى الأصلي الذي تم الحصول عليه من المخزن الطويل الأمد (قاعدة البيانات). حالما يتم الإحضار، فإما سيتم استهلاكه أو سيتم إعادته إلى قاعدة البيانات للتخزين الطويل الأمد. إذا تمتك الموفرات إمكانية الوصول إلى البيانات المخزنة، بعض البيانات يمكن أن يتم نقلها ومعالجتها من خلال تطبيق ما ومجموعة بياناته على قاعدة تخزين قصيرة الأجل.

الموفرات Providers

أنظمة قواعد البيانات database systems الضخمة، مثل مخدم سكول SQL Server وأوراكل Oracle، هي مخدمات قائمة بذاتها تتفاعل مع أدوات وتطبيقات العميل بشكل غير مباشر فقط. هذه الأنظمة تقبل بشكل عام اتصال شبكة من العملاء من خلال منفذ بروتوكول التحكم بالإرسال/بروتوكول الانترنت TCP/IP port أو اتصال مشابه. حالما يتم التصديق *authenticated*، يستطيع الزبون عمل جميع متطلباته من خلال هذا الاتصال قبل قطع الاتصال عن النظام. في عام 1990 نفذت ميكروسوفت (فتح قاعدة البيانات في الحالة المتصلة *ODBC*) (بالاعتماد على معايير موجودة) كنظام عام من خلاله يمكن للعميل أن يتصل بمخدمات قاعدة البيانات، بالإضافة لمصادر البيانات الأخرى الأيسر. لن يقلق العملاء بعد هذا حول جميع بروتوكولات شبكة العمل الضرورية لمحاكاة قاعدة البيانات، كل ذلك الكود تم تضمينه في مشغل ODBC. أطلقت ميكروسوفت فيما بعد نظام اتصال بيانات مشابه يدعى *OLE DB* (ربط وتضمين الكائنات لقاعدة البيانات)، بالاعتماد على تقنية *ActiveX*. حالما ظهرت مشغلات *OLE DB* للأنظمة المشتركة، وعلى الرغم من أنك ماتزال تستطيع الدخول إلى مصادر ODBC من خلال مشغل ODBC الشامل المبني ضمن *OLE DB*.

في الدوت نت، كل من ODBC وOLE DB تم استبدالهما بأكود المكتبات libraries، والموفرات providers التي توفر جميع الاتصالات بين قاعدة البيانات وتطبيقك. إن الموفرات هي الجزء المكمل لآدو دوت نت ADO.NET، وعليك استخدامها للحصول على قواعد بياناتك. لحسن الحظ، تتواجد الموفرات من أجل أنظمة قواعد البيانات الرئيسية، ويتواجد موفر OLE DB من أجل الأنظمة التي هي بدون موفر بنفسها.

من أجل الكائنات الرئيسية التي تكوّن وجهة نظر المبرمج عن الموفر:

كائن الاتصال *The Connection object*

هذه الكائن يوجه الاتصال بين برنامجك ومصدر البيانات. إنه يتضمن خصائص وطرق تتيح لك الإشارة إلى موقع أو وسائط الاتصال لمصدر البيانات. العديد من أوامر العمليات (الإجراءات) يتم إدارتها عند مستوى هذه الكائن.

كائن الأمر *The Command object*

يأخذ هذا الكائن عبارة سكول SQL التي توفرها، ويحضّر لها ليتم نقلها من خلال كائن الاتصال. تستطيع تضمين وسيطات في أمرك لتخزين إجراء ودعم عبارة معقدة.

كائن قارئ البيانات *The DataReader object*

يوفر كائن قارئ البيانات طريقة بسيطة وفعالة لاستخراج نتائج من استعلام بيانات. تستخدمه كائنات أخرى في الدوت نت لاستقبال وإعادة توجيه البيانات للاستخدام ضمن برنامجك، ولكن يستطيع كودك استخدامه مباشرة لمعالجة نتائج عبارة "SELECT" أو إجراء استرجاعي لبيانات أخرى.

كائن محول البيانات *The DataAdapter object*

هذا الكائن هو ما يجعل الاتصال بين مجموعة البيانات وباقي الموفرات ممكنة. واحد من أعماله الرئيسية هو تعديل عبارات معالجة البيانات (عبارات SELECT، و INSERT، و UPDATE، و DELETE) المولدة بواسطة مجموعة البيانات في تنسيق يمكن أن يتم استخدامه بواسطة مصدر البيانات المناسب (ذو الصلة).

استخدام هذا الكائنات مفيد قليلاً، ولكنها ليست صعبة الفهم. للاتصال بقاعدة بيانات علائقية نموذجية، مثل مخدم سكول، ومعالجة البيانات، اتبع الخطوات التالية:

1. أسس اتصال لمصدر بياناتك باستخدام كائن الاتصال.

2. ضمن عبارة سكول في كائن الأمر.

3. نفذ كائن الأمر في سياق الاتصال المؤسس له.

4. إذا كان يجب استعادة أي بيانات، استخدم إما قارئ البيانات DataReader لعمل مسح خلال السجلات، أو دمج (ضم) لكل من "محول بيانات DataAdapter" و "مجموعة بيانات DataSet (أو جدول بيانات DataTable)" لاستخراج أو تخزين النتائج.

5. أغلق كل الكائنات التي قمت بفتحها لمعالجة البيانات.

على الرغم من أن إطار عمل الدوت نت يتضمن موفرات بيانات للعديد من أنظمة البيانات المختلفة، فإن باقي المناقشة في هذا الفصل ستتركز على موفر مخدم سكول فقط، والذي يتم عرضه من خلال فضاء الأسماء System.Data.SqlClient.

ملاحظة: تتضمن سكول سرفر 2005 دعم لميزة تدعى "حالات المستخدم User Instances" للاستخدام مع قواعد بيانات سكول سرفر 2005 بطبعته السريعة. تتيح هذه

الميزة امتياز أدنى low-privilege للمستخدم للوصول لملف قاعدة بيانات سكول سرفر السريع دون الحاجة لمدير administrator من أجل تأسيس إعدادات أمان سكول سرفر لذلك المستخدم. وهذه الميزة مفيدة في البيئات حيث البرمجيات المرتبطة تم تنصيبها من خلال طريقة التوزيع "بنقرة واحدة ClickOnce" (سيتم مناقشتها في الفصل

25 إن شاء الله) دون تضمين للمدير administrator. ويتطلب أيضاً تركيب خاص لتنصيب لسكول سرفر السريع قبل الاستخدام. ولمزيد من المعلومات حول هذه الميزة

راجع موضوع "العمل مع حالات المستخدم" في قسم "الدوت نت Working with User Instances" في مستندات شبكة مطوري ميكروسوفت MSDN. والموفرة مع تنصيب الفيچوال أستوديو إذا كنت قد نصبت هذه المستندات MSDN.

مجموعات البيانات *Data Sets*

إذا كنت في طريقك لعمل أكثر من مسح scan للبيانات الراجعة من استعلام قارئ البيانات DataReader، من المحتمل أنك سوف تستخدم مجموعة البيانات data set لتخزين أو إدارة أو بشكل اختياري تحديث بياناتك. توفر كل مجموعة بيانات عرض غير متصل (بدون اتصال) شامل generic disconnected view للبيانات، سواء كانت بيانات من موفر أو بيانات عملت على بناءها من خلال الكود. على الرغم من أن كل موفر يتم ربطه بمنصة قاعدة بيانات معينة (مثل أوراكل) أو اتصال قياسي (مثل OLE DB)، الكائنات في مملكة مجموعة البيانات هي شاملة، ويمكن التفاعل معها بأي موفرات خاصة بمنصة.

ثلاث كائنات تركيب عالم مجموعات البيانات data sets :

كائن مجموع البيانات *The DataSet object*

يتصرف كل كائن مجموعة بيانات كقاعدة بيانات صغيرة. تستطيع إضافة العديد من الجداول لمجموعة البيانات كما ترغب، وتؤسس علاقات مفاتيح ثانوية foreign-key relationships بين حقول fields هذه الجداول tables. ذاتيات internals كل مجموعة بيانات هي غامضة لا يمكن سبرها unfathomable mystery، ولكن تستطيع تصدير كامل مجموعة بيانات DataSet لـ XML. وتعيد تحميله مرة أخرى فيما بعد لما كان عليه إذا كان يجب عليك ذلك.

كائن جدول البيانات *The DataTable object*

كل جدول table في مجموعة بياناتك DataSet يستخدم كائن جدول بيانات Data Table منفصل، وقابلية الوصول له تتم من خلال تجميع جداول Tables collection مجموعة البيانات DataSet. إن جداول البيانات مفيدة أيضاً ككائنات قائمة بذاتها. إذا خططت لإضافة جدول مفرد لمجموعة بياناتك DataSet، من المحتمل أنك تفضل استخدام فقط كائن جدول بيانات DataTable لوحده بدون مجموعة بيانات DataSet. ضمن كل كائن جدول بيانات DataTable، كائنات عمود بيانات DataColumn وصف بيانات DataRow منفصلة تؤسس تعريفات حقل field وقيم values بيانات حقيقية، على الترتيب.

كائن علاقة البيانات *The DataRelation object*

استخدم كائنات علاقة البيانات DataRelation، المخزنة ضمن تجميع علاقات مجموعة البيانات DataRelation's collection، لتأسيس علاقات على مستوى الحقل field-level relationships وقيود constraints بين الأعمدة في كائنات جدول البيانات DataTable.

على الرغم من أن مجموعات البيانات data sets على الأغلب يتم استخدامها مع الموفرات providers، تستطيع استخدامها بشكل منفصل لبناء تجميع خاص بك في الذاكرة للجداول والعلاقات. وهذا مشابه لـ "مجموعات سجلات العميل client-side record sets" والتي تستطيع بناءها بكائنات آدو السابقة لآدو دوت نت

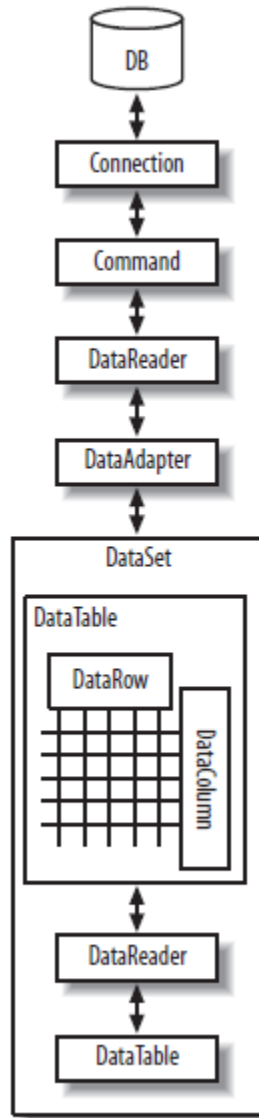
ADO.NET pre-، على الرغم من الميزات المضمنة في ADO.NET التي تجعل مجموعات البيانات data sets أكثر قوة من مجموعات السجلات الأقدم older record sets.

ملاحظة: تتضمن الفيجوال بيسك 2008 ميزة "مجموعات بيانات محددة النوع Typed DataSets" تستخدم لتكامل مجموعة بيانات DataSet مع بيانات معينة أو تنسيق سجل record format. تستخدم تقنية لينكو LINQ الجديدة ميزة مشابهة للمساعدة في تأسيس علاقات بين لينكو وجداول قاعدة البيانات database tables.

مجموعات البيانات مقابل بدون مجموعات بيانات Data Sets Versus No Data Sets

عندما يتم استخدام كل من الموفرات ومجموعات البيانات مع بعضها فإنها تمنح واجهة نهاية (طرف) - إلى- نهاية (طرف) لقيم البيانات المستقلة من حقول في جداول قاعدة البيانات إلى بنود في الذاكرة لسجل صف بيانات DataRow. يبين الشكل التالي تفاعل هذه الكائنات. عندما تتفاعل مع بيانات من قاعدة بيانات خارجية، إنك دائماً تستخدم فئات الموفر provider، ولكن يعود لك فيما إذا كنت تريد استخدام مجموعات البيانات data sets. يوجد محاسن وسيئات لكلا الطريقتين، بعض منها يظهر في الجدول التالي:

بدون مجموعات البيانات Without data sets	مع مجموعات البيانات With data sets
عليك توفير جميع عبارات سكول، في تنسيق يتوقعه الموفر. وهذا صحيح لجميع متطلبات SELECT و INSERT و UPDATE و DELETE.	يعمل كل من مجموعة البيانات DataSet ومحول البيانات DataAdapter مع بعضهما لصنع العديد من عبارات سكول عوضاً عنك.
البيانات المستخرجة من خلال DataReader هي للقراءة فقط عليك أن تصدر أوامر منفصلة لتحديث البيانات.	قراءة البيانات من قاعدة البيانات يمكن أن يتم تعديلها في الذاكرة، ويتم تحديثها كمجموعة مع استدعاء طريقة مفردة.
تنقلات البيانات فعالة جداً، بما أنه ليس هناك حاجة لتكاليف نقل البيانات ضمن تركيب مجموعة بيانات معقد.	يمكن أن يكون هناك أداء ناجح بما أن مجموعة البيانات تبني الكائنات الضرورية المطلوبة لكل سجل منقول.
تجميع الذاكرة يكون محدود لسجل مفرد مساوي لحقول البيانات، زائد بعض التكاليف الصغيرة.	تجميع الذاكرة مطلوب لمجموعة ناتجة بالكامل، زائد التكاليف بالنسبة لكل جدول، حقل، عمود، والصفوف في المجموعة الناتجة.
يمكن أن يتم فتح قارئ بيانات DataReader مفرد فقط في نفس الوقت (مالم يدعم الموفر MARS والذي سأناقشه فيما بعد)	أي عدد من مجموعات البيانات يمكن أن تكون قيد الاستخدام في نفس الوقت.
عمر الاتصال بقاعدة البيانات يتواجد طالما أن قارئ البيانات DataReader قيد الاستخدام. فإذا استغرقت خمس دقائق لمسح مجموعة بيانات ناتجة لأنك تعمل الكثير من تحليل السجلات المسبقة، فإن الاتصال سيبقى فعال كامل الخمس دقائق.	اتصالات البيانات يتم حفظها فقط بطول كافي لنقل البيانات من أو إلى قاعدة البيانات فقط.
تخضر قارئات البيانات DataReaders سجل كل مرة. ويجب أن يتم معالجة السجلات في ترتيب الاستخراج.	تستطيع القفز فوق السجلات في مجموعة البيانات، وتمايز بينهم لتفي بمتطلباتك.
تمضي الكثير من الوقت في العمل مع النصوص (من أجل عبارات سكول) وحقول بيانات الصف.	جميع حقول البيانات يتم تنظيمها بشكل منطقي، تماماً مثل ما كانت في قاعدة البيانات الفعلية. تستطيع التفاعل معها بشكل مباشر.
كل أمر Command واتصال Connection يعمل مع مصدر بيانات يدعم موفر مفرد.	يمكن أن تتصل جداول البيانات DataTables المختلفة ضمن مجموعة بيانات DataSet إلى مصادر بيانات متميزة (مختلفة distinct data sources). وأيضاً، تستطيع صنع البيانات يدوياً لذلك كل صف بيانات DataRow يحتوي بيانات من مصادر مختلفة.
بسبب أنك أنت الذي تدير جميع عبارات سكول SQL، فإن لديك (نسبياً) مستوى محدود (relatively) مستوى عالي من السيطرة على عمليات تفاعل البيانات بالكامل.	بسبب أن عرض البيانات هو مجرد abstracted، فإن لديك (نسبياً) مستوى محدود للسيطرة على عملية تفاعل البيانات بالكامل (على الرغم من أن الاستخدام المتقدم لمجموعات البيانات يعمل على منحك بعض التحكم الإضافي).



الموفرات ومجموعات البيانات أثناء العمل Providers and data sets in action

بالنسبة لي فإن المدخلة الأخيرة في الجدول السابق هي الحاسمة إن عمل المبرمج هو السيطرة على تجربة برمجيات المستخدم، والأكثر تحكماً هو الأفضل. ولهذا السبب فإني أكره المعالجات السحرية wizards و"مولدات الكود code generators" والتي تأخذ التحكم عوضاً عني، أو عوضاً عن المطور. على الرغم من أنني أصبر على القالب المزود بواسطة الفيجوال أستوديو عند إنشاء مشروع جديد. ما يزال وكما سترى، لمحات التحكم الشخصية على كودي في مشروع المكتبة، مع اعتمادي الكبير على قارئ البيانات DataReaders أكثر من مجموعات البيانات DataSets. عندما أعمل على تخزين البيانات على المدى الطويل، فإني عادة ألصق البيانات فقط في جدول بيانات DataTable دون تضمين مجموعة بيانات DataSet.

دعم مجموعات ناتجة عن نشاط متعدد. MARS Support

لقد ذكرت فيما سبق شيء ما يدعى MARS و MARS اختصاراً "مجموعات ناتجة عن نشاط متعدد Multiple Active Result Sets" عادة، يسمح كائن اتصال Connection مفرد لقارئ بيانات DataReader مفرد لأن يكون قيد الاستخدام في وقت معين. وهذا التحديد هو ثنائي الاتجاه bidirectional. إذا كنت تعمل مسح قارئ بيانات باستخدام عبارة "SELECT" فلا تستطيع إصدار عبارات إدراج INSERT، تحديث UPDATE، أو حذف DELETE على نفس الاتصال حتى تغلق قارئ البيانات DataReader. مع تقديم MARS، يمكن لاتصال وحيد الآن معاملة نشاطات انتقال بيانات في أي اتجاه. أضافت سكول سرفر دعم MARS مع الإصدار 2005. وكذلك دعمت أراكل MARS كالميزات منذ الإصدار الأول للدوت نت. تبدو اتصالات MARS مميزة تحتاج لتمكينها باستمرار. ولكنها تعمل على إضافة تكاليف إضافية لتطبيقك تبطئ تطبيقك. وأيضاً لا تمتزج دائماً مع التطبيقات ذات مسارات التنفيذ المتعددة multithreaded applications.

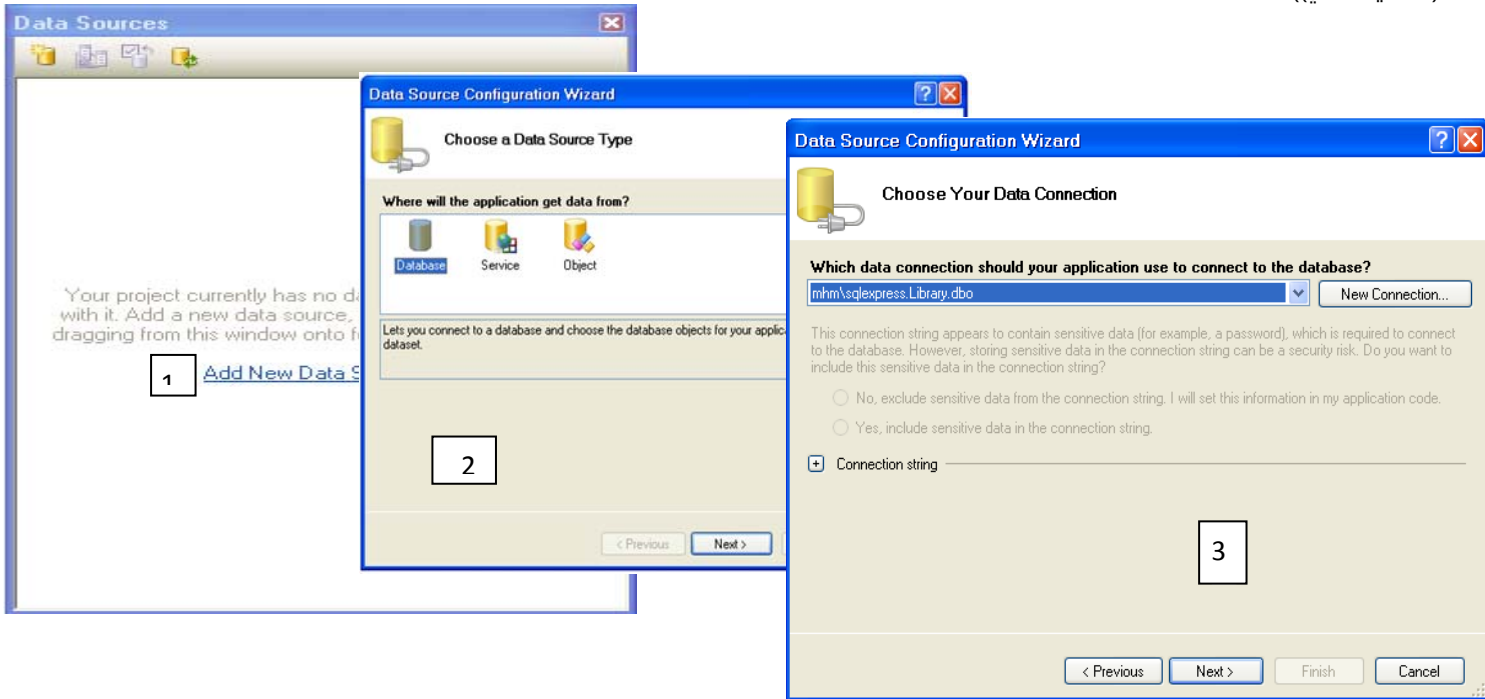
الاتصال إلى سكول سرفر باستخدام فيجوال أستوديو. Connecting to SQL Server with Visual Studio.

إن للفيجوال أستوديو العديد من الأدوات الجاهزة والتي تجعل العمل مع البيانات بسيطاً كإسقاط حسناً، ليس بهذه السرعة. ولكن بالإجابة على السؤال من الأسئلة وسحب وإسقاط بند واحد، تستطيع بناء تطبيق كامل يتيح لك تحديث البيانات في قاعدة بياناتك. لنجرب ذلك معاً.

إنشاء مصدر بيانات Creating a Data Source

أبدأ مشروع نماذج ويندوز جديد في الفيجوال أستوديو، اختر بيانات Data << أمر قائمة "أظهر مصادر البيانات Show Data Sources" مما يحضر لوحة مصادر البيانات، كما هو مبين في الشكل التالي.

إن المشروع الجديد لا يتضمن أي مصدر بيانات بشكل افتراضي. لذلك تحتاج لإضافة مصدر بيانات. انقر على الوصلة "أضف مصدر بيانات جديد Add New Data Source" في لوحة "مصادر البيانات Data Sources". يقولك "المعالج السحري لتركيب مصدر بيانات" خلال عمليات إنشاء مصدر بيانات: 1. تسأل الخطوة الأولى "من أين سيحصل التطبيق على البيانات؟ اختر "قاعدة البيانات Database" وانقر الزر "التالي Next" (كما هو مبين في الشكل التالي (التخطيط الثاني)).



2. تسأل الخطوة التالية، "أي اتصال بيانات يجب أن يستخدمه تطبيقك للاتصال بقاعدة البيانات؟" سيكون علينا إنشاء اتصال جديد لقاعدة بيانات المكتبة والتي عملنا على تصميمها في الفصل الرابع. انقر زر "اتصال جديد New Connection" (المبين في الشكل السابق (التخطيط الثالث)). 3. يظهر حوار اختيار مصدر البيانات. اختر ميكروسوفت سكول سرفر Microsoft SQL Server من قائمة "مصدر البيانات Data source"، ومن ثم انقر زر "استمر Continue". إذا كنت قد وصلت لهذا الحوار من قبل، واخترت حقل الاختبار "دائماً استخدم هذا الاختيار Always use this selection" من المحتمل أن يظهر هذا الحوار على الإطلاق.

4. يظهر حوار إضافة اتصال Add Connection لتجميع تفاصيل الاتصال الجديد.. إذا كان حقل "مصدر البيانات Data source" يحتوي على شيء ما غير "ميكروسوفت سكول سرفر Microsoft SQL Server" انقر على زر "تغيير Change" لتبديل نوع الاتصال باستخدام الحوار المذكور في الخطوة الثالثة. 5. العودة إلى نموذج "إضافة اتصال" وقد تم ملئ حقل "اسم المستخدم Server name" باسم نسخة سكول سرفر SQL Server instance خاصتك. هذه القائمة المنسدلة لديها نسخ مجدولة لأسماء المخدمات، ولكن إذا لم تكن كذلك، عليك إدخالها بنفسك. الاسم الافتراضي بالنسبة لسكول سرفر السريع هو اسم كمبيوترك، مع اللاحقة "SQLEXPRESS". فإذا كان اسم كمبيوترك هو "MYSYSTEM" فاسم النسخة سيكون "MYSYSTEM\SQLEXPRESS" (كما مبين في الشكل السابق (التخطيط 3 فاسم نسختي mhm\ SQLEXPRESS)).

6. ركب إعدادات التصديق (التوثيق) في مقطع "الدخول إلى المخدم Log on to the server". بالنسبة لي فإني استخدم توثيق ويندوز القياسي، ولكنه يعتمد على كيفية تثبيت قاعدة البيانات في الفصل الرابع.

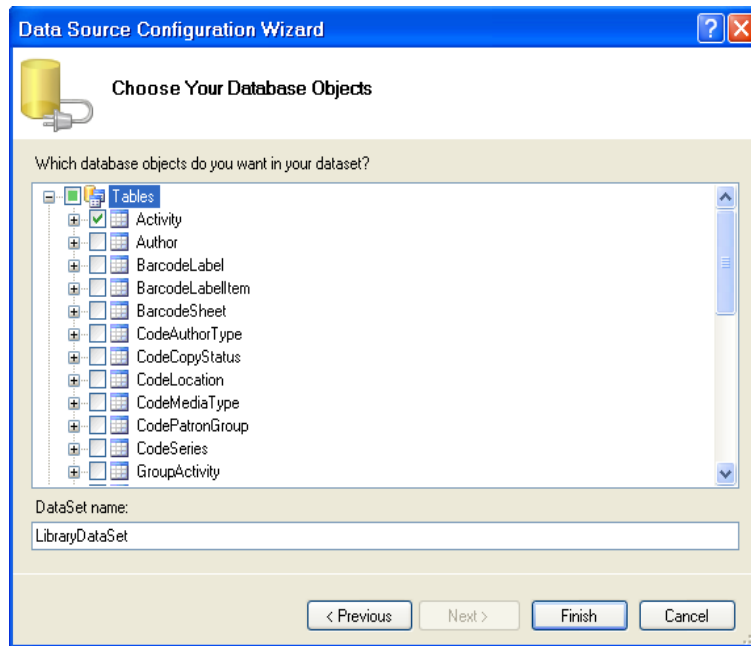
7. في مقطع "اتصال إلى قاعدة البيانات Connect to a database"، إما اختر أو اكتب "Library" من أجل اسم قاعدة البيانات.

8. انقر على زر "اختبر الاتصال Test Connection" للتأكد من أن العمل كله صحيح. عندما تنتهي انقر على زر "موافق OK" لإنشاء الاتصال الجديد..

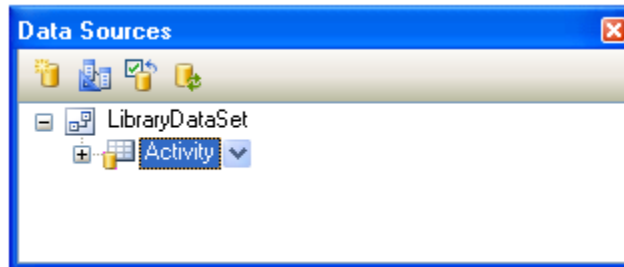
9. بعد النقر على زر "موافق OK" سيتم إعادتنا إلى نموذج المعالج السحري لتركيب مصدر البيانات. والاتصال الذي عملنا على إنشائه يجب أن يظهر الآن في القائمة كما يظهر في الشكل السابق (التخطيط الثالث).

10. اللوحة التالية تسأل فيما إذا سيصبح مصدر البيانات هذا جزء من الإعدادات القابلة للتركيب من أجل هذا المشروع this data source should become part of the configurable settings for this project. سندخل في ميزات الإعدادات الخاصة بالفيجوال بيسك في الفصل 14. ولكن الآن، عليك فقط قبول الافتراضي والنقر على "التالي Next".

11. هذه هي الخطوة الأخيرة في إنشاء مصدر البيانات. تري اللوحة النهائية قائمة بميزات البيانات الناتجة في قاعدة بيانات المكتبة. افتح تفرع الجداول Tables واختر "Activity" كما هو مبين في الشكل التالي. ومن ثم انقر "إنهاء Finish".

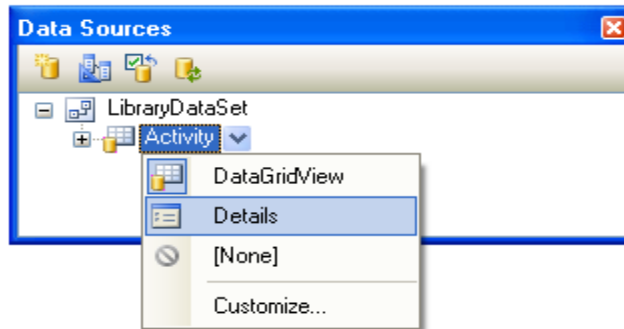


راجع لوحة مصادر البيانات المبينة في الشكل التالي. فهي تتضمن على مصدر بيانات "LibraryDataSet" مع وصلته إلى جدول "Activity".

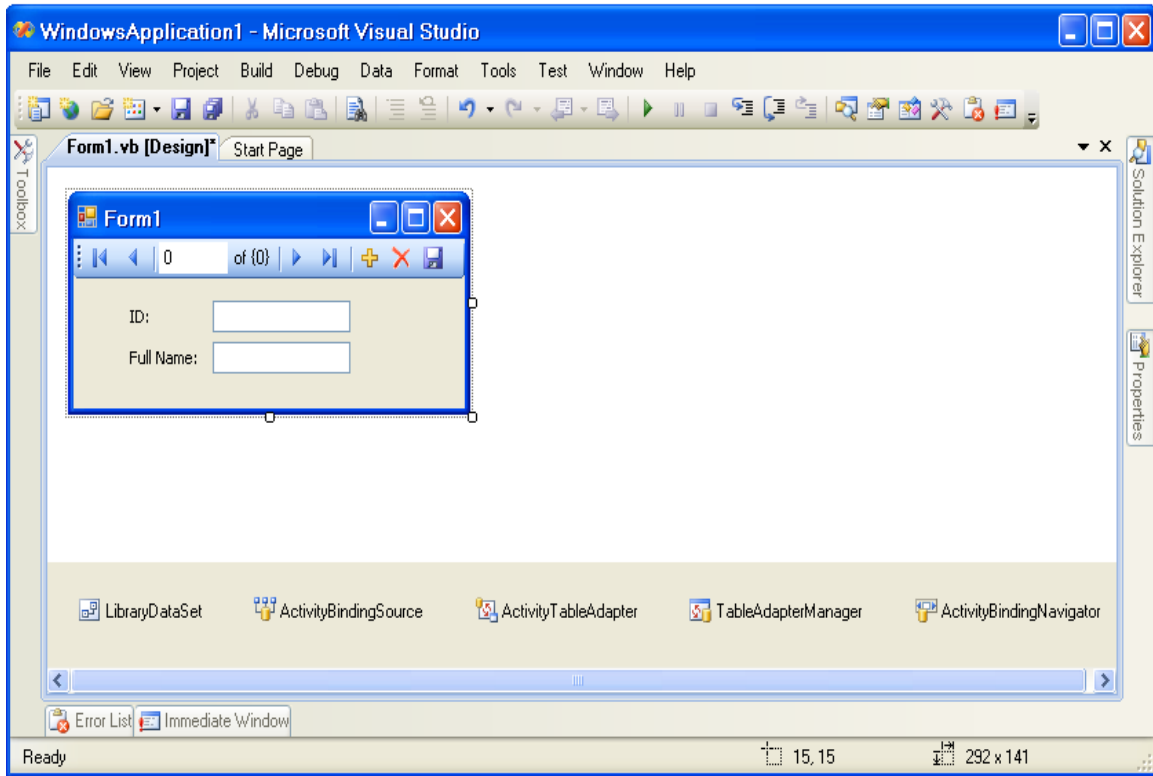


استخدام مصدر بيانات Using a Data Source

على أية حال ما هو مصدر البيانات؟ إنه ببساطة وصلة إلى قسم في قاعدة بياناتك، تم إنجاءه في كائن دوت نت نموذجي. والآن هو جزء من مشروعك وبإمكانك استخدامه للوصول للبيانات في جدول "Activity" من خلال كود مشروعك، أو بواسطة "السحب والإسقاط drag-and-drop". في لوحة مصادر البيانات، ستجد المدخلة "Activity" ولها قائمة منسدلة فعلية. اختر "تفاصيل" من القائمة، وكما هو مبين في الشكل التالي. (يجب أن يتم عرض سطح الفورم Form1 من أجل هذا العمل).



أخيراً، اسحب واسقط مدخلة Activity على سطح Form1. وعندما الاستمرار، ستضيف الفيچوال أستوديو مجموعة من الأدوات للفورم، زائد العديد من الأدوات التي ليس لديها واجهة مستخدم non-userinterface تحت الفورم تماماً. (شاهد الشكل التالي).



بمجرد السحب والإسقاط، تضيف الفيجوال أستوديو جميع الأدوات الضرورية والوصلات ليذور نموذجك ضمن التوربين المشحون لمحرر جدول "Activity". جربه الآن بالضغط على المفتاح F5. في البرنامج المشتغل، تستطيع استخدام تحكم الوصول للسجلات كما في تخطيط برنامج ميكروسوفت أكسس وذلك للتنقل بين السجلات لجدول "Activity" (الأسهم في أعلى الفورم). تستطيع أيضاً تعديل القيم لكل سجل، وإضافة سجلات جديدة، أو حذف سجلات موجودة (ولكن أعد تخزين جميع الأشياء لما كانت عليه، حيث أننا سنستخدمها فيما بعد). تكلم عن القوة، البساطة، الأسطر غير الموظفة! فمن يريد أن يدفع لمبرمجين مثلنا، عندما تستطيع الفيجوال أستوديو فعل كل هذا لك؟

تحميل البيانات Data Binding

في الحقيقة، إن كل ما تعمله الفيجوال أستوديو ليس بالكثير. حيث أنها تستخدم ميزة تدعى "تحميل البيانات Data binding" لربط حقول الفورم مع مصدر البيانات (جدول قاعدة بيانات المكتبة). إن تحميل البيانات هو ميزة جاهزة ضمن أدوات نماذج ويندوز تسمح لهذه الأدوات وبشكل آلي عرض وتحرير القيم من مصدر البيانات المصاحب، مثل قاعدة البيانات. ويتم تصنيف (فرز) الجميع من خلال خاصيات الأداة. اختر أداة FullNameTextBox المضافة لنموذج هذا المشروع، ومن ثم تخصص خاصياتها. في الأعلى تماماً توجد خاصية مسماة "حزم البيانات Data Bindings" وهي خاصية نصية جزئية تحتوي "ActivityBindingSource - FullName" مشيرة إلى الأداة ActivityBindingSource التي ليس لها واجهة مستخدم والتي تم إضافتها بواسطة الفيجوال أستوديو وبشكل آلي. والأداة ActivityBindingSource، في دورها، تحتوي مرجع إلى الكائن LibraryDataSet (مصدر البيانات data source الذي عملنا على إنشائه سابقاً). يتصل مصدر البيانات إلى مخدوم سكول سرفر SQL Server، ومنه إلى قاعدة بيانات المكتبة Library database وأخيراً إلى الجدول Activity. وثم حقله FullName. إذا عملت على إحصاء جميع الكائنات المضمنة في علاقة تحميل البيانات هذه، تصل إلى ما يقارب 5,238 كائن متميز. لا تتعجب من كون الفيجوال أستوديو تقوم بعمل الكثير لأجلك. يوفر تحميل البيانات الكثير من الراحة، ولكنه يأخذ منك الكثير من التحكم والتطوير. على الرغم من أن هناك خاصيات وأحداث تتيح لك إدارة سمات تحميل البيانات وعمليات التحديث الخاصة بها، فإن معظم الكود الأساسي يكون مخفي بعيداً داخل أقسام تحميل البيانات للدوت نت. يمكن أن لا تلمسه، أو تشوّهه، ومجرد كونه كذلك فإنه سيء. فمعتقدني: البرمجيات الجيدة تتضمن تحكم (سيطرة) أعظمية للمطور، وتحكم أقل بالنسبة للمستخدم. كجزء من عملك كمطور توفير بيئة مصنوعة بشكل متقن للمستخدم من أجل الوصول للبيانات الهامة. وهذه المتطلبات التي لديك تتحكم بتجربة المستخدم من خلال كودك المصدري. على وجه التخصص، سنؤجل الكثير من ذلك التحكم إلى تحكم آخر عندما تستخدم أدوات أخرى خارجية. طالما أن هذه الأدوات تسمح لك بالتحكم بمعرفة المستخدم لمستوى تكون راض عنه. هذا عظيم، ولكنني دائماً أشعر بالإحباط من تحميل البيانات، ما عدا وقت تنفيذ عرض بيانات للقراءة فقط من قاعدة البيانات. ميزات مشابهة كانت في الفيجوال بيسك لوقت طويل وقبل أن تصل الدوت نت، وهذه الميزات قد تم جعلها صعبة بالنسبة للمطور من أجل التحكم بتفاعلات البيانات المتنوعة على الفورم.

لحسن الحظ، إذا كنت تتجنب ميزات تحميل البيانات، فإن الفيجوال بيسك ستممر لك مسؤولية إدارة جميع التفاعلات بين قاعدة البيانات والمستخدم.

التفاعل مع سكول سرفر في الكود. Interacting with SQL Server in Code.

الاتصال مع قاعدة البيانات بنفسك هو بالتأكيد أكثر من عمل سحب وإسقاط مصادر البيانات.

بناء نص الاتصال Building the Connection String

الخطوة الأولى على الطريق إلى نمط حياة التحكم بالبيانات هو الاتصال إلى قاعدة البيانات باستخدام نص الاتصال. إذا كنت قد استخدمت ADO، فإنك على معرفة مسبقة بنصوص الاتصال المستخدمة في ADO.NET، بشكل عام نصوص الاتصال connection strings متشابهة من المحتمل أيضاً أنك تعرف أنه من خلال نصوص الاتصال تحفظ ميكروسوفت لجام محكم tight rein على مطوري ويندوز. وهي ليست بهذا التعقيد، وهي لأشياء أكثر من نصوص وسيطات مفصولة بفاصلة منقوطة. ولكن الوسيطات parameters التي يجب أن تضمن، وتنسيقها الدقيق هو قوام عنوان تفسيري stuff of legend. مستندات ميكروسوفت المضمنة MSDN مع الفيجوال

أستوديو توفر بعض الأمثلة لنصوص الاتصال، ولكن ليس مع الكثير من التفاصيل. يوفر مصدر ثالث <http://www.connectionstrings.com> أيضاً عدد ضخم من الأمثلة عن تنسيقات نصوص اتصال صحيحة.

نص الاتصال الذي سنستخدمه للاتصال إلى قاعدة بيانات المكتبة، ولحسن الحظ ليس معقد بإفراط. إذا استخدمت تسجيل دخول ميكروسوفت ويندوز الخاص بك للاتصال إلى قاعدة البيانات، النص التالي سيوفر احتياجاتك (كسطر واحد غير مفصول)

```
Data Source=instance_name;Initial Catalog=Library;Integrated Security=true
```

بالنسبة لنص الاتصال الذي عملت على بناءه للاتصال مع قاعدة بيانات المكتبة هو التالي:

```
Data Source=MHM\SQLEXPRESS;Initial Catalog=Library;Integrated Security=True
```

حيث يتم استبدال *instance_name* باسم بنسخة مخدم سكول الخاص بك أو مصدر البيانات، نص الاتصال الذي عملنا على بناءه سابقاً يستخدم "MYSYSTEM\SQLEXPRESS" كاسم مصدر بياناته.

لاستخدام معرف مستخدم سكول سرفر SQL Server user IDs وكلمة المرور passwords، جرب التنسيق التالي:

```
Data Source=instance_name;Initial Catalog=Library;User ID=sa;Password=xyz
```

بالطبع، استبدل معرف المستخدم user ID (*sa*) وكلمة المرور password (*xyz*) بأعداداتك الخاصة. إذا كنت تريد تضمين دعم MARS في اتصالك، أضف مكونات فاصلة منقوطة أخرى:

```
MultipleActiveResultSets=true
```

ملاحظة: خيارات أخرى لنص الاتصال Other connection string options تتيح لك الاتصال إلى ملف قاعدة بيانات مخدم سكول سرفر السريع SQL Server Express (SSE) بشكل مباشر، غير طريقة "نسخة المستخدم user instancing" (التي تستخدم غالباً مع قواعد بيانات التوزيع بنقرة واحدة ClickOnce-deployed databases)، وأعمل التعديلات الأخرى، على الرغم من أنها مبعثرة نوعاً ما، تستطيع إيجاد هذه الخيارات موثقة في مستندات MSDN التي تأتي مع الفيجوال أستوديو.

تأسيس الاتصال. Establishing the Connection.

استخدم نص الاتصال لإنشاء كائن SqlConnection، ومن ثم افتح الاتصال. أنشئ تطبيق نماذج ويندوز جديد في الفيجوال أستوديو وأضف أداة زر لسطح الفورم. انقر مزدوج على هذا الزر للوصول إلى معالج حدث النقر. ومن ثم أضف الكود التالي لمعالج الحدث هذا.

```
' ----- Assumes:
' Imports System.Data
Dim libraryDB As New SqlClient.SqlConnection("Data Source=mhm\SQLEXPRESS;" & "Initial
Catalog=Library;Integrated Security=true")
libraryDB.Open()
```

تأكد من تبديل "mhm" باسم النظام الخاص بك، فلاسم *mhm* المبين هو اسم النظام الخاص بي. إن مقطع الكود الكامل هذا يبدو سهلاً جداً وأسهل بالنسبة لي من الخطوات العشر أو الخمسة عشرة والتي عليك إتباعها سابقاً عند إعداد وتركيب الاتصال خلال الفيجوال أستوديو.

استخدام عبارات سكول Using SQL Statements

حالما يتم فتح الاتصال، تستطيع إصدار "اختر SELECT"، "تحديث UPDATE"، "حذف DELETE"، "إدراج INSERT" أو أية عبارة من عبارات لغة معالجة البيانات data manipulation language (DML) أو أية عبارة من عبارات لغة تعريف البيانات data definition language (DDL) المقبولة بواسطة قاعدة البيانات. يحضر الكائن SqlCommand عبارة سكول SQL الخاصة بك ليتم استخدامها بواسطة الاتصال المفتوح. إليك عبارة تعود بوصف المدخلة رقم 1 في جدول Activity.

```
SELECT FullName FROM Activity WHERE ID = 1
```

إنشاء كائن SqlCommand والذي يدور حول هذه العبارة سهل. إن المشيد لكائن SqlCommand يأخذ عبارة سكول SQL، راند كائن SqlConnection. أضف الكود التالي لنهاية معالج حدث نقر الزر السابق:

```
Dim sqlStatement As New SqlClient.SqlCommand("SELECT FullName FROM Activity WHERE ID = 1", libraryDB)
```

معالجة النتائج Processing the Results

الشيء الوحيد الباقي والذي عليك عمله هو تمرير عبارة سكول SQL لقاعدة البيانات، بواسطة الاتصال، واستخراج النتائج ككائن SqlDataReader. حالما تحصل على البيانات، عالج كل سجل باستخدام طريقة الكائن Read. إنك تصل إلى الحقول المستقلة بواسطة الاسم من خلال تجميع البنود Item الافتراضي. أضف هذا الكود الإضافي لنهاية معالج حدث نقر الزر السابق:

```
Dim sqlResults As SqlClient.SqlDataReader = sqlStatement.ExecuteReader()
sqlResults.Read()
MsgBox(CStr(sqlResults.Item("FullName")))
' هو الخاصية الافتراضية فإن التالي يعمل أيضاً Item بما أن
' MsgBox(CStr(sqlResults("FullName")))
' والاختصار التالي يعمل أيضاً
' MsgBox(CStr(sqlResults!FullName))
```

مع أخذ كل مقاطع الكود السابقة مع بعضها يعرض الرسالة المبينة في الشكل التالي.



عندما تنتهي. تأكد من أنك أغلقت جميع الاتصالات التي قمت بفتحها. لذلك أضف المقطع الصغير التالي من الكود إلى معالج حدث نقر الزر السابق:

```
sqlResults.Close()
```

```
libraryDB.Close()
```

تعديل البيانات Modifying Data

إن إجراء تغييرات إلى جداول قاعدة البيانات يتم بكتابة الكود تماماً مثل استخراج البيانات، ولكن ليس هناك حاجة "لقارئ بيانات سكول". وبدلاً من استخدام الطريقة `ExecuteReader`، استخدم الطريقة `ExecuteNonQuery` والتي لا تعود بنتائج.

```
Dim sqlStatement As New SqlCommand("UPDATE Activity SET FullName = 'Sleeps all day' & "
WHERE ID = 1", libraryDB)
sqlStatement.ExecuteNonQuery()
```

إن للسكول سرفر 2005 ميزة مريحة بحيث تعود بحقل مفرد من سجل جديد منشئ بواسطة العبارة "إدراج INSERT". إذا رجعت تصميم قاعدة بيانات مشروع المكتبة، ستري أن الحقول ID في عدة جداول يتم توليدها بشكل آلي.

تقليدياً، إذا أردت استخراج حقل ID بشكل مباشر إلى سجل جديد، عليك أولاً إدراج INSERT السجل، ومن ثم إجراء عبارة "اختر SELECT" منفصلة، العودة بحقل ID لسجل جديد:

```
INSERT INTO CodeSeries (FullName)
VALUES ('Children's Books')
SELECT ID FROM CodeSeries
WHERE FullName = 'Children's Books'
```

مخرجات سكول سرفر المدرجة تضم كل من هاتين العبارتين في إجراء وحيد:

```
INSERT INTO CodeSeries (FullName)
```

المخرجات المدرجة لحقل ID.

```
VALUES ('Children's Books')
```

عندما يكتمل الإدراج INSERT، يعمل سكول سرفر على إرجاع حقل ID كمجموعة نتيجة `result set`، تماماً كأنك أصدرت عبارة "اختر SELECT" منفصلة. إن طريقة `ExecuteScalar` لكان أمر سكول `SqlCommand` هي طريقة بسيطة لاستخراج قيمة مفردة من استعلام سكول.

```
sqlStatement = New SqlCommand("INSERT INTO CodeSeries (FullName) " & "OUTPUT INSERTED.ID VALUES "
('Children's Books')", libraryDB)
Dim newID As Integer = CInt(sqlStatement.ExecuteScalar())
```

مداولات قاعدة البيانات Database Transactions

يمكن إجراء المداولات *Transactions* "الجميع أو لا شيء" عبر عبارات سكول متعددة. حال البدء، إما يتم إصدار جميع عبارات سكول ضمن سياق المداولة المكتملة، أو لا يكتمل أي منها. إذا كان لديك 10 تحديثات `updates` لبيانات يجب أن يتم عملها، ولكن قاعدة البيانات فشلت بخمسة منهم، تستطيع التراجع `roll back` عن المداولة بالكامل. تعكس `reverses` قاعدة البيانات العبارات السابقة، وتعيد تخزين البيانات لما كانت عليه قبل أن تبدأ المداولة. (ولكن تحديثات من مستخدم آخر لا تتأثر بالتراجع). إذا نجحت جميع العبارات، تستطيع تنفيذ `commit` المداولة بالكامل، وجعل كل تغييرات المداولة ثابتة `permanent`. بالنسبة لقواعد بيانات سكول سرفر `SQL Server databases`، يتم إدارة المداولات من خلال موفر الكائن `SqlTransaction`. مثل ميزات `ADO.NET` الأخرى فهو سهل الاستخدام. تبدأ المداولة باستدعاء الطريقة `BeginTransaction` على الاتصال `connection`.

```
Public atomicSet As SqlCommand.SqlTransaction = libraryDB.BeginTransaction()
```

لتضمين عبارة سكول في المداولة، أسند كائن `SqlTransaction` إلى خاصية "المداولة Transaction" للكائن `SqlCommand`.

```
sqlStatement.Transaction = atomicSet
```

ومن ثم استدعي الطريقة المناسبة على الأمر. عندما تكتمل جميع الأوامر، استخدم الطريقة `Commit` للمداولة `transaction` لجعل التغييرات ثابتة `permanent`.

```
atomicSet.Commit()
```

إذا، بدلاً من الإدراج، احتجت إلى إجهاض المداولة `transaction`، استخدم طريقة "التراجع Rollback"

```
atomicSet.Rollback()
```

ADONET Entity Framework العمل. كينونة آدو دوت نت لإطار العمل.

الإصدار `ADO.NET` المضمن مع النسخة 3.5 من إطار عمل الدوت نت (وهي النسخة التي تأتي مع النسخة الحالية أي فيجوال أستوديو 2008) تتضمن مكونات جديدة: كينونة إطار عمل آدو دوت نت `ADO.NET Entity Framework`. جزء أداة نمذجة علاقة الكينونة `Part entity-relationship modeling tool`، جزء مولد الكود `part code generator`، وهذه التقنية الجديدة تساعد على صنع عروض بيانات منطقية `logical data views` للبيانات المخزنة في قاعدة البيانات العلائقية الخاصة بك أو مصدر البيانات.

تتيح لك كينونة إطار العمل `Entity Framework` تصميم ثلاث أنواع من الكائنات بالاستناد على بياناتك المخزنة: الكينونات `entities` (مشابهة للجداول)، العلاقات `relationships` (روابط قاعدة البيانات)، ومجموعات الكينونة `entity sets` (الكينونات المتعلقة ذات الصلة). كل نوع يتم تمثيله بواسطة كائنات تعرض أعضاء قاعدة بياناتك الجوهرية (القاعدية) في صيغة أكثر برمجية `in a more programmable fashion`. على سبيل المثال، تستطيع تصميم كينونة كائن تمثل جدول في قاعدة بياناتك، وأعضاء كائن يمكن أن تمثل الحقول `fields` في سجل وحيد `single record`. إنها تبدو مثل بعض الميزات المبنية ضمن كائن جدول البيانات `DataTable` لآدو دوت نت `ADO.NET`، إن ما يجعل كينونة إطار العمل مفيدة جداً هو التالي (1) فهو يشبك البيانات الفعلية بالعروض `views`، (2) إنه دعم لوراثة الكينونة `entity inheritance`، (3) ولها المقدرة على التصرف كموفر `ADO.NET`.

تشبيك البيانات Data mapping

تشبيك البيانات مشابه لإنشاء "عرض view" في قاعدة بيانات علائقية تقليدية، تستطيع إنشاء كينونات والتي يتم تركيبها ببعضها من امتداد سجلات مصادر متعددة عبر جداول قاعدة بيانات متعددة `multiple database tables`. وهذا يتضمن عروض آباء-أبناء للبيانات `parent-child views`، تستطيع إنشاء كينونة "طلبية Order" والتي تشير إلى كل من سجل الطلبية `order` الرئيسي وينود سطر "الطلبية Order" المتعددة المضمنة في الطلبية. منطقياً، ويتم معاملة هذه الكينونة الجديدة كعنصر استعلامي مفرد. عندما تطلب بيانات من خلال الكينونة الجديدة، ليس عليك تعليمها كيفية ربط وإقامة علاقة سببية ومنطقية بين القطع المصدرية المتنوعة للبيانات كل مرة.

وراثة الكينونة Entity inheritance

حالما يكون لديك كينونة معرفة، تستطيع تمديد تركيب الكينونة من خلال الوراثة. مثلاً، من المحتمل أنك تريد إنشاء كينونة تدعى InternalOrder معتمدة على كينونة "الطلبية Order" الأصلية، وإضافة أعضاء بحيث تقتفي أثر البيانات المخصصة لتلييات داخلية. يمكن لهذه الحقول الجديدة أن تكون جدول مخصص، يتم إطلاقه بواسطة إشعار منطقي Boolean على الجدول الرئيسي للطلبية. ولكن هذا ليس بالقضية المهمة: من المستطاع إخفاء الجميع بواسطة منطق الكينونة نفسها. عندما تطلب حالة (نسخة) من InternalOrder، فإنها ستعرف تماماً أنك تعني فقط النوع المشار إليه والمخصص داخلياً للطلبية، وليس الطلبية القياسية.

دعم مزود آدو دوت نت ADO.NET provider support

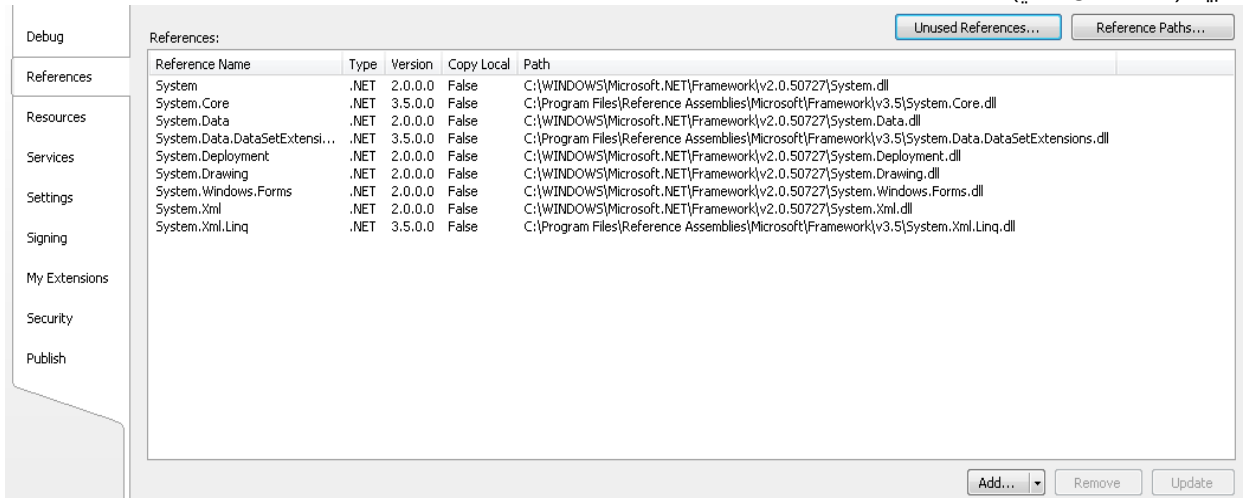
حالما تكون قد أنشأت كينوناتك ومنطق التشبيك المتعلق بها، تستطيع "الاتصال" إلى الكينونات وكأنه تم تخزينها في قاعدة بياناتك الخاصة. بدلاً من الاتصال إلى سكول سرفر والاستعلام عن الجداول مباشرة، فإنك تتصل إلى السياق المشبك والاستعلام على عرض المنطق الجديد لبياناتك. بعض هذه التقنية والتي سأناقشها في الفصل 17 لديها صيغة قريبة للكود المركز على كينونة إطار العمل للدوت نت، ولكن وبشكل خاص ستستهدف بشكل خاص سكول سرفر. إن كينونة إطار العمل لآدو دوت نت هي الطريقة الرئيسية لاتصال تقنية لينكو الجديدة للدوت نت إلى قواعد بيانات غير التابعة لميكروسوفت مثل أوراكل و DB2. لسوء الحظ، قبل عدة أشهر من إطلاق فيجوال أستوديو 2008، أعلنت ميكروسوفت أن كينونة إطار العمل Entity Framework لن تكون جاهزة في الوقت الحالي من أجل المنتجات الرئيسية (مثل أوراكل). فقد تم نقدها لبروغها كإصدار منفصل بعد عدة أشهر من إصدار الفيجوال أستوديو. ويمكن أن يكون أو لا يكون المنتج النهائي متاح عند قراءتك لهذه الأسطر.

مشروع Project

من المحتمل أن أكثر من 50% من كود مشروع المكتبة سيتضمن الوصول المباشر لقاعدة البيانات، أو معالجة البيانات المستخرجة من خلال ADO.NET. باستمرار إنشاء كائنات أمر وقارئ بيانات جديدة، على الرغم من البساطة، ولكن صعبة نوعاً ما على الأصابع. بما أن الكثير من الكود يتم تكراره، فإن الكود في مشروع هذا الفصل سيحاول التركيز على بعض الأساسيات، والكود المعياري boilerplate.

مرجع فضاءات أسماء البيانات Reference the Data Namespaces

لكود مشروع المكتبة مراجع لعدة فضاءات أسماء هامة في الدوت نت مثل System.Windows.Forms، و System.Windows.Forms، و Microsoft.VisualBasic. مهما يكن، ليس له حتى الآن مرجع لأي من فضاءات أسماء آدو دوت نت ADO.NET. (تذكر أن "التزويد بمرجع referencing" يعني إمكانية الوصول إلى مكتبة الربط الديناميكي للدوت نت NET DLL. في المشروع واستخدام أنواعها في كودك) قبل استخدام فضاءات الأسماء هذه في الكود، نحتاج إلى إنشاء مراجع (أو مؤشرات) لها. وهذا يتم عمله من خلال نافذة خصائص المشروع project properties، في تبويب "المراجع References". سترى قائمة بالمجمعات التي تم الإشارة إليها كمراجع سابقاً في تطبيقك (شاهد الشكل التالي).



إضافة مرجع جديد، انقر الزر "إضافة Add" الموجود تحت القائمة مباشرة، واختر مرجع، فإذا طلب منك نوع المرجع الذي سيضاف، على نموذج إضافة مرجع، سيكون تبويب دوت نت فعال سابقاً. ومن المدهش رؤية كيف تم تثبيت العديد من مجمعات الدوت نت على نظامك سابقاً. ولكن لا تجلس وتحقق كالأبله، اختر كل من System.Data و System.Xml من قائمة المكونات، ومن ثم انقر الزر موافق OK. وبالتالي ستتضمن قائمة المراجع الآن في خصائص المشروع كل مكتبات فضاء الأسماء المختارة.

نستطيع الآن الإشارة إلى فئات في فضاء الأسماء System.Data مباشرة، ولكن كتابة System.Data قبل كل استخدام لفئة ذات صلة بالبيانات data-related class سيسبب الملل. نستطيع نشر العبارات "Imports System.Data" لجميع الملفات في المشروع، ولكن توفر الفيجوال أستوديو حل أكثر مركزية. بما أنه ما يزال لديك تبويب المراجع References مفتوح، انظر للأسفل إلى مقطع "فضاءات الأسماء المجلوبة Imported namespaces". تشير قائمة الاختيار الضخمة إلى أي من فضاءات الأسماء سيتم إحضارها بشكل آلي على كامل تطبيقك، وفضاءات الأسماء هذه لا تحتاج إلى عبارات Imports مفصلة في كودك، ولكن يتصرف كودك وكأنك قد عملت على إضافتهم على أية حال. والآن ارجع إلى الأعلى واختر صندوق الاختبار بجانب مدخلة System.Data في هذه القائمة. ومن ثم أغلق نافذة خصائص المشروع project properties.

معظم الكود الجديد لهذا الفصل سيظهر في الملف General.vb، لذلك افتحه الآن. سنستخدم متغيران على مستوى المشروع (عام) لإدارة اتصال قاعدة بيانات رئيسية إلى قاعدة بيانات المكتبة. المتغير الأول LibraryDB، وهو كائن SqlConnection يستخدم نص اتصالنا لقاعدة connection string بيانات المكتبة. الكائن

HoldTransaction ذو الصلة بالمتغير السابق سيحفظ SqlConnection عندما تكون المعاملة قيد التنفيذ. أضف هذين السطرين إلى الوحدة البرمجية. لقد عملت على وضعهما قبل الطريقة الموجودة CenterText تماماً.

```
Public LibraryDB As System.Data.SqlClient.SqlConnection
Public HoldTransaction As System.Data.SqlClient.SqlTransaction
```

الاتصال بقاعدة البيانات Connecting to the Database

بما أن مشروع المكتبة سيعتمد كثيراً على قاعدة البيانات، سنبني كائن SqlConnection عندما يبدأ التطبيق بالتشغيل. ملاحظة:

الاحتفاظ بالاتصال على طول التطبيق يأتي ضد النصيحة التي قدمتها سابقاً أن اتصال قاعدة البيانات يجب أن يكون قصير الأمد. مهما يكن، من أجل الحفاظ على الكود بسيط قدر الإمكان من أجل أهداف التوضيح التدريجية، اخترت هذه المقاربة. وأيضاً لأن مشروع المكتبة تم تصميمه من أجل قاعدة صغيرة، فليس هناك حاجة لأن يكون قابل للتوسيع إلى حد كبير.

يحتوي الإجراء ConnectDatabase جميع الكود الضروري لإنشاء هذا الكائن. أما الآن، فلقد عملت على كتابة نص الاتصال connection string في هذا الروتين. في الفصول اللاحقة سنعمل على تضمين معلومات الاتصال كجزء من نظام الإعداد. أضف الروتين التالي إلى الوحدة البرمجية General. وتأكد من تغيير المرجع (المؤشر MYSYSTEM) لما هو ضروري على نظامك الخاص كما ذكرنا سابقاً.

```
Public Function ConnectDatabase() As Boolean
    ' ----- Connect to the database. Return True on success.
    Dim connectionString As String
    ' ----- Initialize.
    HoldTransaction = Nothing
    ' ----- Build the connection string.
    ' !!! WARNING: Hardcoded for now.
    connectionString = "Data Source=MYSYSTEM\SQLEXPRESS;" & "Initial Catalog=Library;Integrated
Security=true"
    ' ----- Attempt to open the database.
    Try
        LibraryDB = New SqlConnection(connectionString)
        LibraryDB.Open()
    Catch ex As Exception
        GeneralError("ConnectDatabase", ex)
        Return False
    End Try
    ' ----- Success.
    Return True
End Function
```

هذا الروتين هو الروتين الرئيسي للمشروع وهو حدث التطبيق العملي MyApplication_Startup، من ملف الكود المصدري لـ ApplicationEvents.vb، من أجل بناء كائن الاتصال لدى بداية تشغيل التطبيق، أضف الكود التالي لنهاية معالج الحدث هذا.

```
Private Sub MyApplication_Startup(ByVal sender As Object, ByVal e As
Microsoft.VisualBasic.ApplicationServices.StartupEventArgs) Handles Me.Startup
    ' ----- Connect to the database.
    If (ConnectDatabase() = False) Then
        Me.HideSplashScreen()
        e.Cancel = True
        Return
    End If
End Sub
```

عندما يخرج المستخدم من تطبيق المكتبة، فإن الكود سيستدعي الطريقة CleanupProgram للتخلص بشكل مناسب من كائن الاتصال. ارجع إلى الوحدة البرمجية General.vb وأضف تلك الطريقة.

```
Public Sub CleanupProgram()
    ' ----- استعداد للخروج من البرنامج.
    On Error Resume Next
    LibraryDB.Close()
End Sub
```

لتبسيط الأشياء، نستدعي هذا الروتين من معالج حدث التطبيق MyApplication_Shutdown، ارجع إلى ملف ApplicationEvents.vb. وأضف التالي:

```
Private Sub MyApplication_Shutdown(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Shutdown
    ' ----- أغلق قاعدة البيانات.
    CleanupProgram()
End Sub
```

التفاعل مع قاعدة البيانات Interacting with the Database

بعد أن تم تأسيس اتصال قاعدة البيانات، حان الوقت لعمل شيء ما به. تنفذ الروتينات الأربع المتمركزة في البداية الكثير من الكود الذي تم مناقشته سابقاً: إنشاء قارئات بيانات data readers وجداول tables، ومعالجة كود سكول SQL العام. أضف هذه الروتينات إلى الوحدة البرمجية كما يلي:

```
Public Function CreateDataTable(ByVal sqlText As String) As Data.DataTable
    ' ---- تقديم عبارة سكول للرجوع بمجدول بيانات.
    Dim dbCommand As SqlClient.SqlCommand
    Dim dbAdapter As SqlClient.SqlDataAdapter
    Dim dbTable As Data.DataTable

    ' محاولة تشغيل العبارة
    لاحظ أنه لا يوجد عملية إصطيد للأخطاء هنا، ويرجع للروتين المستدعي تثبيت اختبار الأخطاء.
    dbCommand = New SqlClient.SqlCommand(sqlText, LibraryDB)
    If Not (HoldTransaction Is Nothing) Then dbCommand.Transaction = HoldTransaction
    dbAdapter = New SqlClient.SqlDataAdapter(dbCommand)
    dbTable = New Data.DataTable
    dbAdapter.Fill(dbTable)
    dbAdapter = Nothing
    dbCommand = Nothing
    Return dbTable
End Function
```

```
Public Function CreateReader(ByVal sqlText As String) As SqlClient.SqlDataReader
    ' ---- تقديم عبارة سكول للرجوع بقارئ بيانات
    Dim dbCommand As SqlClient.SqlCommand
    Dim dbScan As SqlClient.SqlDataReader
```

```
' محاولة تشغيل العبارة، لاحظ أنه يتم إصطيد الأخطاء هنا.
' وتثبيت اختبار الخطأ راجع إلى الروتين المستدعي.
dbCommand = New SqlClient.SqlCommand(sqlText, LibraryDB)
If Not (HoldTransaction Is Nothing) Then dbCommand.Transaction = HoldTransaction
dbScan = dbCommand.ExecuteReader()
dbCommand = Nothing
Return dbScan
End Function
```

```
Public Sub ExecutesSQL(ByVal sqlText As String)
    ' تقديم عبارة سكول لتشغيلها.
    Dim DBCommand As SqlClient.SqlCommand
```

```
' محاولة تشغيل العبارة. لاحظ أنه لا يوجد إصطيد أخطاء هنا.
' وتثبيت اختبار الخطأ راجع إلى الروتين المستدعي.
DBCommand = New SqlClient.SqlCommand(sqlText, LibraryDB)
If Not (HoldTransaction Is Nothing) Then DBCommand.Transaction = HoldTransaction
DBCommand.ExecuteNonQuery()
End Sub
```

```
Public Function ExecutesSQLReturn(ByVal sqlText As String) As Object
    ' تقديم عبارة سكول، تشغيلها، والعودة بالنتيجة المناسبة
    Dim DBCommand As SqlClient.SqlCommand
```

```
' محاولة تشغيل العبارة. لاحظ أنه لا يوجد إصطيد للأخطاء هنا.
' وتثبيت اختبار الخطأ راجع إلى الروتين المستدعي.
DBCommand = New SqlClient.SqlCommand(sqlText, LibraryDB)
If Not (HoldTransaction Is Nothing) Then DBCommand.Transaction = HoldTransaction
Return DBCommand.ExecuteScalar()
End Function
```

```
Public Sub TransactionBegin()
    ' إنشاء مداولة جديدة لقاعدة البيانات.
    On Error Resume Next
    HoldTransaction = LibraryDB.BeginTransaction()
End Sub
```

```
Public Sub TransactionCommit()
    ' تجاهل حالة عدم وجود مداولة.
    On Error Resume Next
    If (HoldTransaction Is Nothing) Then Return
```

```
' ---- Commit the transaction.
```

```

'نفذ المداولة.'
HoldTransaction.Commit()
HoldTransaction = Nothing
End Sub

Public Sub TransactionRollback()
' ----- Ignore if there is no transaction.
' تجاهل حالة عدم وجود مداولة.
On Error Resume Next
If (HoldTransaction Is Nothing) Then Return

' ----- Rollback the transaction.
' التراجع عن المداولة.
HoldTransaction.Rollback()
HoldTransaction = Nothing
End Sub

```

لنشرح هذه الروتينات السبع:

دالة إنشاء جدول بيانات Function CreateTable

تقديم عبارة سكول، لاستخراج نتائجها من قاعدة البيانات، ووضع جميع النتائج في كائن جدول بيانات DataTable. يتصل الكائن SqlDataAdapter إلى SqlDataReader من خلال جدول البيانات DataTable.

دالة إنشاء قارئ Function CreateReader

تقديم عبارة سكول، لاستخراج نتائجها من قاعدة البيانات، والعودة بكائن قارئ بيانات سكول SqlDataReader المصاحب.

الإجراء تنفيذ سكول Sub ExecuteSQL

إرسال عبارة سكول إلى قاعدة البيانات للمعالجة.

الدالة تنفيذ العودة بسكول Function ExecuteSQLReturn

إرسال عبارة سكول إلى قاعدة البيانات للمعالجة، والعودة بقيمة لنتيجة مفردة.

إجراء بدء المداولة Sub TransactionBegin

بدء مداولة جديدة.

إجراء تنفيذ المداولة Sub TransactionCommit

تنفيذ المداولة، وجعل جميع التغييرات ثابتة permanent.

إجراء مسار التراجع عن المداولة Sub TransactionRollback

السير بطريق الرجوع عن المداولة، وعدم عمل أي تغييرات والتي كانت جزء من المداولة.

ولا واحد من هذه الروتينات يتضمن كود معالجة خطأ خاص بها، فهي إما تقمع (تخمد) الأخطاء بالعبارة "عند الخطأ تابع التالي"، أو الاعتماد على الروتين المستدعي لاصطياد الأخطاء. وهذا يتيح للروتين المستدعي القيام بفعل معين بالاعتماد على نوع الخطأ الناتج. جميع هذه الروتينات متشابهة إلى حد بعيد مع بعضها البعض. ففي كود الروتين CreateReader، جزء واحد مهم وهو استخدام الكائن HoldTransaction عند تكون المداولة قيد التنفيذ.

```

If Not (HoldTransaction Is Nothing) Then dbCommand.Transaction = HoldTransaction

```

معالجة قيم البيانات. Processing Data Values.

يتضمن بناء عبارات سكول يدوياً الكثير من معالجة النصوص، زائد المعالجة الشرطية لها عدة مرات عند احتمال فقدان البيانات. على سبيل المثال، إذا أردت تخزين قيم نصية في قاعدة البيانات، عليك تحضيرها للاستخدام من قبل عبارة سكول (معالجة خاصة من أجل علامات الاقتباس المفردة)، ولكن إذا كانت قيمة النص ذات طول صفري zerolength، فإنك تمرر الكلمة في العبارة عوضاً عن ذلك. جميع هذه التحضيرات للبيانات يمكن أن تعرقل كودك، لذلك لما لا تكثفها (تجعلها مركزية)؟ الروتينات الثمانية في هذا المقطع إما تحضر البيانات للاستخدام في عبارات سكول، أو تضبط البيانات المستخرجة للاستخدام في التطبيق.

```

Public Function DBCombo(ByRef whichField As ComboBox) As String
'تحضير قيمة صندوق مركب للاستخدام في قاعدة البيانات
Dim listItem As Library.ListItemData
'الحصول على البيانات المخزنة لهذا البند.
If (whichField.SelectedItem Is Nothing) Then Return "NULL"
listItem = CType(whichField.SelectedItem, Library.ListItemData)

```

```

'(-1) كقيمة لبند ما لديها معنى خاص: غير مختارة
If (listItem.ItemData = -1) Then
Return "NULL"
Else
Return CStr(listItem.ItemData)
End If
End Function

```

```

Public Function DBDate(ByVal origText As String) As String
' ----- Prepare a date for insertion in a SQL statement.
If (Trim(origText) = "") Then
Return "NULL"

```

```

ElseIf (IsDate(origText)) Then
    Return "" & Format(CDate(origText), "d-MMM-yyyy") & ""
Else
    Return "NULL"
End If
End Function

```

```

Public Function DBDate(ByVal origDate As Date) As String
' ----- Prepare a date for insertion in a SQL statement.
Return "" & Format(origDate, "d-MMM-yyyy") & ""
End Function

```

```

Public Function DBGetDecimal(ByRef dataField As Object) As Decimal
' ----- Return the decimal equivalent of an optional database field.
If (IsDBNull(dataField) = True) Then
    Return 0@
Else
    Return CDec(dataField)
End If
End Function

```

```

Public Function DBGetInteger(ByRef dataField As Object) As Integer
' ----- Return the integer equivalent of an optional database field.
If (IsDBNull(dataField) = True) Then
    Return 0
Else
    Return CInt(dataField)
End If
End Function

```

```

Public Function DBGetText(ByRef dataField As Object) As String
' ----- Return the text equivalent of an optional database field.
If (IsDBNull(dataField) = True) Then
    Return ""
Else
    Return CStr(dataField)
End If
End Function

```

```

Public Function DBNum(ByVal origText As String) As String
' ----- Prepare a number for insertion in a SQL statement.
If (Trim(origText) = "") Then
    Return "NULL"
Else
    Return Trim(origText)
End If
End Function

```

```

Public Function DBText(ByVal origText As String) As String
' ----- Prepare a string for insertion in a SQL statement.
If (Trim(origText) = "") Then
    Return "NULL"
Else
    Return "" & Replace(origText, "", "'') & ""
End If
End Function

```

DBCombo

يأخذ الكود العددي numeric code المصاحب لبيد مختار في أداة صندوق مركب ComboBox ويعود به كنص. إذا لم يكن هناك نص قيد الاختيار أو القيمة هي 1-، يعود الروتين بـ"بدون قيمة NULL".

DBDate(String)

تقديم نص ما يحتوي على تاريخ منسق، والعودة بتاريخ جاهز للاستخدام في عبارة سكول.

DBDate(Date)

تقديم قيمة تاريخ صحيحة، والعودة بتاريخ نصي جاهز للاستخدام في عبارة سكول.

DBGetDecimal

العودة بعدد عشري من مجموعة نتيجة، حتى ولو كان الحقل يحوي قيمة "بدون قيمة NULL".

DBGetInteger

يعود بعدد أنتغر من مجموعة نتيجة، حتى ولو كان الحقل يحوي قيمة "بدون قيمة NULL".

DBGetText

يعود بنص من مجموعة نتيجة، حتى ولو كان الحقل يحوي قيمة "بدون قيمة".

DBNum

يحضر عدد للاستخدام في عبارة سكول .

DBText

يحضر نص للاستخدام في عبارة سكول. راجع الكود في الأعلى من أجل هذا الروتين. فالنص في عبارة سكول يجب أن يكون محاط بعلامات اقتباس مفردة، وأي علامة اقتباس لكي يتم تضمينها يجب أن تضاعف.

التركيب (الإعداد) على مستوى النظام System-Level Configuration

المقطع الأخير من الكود يدعم التحديث والاستخراج السريع لقيم نظام التركيب الواسع المخزنة في جدول قيم النظام SystemValue لقاعدة بيانات المكتبة. يعود الروتين GetSystemValue بالإعداد الحالي لقيمة التركيب عند التزويد بقيمة الاسم value name. يحدث SetSystemValue (أو يضيف، عند الحاجة) قيمة تركيب مسماة (أو محددة). كل من هذين الروتينان يظهران في الوحدة البرمجية General.

```
Public Function GetSystemValue (ByVal valueName As String) As String
    ' ----- Return the data portion of a system value name-data pair.
    Dim sqlText As String
    Dim returnValue As String
    ' ----- Retrieve the value.
    returnValue = ""
    sqlText = "SELECT ValueData FROM SystemValue WHERE UPPER(ValueName) = " & _
        DBText (UCase (valueName))
    Try
        returnValue = DBGetText (ExecuteSQLReturn (sqlText))
    Catch ex As Exception
        GeneralError ("GetSystemValue", ex)
    End Try
    Return returnValue
End Function

Public Sub SetSystemValue (ByVal valueName As String, ByVal valueData As String)
    ' ----- Update a record in the SystemValue table.
    Dim sqlText As String
    Dim dbInfo As SqlConnection.SqlDataReader
    Try
        ' ----- See if the entry already exists.
        sqlText = "SELECT COUNT(*) FROM SystemValue WHERE UPPER(ValueName) = " & _
            DBText (UCase (valueName))
        If (CInt (ExecuteSQLReturn (sqlText)) > 0) Then
            ' ----- Value already exists.
            sqlText = "UPDATE SystemValue SET ValueData = " & DBText (valueData) & _
                " WHERE UPPER(ValueName) = " & DBText (UCase (valueName))
        Else
            ' ----- Need to create value.
            sqlText = "INSERT INTO SystemValue (ValueName, ValueData) VALUES (" & _
                DBText (valueName) & ", " & DBText (valueData) & ")"
        End If
        ' ----- Update the value.
        ExecuteSQL (sqlText)
    Catch ex As System.Exception
        GeneralError ("SetSystemValue", ex)
    End Try
End Sub
```

الروتين GetSystemValue واضح. فهو يستخرج ببساطة قيمة مفردة من قاعدة البيانات.

على الروتين SetSystemValue أولاً اختبار فيما إذا قيمة التركيب (الإعداد configuration values) التي ستستخدم للتحديث موجود مسبقاً في قاعدة البيانات. فإذا كانت كذلك (أي موجودة) فإنه يعدل السجلات. وإلا، فإنه يضيف سجل كامل جديد. لتحديد فيما إذا كان سجل موجود أم لا، فالمطلوب إحصاء السجلات التي تطابق اسم قيمة النظام system value name. ويستعلم (يستفهم) هذا الروتين عن قاعدة البيانات من خلال الطريقة ExecuteSqlReturn، والتي تعود بقيمة وحيدة من استعلام query. في هذه الحالة، القيمة هي عدد السجلات المتطابقة.

```
sqlText = "SELECT COUNT(*) FROM SystemValue WHERE UPPER(ValueName) = " & _
    DBText (UCase (valueName))
```

```
If (CInt(ExecuteSQLReturn(sqlText)) > 0) Then
```

كما ترى استخدام الـ `GetSystemValue` سهل، لذلك لنستخدمه الآن. ارجع إلى معالج حدث `MyApplication_Startup` في `ApplicationEvents.vb` وأضف التالي إلى نهاية معالج الحدث:

```
' ----- Check the database version.
    Dim productionDBVersion As String
    productionDBVersion = Trim(GetSystemValue("DatabaseVersion"))
    If (CInt(Val(productionDBVersion)) <> UseDBVersion) Then
        MsgBox("The program cannot continue due to an incompatible database. " &
            "The current database version is '" & productionDBVersion &
            "'. The application version is '" & UseDBVersion & "'.",
            MsgBoxStyle.OkOnly Or MsgBoxStyle.Critical, ProgramTitle)
        CleanupProgram()
        Me.HideSplashScreen()
        e.Cancel = True
        Return
    End If
```

بين فترة وأخرى، أجد من الضروري تعديل تركيب قاعدة البيانات إلى حد أن الإصدارات القديمة من التطبيق إما لم تعد تعمل، أو أن تسبب البيانات الرئيسية وجع راس. ولمنع هذا، عملت على إضافة إعداد إصدار قاعدة البيانات `DatabaseVersion`، واستخدام مقطع هذا الكود من أجل الاختبار عليه. فإذا لم يوافق البرنامج إصدار قاعدة البيانات المتوقع، فإنه سيرفض التشغيل.

الأمن Security

الأسرار أشياء ممتعة، فمع البلايين من الناس على الكوكب، لا يوجد قحط في الأحداث والقصص الحقيقية، ولكن ولا واحدة منها تحفظ متعنتاً إذا لم يكن هناك سر يستكشف في مكان ما. العديد من الكتب عملت على تضمين الكلمة " Secret " في عناوينها لجعلها وجعل مواضيعها أكثر إثارة، عناوين مثل " أسرار الطبخ الياباني Japanese Cooking Secrets " .

في هذه الفترة increasingly permissive moral era من تكاثف المعلومات information overload والمعايير الأخلاقية المتساهلة يافراط increasingly permissive moral standards على التلغز، تبدو الأسرار نادرة. ولكن كل واحد لديه معلومات مهمة يحتاج حفظها عن الآخرين، وهذا يتضمن مستخدمي برامجك. لحسن الحظ، برامج الدوت نت والبيانات المتعلقة بها يمكن أن تكون آمنة عند الحاجة، إذا استخدمت ميزات الأمان المتاحة لك في إطار عمل الدوت نت.

ميزات الأمان في الدوت نت Security Features in .NET

يتضمن الأمان في الدوت نت العديد من الميزات، ولكنها تقع بشكل عام تحت ثلاث مناطق رئيسية:

الأمن الداخلي Internal security

الغنائم Classes وأعضاء الغنائم في الدوت نت يمكن أن يتم حمايتها بواسطة الأمان المعتمد على المستخدم أو الأمان المعتمد على القواعد. ويتواجد أمن الوصول للكود (CAS) Code Access Security لحفظ المستخدمين الغير مرخص بهم unauthorized من الوصول لميزات المكتبات الفعالة powerful libraries التابعة للدوت نت. و فقط هؤلاء المستخدمين الذين لديهم مجموعة قليلة أو خاصة من الحقوق يمكنهم استخدام هذه الميزات المحمية.

الأمن الخارجي External security

بما أن أي واحد يمكنه أن يطور وينشر تطبيق دوت نت، فمن الهام حماية مصادر النظام من كود مؤذ. وهذه قضية كبيرة، وخاصة مع التقارير الذاهبة (الخارجية) من الهكرز (القراصنة) والتي تستفيد من مشاكل "احتياج الجدار" في البرمجيات الصادرة من ميكروسوفت والباينيين الآخرين. وكما أن "أمن الوصول للكود (CAS) Code Access Security" يحفظ الكود من الوصول لميزات خاصة بالفئة، وهذا الأمان يتفاعل مع نظام التشغيل لحفظ الكود المؤذي (المحتال) من الوصول لبعض أو كل الملفات والأدلة، مدخلات التسجيل، مصادر شبكة العمل، محيطات أجهزة الكمبيوتر المركبة عليه، أو مجمعات الدوت نت الأخرى المعتمدة على سياسات الأمان السارية (النافذة المفعول in-effect security policies).

أمن البيانات Data security

البرامج ومصادر الكمبيوتر ليست الشيء الوحيد الذي يحتاج الحماية. بعض المستخدمين ذوي الألفاظ الطنانة يفكرون أن بياناتهم ثمينة وهامة بحيث تستحق أن يتم حمايتها من خلال وسائل برمجية خاصة. التشفير Encryption والتوقيعات الرقمية digital signatures وميزات تشفير cryptographic أخرى توفر دعم خاص تحتاجه مثل هذه البيانات.

ولأن مشروع المكتبة يتفاعل مع مصادر خارجية رئيسية - قاعدة بيانات سكول سرفر - فهي تتعامل مع قضايا الأمان الخارجي، وأيضاً بطريقة غير مباشرة مع ADO.NET وسياسات أمن الأنظمة system security policies. ولأن الكتاب يركز على توزيع تطبيق عمل نموذجي، فهذا الفصل لن يناقش كل من قضايا الأمان الخارجي والداخلي، ولكنه سيركز على مواضيع أمن البيانات، وخاصة تشفير البيانات.

التشفير وعلم التشفير Cryptography and Encryption

استبداد سر، لدى شخص آخر هي قضية، والتأكد من أن سر ما يأتي من شخص آخر موثوق هي قضية أخرى أيضاً. التأكد من أنني أحصل على أفضل معاملة بالنسبة لتأمين السيارة هي قضية أخرى مختلفة بالكامل.

من الواضح، أمن البيانات هي أكثر من حفظ قطعة من البيانات محمية عن أعين المتطفلين. وليس ما يهمنا فقط أعين المتطفلين.

حفظ الأسرار Keeping Secrets

عندما يفكر الناس بشأن التشفير encryption والأمان البيانات data security، بشكل عام يركزون على سمة "حفظ الأسرار keeping secrets". إمكانية كتابة محتوى بشكل كودي (تحويل الأحرف إلى رموز)، وحفظها بعيدة عن الخصم adversary، مع ذلك تبقى قابلة لفك التشفير من قبلك أو إرفاقها في وقت متأخر هو هام. مجال تقنيات التشفير بدءاً من الانحرافات عن اللغة البسيطة وتبديل الشفرة (استبدال الأحرف letter substitutions، المستخدمة في الألغاز الرمزية (المكتوبة بالرموز cryptogram puzzles)) إلى أنظمة التكويد النوعية للأجهزة التفسيرية enigma-machine-quality encoding systems. البرمجيات التي تمكن التشفير هي جزء من تعلمنا اليومي الآن. عندما تعمل كرت اعتماد (بطاقة ائتمان) شرائية من مواقع انترنت، فإن الفرصة جيدة نسبياً لأن تكون معلومات بطاقة الاعتماد خاصتك يتم تشفيرها وتحويلها إلى دقة سرية في 128- بت.

نموذجياً طرق التشفير encryption تعمل على استخدام واحد أو أكثر من المفاتيح، زائد توحيد دوال العنونة بالكود hashing functions وخوارزميات التشفير encryption algorithms، لتحويل محتوى دقيق sensitive content إلى صيغة ليس من السهل الوصول لها دون المفتاح المناسب أو الأصلي. التكويد المتماثل Symmetric cryptography هو الاسم المستخدم لطرق التشفير encryption التي تستخدم مفتاح سري مفرد single secret key. يعرف أيضاً التشفير بالمفاتيح العامة Public-key encryption بالتشفير الغير متماثل asymmetric cryptography - ويستخدم زوج من المفاتيح لتشفير encrypt وتشفير decrypt البيانات. يمكن أن يتم إعطاء المفاتيح العامة لأي واحد يهتم بالاتصال معك بشكل سري. حتى من الممكن أن تعطي هذه المفاتيح لمنافيسك، فهي عامة.

المفاتيح الخاصة private key المناسبة يتم حفظها بشكل آمن لتستخدمها أنت، ولن تريها لأي كان. المحتوى المشفر باستخدام واحد من المفاتيح (وخوارزمية التشفير) يمكن أن يتم فك تشفيره فيما بعد باستخدام المفتاح الآخر. إذا ما شفر صديقك بعض المعلومات باستخدام مفتاح عام، لا يمكن لأي أن يكون قادر على فكه ما عداك أنت، وسيطلب مفتاح خاص بك. تستطيع أيضاً تشفير البيانات بواسطة مفتاحك الخاص، ولكن أي كان يستطيع فك شفرته بواسطة المفتاح العام. سنرى استخدام هذا الإجراء الغير آمن ظاهرياً بعد قليل.

استقرار البيانات Data Stability

يساعد تشفير البيانات ضمان تكامل مقطع من البيانات، حتى ولو كانت تلك البيانات غير مشفرة. إذا أرسلت لشخص أيمل أثناء عاصفة رعدية، توجد وبشكل خاص فرصة لأن يتم تبديل بعض أو كل محتوى الرسالة الالكترونية بشكل الكتروني قبل وصولها إلى المستقبل. دعنا نقول أن بعض الكهرباء الساكنة في خطوط شبكة النقل حدث لأن تسبب مضاعفة جملة واحدة. كيف ستعرف فيما إذا كان المؤلف يحاول صنع نقطة تذكير أو ببساطة شائبة كمبيوتر؟.

تضمن "اختبار الإضافة checksum" مع المحتوى يساعد في التعرف على مشاكل البيانات أثناء النقل. يدعى "اختبار الإضافة checksum" في بعض الأحيان "قيمة التكرار hash value" - تأخذ المحتوى الأصلي وتممره إلى دالة تولد قيمة مختصرة short value تمثل البيانات الأصلية. دوال "اختبار الإضافة" (أو خوارزميات التكرار hashing algorithms) حساسة جداً حتى للتغيرات في البايث المفرد ضمن المحتوى، فيما إذا تم تبديل ذلك البايث المفرد، تغير موقعه، أضيف، أو تمت إزالته من البيانات الأصلية. بتوليد "اختبار الإضافة" قبل وبعد إرسال البيانات، تستطيع التثبت فيما إذا تغير المحتوى بالكامل أثناء النقل.

تمثل "اختبارات الإضافة" تشفير غير موجه للبيانات الأصلية. من غير الممكن استخدام "اختبار الإضافة" للحصول على محتوى البيانات الأصلية. وهذا صحيح، إذا، وبما أن الهدف من "اختبار الإضافة" ليس لتسليم المحتوى بشكل سري، ولكن تسليمها غير متغيرة. التشفير الثنائي الاتجاه Bidirectional encryption هو ما تحدثت عنه في مقطع "حفظ الأسرار Keeping Secrets". إذا كان لديك مفتاح صحيح و خوارزمية صحيحة، يعيد التشفير الثنائي الاتجاه bidirectional encryption تخزين المحتوى الأصلي من المحتوى المشفر.

إثبات الهوية Identity Verification

لنقول أنك استقبلت رسالة الكترونية تقول شيء ما، كيف تعرف أن هذه الرسالة موثوقة، أو حقيقة من مرسل معين؟ في هذه الحالة، المحتوى لوحده يجب أن يبرهن استحفاقه الثقة trustworthy. ولكن إذا أردت حقاً التحقق من المصدر، ومن غير المتاح الاتصال مع من بعث الرسالة، تستطيع توظيف توافيق رقمية digital signatures لإثبات هوية المرسل.

طريقة وحيدة لاستخدام التوافيق الرقمية التي توظف تشفير بالمفاتيح العامة لنقل رسالة أو كلمة مرور متفق عليها، وتمرر المحتوى المشفر على طول مع رسالة الكترونية أكبر. على سبيل المثال، يمكن أن يشفر المرسل النص "أنا المرسل فلان" باستخدام مفتاحه الخاص. عندما تستقبل الإيميل، يمكنك فك تشفير التوقيع الرقمي باستخدام مفتاح المرسل العام. إذا أنتج فك تشفير الرسالة "أنا المرسل فلان"، ستعلم أن تلك الرسالة فعلية، وفي الحقيقة تأتي من المرسل فلان.

التشفير في الدوت نت. Encryption in .NET

تشفير البيانات وميزات الأمن المضمنة مع الدوت نت تظهر في فضاء الأسماء System.Security.Cryptography. معظم الفئات في فضاء الأسماء هذا تنفذ خوارزميات تشفير معروفة جيداً ومتنوعة والتي تم قبولها بواسطة المنظمات والحكومات ك معايير تشفير معتمدة. على سبيل المثال الفئة DESCryptoServiceProvider توفر ميزات معتمدة على خوارزمية معيار تشفير البيانات Data Encryption Standard (DES) algorithm. الخوارزمية المطورة أصلاً بواسطة شركة IBM في منتصف 1970.

علم التكويد (التشفير) المتماثل Symmetric Cryptography

يستخدم علم التكويد المتماثل مفتاح سري مفرد single secret key لتشفير وفك تشفير مقطع بيانات. على الرغم من أن هذه الخوارزميات وعلى الأغلب سريعة جداً (عند مقارنتها مع التكويد غير المتماثل asymmetric cryptography) الحاجة لتوفير مفتاح سري كامل للأخرين لمشاركة البيانات يمكن أن تجعلها (البيانات) ملازمة لأمن أقل. ما يزال كافي بالنسبة لعدة تطبيقات، "التشفير بمفتاح سري secret key encryption". يتضمن إطار عمل الدوت نت دعم لأربع خوارزميات تشفير متماثل:

• معيار تشفير البيانات Data Encryption Standard (DES)، شفرة مقطع 56-bit مع دعم رئيسي من خلال الفئة DESCryptoServiceProvider. هذه الخوارزمية بشكل عام آمنة، ولكن تبعاً لحجم مفتاحها الصغير (المفاتيح الأصغر أكثر سهولة في الفضح)، وهي غير مناسبة من أجل البيانات العالية الحساسية.

• شفرة ريفست رقم 2 RC2 (Rivest Cipher number 2)، شفرة مقطع 56-bit مع دعم رئيسي من خلال الفئة RC2CryptoServiceProvider.

• ريجدل Rijndael (مشتقة من اسمي مصمميها، ديمين Daemen وريجن Rijmen) شفرة مقطعية ذات بت متغير (بين 128 إلى 256 بت) block cipher مع دعم رئيسي من خلال الفئة RijndaelManaged. ويتم تعليقها إلى خوارزمية مشابهة مسماة "معيار تشفير متقدم Advanced Encryption Standard (AES)"، وهي الخوارزمية الأكثر أمناً من خوارزميات مفتاح السر المزودة بواسطة الدوت نت.

• معيار تشفير البيانات الثلاثي Triple DES، شفرة مقطعية تستخدم الخوارزمية DES algorithm المضمنة ثلاث مرات لتوليد نتيجة أكثر أمناً. مع دعم رئيسي من خلال الفئة TripleDESCryptoServiceProvider. على الرغم من أنها أكثر أمناً من DES البسيطة، فما تزال أقل تحصيلاً vulnerable من ريجدل Rijndael أو القياسي AES.

فئات "المزود provider" المتنوعة هي أدوات يجب أن يتم استخدامها مع بعضها ومع فئات التكويد لتعمل بشكل مناسب. على سبيل المثال، هذه العينة من الكود (المعتمدة على كود موجود في توثيق MSDN) يستخدم الفئة DESCryptoServiceProvider والفئة CryptoStream. كلاهما أعضاء من System.Security.Cryptography، معاً لتشفير وفك تشفير مقطع نصي:

```
Imports System
```

```
Imports System.IO
Imports System.Text
Imports System.Security.Cryptography
Class CryptoMemoryStream
    Public Shared Sub Main()
        ' ----- Encrypt then decrypt some text.
        Dim key As New DESCryptoServiceProvider
        Dim encryptedVersion() As Byte
        Dim decryptedVersion As String
        ' ----- First, encrypt some text.
        encryptedVersion = Encrypt("This is a secret.", key)
        ' ----- Then, decrypt it to get the original.
        decryptedVersion = Decrypt(encryptedVersion, key)
    End Sub
    Public Shared Function Encrypt(ByVal origText As String,
        ByVal key As SymmetricAlgorithm) As Byte()
        ' ----- Uses a cryptographic memory stream and a
        ' secret key provider (DES in this case)
        ' to encrypt some text.
        Dim baseStream As New MemoryStream
        Dim secretStream As CryptoStream
        Dim streamOut As StreamWriter
        Dim encryptedText() As Byte
        ' ----- A memory stream just shuffles data from
        ' end to end. Adding a CryptoStream to it
        ' will encrypt the data as it moves through
        ' the stream.
        secretStream = New CryptoStream(baseStream,
            key.CreateEncryptor(), CryptoStreamMode.Write)
        streamOut = New StreamWriter(secretStream)
        streamOut.WriteLine(origText)
        streamOut.Close()
        secretStream.Close()
        ' ----- Move the encrypted content into a useful
        ' byte array.
        encryptedText = baseStream.ToArray()
        baseStream.Close()
        Return encryptedText
    End Function
    Public Shared Function Decrypt(ByVal encryptedText() As Byte,
        ByVal key As SymmetricAlgorithm) As String
        ' ----- Clearly, this is the opposite of the
        ' Encrypt() function, using a stream reader
        ' instead of a writer, and the key's
        ' "decryptor" instead of its "encryptor."
        Dim baseStream As MemoryStream
        Dim secretStream As CryptoStream
        Dim streamIn As StreamReader
        Dim origText As String
        ' ----- Build a stream that automatically decrypts
        ' as data is passed through it.
        baseStream = New MemoryStream(encryptedText)
        secretStream = New CryptoStream(baseStream,
            key.CreateDecryptor(), CryptoStreamMode.Read)
        streamIn = New StreamReader(secretStream)
        ' ----- Move the decrypted content back to a string.
        origText = streamIn.ReadLine()
        streamIn.Close()
        secretStream.Close()
        baseStream.Close()
        Return origText
    End Function
End Class
```

يضم هذا الكود فئة DES encryption مع *stream*، أداة مشتركة في تطبيقات الدوت نت لنقل البيانات من حالة أو موقع لآخر. ("الجداول) أو التجميعات) Streams هي طريقة رئيسية مستخدمة لقراءة وكتابة الملفات (files) الجداول Streams ليست بأداة صعبة الاستخدام، ولكن ماتزال تبدو معقد قليلاً. لماذا لا تتضمن الفئة DESCryptoServiceProvider طرق تشفير Encrypt وفك التشفير Decrypt ببساطة؟ هذا سؤال، على

الأقل، إنني متأكد من أن لديها شيء ما يجب عمله مع حفظ الفئة "شاملة generic" للاستخدام في بيئات متعددة. ما يزال، هذا الكود كثيف، من المؤكد أنه أكثر سهولة من كتابة كود التشفير بنفسه. وهو عام بشكل كافي بحيث أستطيع مبادلة خوارزميات مفتاح سري دون تغيير كبير في الكود.

التكويد غير المتماثل Asymmetric Cryptography

في تكويد المفتاح السري، تستطيع استخدام أي مفتاح قديم تريده لدعم عملية التشفير وفك التشفير. طالما أنك تحفظه سراً، في الحقيقة محتوى المفتاح نفسه ليس مهم كثيراً. لا يمكن قول المثل على التكويد الغير متماثل (المفتاح العام). لأن المفاتيح المفصولة يتم استخدامها لتشفير وفك تشفير البيانات، المفاتيح الخاصة والعامية النوعية يجب أن يتم صنعها بشكل خاص كزوج. لا تستطيع اختيار فقط مفاتيح خاصة وعامية عشوائية وتأمل أن تعمل مع بعضها. المكونات المستخدمة لدعم التكويد الغير متماثل تتضمن "المولدات generators" والتي تصدر أزواج مفاتيح خاصة وعامية. حالما يتم التوليد، هذه المفاتيح يمكن أن يتم استخدامها في كودك لتقيع mask البيانات الحساسة. وتبعاً لحجم المفتاح الأكبر، فمن الصعب بالنسبة لكل واحد أن يدخل تعديل على بياناتك المشفرة.

تشفير المفتاح العام مشهور notoriously ببطئته، فهو يأخذ دائماً يوم لكتابة كود كميات ضخمة من البيانات باستخدام المفتاح المصدر. بسبب بطء sluggish أداء التشفير الغير المتماثل، فالعديد من أنظمة أمن البيانات تستخدم جمع من تشفير مفتاح عام ومفتاح سري لحماية البيانات. التراخيص الأولية تحدث مع عمليات المفتاح العام، ولكن حالما تفتح قناة الأمن، البيانات الممررة بين الأنظمة التي تحصل على التشفير باستخدام طرق المفتاح السري الأسرع. تتضمن الدوت نت فنتي تكويد تفي برغبتك في التشفير وفك التشفير:

خوارزمية التوقيع الرقمي (DSA) *Digital Signature Algorithm (DSA)*، خوارزمية صممت بواسطة الحكومة الأمريكية لاستخدام التوقيعات الرقمية، مع دعم رئيسي من خلال الفئة DSACryptoServiceProvider.

خوارزمية RSA (موجدها Ron Rivest و Adi Shamir و، و Adleman)، خوارزمية غير متماثلة الأمن أقدم مع دعم رئيسي من خلال الفئة RSACryptoServiceProvider.

سوف نستخدم التشفير الغير متماثل في مشروع المكتبة، ولكن ليس حتى آخر فصل. حتى ذلك الحين، لن أدخل بالكثير من التفاصيل حول أعمال التشفير الغير متماثل.

البعثرة (أو الخلط) Hashing

على الرغم من أن خوارزميات البعثرة (خوارزميات العنونة بالكود (أو الرموز)) hashing algorithms لا تمنحك إمكانية تشفير وفك تشفير البيانات كما ترغب، ولكنه مفيدة في دعم الأنظمة التي تؤمن وتحقق من محتوى البيانات. سننجز بعض البعثرة على البيانات في كود المشروع من أجل هذا الفصل، لذلك ابق يقظ.

ييجاد حل خوارزمية البعثرة سهل. فهي أخذت أفضل عقول وكالة الأمن العالمية ومعهد ولاية ماسشتس للتكنولوجيا لاختراع أنظمة التشفير بمفتاح عام ومفتاح خاص، ولكنك تستطيع تطوير خوارزمية البعثرة بعدة دقائق فقط. إليك خوارزمية بسيطة قمت بعملها:

```
Public Function HashSomeText(ByVal origText As String) As Long
    ' ----- Create a hash value from some data.
    Dim hashValue As Long = 0&
    Dim counter As Long
    For counter = 1 To Len(origText)
        hashValue += Asc(Mid(origText, counter, 1))
        If (hashValue > (Long.MaxValue * 0.9)) Then
            hashValue /= 2
        End If
    Next counter
    Return hashValue
End Function
```

في الكود عملت فقط على جمع قيم الأسكي لكل حرف في السلسلة الحرفية للنص، وعملت على إعادة النتيجة. وقد عملت اختبار في الحلقة للتأكد من أنني لا أريد عن 90% من قيمة الطول الأعظمي، ولا أريد تجاوز المتغير hashValue وتوليد خطأ. على الرغم من أن الدالة HashSomeText تعمل على توليد تمثيل مبعثر hashed representation للبيانات المدخلة، لديها أيضاً بعض العيوب deficiencies:

• من السهل جداً التخمين من قيمة التكرار hash فيما إذا المحتوى القادم كان قصير أو طويل. المحتوى الأقصر سيولد عدد صغير، وقيم المخرجات الأكبر تميل إلى الإشارة إلى محتوى إدخال أكبر.

• ليست ذات حساسية عالية لبعض أنواع تغيرات المحتوى. على سبيل المثال، إذا رتبنا عدة حروف في المحتوى، من المحتمل أن لا تؤثر على قيمة البعثرة. تغيير الحرف سيؤثر على القيمة ولكن إذا غيرت حرف من A إلى B وحرف آخر مجاور من T إلى S فإن قيمة البعثرة ستبقى نفسها ولن تتغير.

• كلما قصر المحتوى كلما زادت فرص توليد نفس قيمة البعثرة بالنسبة لإدخالين.

من المحتمل أنك تحتاج شيء ما أقوى. إذا كان الأمر كذلك، تتضمن الدوت نت العديد من أدوات البعثرة:

1. بعثرة معتمدة على كود ترخيص (توثيق) الرسالة (HMAC) *Hash-based Message Authentication Code (HMAC)* يتم حسابه باستخدام دالة بعثرة خوارزمية أمن البعثرة رقم 1 (SHA-1) Secure Hash Algorithm number 1، يمكن جعلها متاحة من خلال الفئة HMACSHA1. وتستخدم كود بعثرة 160 بت. لا يوجد قيود restrictions خاصة على طول مفتاح السر secret key المستخدم في الحساب. على الرغم من أنها مناسبة لحالات الخطر المنخفضة، فإن الخوارزمية SHA-1 عرضة للهجوم (الغزو).

2. كود توثيق الرسالة (MAC) Message Authentication Code يتم حسابه باستخدام خوارزمية المفتاح Triple-DESsecret (تم شرحها سابقاً)، يمكن جعلها متاحة من خلال الفئة TripleDES MAC. مفتاح السر secret key المستخدم في الحساب هو إما بطول 16 أو 24 بايت، والقيمة المولدة بطول 8 بايت.
 3. حساب البعثة لـ خوارزمية استيعاب الرسالة رقم 5 (MD5) Message-Digest algorithm number 5، يمكن جعلها متاحة من خلال الفئة MD5CryptoServiceProvider. إن MD5 ما تزال خوارزمية سرية عالية المستوى صممت من قبل رون ريفيست Ron Rivest، ولكنها أظهرت احتواءها على بعض العيوب التي تجعل من أمن التشفير خطر encoding security risk. قيمة التكرار الناتجة هي بطول 128 بت.
 4. مثل الفئة HMACSHA1، تحسب الفئة SHA1Managed class قيمة البعثة باستخدام دالة البعثة SHA-1. مهما تكن، تم كتابتها باستخدام كود الدوت نت المدار NET managed code. فقط إن HMACSHA1 وبعض ميزات التكويد الأخرى في الدوت نت بكل بساطة تدور حول (تغليف) Cryptography API (CAPI) الأقدم، ومكتبات الربط الديناميكي للدوت نت السابقة NET DLL pre-.SHA1 المدارة تستخدم كود بعثة 160 بت.
 5. فئات ثلاث أخرى: SHA256 المدارة، SHA384 المدارة، SHA512 المدارة، مشابهة لفئة SHA1 المدارة، ولكن تستخدم أكواد بعثة 256 بت 384 بت، و512 بت على الترتيب.
- كل من هذه الخوارزميات تستخدم مفتاح السر الذي يجب أن يتم تضمينه كل مرة يتم توليد البعثة على نفس المجموعة من البيانات المدخلة. طالما أن البيانات المدخلة لم تتغير، ومفتاح السر هو نفسه، فقيمة التكرار الناتج ستبقى غير متغيرة. تصميمياً، حتى أصغر تغيير في البيانات المدخلة يولد تغيرات رئيسية في قيمة البعثة المخرجة.

مميزات أمن أخرى Other Security Features

يوجد العديد من ميزات الأمن الهامة والتي لن أناقشها هنا بالتفصيل، ولكن تستحق على الأقل أن أشير إليها.

ترخيص المستخدم و (My.User). User Authentication and My.User

يتضمن كائن الفيچوال بيسك My.User العديد من ميزات الترخيص (التوثيق authentication) والتي يمكن أن تساعد على تصميم كود تمكين الأمن. عضو واحد مفيد هو خاصية "الاسم Name"، والتي تزود باسم المستخدم المرخص حالياً. تخبرك الطريقة IsInRole فيما إذا المستخدم الفعال يتم تضمينه، فيما يدعى مجموعة أمن المدراء the Administrators security group. من أجل تطبيقات نماذج ويندوز، استشير أعضاء My.User بشكل نموذجي إلى تسجيل دخول مستخدم ويندوز. مهما يكن، تستطيع استخدام أنظمة توثيق authentication systems أخرى والتي تلبى احتياجات التطوير الخاصة بك. تتضمن الخيارات Options استخدام الانترنت المعتمد على نظام "معرف ويندوز لايف" Windows Live "Internet-based ID" من ميكروسوفت، وأنظمة توثيق ثانوية مشاركة أخرى third-party authentication systems، أو نظام إدارة المستخدم المخصص التصميم من قبلك.

فئة أمن النصوص The SecureString Class

إنه لمن المميز فمع كل هذه الأدوات المتقدمة، ما يزال المستخدمين يمضون الكثير من الوقت في البناء وتفسير البيانات النصية. لحسن الحظ، تتضمن الدوت نت وفرة من أدوات مفيدة لمعالجة النصوص. ولكن لسوء الحظ، هي ليست آمنة كثيراً. من المحتمل أنك تتذكر أن نصوص الدوت نت ثابتة immutable، حالما يتم إنشائها، فهي لن تتغير. أخيراً، سيتم تدميرها بواسطة معالجة مجمع النفايات. ولكن حتى ذلك الحين، فإنها تستقر في الذاكرة، منتظرة أن يتم عمل مسح بواسطة كود المصمم من قبل هكر ما. بشكل داخلي، البيانات النصية يتم تخزينها كنصوص سهلة، لذلك إذا ما استطاع شخص ما الحصول على الذاكرة، يستطيع نسخ المحتوى لأهداف شريرة nefarious. تتيج لك الفئة System.Security.SecureString تخزين النصوص والحصول عليها مرة أخرى، ولكن داخلياً، محتوى النص يتم تشفيره. إذا ما حصل شخص ما على محتوى داخلي لحالة فئة، سيبدو مثل الترتبة gibberish.

مشروع Project

هذا الفصل سيبين ميزات متحورة حول مواضيع الأمن التالية والمضافة لمشروع المكتبة: فورم تسجيل الدخول "login"، والتي توثق (تثبت صحة) أمان المكتبة، والمستخدمين الإداريين الآخرين. نماذج مجموعة الأمن وإدارة المستخدم. دالة function تعمل على تشفير كلمة المرور المزودة من قبل المستخدم user-supplied password. تفعيل بعض ميزات التطبيق والتي تعتمد على توثيق (تصديق authentication) المستخدم.

دعم التصديق (الترخيص) Authentication Support

بما أن جميع بيانات المكتبة يتم تخزينها في قاعدة بيانات سكول سرفر، فقد استخدمنا سابقاً إما ويندوز Windows أو أمن مخد سكول SQL Server security (أمن سكول سرفر) لتقييد restrict الوصول إلى البيانات نفسها. ولكن حالما نتصل بقاعدة البيانات، سنستخدم نظام تصديق (توثيق) مخصص لتمكين وعدم تمكين ميزات في التطبيق. فهو موجود هناك بحيث نضع بعض ميزات تشفير دوت نت قيد الاستخدام. قبل إضافة الكود المهم، نحتاج لإضافة بعض المتغيرات العامة والتي تدعم الأمن على كامل التطبيق. جميع هذه العناصر العامة تظهر في الملف General.vb، ضمن كود الوحدة البرمجية.

```
Public LoggedInUserID As Integer
Public LoggedInUserName As String
Public LoggedInGroupID As Integer
```

```
Public SecurityProfile (MaxLibrarySecurity) As Boolean
```

على الرغم من أننا عملنا على إضافة العداد LibrarySecurity سابقاً، فهو جزء هام من نظام الأمن. وعناصره تطابق ما يتواجد في جدول Activity في قاعدة بيانات المكتبة. كل قيمة للعداد توافق عنصر في المصفوفة SecurityProfile والتي أضفناها الآن إلى الكود.

'قيم الأمن

```
Public Enum LibrarySecurity As Integer
```

```
    ManageAuthors = 1
    ManageAuthorTypes = 2
    ManageCopyStatus = 3
    ManageMediaTypes = 4
    ManageSeries = 5
    ManageGroups = 6
    ManageItems = 7
    ManagePatrons = 8
    ManagePublishers = 9
    ManageValues = 10
    ManageUsers = 11
    ProcessFees = 12
    ManageLocations = 13
    CheckOutItems = 14
    CheckInItems = 15
    AdminFeatures = 16
    DailyProcessing = 17
    RunReports = 18
    PatronOverride = 19
    ManageBarcodeTemplates = 20
    ManageHolidays = 21
    ManagePatronGroups = 22
    ViewAdminPatronMessages = 23
```

```
End Enum
```

```
Public Const MaxLibrarySecurity As LibrarySecurity = LibrarySecurity.ViewAdminPatronMessages
```

جميع المتغيرات العامة الجديدة المضافة تخزن معلومات مطابقة للمدير administrator الفعال. عندما يكون زبون هو المستخدم الفعال، يضع البرنامج هذه القيم لإعداداتها الافتراضية. بما أن هذا يجب عمله عندما يبدأ البرنامج للمرة الأولى، سنضيف روتين InitializeSystem يتم استدعاه عند البدء. ويظهر هذا الروتين أيضاً في الوحدة البرمجية General.

```
Public Sub InitializeSystem()
```

'التمهيد للمتغيرات العامة هنا.

```
    Dim counter As Integer
```

'إزالة قيم الأمن ذات الصلة.

```
    LoggedInUserID = -1
```

```
    LoggedInUserName = ""
```

```
    LoggedInGroupID = -1
```

```
    For counter = 1 To MaxLibrarySecurity
```

```
        SecurityProfile(counter) = False
```

```
    Next counter
```

```
End Sub
```

(لدى مصفوفة SecurityProfile بنود تتراوح من 0 إلى MaxLibrarySecurity، ولكن الحلقة عند نهاية هذا الكود تبدأ من العنصر 1، لأن جدول Activity يبدأ العد من 1، فقررت تجاوز العنصر 0). يتم استدعاء الطريقة InitializeSystem من الحدث MyApplication_Startup في ملف ApplicationEvents.vb، تماماً قبل تأسيس الاتصال إلى قاعدة البيانات. لذلك دعنا نضيف الكود الآن.

'عمل تمهيد عام

```
InitializeSystem()
```

كل مرة يحاول مدير استخدام النظام، وكل مرة يسجل خروج ويعود البرنامج إلى نمط الزبون (العميل)، فإن جميع المتغيرات العامة ذات الصلة يجب أن يتم إعادة وضعها. وهذا يتم عمله في الطريقة ReprocessSecuritySet، أضف هذه الطريقة إلى الوحدة البرمجية General.

```
Public Sub ReprocessSecuritySet()
```

'إعادة التحميل في مجموعة الأمن للمستخدم الحالي.

'لو لم يكن مستخدم مسجل الدخول، نطف جميع الإعدادات

```
    Dim counter As Integer
```

```
    Dim sqlText As String
```

```
    Dim dbinfo As SqlConnection.SqlDataReader = Nothing
```

'أزل البنود الموجودة

```
    For counter = 1 To MaxLibrarySecurity
```

```
        SecurityProfile(counter) = False
```

```
    Next counter
```

'أخرج إذا لم يكن هناك مستخدم قد سجل الدخول.

```
    If (LoggedInUserID = -1) Or (LoggedInGroupID = -1) Then Return
```

```
Try
```

```

التحميل في عناصر الأمن لهذا المستخدم
sqlText = "SELECT ActivityID FROM GroupActivity WHERE GroupID = " & LoggedInGroupID
dbInfo = CreateReader(sqlText)
Do While (dbInfo.Read)
    SecurityProfile(CInt(dbInfo!ActivityID)) = True
Loop
dbInfo.Close()
Catch ex As Exception
    GeneralError("ReprocessSecuritySet", ex)
    If (dbInfo IsNot Nothing) Then dbInfo.Close()
    'عدم دخول المدير يؤدي إلى اللجوء إلى التكرار المستمر
    LoggedInUserID = -1
    LoggedInGroupID = -1
    ReprocessSecuritySet()
Finally
    dbInfo = Nothing
End Try
End Sub

```

يستخدم هذا الروتين كود تم بناءه في الفصل العاشر وفصول سابقة. فعندما يكتشف هذا الروتين مستخدم موثوق (مصرح به) (أي المتغير LoggedInUserID) فإنه ينشئ الكائن SqlDataReader مع ميزات الأمن المسموحة للمستخدم، ويخزن تلك الإعدادات في المصفوفة SecurityProfile. حالما يتم تحميلها، فيمثل أي عنصر للمصفوفة عندما يكون صحيح True ميزة التطبيق أن المدير administrator مصرح به كي يستخدم التطبيق. سأناقش جدول GroupActivity بعد قليل في هذا الفصل.

إذا ما حدث خطأ قاعدة بيانات خلال المعالجة، يعمل الكود على إعادة وضع كل شيء إلى نمط العمل، وعمل استدعاء دوري Recursive. (يحصل التكرار المستمر Recursion عندما يستدعي روتين نفسه بشكل مباشر أو غير مباشر).

تشفير كلمات المرور Encrypting Passwords

سأستخدم واحدة من طرق خلط (بعثرة) الدوت نت لتشفير كلمة المرور المزودة من قبل المدير قبل تخزينها في قاعدة البيانات. واحد من جداول قاعدة بيانات المكتبة، جدول Username، يخزن قالب profile الأمن الأساسي لكل مكتبي أو مستخدم إداري آخر، ومن ضمنها كلمات المرور. بما أن أياً كان يستطيع الدخول إلى قاعدة البيانات سنكون قادرين على رؤية كلمات المرور المخزنة في هذا الجدول، فسنعلم على تشفيرها لجعلها أقل جذاباً. (من أجل الزبائن الذين يستخدمون البرنامج ببساطة، فليس من الواجب عليهم الوصول مباشرة إلى قاعدة البيانات بعيداً عن التطبيق، ولكنك لاتعلم هؤلاء الزبائن اللعوبين).

لحفظ الأشياء آمنة، فسنشوش (نمزج scramble) كلمة المرور المدخلة من قبل المستخدم، واستخدامها لتوليد قيمة خلط (بعثرة) وتخزين قيم البعثرة في حقل كلمة المرور password لقاعدة البيانات من أجل المستخدم. فيما بعد، عندما يريد مستخدم مدير الحصول على إمكانية الوصول إلى الميزات المحسنة، فإن البرنامج سيحول مرة أخرى كلمة المرور المدخلة في قيمة البعثرة (الخلط)، ويقارن تلك القيمة بكلمة المرور المخلوطة (المشفرة) في السجل. كل دالة بعثرة (خلط) للدوت نت تعتمد على كود سري. بما أن مشروع المكتبة سيعمل فقط تشفير غير مباشر، ولن يسأل أي برنامج آخر أبداً لإعادة تشفير كلمة المرور، فإننا سنستخدم فقط اسم تسجيل الدخول للمستخدم كمفتاح "سري secret". قررت استخدام فئة البعثرة HMACSHA1، على الأغلب من أجل قدرتها على قبول مفتاح متغير الحجم variablesize. على الرغم من أنها مقررّة لأن تحتوي قضايا الأمن، فإن هذا لن يكون مشكلة بالنسبة للطريقة التي سنستخدمها بها. أعني، إذا ما حاول أحدهم الدخول لقاعدة البيانات فعلياً لفك تشفير كلمات المرور المخزنة في جدول Username، فسيكون لهذه الفئة إمكانية وصول كاملة لكل شيء في نظام المكتبة سابقاً.

بالطبع، يتطلب كود التشفير مرجع إلى فضاء الأسماء System.Security.Cryptography. وسنحتاج أيضاً مرجع إلى System.Text من أجل بعض الدعم للكود. أصف عبارات Import ذات الصلة إلى أعلى ملف كود General.vb.

```

Imports System.Text
Imports System.Security.Cryptography

تحدث البعثرة (التشويش) الحقيقي لكلمة المرور في الروتين EncryptPassword، لذلك عمل على إضافته إلى كود الوحدة البرمجية General.
Public Function EncryptPassword(ByVal loginID As String, ByVal passwordText As String) As String
    'إدخال اسم المستخدم وكلمة المرور، تشفير كلمة المرور بحيث تصبح من صعب فك تشفيرها.
    'لا يوجد حد لطول كلمة المرور بما أنها تصبح مشفرة بأي طريقة.
    Dim hashingFunction As HMACSHA1
    Dim secretKey() As Byte
    Dim hashValue() As Byte
    Dim counter As Integer
    Dim result As String = ""
    'تحضير مفتاح السر. وإجباره ليكون بالحالة الكبيرة من أجل الاستقرار.
    'ومن ثم حشره في مصفوفة بايت.
    secretKey = (New UnicodeEncoding).GetBytes(UCCase(loginID))
    'إنشاء مكونات خلط (تشفير) تستخدم SHA-1 المدار
    hashingFunction = New HMACSHA1(secretKey, True)

```

```

'حساب قيمة التشفير (الخلط). سطر واحد بسيط من الكود.
    hashValue = hashingFunction.ComputeHash((New UnicodeEncoding).GetBytes(passwordText))
'قيمة البعثة جاهزة، ولكنني أحب الأشياء في نص بسيط،
'وعندما يكون مستحيل. لنحوله إلى نص ست عشري طويل
    For counter = 0 To hashValue.Length - 1
        result &= Hex(hashValue(counter))
    Next counter
'كلمات المرور المخزنة محدودة إلى 20 حرف.
    Return Left(result, 20)
End Function

```

الطرق الرئيسية للتفاعل مع مزودات الأمن security providers في الدوت نت هي إما بواسطة مصفوفة بايت أو تجميع (ستيريم). أثرت استخدام طريقة مصفوفة بايت، بتحويل القيم النصية القادمة من خلال الطريقة GetBytes للكائن UnicodeEncoding. حالما يتم تخزين مصفوفة بايت، قمت بتمرير معرف الدخول login ID وكلمة المرور password كمعاملات نسبية arguments إلى ميزات الفئة HMACSHA1.

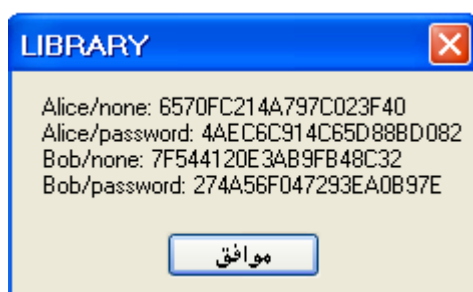
على الرغم من أنني أستطيع تخزين مخرجات الطريقة ComputeHash مباشرة في حقل قاعدة البيانات. قررت تحويل النتيجة إلى رموز الآسكي ASCII القابلة للقراءة لذلك فإن تلك الأشياء لن تبدو مقلقة عند إصدار issued عبارات سكول على جدول UserName. التحويل الذي قمت به أساسي basic: تحويل كل بايت byte إلى مكافئه الست عشري القابل للطباعة printable hexadecimal equivalent باستخدام دالة الفيچوال بيسك Hex. ومن ثم سلسلة فقط النتائج مع بعضها البعض. حقل كلمة المرور لجدول UserName يحفظ فقط 20 حرف، لذلك قطعت (بترت chop off) أي شيء أطول. ومن أجل التأكد فقط أن هذه الخوارزمية تولد مخرجات معقولة، استدعيت EncryptPassword مع عدة مخرجات مختلفة.

```

MsgBox("Alice/none: " & EncryptPassword("Alice", "") & vbCrLf & "Alice/password: " & EncryptPassword("Alice",
"password") & vbCrLf & "Bob/none: " & EncryptPassword("Bob", "") & vbCrLf & "Bob/password: " &
EncryptPassword("Bob", "password"))

```

يولد الكود السابق الرسالة المبينة في الشكل التالي:



التراجع عن بعض التغييرات السابقة Undoing Some Previous Changes

تعرف الجداول UserName و GroupName و GroupActivity في قاعدة البيانات أشكال الأمن لكل مستخدم إداري. كل مستخدم (سجل في جدول UserName) هو جزء من مجموعة أمن (سجل GroupName). كل مجموعة تتضمن إمكانية وصول إلى صفر zero أو عدد أكبر لميزات التطبيق المحسنة، يعرف الجدول GroupActivity أي الميزات التي تتطابق مع كل سجل مجموعة أمن من جدول GroupName. لإدارة هذه الجداول، نحتاج إلى إضافة نماذج خاصية property forms والتي تحدث حقول سجل مفرد لقاعدة البيانات. لقد كتبنا سابقاً بعض الكود الخلفي منذ فترة قليلة. عرّف الفصل الثامن ملف BaseCodeForm.vb، قالب من أجل النماذج والتي تحدث سجلات مفردة لقاعدة البيانات. ونفس الفصل قدم ملف ListEditRecords.vb، الفورم الرئيسي الذي يعرض قائمة بسجلات قاعدة البيانات المسبقة التعريف. محرر سجلنا من أجل كل المستخدمين users ومجموعات الأمن security groups سيستخدم ميزات في هذين النموذجين الموجودين.

عندما صممنا الكود من أجل BaseCodeForm.vb في الفصل الثامن، كان هدفي توضيح لك ميزات الفئة MustInherit و MustOverride المضمنة في الفيچوال بيسك. وهي ميزات مفيدة نوعاً ما. لسوء الحظ، إنهما فقط لا يمتزجان بشكل جيد مع عناصر واجهة المستخدم، وإليك لماذا: تعمل الفيچوال أستوديو فعلياً على إنشاء حالات (نسخ instances) من نموذجك وقت التصميم design time بحيث تستطيع التفاعل معها داخل المحرر. إذا كنت تريد أن تسبر الكود المصدري، صرح عن أداة صندوق نص TextBox، فستجد كود خاص يتعامل مع تقدمه وقت التصميم للأداة. هل هو هام؟ نعم، من؟ نعم، مثالي في كل الحالات؟ لا.

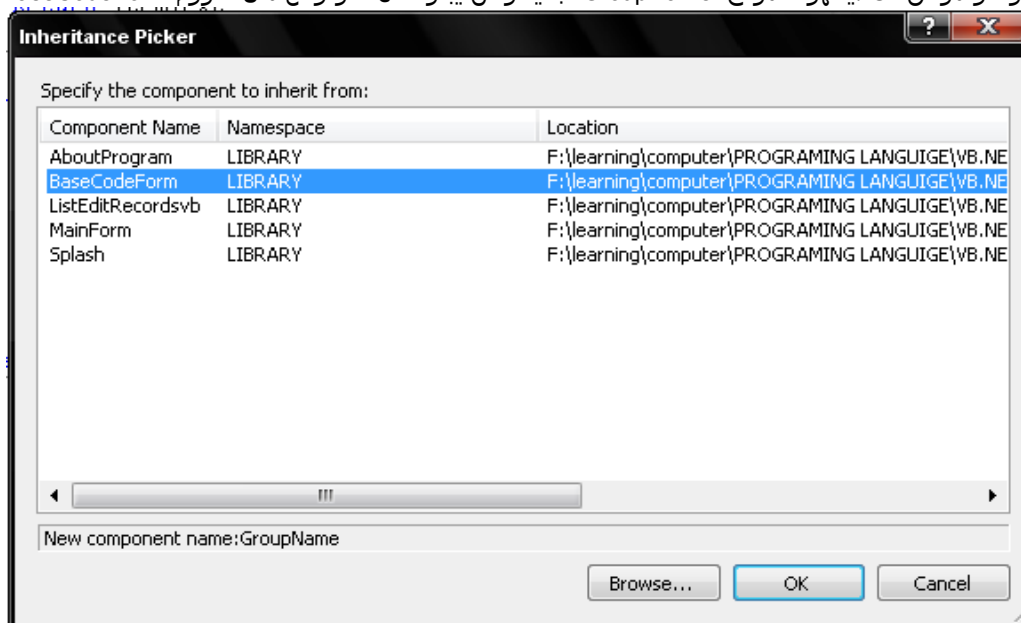
المشكلة، والمشكلة هي وضعه بشكل معتدل، هل لأن فيچوال بيسك لا تستطيع إنشاء حالة فئة معرفة كـ (يجب أن تشتق MustInherit) وهذا لأنك يجب أن ترثها من خلال فئة أخرى أولاً قبل أن تنشئ حالات. ماذا يعني هذا لك؟ هذا يعني إذا حاولت تصميم نموذج يشتق من قالب نموذج "يجب وراثته MustInherit"، فإن الفيچوال بيسك لن تجلب حصة واجهة المستخدم للفورم من أجل تحريك الخاص. وما يزال بإمكانك الوصول إلى الكود المصدري للفورم، وإذا ما كانت هذه هي الطريقة التي تريد تصميم نموذج مشتق inherited form، فهذا جميل. ولكننا نبحث عن البساطة في البرمجة، لذلك سوف نستخدم بالتأكيد أدوات الفيچوال أستوديو الخاصة بالفورم لتحرير نماذجنا المرئية. زبدة الكلام، أن علينا تغيير ملف BaseCodeForm.vb، وإزالة الكلمات المحجوزة MustInherit و MustOverride، وعمل التعديلات المناسبة الأخرى. لقد عملت على تعديل التغييرات سابقاً.

هذا جزء من حقيقة البرمجة في الأنظمة المعقدة مثل الفيچوال أستوديو. في بعض الأحيان، حتى بعد أن تعمل جميع أبحاثك وتشيك بحذر ميزات التطبيق والتراكيب، إنك تُشغل ضمن مصمم ما أو مترجم سلوك معين بحيث يجبرك على صنع بعض التغيير. حالما تتعلم تجنب المشاكل الرئيسية، ستجد أن هذا لا يحدث كثيراً. ولكن عندما يحدث، فسيكون وقت عظيم لتغضب.

إدارة مجموعات الأمن. Managing Security Groups.

لذلك، لنرجع إلى محرر سجل GroupName. لم أعمل على إضافته إلى المشروع بعد، لذلك لنعمل على إضافته الآن. لأنه سيرث من نموذج آخر في المشروع، علينا أن نسمح للفيچوال أستوديو استنساخ النموذج القاعدي بترجمة التطبيق أولاً. وهذا يتم عمله بسهولة من خلال بناء Build << أمر قائمة بناء المكتبة Build Library.

لإنشاء فورم جديد، اختر مشروع Project << أمر القائمة أضف نموذج ويندوز Add Windows Form. عندما يظهر بند ويندوز الجديد، اختر لإنشاء فورم form جديد، اختر مشروع Project << أمر القائمة أضف نموذج ويندوز Add Windows Form. عندما يظهر نموذج ناخب الوراثة Inheritance Picker form (شاهد الشكل التالي)، اختر من القائمة Windows Forms من قائمة التصنيفات Categories، المتبوعة بواسطة Inherited Form من قائمة القوالب Templates. أدخل في حقل الاسم GroupName.vb، ومن ثم انقر الزر Add. عند يظهر نموذج ناخب الوراثة Inheritance Picker form (شاهد الشكل التالي)، اختر من القائمة BaseCodeForm، وانقر موافق OK. يظهر النموذج GroupName الجديد، ولكن يبدو على نحو رائع مثل الفورم BaseCodeForm.



أضف أداتي عنوان (لصاقة Label)، وأداتي صندوق نص TextBox، وأداتي زر Button، وأداة صندوق قائمة اختبار CheckBox من صندوق الأدوات، وضع خاصياتها باستخدام الإعدادات (راجع المشروع المخصص لهذا الفصل).

ولا تنسى تعديل ترتيب التنقل (مفتاح) للأدوات على الفورم، لنعمل على إضافة الكود كله دفعة واحدة، أضف الكود التالي لجسم الكود المصدري للفتة:

```
Public Class GroupName
    Private ActiveID As Integer
    Private StartingOver As Boolean
    Public Overrides Function AddRecord() As Integer
        ' إضافة سجل جديد.
        ActiveID = -1
        PrepareFormFields()
        Me.ShowDialog()
        If (Me.DialogResult = Windows.Forms.DialogResult.OK) Then
            Return ActiveID Else Return -1
        End Function
    Public Overrides Function CanUserAdd() As Boolean
        ' اختبار المستخدم من أجل إمكانية الوصول الآمن: إضافة.
        Return SecurityProfile(LibrarySecurity.ManageGroups)
    End Function
    Public Overrides Function CanUserEdit() As Boolean
        ' اختبار المستخدم من أجل إمكانية الوصول الآمن: تحديث.
        Return SecurityProfile(LibrarySecurity.ManageGroups)
    End Function
    Public Overrides Function CanUserDelete() As Boolean
        ' اختبار المستخدم من أجل إمكانية الوصول الآمن: حذف.
        Return SecurityProfile(LibrarySecurity.ManageGroups)
    End Function
    Public Overrides Function DeleteRecord(ByVal recordID As Integer) As Boolean
        ' يريد المستخدم حذف السجل.

```



```

Dim sqlText As String
On Error GoTo ErrorHandler
' ----- Confirm with the user.
If (MsgBox("هل تريد فعلاً حذف مجموعة الأمن؟", _
    MsgBoxStyle.YesNo Or MsgBoxStyle.Question, ProgramTitle) <> MsgBoxResult.Yes) _
    Then Return False
'التأكد من أن هذا السجل ليس قيد الاستخدام
sqlText = "SELECT COUNT(*) FROM UserName WHERE GroupID = " & recordID
If (CInt(ExecuteSQLReturn(sqlText)) > 0) Then
    MsgBox("لا نستطيع حذف السجل لأنه قيد الاستخدام", _
        MsgBoxStyle.OkOnly Or MsgBoxStyle.Exclamation, ProgramTitle)
    Return False
End If
'حذف السجل.
TransactionBegin()
sqlText = "DELETE FROM GroupActivity WHERE GroupID = " & recordID
ExecuteSQL(sqlText)
sqlText = "DELETE FROM GroupName WHERE ID = " & recordID
ExecuteSQL(sqlText)
TransactionCommit()
Return True
ErrorHandler:
GeneralError("GroupName.DeleteRecord", Err.GetException())
TransactionRollback()
Return False
End Function

Public Overrides Function EditRecord(ByVal recordID As Integer) As Integer
'تحديث سجل موجود.
ActiveID = recordID
PrepareFormFields()
Me.ShowDialog()
If (Me.DialogResult = Windows.Forms.DialogResult.OK) Then
    Return ActiveID Else Return -1
End Function

Public Overrides Sub FillListWithRecords(ByRef destList As ListBox, _
    ByRef exceededMatches As Boolean)
'تقديم قائمة، ملئها بجميع البنود المحددة.
Dim sqlText As String
Dim dbInfo As SqlClient.SqlDataReader
On Error GoTo ErrorHandler
'تنظيف أية قيم موجودة.
destList.Items.Clear()
exceededMatches = False
'استخلاص القيم من قاعدة البيانات.
sqlText = "SELECT * FROM GroupName ORDER BY FullName"
dbInfo = CreateReader(sqlText)
Do While dbInfo.Read
'إضافة بند واحد.
destList.Items.Add(New ListItemData(CStr(dbInfo!FullName), CInt(dbInfo!ID)))
Loop
dbInfo.Close()
dbInfo = Nothing
Return
ErrorHandler:
GeneralError("GroupName.FillListWithRecords", Err.GetException())
On Error Resume Next
If Not (dbInfo Is Nothing) Then dbInfo.Close() : dbInfo = Nothing
Return
End Sub

Public Overrides Function FormatRecordName(ByVal recordID As Integer) As String
'تقديم معرف، للعودة بالاسم.
Dim sqlText As String
On Error GoTo ErrorHandler
'استخلاص القيمة من قاعدة البيانات.
sqlText = "SELECT FullName FROM GroupName WHERE ID = " & recordID
Return CStr(ExecuteSQLReturn(sqlText))
ErrorHandler:
GeneralError("GroupName.FormatRecordName", Err.GetException())
Return "Error"

```

Mhm76

```

End Function
Public Overrides Function GetEditDescription() As String
    ' إرجاع عنوان عرض نموذج القائمة
    Return " & استخدم هذه الأكواد لتأسيس مجموعات أمن ضمن أي من " &
    " المستخدمين الإداريين الذين تم وضعهم. كل مجموعة يمكن أن تستقبل " &
    " الوصول إلى أجزاء مختلفة من النظام "
End Function
Public Overrides Function GetEditTitle() As String
    ' إرجاع القائمة من عرض العنوان.
    Return "Security Groups"
End Function
Private Sub PrepareFormFields()
    ' التعينة بالبيانات.
    Dim sqlText As String
    Dim dbInfo As SqlConnection.SqlDataReader
    Dim matchPos As Integer
    On Error GoTo ErrorHandler
    ' التحميل في قائمة مجموعات الأمن.
    If (ActivityList.Items.Count = 0) Then
        ' -- هذه هي المرة الأولى، التحميل من قاعدة البيانات.
        ActivityList.Items.Clear()
        sqlText = "SELECT * FROM Activity ORDER BY FullName"
        dbInfo = CreateReader(sqlText)
        Do While (dbInfo.Read)
            ActivityList.Items.Add
                (New ListItemData(CStr(dbInfo!FullName), CInt(dbInfo!ID)))
        Loop
        dbInfo.Close()
        dbInfo = Nothing
    Else
        ' تحديث الكود المتتالي. تنظيف الاختيارات.
        ' لاتوجد طريقة تنظيف لعمل هذا.
        Do While (ActivityList.CheckedItems.Count > 0)
            ActivityList.SetItemChecked(ActivityList.CheckedIndices(0), False)
        Loop
        If (ActivityList.Items.Count > 0) Then ActivityList.TopIndex = 0
    End If
    ' تنظيف البيانات الموجودة.
    StartingOver = True
    RecordFullName.Text = ""
    If (ActiveID = -1) Then Return
    ' تحميل القيم المخزنة في قاعدة البيانات.
    sqlText = "SELECT FullName FROM GroupName WHERE ID = " & ActiveID
    RecordFullName.Text = CStr(ExecuteSQLReturn(sqlText))
    ' التحميل في إعدادات الأمن الموجودة.
    sqlText = "SELECT ActivityID FROM GroupActivity WHERE GroupID = " & ActiveID
    dbInfo = CreateReader(sqlText)
    Do While (dbInfo.Read)
        matchPos = ActivityList.Items.IndexOf(CInt(dbInfo!ActivityID))
        If (matchPos <> -1) Then ActivityList.SetItemChecked(matchPos, True)
    Loop
    dbInfo.Close()
    dbInfo = Nothing
    Return
ErrorHandler:
    GeneralError("GroupName.PrepareFormFields", Err.GetException())
    On Error Resume Next
    If Not (dbInfo Is Nothing) Then dbInfo.Close() : dbInfo = Nothing
    Return
End Sub
Private Function SaveFormData() As Boolean
    ' يريد المستخدم حفظ التغييرات. الرجوع بصواب في حالة النجاح.
    Dim sqlText As String
    Dim newID As Integer = -1
    On Error GoTo ErrorHandler
    ' التحضير لحفظ البيانات.
    Me.Cursor = Windows.Forms.Cursors.WaitCursor
    TransactionBegin()
    ' حفظ البيانات.

```

```

If (ActiveID = -1) Then
    'إنشاء مدخلة جديدة.
    sqlText = "INSERT INTO GroupName (FullName) OUTPUT INSERTED.ID VALUES (" &
        DBText(Trim(RecordFullName.Text)) & ")"
    newID = CInt(ExecuteSQLReturn(sqlText))
Else
    'تحديث المدخلة الموجودة.
    newID = ActiveID
    sqlText = "UPDATE GroupName SET FullName = " &
        DBText(Trim(RecordFullName.Text)) & " WHERE ID = " & ActiveID
    ExecuteSQL(sqlText)
End If
'تنظيف أي إعدادات أمن موجودة.
sqlText = "DELETE FROM GroupActivity WHERE GroupID = " & newID
ExecuteSQL(sqlText)
' ---- Save the selected security settings.
'حفظ إعدادات الأمن المختارة.
For Each itemChecked As ListItemData In ActivityList.CheckedItems
    sqlText = "INSERT INTO GroupActivity (GroupID, ActivityID) VALUES (" &
        newID & ", " & itemChecked.ItemData & ")"
    ExecuteSQL(sqlText)
Next itemChecked
'إنهاء جميع التغييرات.
TransactionCommit()
ActiveID = newID
'يمكن لهذا التغيير أن يؤثر على هذا المستخدم.
If (LoggedInGroupID = ActiveID) Then ReprocessSecuritySet()
' -- بنجاح.
ActiveID = newID
Me.Cursor = Windows.Forms.Cursors.Default
Return True
ErrorHandler:
Me.Cursor = Windows.Forms.Cursors.Default
GeneralError("GroupName.SaveFormData", Err.GetException())
TransactionRollback()
Return False
End Function
Private Function ValidateFormData() As Boolean
    ' -- تأكد من أن المستخدم يزود بالبيانات الصحيحة. الرجوع بصواب في حالة النجاح.
    Dim sqlText As String
    On Error GoTo ErrorHandler
    ' ---- Name is a required field.
    ' -- الاسم هو الحقل المطلوب.
    If (Trim(RecordFullName.Text) = "") Then
        MsgBox("Please supply the security group name.",
            MsgBoxStyle.OkOnly Or MsgBoxStyle.Exclamation, ProgramTitle)
        RecordFullName.Focus()
        Return False
    End If
    ' -- تأكد من الاسم ليس قيد الاستخدام سابقاً.
    sqlText = "SELECT COUNT(*) FROM GroupName WHERE UPPER(FullName) = " &
        DBText(UCASE(Trim(RecordFullName.Text)))
    If (ActiveID <> -1) Then sqlText &= " AND ID <> " & ActiveID
    If (CInt(ExecuteSQLReturn(sqlText)) > 0) Then
        MsgBox("A record already exists with the name " & Trim(RecordFullName.Text) &
            ".", MsgBoxStyle.OkOnly Or MsgBoxStyle.Exclamation, ProgramTitle)
        RecordFullName.Focus()
        Return False
    End If
    ' -- بنجاح.
    Return True
ErrorHandler:
GeneralError("GroupName.ValidateFormData", Err.GetException())
Return False
End Function
Private Sub GroupName_Activated(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Activated
    ' -- العودة بالتركيز إلى الحقل الرئيسي.
    If (StartingOver) Then RecordFullName.Focus()

```

```

StartingOver = False
End Sub
Private Sub RecordFullName_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles
RecordFullName.Enter
' _ علم كامل النص.
RecordFullName.SelectAll()
End Sub
Private Sub ActOK_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ActOK.Click
' _ احفظ التغيرات وارجع إلى نموذج الملخص.
If (ValidateFormData() = False) Then Return
If (SaveFormData() = False) Then Return
Me.DialogResult = Windows.Forms.DialogResult.OK
End Sub
End Class

```

تتضمن الفئة عضوين خاصين. يحفظ ActiveID رقم ID لسجل GroupName لقاعدة البيانات المعروض، أو 1- عند تحرير سجل جديد. العلامة StartingOver أكثر أهمية نوعاً ما. تذكر أننا نستخدم نموذج ملخص متشارك لعرض جميع سجلات GroupName المدخلة سابقاً. للسماح لهذه الفورم الشاملة ListEditRecords.vb، من العمل مع محررات السجلات المختلفة، نمرر حالة من نموذج التفاصيل (GroupName.vb) في هذه الحالة) إلى نموذج الملخص:

```
ListEditRecordsvb.ManageRecords(New LIBRARY.GroupName)
```

ضمن كود النموذج ListEditRecordsvb، يتم استخدام حالة من GroupName مرةً بعد مرةً. كل مرة يريد المستخدم إضافة أو تحديث سجل قاعدة البيانات GroupName. إذا حدث المستخدم سجل واحد، ومن ثم حاول تحديث واحد آخر، فإن المتبقي من السجل الأول سيبقى موجود في حقول نموذج التفاصيل. وهكذا، سيكون علينا تنظيفها كل مرة نضيف أو نحدث سجل مختلف. يساعد الدليل StartingOver بهذه العملية وذلك بإعادة وضع التركيز إلى حقل نموذج التفاصيل الأول في حدث Activated للنموذج.

```

Private Sub GroupName_Activated(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Activated
' _ العودة بالتركيز إلى الحقل الرئيسي.
If (StartingOver) Then RecordFullName.Focus()
StartingOver = False
End Sub

```

الطريقة الخاصة PrepareFormFields تعمل بالتنظيف الحقيقي وتخزن البيانات مع كل استدعاء لتحديث أو إضافة جديدة من أجل السجلات الجديدة، من البساطة تنظيف كل البيانات المدخلة على الفورم. عند تحديث سجل موجود، فإنه يستخلص البيانات ذات الصلة من قاعدة البيانات، ويخزن القيم المحفوظة في حقول متنوع على الفورم. العبارات التالية تعرض اسم المجموعة المخزنة في حقل RecordFullName، أو أداة صندوق نص TextBox.

```

' تحميل القيم المخزنة في قاعدة البيانات.
sqlText = "SELECT FullName FROM GroupName WHERE ID = " & ActiveID
RecordFullName.Text = CStr(ExecuteSQLReturn(sqlText))

```

معظم الروتينات في نموذج GroupName توفر إعادة قيادة overrides بسيطة من أجل الأعضاء القاعدية لفئة BaseCodeForm. الطريقة CanUserAdd والتي تعود ببساطة بـ False في الفئة القاعدية، تتضمن منطق حقيقي في الفئة المشتقة. إنها تستخدم المصفوفة SecurityProfile التي أضفناها سابقاً لتحديد أي مستخدم حالي مسموح له إضافة سجلات مجموعة group records.

```

Public Overrides Function CanUserAdd() As Boolean
' اختبار المستخدم من أجل إمكانية الوصول الآمن: إضافة.
Return SecurityProfile(LibrarySecurity.ManageGroups)
End Function

```

إذا نظرت إلى الكود المضاف، ستجد إعادة قيادة overrides لجميع أعضاء BaseCodeForm ما عدا الطرق SearchForRecord و UsesSearch. الفئة المشتقة تقبل الفعل الافتراضي لهذين العضوين.

يضيف المستخدم adds، يحدد edits، ويحذف deletes سجلات اسم مجموعة من خلال إعادة قيادة AddRecord، و EditRecord و DeleteRecord. على الترتيب، لكل استدعاء من قبل الكود في فورم ListEditRecords. إليك كود EditRecord.

```

Public Overrides Function EditRecord(ByVal recordID As Integer) As Integer
' تحديث سجل موجود.
ActiveID = recordID
PrepareFormFields()
Me.ShowDialog()
If (Me.DialogResult = Windows.Forms.DialogResult.OK) Then
Return ActiveID Else Return -1
End Function

```

بعد تخزين معرف ID للسجل للتحديث في الحقل الخاص ActiveID، يحمل الكود البيانات من خلال الطريقة PrepareFormFields، ويبحث المستخدم على تحديث السجل باستدعاء Me.ShowDialog حتى يضع كود ما أو أداة خاصة الفورم DialogResult. وهذا يتم عمله في الحدث

ActOK_Click، وأيضاً من خلال الخاصية DialogResult للزر ActCancel، والتي ستعمل الفيچوال أستوديو على إسناده إلى الفورم بشكل آلي عندما ينقر المستخدم الزر ActCancel.

إن الروتينين AddRecord يشبه كثيراً الروتين EditRecord، ولكنه يسند -1 إلى العضو ActiveID ليشير لسجل جديد. الروتين DeleteRecord أكثر تعقيداً، ويستخدم بعض كود قاعدة البيانات الذي كتبناه في الفصل السابق (راجع كود الدالة DeleteRecord) بعد تأكيد الحذف من قبل المستخدم، يحدد تفحص سريع فيما إذا المجموعة group تمازال قيد الاستخدام في مكان في جدول UserName. إذا كان كل شيء تم مراجعته على مايرام، يتم حذف السجل باستخدام عبارة سكول DELETE. بما أننا نحتاج إلى حذف البيانات في جدولين، فقد ضمنتها كله في مداولة transaction. إذا ما حدث خطأ، فإن معالج الخطأ عند نهاية الروتين سيتراجع عن المداولة من خلال TransactionRollback.

ملاحظة: تحذير تكامل قاعدة البيانات. Database Integrity Warning.

إذا كان لديك خلفية في تطوير قاعدة البيانات، فقد تكون قد رأيت سابقاً الخلل في كود الحذف delete. على الرغم من أنني أخذت وقتي في إثبات أن السجل ليس قيد الاستخدام قبل حذفه، من المحتمل أن يستخدمه بعض المستخدمين الآخرين بين الوقت الذي أختبر فيه استخدام السجل والوقت الذي أحذفه فعلياً. بالاعتماد على إعداد (تركيب) الكود وقاعدة البيانات الذي جلبته حتى الآن، ستكون هناك مشكلة حقيقية. عندما صممت هذا النظام، توقعت أن أمين مكتبي مفرد سيدير مهمات إدارية كهذه، لذلك لم ألق حول تعارضات conflicts كهذه. إذا كنت مهتم حول احتمالات حذف سجلات قيد الاستخدام من خلال كود مثل هذا، تستطيع تمكين التكامل المرجعي referential integrity على العلاقات في قاعدة البيانات. لقد أسست علاقة بين الحقول GroupName.ID و UserName.GroupID، ولكن كانت من أجل الأهداف التعليمية فقط. تستطيع إعادة تركيب هذه العلاقة ليملك سكول سرفر القدرة على إجبار العلاقة بين الجداول. إذا ما عملت هذا، فلن يكون من الممكن حذف سجل قيد الاستخدام، وسيحدث خطأ في البرنامج عندما تحاول فعل هذا. إن هذا يبدو جيداً، وهو كذلك، ولكن الإفراط في استخدام التكامل المرجعي يمكن أن يبطل إمكانية الوصول لبياناتك. وسأترك هذا التركيب (الإعداد) اختياري بالنسبة لك. عندما يتم عمل تغييرات من قبل المستخدم إلى السجل، فإن النقر على الزر "موافق OK" يدفع البيانات المخرجة من قاعدة البيانات إليها مرة أخرى. يتحقق معالج الحدث ActOK_Click من البيانات، ويحفظها.

```
' _ احفظ التغييرات وارجع إلى نموذج الملخص.
If (ValidateFormData() = False) Then Return
If (SaveFormData() = False) Then Return
Me.DialogResult = Windows.Forms.DialogResult.Ok
```

تعمل الطريقة ValidateFormData بعض الاختبارات البسيطة للتحقق من البيانات، مثل الطلب من المستخدم إدخال اسم مجموعة الأمن security group name. وهذا مميز. إذا ما بدا كل شيء على مايرام، يبين الروتين SaveFormData عبارات سكول التي تحفظ البيانات (راجع كود الدالة SaveFormData). تأكد من مراجعة الروتينات الأخرى في نموذج GroupName، فهي موجودة لدعم وتحسين خبرة المستخدم.

إدارة المستخدمين. Managing Users.

نحتاج أيضاً إلى فورم لإدارة سجلات جدول UserName. بما أن كود تلك الفورم بشكل عام يتبع ما رأيناه سابقاً في نموذج GroupName، فلن أثقل عليك بالتفاصيل. لقد عملت على إضافة UserName.vb لمشروعك، ولكن لمنع الأخطاء في كودك بينما تكون في وسط التطوير، فقد عملت على تعطيله. ولتمكينه، اختر هذا الملف في نافذة مستكشف الحلول Solution Explorer. ومن ثم في نافذة الخصائص Properties، غير خاصية Build Action من None إلى Compile.

الكود الوحيد المهم في هذه الفورم والذي هو مختلف نوعاً ما عن فورم GroupName هو معالجة كلمة المرور password. وللحفاظ على الأشياء سرية قدر الإمكان، فلن أحمل فعلياً كلمة المرور المحفوظة ضمن حقل كلمة المرور على الفورم. ولن أعمل أية طريقة جيدة منذ أن عملت على تخزين نسخة (إصدار) الخلط hashed version في قاعدة البيانات. بما أنني استخدم معرف تسجيل الدخول Login ID للمستخدم كمتفتاح سري عند تشفير كلمة المرور، يجب عليّ أن أعيد توليد كلمة المرور password ولو غير المستخدم معرف تسجيل الدخول Login ID. يحفظ الحقل الخاص OrigLoginID نسخة من معرف تسجيل الدخول Login ID عند فتح الفورم للمرة الأولى، ويتفحص من أجل أي تغييرات عند إعادة حفظ السجل. إذا ما حدثت تغييرات، فإنه يعيد توليد كلمة المرور.

```
passwordResult = EncryptPassword(Trim(RecordLoginID.Text), Trim(RecordPassword.Text))
```

استخدام نماذج التحرير UserName و GroupName يتطلب بعض الكود الإضافي في الفورم الرئيسي main form. أضفه إلى جسم الكود المصدري للفورم الرئيسي:

```
Private Sub AdminLinkGroups_LinkClicked(ByVal sender As Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles AdminLinkGroups.LinkClicked
' ----- Let the user edit the list of security groups.
' دع المستخدم يحرر قائمة مجموعات الأمن.
If (SecurityProfile(LibrarySecurity.ManageGroups) = False) Then
MsgBox(NotAuthorizedMessage, MsgBoxStyle.OkOnly Or
MsgBoxStyle.Exclamation, ProgramTitle)
Return
End If
' ----- Edit the records.
' تحرير السجلات.
ListEditRecords.ManageRecords(New Library.GroupName)
ListEditRecords = Nothing
```

Mhm76

```

End Sub
Private Sub AdminLinkUsers_LinkClicked(ByVal sender As Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles AdminLinkUsers.LinkClicked
' ----- Let the user edit the list of administrative users.
دع المستخدم يحرر قائمة المستخدمين الإداريين.
If (SecurityProfile(LibrarySecurity.ManageUsers) = False) Then
MsgBox(NotAuthorizedMessage, MsgBoxStyle.OkOnly Or
MsgBoxStyle.Exclamation, ProgramTitle)
Return
End If
' ----- Edit the records.
تحرير السجلات.
ListEditRecords.ManageRecords(New Library.UserName)
ListEditRecords = Nothing
End Sub

```

الأدوات AdminLinkUsers و AdminLinkGroups هي لصاقات وصل نمطية الويب web-style link labels والتي أضفناها للبرنامج من عدة فصول سابقة. الحدث LinkClicked وليس الحدث Clicked هو من يطلق العرض لمحرر الكود. إليك الكود الذي يحرر جدول GroupName.

```

Private Sub AdminLinkGroups_LinkClicked(ByVal sender As Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles AdminLinkGroups.LinkClicked
' ----- Let the user edit the list of security groups.
دع المستخدم يحرر قائمة مجموعات الأمن.
If (SecurityProfile(LibrarySecurity.ManageGroups) = False) Then
MsgBox(NotAuthorizedMessage, MsgBoxStyle.OkOnly Or
MsgBoxStyle.Exclamation, ProgramTitle)
Return
End If
' ----- Edit the records.
تحرير السجلات.
ListEditRecords.ManageRecords(New Library.GroupName)
ListEditRecords = Nothing
End Sub

```

تجربة كل مستخدم. Per-User Experience.

والآن بما أن لدينا كل كود دعم الأمن المضاف إلى المشروع، نستطيع البدء باستخدام هذه الميزات لتغيير تجربة التطبيق من أجل الزبائن والمدراء. ليس من الخلق إغراء الناس بالقوة الهائلة، لذلك من الأفضل إخفاء هذه الميزات والتي هي غير قابلة للوصول بالنسبة للعملاء المستخدمين المتواضعين وبشكل أساسي أقل قوة. أولاً، لنوفر القوة المتنوعة بإضافة نموذج تسجيل دخول إداري، المبين في الشكل التالي.

لقد عملت سابقاً على إضافة النموذج ChangeUser.vb إلى المشروع. معظم العمل الصعب في الفورم يحدث في معالج الحدث ActOK_Click. إذا ما اختار المستخدم "الرجوع إلى نمط العميل"، جميع قيم الأمن يتم تنظيفها، والفورم الرئيسي تخفي معظم الميزات (من خلال كود سيضاف فيما بعد)

```

LoggedInUserID = -1
LoggedInUserName = ""
LoggedInGroupID = -1
ReprocessSecuritySet()

```

تعمل هذه الفورم على الاتصال إلى التطبيق من خلال الفورم الرئيسي وذلك من خلال الحدث ActLogin_Click. افتح ملف الفورم الرئيسي MainForm.vb، انقر مزدوج على زر Login في الزاوية العلوية اليسارية، وأضف الكود التالي لحدث النقر الذي يظهر.

```

Private Sub ActLogin_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
ActLogin.Click

```

```

' حدث المستخدم على اختيار نمط العميل أو النمط الإداري
ShowLoginForm()
End Sub

```

أضف الكود التالي للفورم أيضاً:

```

Private Sub ShowLoginForm()
' حدث المستخدم على اختيار نمط العميل أو النمط الإداري
Dim userChoice As Windows.Forms.DialogResult
userChoice = ChangeUser.ShowDialog()
ChangeUser = Nothing
If (userChoice = Windows.Forms.DialogResult.OK) Then UpdateDisplayForUser()
End Sub

```

لنمكن أيضاً المفتاح F12 ليتصرف كمطلق للدخول login. أضف الكود التالي إلى عبارة "اختر حالة Select Case" في معالج حدث MainForm_KeyDown.

```

Case Keys.F12
' حدث المستخدم من أجل نمط العميل أو النمط الإداري.
ShowLoginForm()
e.Handled = True

```

يستدعي الروتين ShowLoginForm طريقة أخرى، UpdateDisplayForUser، والتي تخفي وتظهر عناصر العرض المتنوعة على الفورم الرئيسي بالاعتماد على شكل الأمن security profile للمستخدم الحالي. أضفها إلى كود فئة النموذج الرئيسي MainForm. وهو ينظر بشكل أساسي إلى المتغير LoggedInUserID، وإذا ما تم وضعه إلى -1، فإنه يخفي جميع الأدوات بالنسبة للميزات المتقدمة.

```

Private Sub UpdateDisplayForUser()
' تحديث العرض بالاعتماد على إعدادات المستخدم
Dim sqlText As String
If (LoggedInUserID = -1) Then
' إعداد العرض من أجل نمط العميل
ActiveMode.Text = "مرحباً بك زبون المكتبة"
LabelTasks.Visible = False
LineTasks.Visible = False
PicCheckOut.Visible = False
ActCheckOut.Visible = False
PicCheckIn.Visible = False
ActCheckIn.Visible = False
PicAdmin.Visible = False
ActAdmin.Visible = False
PicProcess.Visible = False
ActProcess.Visible = False
PicReports.Visible = False
ActReports.Visible = False
ActLogin.Visible = True
' بنود البحث بشكل افتراضي
TaskLibraryItem()
Else
' إعداد العرض من أجل النمط الإداري
Try
sqlText = "SELECT FullName FROM UserName WHERE ID = " & LoggedInUserID
ActiveMode.Text = "Administrative User: " & CStr(ExecuteSQLReturn(sqlText))
Catch ex As Exception
GeneralError("MainForm.UpdateDisplayForUser", Err.GetException())
ActiveMode.Text = "Administrative User: Unknown"
End Try
LabelTasks.Visible = True
LineTasks.Visible = True
PicCheckOut.Visible = True
ActCheckOut.Visible = True
PicCheckIn.Visible = True
ActCheckIn.Visible = True
PicAdmin.Visible = True
ActAdmin.Visible = True
PicProcess.Visible = True
ActProcess.Visible = True
PicReports.Visible = True
ActReports.Visible = True
ActLogin.Visible = True
End If
End Sub

```

حالياً، عندما نشغل التطبيق، جميع الميزات المتقدمة تظهر، حتى ولو لم يكن قد زود مدير بالمعرف ID أو كلمة المرور password. استدعاء MainForm_Load. UpdateDisplayForUser عندما يظهر الفورم الرئيسي للمرة الأولى يحل المشكلة. أضف الكود التالي إلى نهاية الطريقة

```
التحضير للمستخدم العميل
UpdateDisplayForUser()
```

التحديث الأخير (عملياً، خمس تحديثات) يتضمن وضع حدود المقاطع الرئيسية للفورم لضبط المدراء المرخص لهم. على سبيل المثال، فقط المدراء المرخص لهم بتشغيل التقارير يجب أن يكونوا قادرين على الوصول إلى لوحة التقارير على الفورم الرئيسي. أوجد الطريقة TaskReports في الفورم الرئيسي، وأوجد السطر الذي يعرض اللوحة.

```
PanelReports.Visible = True
```

بدل هذا السطر بالكود التالي.

```
If (SecurityProfile(LibrarySecurity.RunReports)) Then PanelReports.Visible = True
```

نحتاج لعمل نفس الشيء في الطرق TaskCheckOut ، TaskAdmin و TaskProcess. في كل حالة بدل السطر الذي يقرأ:

```
Panel???.Visible = True
```

بالكود الذي يختبر إعدادات الأمن قبل إظهار اللوحة.

فبدل السطر التالي PanelCheckOut.Visible = True بالكود التالي من أجل TaskCheckOut.

```
If (SecurityProfile(LibrarySecurity.CheckOutItems)) Then PanelCheckOut.Visible = True
```

وبدل السطر التالي PanelCheckIn.Visible = True بالكود التالي من أجل TaskCheckIn.

```
If (SecurityProfile(LibrarySecurity.CheckInItems)) Then PanelCheckIn.Visible = True
```

وبدل السطر التالي PanelAdmin.Visible = True بالكود التالي من أجل الطريقة TaskAdmin.

```
If (SecurityProfile(LibrarySecurity.AdminFeatures)) Then PanelAdmin.Visible = True
```

بدل السطر التالي PanelProcess.Visible = True بالكود التالي من أجل الطريقة TaskProcess.

```
If (SecurityProfile(LibrarySecurity.DailyProcessing)) Then PanelProcess.Visible = True
```

شغّل البرنامج وسترى أن بدايته تبدو كتطبيق حقيقي. إذا حاولت الوصول إلى الميزات المحسنة، جرب معرف تسجيل الدخول " admin " وبدون كلمة مرور. تستطيع تغيير ذلك من خلال نموذج UserName إذا أردت.

بما أنه لدينا طريقة لتأمين الوصول إلى البيانات والميزات بالنسبة لمشروع المكتبة، لذلك لننتقل إلى الفصل التالي ونبدأ بالتركيز على البيانات، النقطة المركزية لأي تطبيق عمل.

إعادة التعريف والامتدادات Overloads and Extensions

بدايةً، إن الفيچوال بيسك تدعم إعادة تعريف *operator overloading* المعاملات مثل الجمع (+). هذا الفصل سيريك كيف تستطيع توجيه عملية تحسين البناء الجسدي من أجل معاملات الفيچوال بيسك المتنوعة. وسأقدم لك أيضاً طرق الامتداد (التوسيع *extension*)، والتي تتيح لك بشكل مشابه تحسين الفئات، حتى ولو لم يكن عليك الوصول إلى الكود المصدرى الأصلي لهذه الفئات.

ماذا نعني بإعادة تعريف المعامل؟ What Is Operator Overloading?

تتيح عملية إعادة تعريف معاملاً لكدوك تحسين معاملات الفيچوال بيسك الأساسية وتمنحهم إمكانيات غير متوفرة لديهم سابقاً بواسطة المترجم. إعادة التعريف لا تغير البناء المستعمل عند توظيف المعاملات، ولكنها تغير نوع الكائنات التي يديرها كل معاملاً. على سبيل المثال، معاملاً الضرب (*) عادة يتفاعل مع الأعداد فقط، ولكن تستطيع تضخيمه للعمل مع فئة مثل "النحلة الطنانة Bumblebee" التي تخصصها أنت.

```
Dim swarm As Bumblebee
```

```
Dim oneBumblebee As New Bumblebee
```

```
Dim twoBumblebee As New Bumblebee
```

```
swarm = oneBumblebee * twoBumblebee
```

المعنى الذي تطيقه على المعامل المعاد تعريفه عائد لك. على الرغم من أنك عندما تريد أن تحتفظ بالنوع المضاف لمعامل الجمع (الإضافة) عند إعادة تعريفه، فليس عليك عمل هذا. في الحقيقة، تستطيع إعادة تعريف معاملاً الإضافة بحيث يعمل على طرح قيمة من أخرى. ولكنني سأطردك من العمل عندي إذا عملت ذلك، هذا فقط كي تعلم أنه ممكن.

جميع ميزات إعادة تعريف معاملاً مرتبطة بشكل مباشر بواحد أو أكثر من فئاتك. تبدو الميزات المعاد تعريفها مثل أعضاء دالة قياسية تقريباً، وتظهر كأعضاء لفئاتك. تتضمن الفيچوال بيسك نوعين من المعاملات: أحادي *unary* والثنائية *binary*، معرفة تبعاً لعدد العوامل (أطراف المعامل) المميزة بواسطة المعامل. تقبل المعاملات الأحادية طرف (عامل) واحد فقط، والذي يظهر دائماً على يمين اسم المعامل أو الرمز *symbol*. فمنطق المعامل "ليس Not" هو معاملاً أحادي:

```
oppositeValue = Not originalValue
```

تقبل المعاملات الثنائية عاملين (طرفين) واحد على كل جهة من المعامل. فمعاملاً الضرب هو معاملاً ثنائي:

```
ten = two * five
```

طبيعة معاملاً ما هي ما يعمل مباشرة، في الحقيقة يتم استبدال المعامل وعوامله (عامله) المدخلة بالنتيجة المحسوبة. فمثلاً التعبير (10/5) يتم استبداله بالنتيجة المحسوبة (2)، وهذه النتيجة يتم استخدامها لإتمام عبارة ما أو تعبير عملية أصلية يظهر ضمنه المعامل وعوامله. فهو يعمل تماماً كدالة:

```
'السطرين التاليين يعملان نفس الشيء
```

```
'النتيجة المحسوبة في المتغير theAnswer
```

```
theAnswer = 2 * 5
```

```
theAnswer = DoubleIt(5)
```

ليكون معاملاً جاهز لإعادة التعريف، غير تفكيرك لتري المعاملات كدوال. انظر عبر حدود بيئة معاملك، واقتح عقلك إلى حقيقة أن المعاملات والدوال هي واحدة. تعريف إعادة تعريف المعاملات في الفيچوال بيسك كما يلي، إذا كان عليك ترجمة تعريف الضرب إلى دالة فيچوال بيسك، يمكن أن تبدو كالتالي:

```
Public Shared Function *(ByVal firstOperand As Integer, ByVal secondOperand As Integer) As Integer
```

المعامل (*) يصبح اسم الدالة. مع العوامل *operands* التي تلعب دور وسيطات الدالة. أخيراً، توليد القيمة المعروضة من خلال القيمة المعادة بالدالة. على الرغم من أن المعاملات لا يتم تعريفها كدوال بهذه الطريقة في الفيچوال بيسك، ولكن إعادة تعريف هذه المعاملات يتم تعريفها كدوال.

لإعادة تعريف معاملاً الضرب في فئة Bumblebee الخيالية، نستخدم الكلمة المحجوزة *Operator* لتعريف "دالة الضرب multiplication function" من أجل عوامل الفئة Bumblebee.

```
Partial Class Bumblebee
```

```
Public Shared Operator *(ByVal operand1 As Bumblebee, _
```

```
ByVal operand2 As Bumblebee) As Bumblebee
```

```
' ----- Multiply two bumblebees together.
```

```
Dim finalResult As New Bumblebee
```

```
' ----- Add special "multiplication" code here, then...
```

```
Return finalResult
```

```
End Operator
```

```
End Class
```

والآن عندما تضرب حالي Bumblebee مع بعضهما بمعاملاً الضرب، تميز الفيچوال بيسك النموذج "عامل1*عامل2" كمطابقة لإعادة تعريف معاملاً الضرب مع معاملين نسيبين من نوع Bumblebee، وتستدعي هذه الفئة بالاعتماد على دالة المعامل للحصول على النتيجة.

جميع التصريحات عن المعاملات يجب أن تتضمن الكلمة المحجوزة *Public* والكلمة المحجوزة *Shared*. فإذا لم تكن مشاركة، *shared* سيكون على الفيچوال بيسك طلب إنشاء حالة إضافية من الفئة من أجل الوصول لكود إعادة تعريف المعامل فقط، وذلك لن يكون فعال جداً.

ما الذي تستطيع إعادة تعريفه؟ What Can You Overload?

تستطيع إلى حد ما إعادة تعريف أي معاملات قياسية للفيچوال بيسك (ما عدا المعامل *Is* والمعامل *IsNot*) زائد عدة ميزات أخرى. يشرح هذا الفصل كل معاملاً قابل لإعادة التعريف، مصنّف بواسطة النوع العام. كل مقطع يتضمن جدولاً بالمعاملات. لإعادة تعريف معاملاً في فئة، استخدم الاسم في عمود المعامل كاسم للدالة. إذا كان هناك معاملاً مسمى *XX*، ستكون عبارة المعامل المطابقة كما يلي:

```
Public Shared Operator XX(...)
```

المعاملات الرياضية Mathematical Operators

تعرف الفيچوال ببيسك 10 معاملات رياضية أو معاملات رياضية مستعارة (شبه رياضية) pseudo-mathematical. جميعها لمعاملة الأعداد ما عدا واحد، المعامل الباقي هو مسلسل النصوص، string concatenation (&) ولكن يبدو مثل باقي المعاملات الرياضية في بناءه واستخدامه. معاملين من المعاملات، الزائد (+) والناقص (-)، هما معاملات أحادية unary وثنائية binary. إشارة الناقص minus sign تعمل كمعامل "نفي negation" أحادي كما في ("5-") وكمعامل "طرح subtraction" ثنائي (البناء "2-5" المعروف). عندما إعادة تعريف هذه المعاملات، الاختلاف الموضوع في عدد المعاملات النسبية المضمنة في توقيع المعامل يحدد أي منها أحادي أم ثنائي.

Public Shared Operator - (ByVal operand1 As SomeClass) As SomeClass

' ----- This is the unary "negation" version.

End Operator

تجدول القائمة التالية المعاملات الرياضية التي تدعم إعادة التعريف:

المعامل	Operator	النوع Type	التعليق Comments
+		Unary	المعامل "زائد" الأحادي. تستطيع استخدام هذا المعامل مع الأعداد، كما في (+5). ولكن إذا أدخلت هذه القيمة في الفيچوال أستوديو، فإنه سيتم تجريد (إزالة) معامل الزائد، بما أنه يعتبر فائض (زائد عن الحاجة). مهما يكن، إذا عملت على إعادة قيادة هذا المعامل على فئة خاصة بك، فإن الفيچوال أستوديو سيحفظ الصيغة الأحادية لهذا المعامل عند الاستخدام في الكود.
+		Binary	بما أن هذا المعامل أحادي، يتضمن فقط معامل نسبي واحد عند تعريف طريقة المعامل Operator. معامل الإضافة (الجمع) القياسي. تذكر، فقط لأن المعامل يدعى معامل "إضافة" لا يعني أن عليك حفظ ذلك المفهوم، مهما يكن، عليك المحاولة في إعادة تعريف المعاملات كأقرب ما يمكن لمعناه الأصلي قدر الإمكان، الفيچوال ببيسك نفسها تعيد قيادة هذا المعامل لتسمح له بالتصرف قليلاً مثل معامل سلسلة النصوص.
-		Unary	هذا معامل "سليبي (نفي negation)" أحادي والذي يأتي تماماً قبل قيمة أو تعبير.
-		Binary	معامل الطرح subtraction.
*		Binary	معامل الضرب multiplication.
/		Binary	معامل القسمة القياسي standard division operator.
\		Binary	معامل القسمة الصحيحة integer division operator. تذكر أنك لست بحاجة للاحتفاظ بأي شيء "عددي صحيح integer" في هذا المعامل إذا لم يوافق احتياجاتك.
^		Binary	معامل الدليل أو الأس exponentiation ("لقوة من" to the power of)
Mod		Binary	معامل الباقي modulo، بعض الأحيان يدعى معامل الباقي remainder operator لأنه يعود بالباقي من عملية القسمة.
&		Binary	معامل سلسلة النصوص string concatenation operator.

معاملات المقارنة Comparison Operators

تتضمن الفيچوال سبع معاملات مقارنة أساسية، معظمها يتم استخدامه في عبارة If والتعبير المشابهة والتي تتطلب حساب شروط منطقية Boolean. طرق المعامل Operator من أجل معاملات المقارنة هذه لها نفس البناء الذي تم استخدامه في المعاملات الرياضية، ولكن معظمها يجب أن يتم معالجته بشكل زوجي. على سبيل المثال، إذا أعدت تعريف المعامل (أقل من <)، تطلب منك الفيچوال ببيسك إعادة تعريف المعامل (أكبر من >) أيضاً ضمن نفس الفئة، ومن أجل نفس توقيع المعامل النسبي argument signature.

جميع معاملات المقارنة هي معاملات منطقية. على الرغم من أنك تستطيع تبديل نوع البيانات للمعاملات النسبية الممررة للمعامل، ويجب أن تعود كلها بقيمة منطقية.

Public Shared Operator <= (ByVal operand1 As SomeClass, ByVal operand2 As SomeClass) As Boolean

' ----- The <= operator returns a Boolean result.

End Operator

القائمة التالية تجدول سبع معاملات مقارنة أساسية والتي تستطيع إعادة تعريفها. وكل مدخلة تتضمن قيمة "المصاحب buddy" والتي تحدد المعامل الموافق والذي يجب أن يتم إعادة تعريفه.

المعامل	Operator	المصاحب Buddy	التعليق Comments
=	<>		يقارن معاملاً المساواة عاملين operands من أجل التساوي equivalence، ويعود بصواب True إذا كانا متساويين.
<>	=		يقارن معاملاً عدم المساواة عاملين من أجل عدم التساوي، ويعود بصواب إذا كانا غير متطابقين.
<	>		يعود المعامل أقل من بصواب إذا كان العامل الأول "أقل من" less than الثاني.
>	<		يعود المعامل أكبر من بصواب إذا كان العامل الأول "أكثر من" greater than الثاني.
<=	>=		يعود المعامل أقل من أو يساوي إلى بصواب إذا كان العامل الأول "أقل من أو يساوي ل" less than or equal to الثاني.
>=	<=		يعود المعامل أكثر من أو يساوي ل، بصواب إذا كان العامل الأول "أكبر من أو يساوي ل" greater than or equal to الثاني.

معامل المقارنة السايغ هو Like. في الفيچوال بيسك القياسي، يعمل هذا المعامل على مقارنة العامل الأول إلى "نموذج pattern نصي، والذي هو مجموعة من الحروف المتطابقة والقيم الشاملة wildcards .

If (someValue Like somePattern) Then

ليس عليك استخدام نفس قواعد النموذج pattern عند إعادة تعريف المعامل Like وتستطيع أن تقبل أي نوع بيانات من أجل عامل النموذج pattern ، ولكن يجب عليك دائماً العودة بنتيجة منطقية Boolean .

Public Shared Operator Like(ByVal operand1 As Bumblebee, ByVal operand2 As Integer) As Boolean

' ----- See if Bumblebee matches an Integer pattern.

End Operator

لا يوجد معامل "مرافق" buddy " والذي يجب عليك معاملته عند إعادة قيادة المعامل Like.

المعاملات المنطقية Logical و Bitwise على مستوى البت Bitwise and Logical Operators

توجد أربع معاملات من بين المعاملات المنطقية logical والتي على مستوى البت (الوحدة التخزينية) المضمنة في الفيچوال بيسك، تنجز واجب مضاعف كمعاملات معاد تعريفها. تقبل المعاملات على مستوى البت (وحدة تخزينية) وهي (And, Or, Xor و Not) عوامل عديدة صحيحة integer ، وتولد نتائج عديدة مع القيم المنقولة (المرسلة) على مستوى الوحدة التخزينية المستقلة. وهي تعمل أيضاً كمعاملات منطقية، تقبل وتعيد قيم منطقية Boolean، على الأغلب في العبارات الشرطية. ولكن بإمكانها معالجة الضغط كونها قابلة لإعادة القيادة أكثر بقليل. عندما تعمل على إعادة قيادة override هذه المعاملات الأربع، فأنت تعيد قيادة إصدارات على مستوى البت (الوحدة التخزينية) bitwise versions وليس الإصدارات المنطقية logical . بشكل أساسي، هذا يعني أن لديك تحكم على القيمة المعادة، وليس مطلوب جعلها منطقية Boolean . تجدول القائمة التالية ثمانية معاملات على مستوى الوحدة التخزينية والمنطقية القابلة لإعادة التعريف.

المعامل Operator التعليق Comments

< > يعمل معامل الرفع اليساري shift left وحدة تخزينية على قيمة عددية صحيحة مصدرية، نقل الوحدات التخزينية bits إلى اليسار بواسطة عدد معين من المواضع. على الرغم من أنه ليس عليك استخدام هذا المعامل لعمل رفع وحدة تخزينية صحيحة، عليك قبول كمية رفع (ما عددي صحيحة Integer) كعامل ثاني.

Public Shared Operator <<(ByVal operand1 As Bumblebee, ByVal operand2 As Integer) As Bumblebee

' ----- Add shifting code here.

End Operator

> > يعمل معامل الرفع اليميني shift right وحدة تخزينية تماماً مثل معامل الرفع اليساري، ولكن تنقل الوحدات التخزينية في اتجاه "اليمين right". إنني أظن أن هذا سيجعل هذه الوحدات التخزينية أكثر محافظة conservative. يمكن لكودك أن يجعل القيمة المعادة أكثر تساهلاً إذا أردت، ولكن كما مع معامل الرفع اليساري، يتوجب عليك قبول عدد صحيح Integer كعامل ثاني.

Not معامل نفي على مستوى الوحدة التخزينية وهو أحادي، يقبل فقط معامل نسبي كعامل مفرد فقط.

And معامل ربط conjunction على مستوى الوحدة التخزينية يضع وحدة تخزينية bit في القيمة المعادة إذا ما تم إعداد وحدتي تخزين على حد سواء متوضعين في عوامل المصدر.

Or يضع معامل الفصل disjunction على مستوى الوحدة التخزينية بت (وحدة تخزينية) في القيمة المعادة إذا ما تم إعداد الوحدات التخزينية المتوضعة بالتساوي في عوامل المصدر.

Xor معامل الاستثناء على مستوى البت bitwise exclusion operator ، يضع بت في القيمة المعادة إذا فقط إذا كان واحد من البتات المتوضعة بالتساوي في عوامل المصدر تم إعداده.

IsTrue إعادة تعريف المعامل Or لا يعمل على إعادة تعريف المعامل OrElse المتعلق به بشكل آلي. لاستخدام OrElse يتوجب عليك إعادة تعريف المعامل IsTrue الخاص. وهو ليس معامل فيچوال بيسك حقيقي، ولا تستطيع استدعاه بشكل مباشر حتى عندما يتم إعادة تعريفه. ولكن عندما تستخدم المعامل OrElse مكان المعامل Or المعاد تعريفه، تستدعي الفيچوال بيسك المعامل IsTrue عند الحاجة. يوجد عدة قواعد عليك إتباعها لاستخدام إعادة تعريف IsTrue.

المعامل Or المعاد تعريفه يجب أن يعود بنوع الفئة للفئة التي بها تم تعريفه. إذا أردت استخدام OrElse على الفئة Bumblebee، فإن إعادة تعريف المعامل Or في تلك الفئة يجب أن يعود بقيمة من نوع Bumblebee.

المعامل IsTrue المعاد تعريفه يجب أن يقبل عامل مفرد من نوع الفئة المحتوية (Bumblebee)، ويعود بمنطقية Boolean . يتوجب عليك إعادة تعريف المعامل IsFalse .

كيف تحدد الصحة truth أو الزيف من Bumblebee يعود إليك.

IsFalse إعادة تعريف IsFalse يعمل تماماً مثل IsTrue ولديه نفس القواعد، ولكن يقدم إلى المعاملات And و AndAlso .

المعامل "تحويل النوع" CType Operator

تبدو ميزة الفيچوال بيسك CType أكثر شبيهاً لدالة منها لمعامل:

result = CType(source, Type)

ولكن المظاهر خداعة looks are deceiving. فهي ليست دالة حقيقية، وكما مع دوال التحويل الأخرى (مثل CInt)، يتم معالجتها عملياً وقت الترجمة. قبل تشغيل البرنامج بوقت طويل. من خلال السماح لك بإعادة تعريفها كالمعامل، تمكّنك الفيچوال بيسك من إنشاء تحويلات conversions نوعية وخاصة بين أنواع البيانات التي لا تبدو منسجمة compatible. قالب الطريقة التالية يحول قيمة من نوع Bumblebee إلى عددي صحيح Integer .

Public Shared Operator CType(ByVal operand1 As Bumblebee) As Integer

' ----- Perform conversion here, returning an Integer.

End Operator

إذا حاولت كتابة المقطع الأخير من الكود ضمن الفيچوال بيسك، سيعترض complain أنك تفقد إما الكلمة المحجوزة Widening أو الكلمة المحجوزة Narrowing (شاهد الشكل التالي).

```
Conversion operators must be declared either 'Widening' or 'Narrowing'.
Public Shared Operator CType (ByVal operand1 As Bumblebee) As Integer
    ' ----- Perform conversion here, returning an Integer.
End Operator
```

لقد ذكرت تحويلات التوسيع narrowing والتضييق widening في الفصل الثاني، ولكن لنختبرها هنا بعمق أكثر. عندما تحول بين بعض أنواع البيانات الجوهرية في الفيچوال بيسك، يوجد فرصة لأن يفشل التحويل في بعض الأحيان لأن القيمة المصدرية لا تتناسب في القيمة المقصودة. وهذا صحيح عند تحويل قيمة لنوع قصير Short إلى بايت Byte.

```
Dim quiteBig As Short = 5000
Dim quiteSmall As Byte
' سيفشل السطرين التاليين
quiteSmall = quiteBig
quiteSmall = CType(quiteBig)
```

وإنه من الواضح لما يفشل التحويل: المتغير من النوع بايت لا يمكن أن يحفظ قيمة 5000. ولكن ماذا حول هذا الكود؟

```
Dim quiteBig As Short = 5
Dim quiteSmall As Byte
' ----- These next two lines will succeed.
quiteSmall = quiteBig
quiteSmall = CType(quiteBig)
```

إنه يعمل بشكل جيد، بما أن 5 تتناسب ضمن المتغير بايت مع مساحة احتياطية. (إذا كان خيار التدقيق Option Strict تم وضعه إلى فعال On، فإن الإسناد الأول سيفشل في الترجمة)، مع ذلك، لا يوجد شيء يوقفني عن إسناد قيمة 5000 إلى المتغير quiteBig وتجريب الإسناد مرة أخرى. هذا احتمالي potential للفشل خلال التحويل وهذه هي القضية.

عندما يكون للتحويل احتمال الفشل تبعاً لكون البيانات المصدرية غير قادرة على أن تتناسب بالكامل في المتغير المستهدف، فإنه يدعى تحويل التضييق narrowing conversion. تحويلات التضييق narrowing conversion هي حقيقة، وطالما أنك قد اختبرت البيانات قبل التحويل، فلن يكون هناك أي سبب لتقييد مثل هذه التحويلات بشكل ثابت.

تذهب تحويلات التوسيع Widening conversions في اتجاه معاكس. فهي تحدث عندما تتناسب أي قيمة مصدرية في نوع البيانات المصدرية في النوع الهدف دائماً. تحويلات التوسيع ستنتج دائماً طالما أن البيانات المصدرية محققة. تسمح الفيچوال بيسك للتحويلات التوسيع أن تحدث بشكل آلي، وبشكل ضمني. ليس عليك بشكل صريح استخدام CType لإجبار التحويل. إذا كان لديك تحويل توسيع من فئة Bumblebee إلى العددي الصحيح Integer، وقد وضعت خيار التدقيق Option Strict إلى فعال On، فإن الكود التالي سيعمل جيداً:

```
Dim sourceValue As New Bumblebee
Dim destValue As Integer = sourceValue
```

إذا كان التحويل من إلى تضييق، فسيكون عليك إجبار التحويل باستخدام CType وذلك للتأكيد على الفيچوال بيسك أنك تريد عمل هذا:

```
Dim sourceValue As New Bumblebee
Dim destValue As Integer = CType(sourceValue, Integer)
```

عندما تعمل على إنشاء تحويل مخصص مع المعامل CType المعاد تعريفه، يتوجب عليك إعلام الفيچوال فيما إذا التحويل هو توسيع أو تضييق وذلك بإدخال إما الكلمة المحجوزة Widening أو الكلمة المحجوزة Narrowing بين الكلمة المحجوزة Shared والكلمة المحجوزة Operator.

```
Public Shared Narrowing Operator CType (ByVal operand1 As Bumblebee) As Integer
    ' ----- Perform narrowing conversion here.
End Operator
```

Other Operator Overloading Issues.

قضايا إعادة تعريف معامل أخرى. يوجد أيضاً العديد من القواعد التي يجب عليك إتباعها عند إعادة تعريف المعاملات، ولكن في البداية لنلقي نظرة على شبه الفئة Bumblebee المفيدة.

```
Class Bumblebee
    Public Bees As Integer
    Public Sub New()
        ' ----- Default constructor.
        Bees = 0
    End Sub
    Public Sub New (ByVal startingBees As Integer)
```

Mhm76

```
' ----- Assign an initial number of bees.
Bees = startingBees
End Sub
Public Shared Operator +(ByVal operand1 As Bumblebee, ByVal operand2 As Bumblebee) As Bumblebee
' ----- Join bumblebee groups.
Dim newGroup As New Bumblebee
newGroup.Bees = operand1.Bees + operand2.Bees
Return newGroup
End Operator
Public Shared Operator -(ByVal operand1 As Bumblebee, ByVal operand2 As Bumblebee) As Bumblebee
' ----- Separate bumblebee groups.
Dim newGroup As New Bumblebee
newGroup.Bees = operand1.Bees - operand2.Bees
If (newGroup.Bees < 0) Then newGroup.Bees = 0
Return newGroup
End Operator
Public Shared Operator *(ByVal operand1 As Bumblebee, ByVal operand2 As Bumblebee) As Bumblebee
' ----- Create a swarm.
Dim newGroup As New Bumblebee
newGroup.Bees = operand1.Bees * operand2.Bees
Return newGroup
End Operator
Public Shared Widening Operator CType(ByVal operand1 As Bumblebee) As Integer
' ----- Perform conversion here.
Return operand1.Bees
End Operator
End Class
```

الفئة بسيطة لحد ما، فهي موجودة لحفظ عدد بسيط من النحلات. ولكن بإعادة تعريف المعاملات الإضافة، الطرح، الضرب، و"تحويل النوع"، نستطيع استخدام حالات من النحلات مع بناء أكثر واقعية.

```
Dim quiteBig As Short = 5
Dim studyGroup1 As New Bumblebee(20)
Dim studyGroup2 As New Bumblebee(15)
Dim swarmGroup As Bumblebee = studyGroup1 * studyGroup2
MsgBox("The swarm contains " & Cint(swarmGroup) & " bees.")
```

شغل الكود بشكل صحيح يولد الرسالة المبينة في الشكل التالي:



تضمن إعادة تعريف CType والتي تولد عدد صحيح Integer يسمح لي تحويل Bumblebee باستخدام المعامل CInt. وأستطيع أيضاً تغيير السطر الأخير لاستخدام المعامل CType الصحيح.

```
MsgBox("The swarm contains " & CType(swarmGroup, Integer) & " bees.")
```

متطلبات التصريح Declaration Requirements

كما ذكرت سابقاً، يتوجب عليك جعل طرق المعامل عامة ومشاركة Public Shared. ولأن المعاملات المعاد تعريفها تحتاج نوعاً ما اتصال حميم بالفئة الحاوية عليهم، على الأقل واحد من العوامل أو القيمة المعادة يجب أن تطابق نوع الفئة المحتوية. (في بعض إعادة التعريف، تتطلب الفيچوال بيسك أن يكون واحد من المعاملات متطابقة) أي من إعادة التعريف التالي سيعمل تماماً وبشكل جيد، بما أن الـ Bumblebee يتم استخدامها لواحد من المعاملات:

```
Public Shared Operator <=(ByVal operand1 As Bumblebee, ByVal operand2 As Integer) As Boolean
' ----- Compare a bumblebee to a value.
End Operator
Public Shared Operator <=(ByVal operand1 As Date, ByVal operand2 As Bumblebee) As Boolean
```

Mhm76

```
'----- Compare a date to a bumblebee.
```

```
End Operator
```

مهما يكن لا تستطيع وضع كلا العاملين إلى نوع غير النوع Bumblebee في نفس الوقت، وتبقى محتفظاً بإعادة التعريف في فئة Bumblebee.

```
Class Bumblebee
```

```
Public Shared Operator <=(ByVal operand1 As Date,ByVal operand2 As Integer) As Boolean
```

```
'----- هذا لن يتم ترجمته
```

```
End Operator
```

```
End Class
```

إعادة تعريف ما تم إعادة تعريفه. Overloading Overloads

تستطيع إعادة تعريف المعاملات المعاد تعريفها. فيمكنك إضافة تنوعات لعدة توافيق للقيم المعادة والمعاملات النسبية لمعامل معاد تعريفه لفئة مفردة.

```
Public Shared Widening Operator CType(ByVal operand1 As Bumblebee) As Integer
```

```
'----- اعمل تحويل إلى عدد صحيح هنا
```

```
End Operator
```

```
Public Shared Widening Operator CType(ByVal operand1 As Bumblebee) As Date
```

```
'----- اعمل تحويل إلى تاريخ هنا
```

```
End Operator
```

طالما أن توافيق المعامل النسبي argument signatures أو القيمة المعادة مختلف، تستطيع إضافة العديد من إعادة تعريف المعامل كما تشاء. لا تحتاج إلى استخدام الكلمة المحجوزة Overloads أيضاً

طرق التوسيع (التمدد) Extension Methods

ماذا بشأن الرغبة في تعديل سلوك فئة ما، ولكن ليس لديك إمكانية الوصول إلى كودها المصدري؟ تستطيع الاشتقاق منها وتبني فئة جديدة، ولكن هذا ليس مناسباً دائماً. تستطيع الاتصال بالمطور الأصلي وتترجاه من أجل الكود، ولكن بعض هؤلاء المبرمجين بخيلين جداً tight-fisted عندما يتعلق الأمر ببرمجياتهم. يوجد خيار آخر وهو استخدام ميزة الفيچوال بيسك 2008 الجديدة: وهي طرق التوسيع (التمديد) *extension methods*. إليك كيفية العمل:

1. إنك تقرر أي فئة تريد توسيعها بالطرق الجديدة.
 2. تكتب هذه الطرق ضمن وحدة برمجية Module قياسية في كودك المصدري.
 3. تبدأ باستخدام الطرق الجديدة وكأنه تم تضمينها في تعريف الفئة.
- يتضمن نوع بيانات السلسلة الحرفية String (النصية) العديد من الطرق الجاهزة (المبنية داخلياً) والتي تعود بنسخة معدلة لنسخة سلسلة حرفية. على سبيل المثال، في هذا الكود:

```
Dim bossyString As String = "are you talking to me?"
```

```
MsgBox(bossyString.ToUpper())
```

النص الذي يظهر في صندوق الرسالة سيكون كله في الحالة الكبيرة uppercase لأن الطريقة ToUpper تعود بنسخة حالة كبيرة جديدة من حالة (نسخة) النص الأصلي. الطريقة المطابقة ToLower تعمل بطريقة أخرى (تحويل الأحرف إلى الحالة الصغيرة)، ولكن ما أريده بالفعل هو الطريقة "إلى عنوان ToTitle" والتي تكبر فقط الحرف الأول لكل كلمة.

```
MsgBox(bossyString.ToTitle())
```

ولكن فئة السلسلة الحرفية لا تتضمن الطريقة "ToTitle"، ولكن نستطيع إضافتها بفضل طرق التوسيع (التمدد). لإنشاء طريقة تمدد، أنشئ طريقة ضمن وحدة برمجية module قياسية بحيث تقبل نوع البيانات المستهدفة كوسيط أول لها.

```
Module MyExtensions
```

```
<System.Runtime.CompilerServices.Extension()> Public Function ToTitle(ByVal sourceText As String) As String
```

```
Return StrConv(sourceText, VbStrConv.ProperCase)
```

```
End Function
```

```
End Module
```

بالعادة، ستستدعي هذه الدالة كتمرير في السلسلة الحرفية الأصلية

```
MsgBox(ToTitle(bossyString))
```

وذلك الكود سيعمل تماماً، ولكن الزيادة في مواصفة التوسيع (من فضاء الأسماء System.Runtime.CompilerServices) تحول ToTitle إلى طريقة توسيع، توسع نوع البيانات النصية. إن كودك في الحقيقة لا يعدل نص. خلف الكواليس، يعمل مترجم الفيچوال بيسك على تحويل بناء الطريقة المشابهة الجديدة new method-like syntax إلى بناء الدالة المشابهة القديمة old function-like syntax في كل استخدام لـ ToTitle. بنفسها طرق التوسيع لا تعمل الكثير. استدعاء ToTitle(bossyString) لا يختلف عن bossyString.ToTitle(). ولكن كما مع العديد من الميزات الجديدة في الفيچوال بيسك 2008، طرق التوسيع تم إضافتها فقط لرفع raise سعر المنتج. إن ميزة طرق التوسيع الجديدة دعم هام من أجل تقنية لينكو LINQ الجديدة.

مشروع. Project

سنضيف الكثير من الكود في مشروع هذا الفصل إلى تطبيق المكتبة، أكثر من 25% من أساس الكود الكامل. معظمه مشابه للكود الذي أضفناه في الفصول السابقة، يوجد الكثير للقراءة هنا، ولكن كلما أضفت فورم جديد للمشروع تأكد من إلقاء نظرة على كودها لتصبح على اطلاع بعملها الداخلي.

إعادة تعريف تحويل Overloading a Conversion

إن إعادة تعريف معامل أداة مفيدة، وأنا مولع خاصة بإعادة تعريف CType. لنعمل على إضافة إعادة تعريف CType إلى الفئة ListItemData التي صممناها في الفصل الثامن. تعرض هذه الفئة كل من الخاصيات ItemData و ItemText، موفرة إمكانية الوصول إلى السمات النصية textual والعديدية numeric لمحتوى الفئة. وإن هدفها الرئيسي دعم تتبع أرقام ID في أدوات الصندوق المركب وصندوق القائمة. إذا كنا بحاجة لمعرفة رقم ID لبند مختار في أداة صندوق قائمة (لنسمه SomeList)، فإننا نستخدم كود مشابه للتالي:

```
Dim recordID As Integer = CType(SomeList.SelectedItem, ListItemData).ItemData
```

لا يوجد شيء خاطئ في هذا الكود. ولكن إنني أعتقد، "أليس من الجميل تحويل حالة إلى انتغز باستخدام الدالة CInt، وليس علينا التورط بمتغيرات عضو مثل ItemData"؟

```
Dim recordID As Integer = CInt(CType(SomeList.SelectedItem, ListItemData))
```

الكود غير مختلف. ولكن لدعم هذا التحويل، نحتاج إلى إضافة إعادة تعريف CType إلى فئة ListItemData. أفتح ملف تلك الفئة، وأضف الكود التالي كعضو للفئة.

```
Public Shared Widening Operator CType(ByVal sourceItem As ListItemData) As Integer
    ' للتحويل إلى انتغز، ببساطة استخرج عنصر انتغز
    Return sourceItem.ItemData
End Operator
```

هذا بسيط جداً، هذا التحويل الموسع من ListItemData إلى انتغز يعمل على إعادة قسم الانتغز من الحالة. يوجد حوالي أربع أو خمس أمكنة في مشروع المكتبة الحالي بإمكانها الوصول مباشرة إلى العضو ItemData، وهي ليست بتلك الأهمية للعودة وتغييرها. ولكن سنستخدم إعادة تعريف التحويل بشكل متكرر في الكود الجديد المضاف في هذا الفصل.

مميزات الدعم الشاملة. Global Support Features.

نحتاج إلى إضافة العديد من المتغيرات والروتينات الشاملة لدعم الميزات المتنوعة المستخدمة على طول التطبيق. سوف يتتبع متغيران شاملان الإعدادات المخزنة في جدول SystemValue لقاعدة البيانات. أضفهم كأعضاء إلى الوحدة البرمجية General (في General.vb)

```
Public DefaultItemLocation As Integer
Public SearchMatchLimit As Integer
```

يطابق برنامج المكتبة الكتب والبند الأخرى كما تم تخزينها في المواضيع المتعددة مثل التشعبات (الفروع) المتعددة أو مساحات التخزين. يشير DefaultItemLocation إلى أي واحد من هذه المواضيع، من جدول CodeLocation، وهو الافتراضي. تخزن مدخله DefaultLocation لجدول قاعدة البيانات هذه القيمة بشكل دائم. عند البحث عن كتب، مؤلفين، أو أشياء أخرى يمكن أن تنتج آلاف المتطابقات، يدل SearchMatchLimit على العدد الأكبر من المطابقات المعادة بعمليات بحث مثل هذه. ويتم تخزينها كقيم نظام محددة البحث SearchLimit. بما أننا حالياً في الوحدة البرمجية General، أضف دالتي مساعدة إضافيتين:

```
Public Function ConfirmDefaultLocation() As Boolean
    ' العودة بصواب إذا تم تحديد الموضوع الافتراضي.
    Dim locationID As String
    ' الحصول على الموضوع الافتراضي عند الحاجة.
    If (DefaultItemLocation = -1) Then
        locationID = GetSystemValue("DefaultLocation")
        If (locationID = "") Then locationID = "-1"
        DefaultItemLocation = CInt(locationID)
    End If
    ' العودة بنتيجة الحالة.
    If (DefaultItemLocation = -1) Then
        MsgBox("You must first define the default item location.",
            MsgBoxStyle.OKOnly Or MsgBoxStyle.Exclamation, ProgramTitle)
        Return False
    Else
        Return True
    End If
End Function
```

```
Public Function GetCopyDisposition(ByVal copyID As Integer) As String
    ' ----- Given the ID of an item copy, return it's current status.
    ' The available results are:
    ' New Item Copy Error or missing record
    ' Checked In Copy is checked in and available
    ' Checked Out Copy is checked out by a patron
    ' Overdue Copy is checked out and late
    ' Missing Copy has been reported missing
    ' Reference Copy is a reference copy
    Dim sqlText As String
    Dim dbInfo As SqlClient.SqlDataReader
    Dim response As String
    On Error GoTo ErrorHandler
    ' ----- Retrieve information about the book.
    sqlText = "SELECT IC.Reference, IC.Missing, PC.Missing AS PatronMissing, " &
        "PC.DueDate, PC.Patron FROM ItemCopy AS IC LEFT JOIN PatronCopy AS PC " &
        "ON IC.ID = PC.ItemCopy WHERE IC.ID = " & copyID &
        " AND PC.Returned = 0"
```

```

dbInfo = CreateReader(sqlText)
If (dbInfo.Read = False) Then
    ' ----- Perhaps the database, due to the JOIN clause, could not return
    '         the base record of a checked-in book.
    dbInfo.Close()
    sqlText = "SELECT Reference, Missing, NULL AS PatronMissing, " &
              "NULL AS DueDate, NULL AS Patron FROM ItemCopy WHERE ID = " & copyID
    dbInfo = CreateReader(sqlText)
    If (dbInfo.Read = False) Then
        ' ----- What happened to the record?
        dbInfo.Close()
        dbInfo = Nothing
        Return "New Item Copy"
    End If
End If
' ----- We found the record. Check its details.
If (IsDBNull(dbInfo!Patron) = False) Then
    ' ----- This item is checked out.
    If (CBool(dbInfo!Missing) = True) Or (CBool(dbInfo!PatronMissing) = True) Then
        response = "Missing"
    ElseIf (CDate(dbInfo!DueDate) < Today) Then
        response = "Overdue"
    Else
        response = "Checked Out"
    End If
Else
    ' ----- This book is currently checked in.
    If (CBool(dbInfo!Missing) = True) Then
        response = "Missing"
    ElseIf (CBool(dbInfo!Reference) = True) Then
        response = "Reference"
    Else
        response = "Checked In"
    End If
End If
dbInfo.Close()
dbInfo = Nothing
Return response
ErrorHandler:
GeneralError("GetCopyDisposition", Err.GetException())
On Error Resume Next
If Not (dbInfo Is Nothing) Then dbInfo.Close() : dbInfo = Nothing
Return "New Item Copy"
End Function

```

ConfirmDefaultLocation

يتحقق هذا الروتين من وجود مدخلة الموضع الافتراضي DefaultLocation في الجدول SystemValue. ويعمل على إرجاع صواب في حالة النجاح.

GetCopyDisposition

يوفر هذا الروتين وصف قصير للحالة الحالية من نسخة بند مكتبة معين. إنه يحل سجلات العميل والبند، ويعيد واحد من حالات نصوص الكود: نسخة بند جديد New Item Copy، المدخلات Checked In، المخرجات Checked Out، متأخرات Overdue، مفقود Missings، أو دليل (مرجع Reference).

Extending a Framework-Supplied Class. توسيع فئة مزودة من قبل إطار العمل.

ماذا يوجد في الاسم؟، حسناً، إذا كان اسم المؤلف author في مشروع المكتبة، يمكن أن يتضمن الاسم الأول والاسم الأخير first and last names، السابقات prefixes مثل ("د" Dr) ولاحقات suffixes مثل ("القروي")، وتواريخ dates من أجل الميلاد والموت birth and death. بعض من هذه الأجزاء اختيارية، لذلك تنسيق اسم المؤلف عملية متعددة الخطوات. بما أن التطبيق سيحتاج إلى تنسيق أسماء المؤلفين في عدة أماكن على طول الكود، دعنا نضيف روتين مركزي، FormatAuthorName والذي يقوم بهذا العمل.

```

<System.Runtime.CompilerServices.Extension()> Public Function FormatAuthorName(ByRef dbInfo As
SqlClient.SqlDataReader) As String
    ' ----- Given an author record, return the formatted name.
    Dim authorName As String
    On Error Resume Next
    ' ----- Format the name.
    authorName = CStr(dbInfo!LastName)
    If (IsDBNull(dbInfo!FirstName) = False) Then
        authorName &= ", " & CStr(dbInfo!FirstName)
        If (IsDBNull(dbInfo!MiddleName) = False) Then
            authorName &= " " & CStr(dbInfo!MiddleName)
        End If
    End If

```



```

If (IsDBNull(dbInfo!Suffix) = False) Then
    authorName &= ", " & CStr(dbInfo!Suffix)
' ----- Add in the birth and death years.
If (IsDBNull(dbInfo!BirthYear) = False) Or
    (IsDBNull(dbInfo!DeathYear) = False) Then
    authorName &= " ("
    If (IsDBNull(dbInfo!BirthYear) = True) Then
        authorName &= "???"
    Else
        authorName &= CStr(Math.Abs(CInt(dbInfo!BirthYear)))
        If (CInt(dbInfo!BirthYear) < 0) Then authorName &= "BC"
    End If
    authorName &= "-"
    If (IsDBNull(dbInfo!DeathYear) = False) Then
        authorName &= CStr(Math.Abs(CInt(dbInfo!DeathYear)))
        If (CInt(dbInfo!DeathYear) < 0) Then authorName &= "BC"
    End If
    authorName &= ")"
End If
' ----- Finished.
Return authorName
End Function

```

هذا الروتين هو طريقة توسيع تعمل على تمديد الفئة SqlDataReader.SqlClient. تصنع المواصفة Extension المؤهلة بالكامل الاتصال بين التوسع الخاص بنا وفئة إطار العمل المحددة SqlDataReader. منح SqlDataReader المبنية من سجلات في جدول Author، ودالة التنسيق والعودة باسم المؤلف بشكل لائق في التنسيق "Public, John Q, Jr. (1900–1999)". في مكان آخر من التطبيق، يتم استدعاء هذا الروتين وكأنه عضو من حالة قارئ بيانات:

```

dbInfo.FormatAuthorName ( )
FormatAuthorName (dbInfo)

```

نستطيع أن نهمل ميزات طريقة التمديد بالكامل بحذف مواصفة التمديد Extension attribute بكل بساطة. ومن ثم، سيبدو استدعاء تنسيق المؤلف كالتالي:

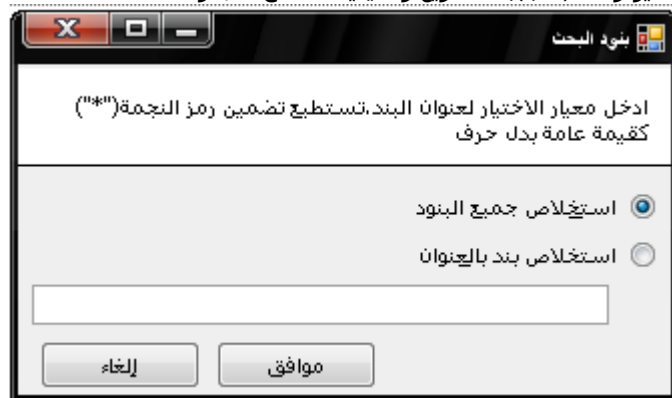
محررات السجلات ودعم النماذج. Record Editors and Supporting Forms.

الآن بدأت الأشياء بالوثوب. سنعمل على إضافة 23 فورم إلى التطبيق في هذا الفصل. معظمها تنفذ محررات الكود القاعدي، مشابهة لملفات *UserName.vb* و *GroupName.vb* التي بنيناها في الفصل 11. نماذج أخرى موجودة لتوفير دعم إضافي لمحررات السجلات هذه.

نماذج تحديد البحث Search-limiting forms

تسمح النماذج الأربع الأولى لأمين المكتبة تحديد إعادة تعريف (تحميل) المعلومات التي تأتي من خلال استخدام قاعدة البيانات مع الآلاف من الكتب books، المؤلفين authors، والناشرين publishers. من المحتمل أنك تتذكر أن النموذج ListEditRecords الشامل يعرض جميع السجلات الموجودة من سجلات جدول بشكل افتراضي. وهذا يعمل بشكل جيد من أجل مجموعات الأمن security groups المخزنة في جدول GroupName بما أنه ومن المحتمل ليس لديك الكثير منها. ولكن جدول جميع الكتب في مكتبة حتى ولو كانت صغيرة يمكن أن يولد قائمة مفروضة تماماً. وبالاعتماد على سرعة كمبيوترك، فيمكن أن يأخذ فترة لتحميل عناوين الكتب ضمن القائمة.

تساعد النماذج الأربع "تحديد-البحث" على تخفيض عدد السجلات التي تظهر في القائمة في الحال. عندما يتمكن أمين المكتبة librarian الوصول إلى قائمة الكتب وينود المكتبة الأخرى، فإن النموذج ItemLimit يوفر بحث (طلب) بحث سريع والذي يقلل النتائج المجدولة.



تتيح هذه الفورم للمستخدم استخلاص جميع السجلات، أو بنود معينة بالاعتماد على اسم بند (مع دعم القيمة الشاملة). حالما يتم تحميل المتطابقات، يمكن للمستخدم الوصول إلى هذه الفورم مرة أخرى بالنقر على "بحث Lookup" على النموذج ListEditRecords من أجل هذه الأنواع من محررات الكود التي تدعم البحث عن المؤلفين، بنود، زبائن، والناشرين). نحن الآن جاهزين لتضمين نماذج تحديد البحث الأربعة في المشروع:

AuthorLimit.vb

تحدد الفورم سجلات المؤلف كما تم تحميلها من جدول المؤلف *AuthorLimit.vb*.

ItemLimit.vb

هذه هي الفورم التي تحدثنا عنها سابقاً. فهي تحدد عرض بنود المكتبة من جدول *NamedItem.vb*.

PatronLimit.vb

في حال احتشاد الزبائن على مكتبك، تتيح لك هذه الفورم تحديد السجلات المحملة من جدول الزبون Patron.

PublisherLimit.vb

تحدد هذه الفورم السجلات من جدول الناشرين Publisher.

محررات الموضوع والكلمات المفتاحية. Keyword and subject editors.

على الرغم من أن معظم محررات السجلات توفر تجربة تحرير كاملة من خلال الفورم ListEditRecords، بعضها خاضع لنماذج تحرير أخرى. فالكلمات المفتاحية Keywords والمواضيع subjects هي مثال جيد. حيث أن لكل منها جداوله الخاصة (الكلمة المفتاحية والموضوع)، اخترت السماح بتحريرها من خلال النموذج الذي يحرر بنود المكتبة المستقلة، أي النموذج NamedItem (سنضيفه فيما بعد). تدير تلك الفورم جميع التفاعلات بين سجلات الموضوع والكلمة المفتاحية وجدول NamedItem. كل هذا من خلال الجداول الوسيطة ItemKeyword و ItemSubject لعلاقة عديد إلى عديد. توفر النموذج KeywordAdd والنموذج SubjectAdd مدخلة نص بسيط من أجل كلمة مفتاحية أو موضوع. اعمل على تضمين كل هذه النماذج الآن في المشروع:

• النموذج إضافة كلمة مفتاحية. *KeywordAdd.vb*

• النموذج إضافة موضوع. *SubjectAdd.vb*

نماذج إضافية لدعم البنود المسماة. More named item support forms.

كما سنرى فيما بعد، إن نموذج NamedItem واحد من النماذج الأكثر تعقيداً المضافة إلى مشروع المكتبة حتى الآن. فهو يدير كل شيء يخص بند مكتبة مولد (مثل كتاب ما). كل بند يمكن أن يكون لديه عدة نسخ، مؤلفين، كلمات مفتاحية، مواضيع، وهكذا. من أجل التحرير الفعال فمن البساطة تضمينها جميعاً على نموذج مفرد. عملنا سابقاً على إضافة نموذجين تابعين (أو ثانويين): KeywordAdd و SubjectAdd. دعنا نعمل على إضافة خمس نماذج دعم إضافية.

AuthorAddLocate.vb

تمثل هذه الفورم واجهة مشابهة لمعالج سحري يتيح للمستخدم إضافة سجل مؤلف جديد أو موجود لبند. والمؤلفين في برنامج المكتبة مصطلح شامل يشير إلى المؤلفين authors، المحررين editors، المفسرين illustrators، الممثلين أو الفنانين performers، وهكذا. خطوات المعالج السحري الثلاث للنموذج تسمح للمستخدم (1) الإشارة إلى نوع المؤلف بواسطة جدول CodeAuthorType. (2) عمل بحث عن مؤلف موجود بالاسم، و (3) اختيار أسماء مؤلفين من قائمة مطابقة. إذا لم يكن المؤلف المطلوب في قاعدة البيانات حتى الآن، تسمح الخطوة الأخيرة بإضافة مؤلف جديد يبين الشكل التالي أو خطوتين من الخطوات الثلاث.

يتم التحكم بمعظم المنطق من خلال معالج حدث زر التالي Next. تشكيلة الكود في هذا الروتين تعتمد على لوحة المعالج الحالي التي هي قيد العرض (كما تم الإشارة إليها من قبل المتغير ActiveForm الذي على مستوى الفئة). إليك الكود الذي يعمل عندما ينقر المستخدم على زر التالي بعد اختيار نوع المؤلف author type :

```
If (CInt(CType(NameType.SelectedItem, ListItemData)) = -1) Then
    MsgBox("من فضلك اختر نوع الاسم من القائمة", MsgBoxStyle.OkOnly Or
    MsgBoxStyle.Exclamation, ProgramTitle)
    NameType.Focus()
    Return
End If
الانتقال إلى لوحة البحث
ActivePanel = PanelCriteria
SecondPanel.Visible = True
FirstPanel.Visible = False
ActBack.Enabled = True
LastName.Focus()
```

الآن ترى سطر المنطق الأول في ذلك الكود؟ إننا نستخدم دالة التحويل CInt للحصول على قيمة ItemData من بند قائمة. وهو استدعاء لمعامل CType الذي عملنا على إعادة تعريفه في فئة ListItemData.

PublisherAddLocate.vb

تشبه هذه الفورم تماماً الفورم السابقة، ولكن تركز على الناشرين. ومعالجها السحري لديه خطوتين فقط بما أن الناشرين غير مجمعين بواسطة النوع type. إنها تعمل على إيجاد أو إضافة سجلات في جدول الناشر Publisher. عندما يحين الوقت لإضافة ناشر إلى بند ما، يستدعي فورم محرر البند الدالة PublisherAddLocate.PromptUser. وتعود هذه الدالة بالمعرف ID لسجل الناشر المختار، أو -1 لإجهاض إضافة ناشر. القيمة الراجعة -2 تعمل على تنظيف أي اختيار سابق.

SeriesAddLocate.vb

تشابه هذه الفورم الفورم PublisherAddLocate، ولكن تنتبئ من أجل السجلات من جدول CodeSeries.

ItemAuthorEdit.vb

حالما يتم إضافة مؤلف إلى بند ما، الطريقة الوحيدة لتغييره إلى مؤلف مختلف هي إزالة المؤلف الغير صحيح، وإضافة المؤلف الصحيح بشكل منفصل من خلال الفورم AuthorAddLocate. ولكن إذا اختار المستخدم ببساطة نوع مؤلف خطأ (مثل "محرر Editor" بدل "مفسر أو شارح Illustrator")، فمن العبء البحث مرة أخرى عن اسم المؤلف لتغيير النوع فقط. يتيح الفورم ItemAuthorEdit للمستخدم من تعديل نوع المؤلف المضاف سابقاً لبند ما. فهي تعدل حقل قاعدة البيانات ItemAuthor.AuthorType.

ItemCopy.vb

من المحتمل أن يكون لدى المكتبة عدة نسخ من كتاب معين، أو سيدي، أو أي بند آخر في مشروع المكتبة، هذا يعني أن كل سجل NamedItem يمكن أن يكون لديه أكثر من سجل ItemCopy مرافق له. كل نسخة يتم تحريرها من خلال الفورم (شاهد الشكل التالي).

على الرغم من أن هذا الكود غير مشتق من BaseCodeForm كما مع المحررات الأخرى، فهو ما يزال لديه العديد من ميزات تلك النماذج، من ضمنها الروتين SaveFormData الذي يكتب السجلات إلى قاعدة البيانات.

واحد من الأشياء المهمة التي تعملها هذه الفورم هو أن لديها دعم لقراءة الأكواد القابلة للقراءة من قبل الآلة. تتصرف العديد من قارئات أكواد الآلة كمقحم wedge إدخال نص كود الآلة للبحث ضمن جدول إدخال لوحة مفاتيح keyboard input stream الكمبيوتر. فأى برنامج يعرض كود الآلة عليه عرض إدخال نص طبيعي. ماسحات حشر أكواد الآلة تزيل (أو تعمل على إلحاق) عودة المشيرة (المفتاح إتر Enter) إلى نهاية أكواد الآلة المنقولة. يسمح هذا للبرنامج من اكتشاف نهاية رقم كود الآلة. ولكن في معظم نماذج برنامج المكتبة، المفتاح إتر Enter يعمل على إطلاق الزر "موافق OK" ويغلق الفورم. نحن لا نريد أن يحدث هذا هنا. لمنع هذا، سنعمل على إضافة بعض الكود لهذه الفورم ليعمل على تعطيل النقر الآلي على الزر "موافق OK" كلما كانت نقطة الإدخال في حقل إدخال نص كود الآلة.

```
Private Sub RecordBarcode_Enter(ByVal sender As Object, ByVal e As System.EventArgs)
Handles RecordBarcode.Enter
    'تعليم كامل النص'
    RecordBarcode.SelectAll()
    'لاتسمح لانتر بإغلاق الفورم'
    Me.AcceptButton = Nothing
End Sub
Private Sub RecordBarcode_Leave(ByVal sender As Object, ByVal e As System.EventArgs)
Handles RecordBarcode.Leave
    'اصح لانتر بإغلاق الفورم مرة أخرى'
    Me.AcceptButton = ActOK
End Sub
Private Sub RecordBarcode_KeyPress(ByVal sender As Object,
ByVal e As System.Windows.Forms.KeyPressEventArgs)
Handles RecordBarcode.KeyPress
    'تجاهل مفتاح انتر'
    If (e.KeyChar = ChrW(Keys.Return)) Then e.Handled = True
End Sub
```

مع هذا الكود، عندما يضغط المستخدم على المفتاح إتر في حقل كود الآلة بشكل يدوي، فإن الفورم لن يتم إغلاقها. ولكنه ثم بسيط تدفعه من أجل دعم كود الآلة.

محررات الكود المشتق. Inherited code editors.

12 من النماذج المضافة في هذا الفصل تُشتق مباشرة من فئة الفورم BaseCodeForm. أضفها إلى المشروع كما ساراجع كل منها:

Author.vb

يحدث نموذج المؤلف Author السجلات في جدول قاعدة البيانات Author. وكفئة مشتقة من الفورم BaseCodeForm. فإنها تعمل على إعادة قيادة العديد من العناصر العامة لفتتها القاعدية. اثنان من إعادة القيادة لم نعمل على استخدامهما من قبل في الفصول السابقة وهما الطريقة UsesSearch و SearchForRecord. تسمح هذه الطرق لمستخدم نموذج ListEditRecords تحديد المؤلفين المعروضين من خلال انبثاق النموذج AuthorLimit الذي تم شرحه سابقاً في هذا الفصل. (إعادة قيادة FillListWithRecords يستدعي أيضاً SearchForRecord للطلب من المستخدم من أجل القائمة الأولية من المؤلفين لكي يتم عرضها). في SearchForLimit، استدعاء إلى الطريقة AuthorLimit.PromptUser يعمل على إرجاع نص مفصول بفواصل في تنسيق "الأول First"، "الأخير Last".

```
'الطلب من المستخدم تحديد اسم المؤلف'
exceededMatches = False
userLimit = (New AuthorLimit).PromptUser()
If (userLimit = "") Then Return
```

يمكن للمستخدم تضمين رمز النجمة (*) كقيمة شاملة wildcard في بداية أو نهاية أجزاء الاسم. لقد أصبحت علامة النجمة حرف عام للاستخدام في جميع أنواع بحث القيمة الشاملة wildcard. لسوء الحظ، إنها غير مدعومة في عبارات السكول سيرفر "اختر SELECT". يستخدم السكول سيرفر رمز النسبة المئوية (%) من أجل القيمة الشاملة (كما تعمل العديد من منصات قاعدة البيانات الخاضعة لسكول) بما أن SearchForLimit يستخرج الأسم الأول والأخير، فهو يضمن استخدام حرف شامل مناسب.

```
'استخدم التعديلات للمساعدة في تحضير النص'
limitLast = Trim(GetSubStr(userLimit, ",", 1))
limitFirst = Trim(GetSubStr(userLimit, ",", 2))
If ((limitLast & limitFirst) = "") Then Return
If (InStr(limitLast, "*") = 0) Then limitLast &= "*"
If (InStr(limitFirst, "*") = 0) Then limitFirst &= "*"
limitLast = Replace(limitLast, "*", "%")
limitFirst = Replace(limitFirst, "*", "%")
```

هذا الكود يستخدم روتيننا الخاص GetSubStr الذي أضفناه سابقاً إلى الوحدة البرمجية General. حالما يتم استخراج أجزاء الاسم، تعمل دالة الفيچوال بيسك Replace على استبدال جميع نسخ * ب % . ستجد كود مشابه في محررات السجلات الأخرى التي تسمح بتحديدات على قائمة السجلات. مثل نموذج المؤلف Publisher الذي سنضيفه فيما بعد.

بينما يكون الكود المصدري لهذه الفورم مفتوح انتقل للأعلى. ستجد هناك عبارة Imports الهامة.

```
Imports MVB = Microsoft.VisualBasic
```

عادة تكون عبارة Imports متبوعة مباشرةً بفضاء أسماء. هذا التنوع يتضمن السابقة MVB =، والتي تعرف مختصر لفضاء الأسماء Microsoft.VisualBasic من أجل الكود في هذا الملف. مع إحضار الفيچوال بيسك العديد من فضاءات الأسماء ضمن فئة موجودة تعرف أيضاً الكثير من الأعضاء العامة، يوجد حصر (تقييد) لأن تحصل تضاربات في اسم عضو. في هذه الحالة، التضارب هو عضو الفورم Left أو RIGHT. بما أن هذا الكود المصدري لنموذج المؤلف Author يرى كل شيء من خلال منشور تلك الفورم، عندما تضمن الكلمة المحجوزة Left في منطقتك، فإن الكود عادة يفترض أنك تعني خاصية الفورم Left، والتي تعمل على وضع الموقع اليساري للفورم. المشكلة أن Left هي دالة معالجة نصوص مشتركة تستخرج حروف الجهة اليسارية من سلسلة حرفية أكبر.

```
smallerString = Left(largerString, 5)
```

في نموذج ما، هذا الكود يولد خطأ بما أن الاعتقاد أن المقصود منه Me.Left لاستخدام نسخة السلسلة الحرفية ل Left، عليك اسبقها بفضاء أسمائها.

```
smallerString = Microsoft.VisualBasic.Left(largerString, 5)
```

تتيح لك عبارة Imports الخاصة استخدام اسم أقصر بديل من فضاء الأسماء Microsoft.VisualBasic الطويل نوعاً ما.

```
smallerString = MVB.Left(largerString, 5)
```

ستجد عدة حالات مشابه لهذا الكود في نماذج أخرى تتضمن الاختصار MVB.

نموذج المؤلف لديها عنصر أكثر أهمية، تظهر لصق مطابقة الاسم Name Matches قرب الحافة السفلية للفورم. كما هو مبين في الشكل التالي:



يساعد هذا الحقل المستخدم على تجنب إضافة نفس المؤلف إلى قاعدة البيانات مرتين. فعندما يتم عمل تغييرات على حقول الاسم الأخير والأول، يطابق الاسم حقل يتم تحديثه بأسماء المؤلفين المتطابقة الموجودة في جدول المؤلف Author. يحصي الروتين RefreshMatchingAuthors عدد المؤلفين المتطابقة من خلال الكود التالي:

```
sqlText = "SELECT COUNT(*) AS TheCount " & "FROM Author WHERE LastName LIKE " &
DBText(Trim(RecordLastName.Text))
If (Trim(RecordFirstName.Text) <> "") Then
sqlText &= " AND FirstName LIKE " & DBText(MVB.Left(Trim(RecordFirstName.Text), 1) & "%")
End If
matchCount = CInt(ExecuteSQLReturn(sqlText))
```

وهذا مشابه لكود البحث في الروتين SearchForLimit، ولكنه يضيف فقط قيمة شاملة wildcard إلى الاسم الأول قبل عمل البحث.

CodeAuthorType.vb

تحرر هذه الفورم السجلات المتعلقة بجدول CodeAuthorType.

CodeCopyStatus.vb

تحرر هذه الفورم السجلات في جدول قاعدة البيانات CodeCopyStatus.

CodeLocation.vb

كما هو متوقع، تحرر هذه الفورم السجلات في جدول CodeLocation. حالما تعمل على إضافة سجل واحد على الأقل على ذلك الجدول، ستكون قادر على وضع الموقع الافتراضي لقاعدة البيانات. سأناقشه مرة أخرى فيما بعد في هذا الفصل.

CodeMediaType.vb

تحرر هذه الفورم السجلات في الجدول CodeMediaType. تتضمنه القليل من السجلات الإضافية أكثر من محررات جداول الكود Code. معظم الحقول تقبل إدخال عددي، على الرغم من أنني أعمل تفحص نهائي من أجل البيانات العددية المحققة (أو الصحيحة) تماماً قبل كتابة سجل إلى قاعدة البيانات، أحاول منع أي بيانات غير

عديدة من الظهور في الموضوع الاول وذلك بتقييد ضربات المفاتيح المقبولة. على سبيل المثال، حدث ضغط مفتاح حقل نص RecordCheckoutDays text field's KeyPress يتضمن الكود التالي:

```
'السماح للأرقام والباك سبيس فقط
If (e.KeyChar = vbBack) Or (sNumeric(e.KeyChar)) Then Return
e.Handled = True
```

وضع الخاصية e.Handled إلى صواب True يوقف الفيجوال بيسك عن عمل أي شيء آخر مع المفتاح. إننا طريقة سهلة وسريعة للتخلص من المفاتيح المدخلة من قبل المستخدم.

CodePatronGroup.vb

تحرر هذه الفورم السجلات في جدول CodePatronGroup.

CodeSeries.vb

هذا المحرر يدير السجلات في جدول CodeSeries. ذكرت سابقاً كيف يتم اتباع الكلمات المفتاحية keywords وأسماء السلاسل series names بنود مسماة named items. ولكن كان من الواضح لي أيضاً توفير إدارة مباشرة لأسماء السلاسل. في حال أردت تركيب قائمة مشتركة قبل إضافة بنود مكتبة مستقلة. لذلك، تعمل هذه الفورم واجبين مضاعفين: تستطيع الوصول لها كمحرر سجلات قياسي من خلال الفورم ListEditRecords، ويتم استخدامها أيضاً من أجل بند مسمى بعينه من خلال نموذج NamedItem الغير مضافة حتى الآن.

عند تحرير أسماء سلسلة ليند معين، يعمل المستخدم أولاً بحث عن اسم السلسلة بكتابتها. بما أنني لا أريد من المستخدم إعادة كتابة اسم السلسلة مرة أخرى في نموذج هذا المحرر، فأردت تمرير اسم السلسلة المكتوبة إلى نموذج CodeSeries، ولكن ولا واحد من الطرق المعاد قيادتها تدعم هذا. لذلك سنحتاج إلى إضافة طريقة جديدة تقبل الاسم المكتوب. العضو AddRecord الذي يعيد قيادة الدالة القاعدية لنفس الاسم.

```
Public Overrides Function AddRecord() As Integer
' إضافة سجل جديد
ActiveID = -1
PrepareFormFields()
Me.ShowDialog()
If (Me.DialogResult = Windows.Forms.DialogResult.OK) Then
Return ActiveID Else Return -1
End Function
```

دعنا نضيف إعادة تعريف لهذه الدالة بحيث تتضمن معاملاً نسبي نصي. سيمرر المستدعي النص المكتوب الأصلي إلى هذا المعامل النسبي. سنعمل على إسناده إلى خاصية النص Text للأداة RecordFullName بحيث تظهر بشكل ألي عند فتح الفورم.

```
Public Overloads Function AddRecord(ByVal seriesText As String) As Integer
ActiveID = -1
PrepareFormFields()
RecordFullName.Text = seriesText
Me.ShowDialog()
If (Me.DialogResult = Windows.Forms.DialogResult.OK) Then Return ActiveID Else Return -1
End Function
```

كان بإمكاننا استخدام اسم ما بدل AddRecord لهذه الدالة وتجنب إضافة إعادة التعريف overload. ولكن من الجميل الحفاظ على الأشياء مستقرة (أو متطابقة). Holiday.vb

تدير هذه الفورم السجلات في جدول. فيما بعد في فصل لاحق سنعمل على إضافة مجموعة من العطل ضمن البرنامج من أجل سرعة الوصول. Patron.vb

نموذج الزبون توفر خدمات تحرير للسجلات في جدول الزبون Patron، وتظهر في الشكل التالي.

The screenshot shows a Windows form titled 'تحرير الزبون' (Edit Patron). It contains several input fields and a checkbox. At the top, there's a checkbox labeled 'تفعيل' (Activate) which is checked. Below it, there are two text boxes for 'الاسم الأول:' (First Name) and 'الاسم الأخير:' (Last Name). A green checkmark indicates 'لتوجد أسمه مطابقة' (Name found). Below that, there's a label 'الفعاليات الأخيرة: غير متاح' (Recent activities: Not available). There are two tabs: 'الرسائل' (Messages) and 'عام' (General), with 'الرسائل' selected. The form has several text boxes for 'قيمة شاملة:' (Full value), 'كلمة المرور:' (Password), 'تأكيدها:' (Confirm), 'الهاتف:' (Phone), 'البريد:' (Email), 'الحيوان:' (Pet), 'مدينة، ش.ص.ب.' (City, P.O. Box), and 'مجموعة الزبون:' (Patron group). At the bottom, there are three buttons: 'إلغاء' (Cancel), 'موافق' (OK), and 'إدارة بنود الزبون...' (Manage patron items...).

تتضمن هذه الفورم أداة تبويب TabControl للمساعدة على تقسيم عدد الحقول التي على المستخدم تجربتها في الحال. إذا كنت قد استخدمت أداة التبويب tab المضمنة مع الفيجوال بيسك 6، ستقدر بسرعة استبدال الدوت نت. فهو يدير جميع منطق تبديل اللوحات بشكل ألي عندما يختار المستخدم تبويب مختلف. كل لوحة هي حالة فنة لصفحة تبويب tabPage منفصلة. في كودك، إجبار أداة التبويب على عرض تبويب مختلف سهل كإسناد حالة صفحة تبويب tabPage مناسبة إلى الخاصية SelectedTab للكتل TabControl. كما مع سطر الكود من الدالة ValidateFormData.

```
TabPatron.SelectedTab = TabGeneral
```

على الرغم من أن هذه الفورم قد تبدو معقدة تماماً، فهي مركبة بالكامل من معظم الكود الذين رأينا في النماذج الأخرى. خلف إعادة القيادة لأعضاء BaseCodeForm، تتضمن هذه الفورم دعم تفحص (أو مسح) كود التعريف bar code (كود تعريف الآلة على أداة ما) المستعارة من النموذج ItemCopy، منطوق كلمة السر password المسروقة من فورم UserName، والكود المطابق لاسم مشابه إلى ما هو مستخدم في نموذج Author. عملت على تضمين زر إدارة بنود زبون Manage Patron's Items على الفورم، ولكن لن نعمل على إضافة منطقة حتى فصل متأخر. وعند ذلك الوقت ستصبح الدالة العامة الإضافية EditRecordLimited هامة في ذلك الوقت.

Publisher.vb

تتيح هذه الفورم للمستخدم تحرير سجلات جدول الناشر Publisher. وهي نموذج بسيط نسبياً مع حقلي إدخال للبيانات فقط. يشير حقل "الحالة Status" إلى كيفية اتصال عدة سجلات "بند مسمى NamedItem" إلى هذا الناشر. يظهر زر صغير إلى اليسار من حقل إدخال النص من أجل موقع الناشر على الويب. وهذا الزر "أظهر موقع الويب ShowWeb"، وعند نقره، فإنه يظهر صفحة الويب الموفرة إلى المستعرض الافتراضي للمستخدم. لتمكين هذا الزر، أضف الكود التالي على معالج حدث نقر زر "أظهر الويب".

```
' أظهر موقع الانترنت المعروف في الحقل
Dim newProcess As ProcessStartInfo
On Error Resume Next
If (Trim(RecordWeb.Text) = "") Then Return
newProcess = New ProcessStartInfo(Trim(RecordWeb.Text))
Process.Start(newProcess)
```

SystemValue.vb

يعالج محرر هذا الكود البنود في جدول SystemValue. على الرغم من أننا سنصله إلى وصلة على نموذج المكتبة الرئيسي في هذا الفصل، سنعمل على تغيير طريقة إمكانية الوصول في فصل لاحق. حسناً، هذه 11 فورم من 12 فورم مشتقة. الفورم الأخيرة هي الفورم NamedItem، المبينة في الشكل التالي.



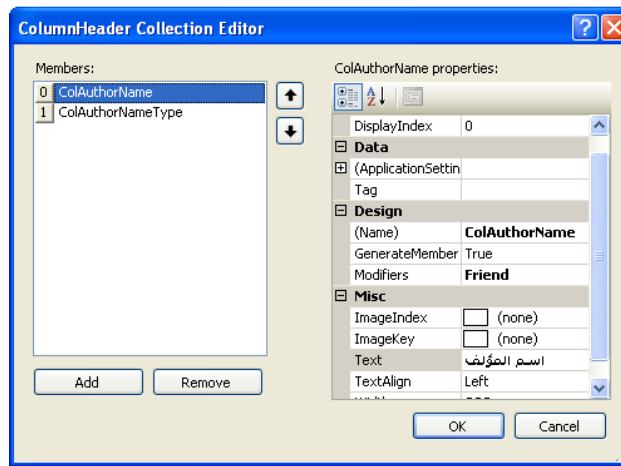
إن الفورم NamedItem هي الفورم الأكبر والأكثر تعقيداً من بين النماذج المشتقة من النموذج BaseCodeForm. فهي تحرر بنود المكتبة الرئيسية المسجلة في جدول قاعدة البيانات NamedItem. وهي معقدة لأنها تدير بشكل مباشر أو غير مباشر السجلات في جداول تابعة (أو ثانوية) أخرى: ItemCopy، ItemAuthor، ItemKeyword، ItemSubject، و Publisher، Keyword، Author، و Subject. جميع الحقول على التتويب "عام" وتبويبات التصنيف تعتمد على حقول المدخلات التي تتدفق مباشرة إلى جدول NamedItem، تماماً كما تم عمله مع نماذج تحرير السجلات الأخرى. تستخدم حقول السلاسل Series والناشر Publisher نماذج اختيار منفصلة (PublisherAddLocate و SeriesAddLocate) للحصول على قيم المعرف ID المخزنة في NamedItem. إليك الكود الذي يبحث عن الناشر:

```
' سؤال المستخدم
newPublisher = (New PublisherAddLocate).PromptUser()
If (newPublisher = -1) Then Return
' الاختيار لتنظيف الناشر
If (newPublisher = -2) Then
RecordPublisher.Text = "Not Available"
PublisherID = -1
Return
End If
```

التبويبات الأربع الباقية - المؤلفين/الأسماء Authors/Names، المواضيع Subjects، الكلمات المفتاحية Keywords، والنسخ Copies - تدير السجلات الثانوية. الكود ثابت (أو متطابق) إلى حد ما بين التبويبات الأربع، لذلك سوف سأقتصر في تعليقي على التتويب مؤلفين/أسماء (شاهد الشكل التالي).



الأدوات على هذا التيوب مشابهة لتلك التي على النموذج ListEditRecords، فهي موجودة لإدارة مجموعة من السجلات في جدول ما. في هذه الحالة، إنه الجدول ItemAuthor. من أجل تقديم القائمة، سأختار استخدام الأداة "عرض القائمة ListView" بدلاً من أداة صندوق القائمة ListBox القياسية. بوضع خاصية عرض View لأداة "عرض القائمة ListView" إلى "تفاصيل Details"، بوضع حقلها FullRowSelect إلى صواب True، وتعديل مجمع أعمدها Columns (شاهد الشكل التالي) تستطيع بسرعة تبديلها إلى صندوق قائمة listbox متعدد الأعمدة multicolumn.



عندما تعمل على إضافة بند ما إلى هذه القائمة، سيكون عليك أيضاً إضافة "نود فرعية subitems" ليكون ليظهر لديك كل شيء ولكن جميعها في العمود الأول.

```
Dim newItem As Windows.Forms.ListViewItem = AuthorsList.Items.Add("John Smith")
newItem.SubItems.Add("Illustrator")
```

يعمل الزر "إضافة Add" على إظهار النموذج AuthorAddLocate، بينما يعرض الزر "خصائص Properties" الفورم ItemAuthorEdit. قبل أن يتم إضافة أي سجل ثانوي، يجب أن يتواجد السجل "الأب parent" في قاعدة البيانات. وهذا لأن السجلات "الأبناء child" تتضمن رقم المعرف ID لسجل الأب، وبدون سجل الأب، فلا يوجد رقم معرف ID الأب. إذا نظرت على كل روتينات الزر "إضافة Add" على هذه الفورم، ستجد كود مشابه للتالي:

```
' يجب ان يتم حفظ السجل أولاً
If (ActiveID = -1) Then
    ' التأكيد على المستخدم
    قبل أن يتم إضافة المؤلفين أو " & " يجب ان يتم حفظ البند إلى قاعدة البيانات"
    If (MsgBox("هل تريد حفظ السجل الآن؟", MsgBoxStyle.YesNo Or MsgBoxStyle.Question,
    ProgramTitle) <> MsgBoxResult.Yes) Then Return
    ' التحقق وحفظ البيانات
    If (ValidateFormData() = False) Then Return
    If (SaveFormData() = False) Then Return
End If
```

إذا كان هذا سجل NamedItem جديد تماماً (ActiveID = -1) فإن هذا الكود سيحفظه قبل السماح للمستخدم إضافة سجل تابع. أي بيانات غير صحيحة تمنع من أن يتم حفظ سجل سيتم التقاطها في الاستدعاء للروتين ValidateFormData الفعلي، والاستدعاءات لكلا الروتينين ValidateFormData و SaveFormData يحدث نفس الشيء عندما ينقر المستخدم على الزر "موافق OK". عادةً، يعمل هذا على إطلاق إعادة رقم معرف ID سجل جديد على النموذج المستدعي. ولكن ماذا بخصوص إذا ما تم استدعاء SaveFormData بواسطة إضافة مؤلف، ولكن بعدها نقر المستخدم الزر "إلغاء Cancel" (والذي يعيد عادةً القيمة 1- ليشير على أنه لم يتم إضافة سجل)؟ لتجنب ذلك، تعمل الدالة SaveFormData على وضع متغير على مستوى الفئة مسمى SessionSaved.

```
SessionSaved = True
```

يتم تنظيف هذه العلامة عندما يتم فتح الفورم للمرة الأولى، ولكن يتم وضعها إلى صواب نسبياً في أي وقت يتغير سجل ثانوي. تعمل النموذج NamedItem على إعادة قيادة الدوال AddRecord و EditRecord وتتفحص هذه العلامة قبل إعادة قيمة إلى النموذج المستدعي.

```
If (Me.DialogResult = Windows.Forms.DialogResult.OK) Or (SessionSaved = True) Then Return
ActiveID Else Return -1
```

يوجد الكثير من الكود الهام في الفورم NamedItem. سأترك لك تفحص كودها.

ربط المحررات إلى الفورم الرئيسي. Connecting the Editors to the Main Form.

إذا شغلت التطبيق الآن، لن ترى أي اختلاف على الإطلاق. ما نزال نحتاج إلى وصل جميع محررات السجلات إلى الفورم الرئيسي. جميعها ترتبط من خلال الأدوات LinkLabel على لوحة الإدارة للفورم الرئيسي (PanelAdmin). نحتاج إلى إضافة 12 معالج حدث نقر الوصلات LinkClicked للتمكن من الوصول إلى جميع النماذج المتنوعة الجديدة. تقدم للأمام وأضفها الآن إلى الفئة الرئيسية.

كل من معالجات حدث LinkClicked على الأغلب صورة عن الأخرى، ما عدا القليل من أسماء حالة كائن هنا وهناك. إليك الكود الذي يعالج النقر على وصلة الناشر:

```
Private Sub AdminLinkPublishers_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs)
Handles AdminLinkPublishers.LinkClicked
    If (SecurityProfile(LibrarySecurity.ManagePublishers) =
False) Then
        MsgBox(NotAuthorizedMessage, MsgBoxStyle.OkOnly Or
MsgBoxStyle.Exclamation, ProgramTitle)
        Return
    End If
    ListEditRecords.ManageRecords(New Library.Publisher)
    ListEditRecords = Nothing
End Sub
```

بعد عمل اختبار سريع للأمن، فإن الكود يستدعي نموذج ListEditRecords القياسي، ممرراً حالة من محرر سجلاته ليتم استخدامها. ما تزال هناك وصلات غير فعالة على لوحة الإدارة والتي سنعمل على تمكينها في فصول لاحقة.

إعداد الموقع الافتراضي. Setting the Default Location.

إن البرنامج الآن جاهز للتشغيل مع جميع الميزات الجديدة. بما أننا عملنا على إضافة الميزات الإدارية، عليك نقر الزر "تسجيل الدخول Login" في الزاوية العلوية اليسارية للفورم الرئيسية قبل أن تتمكن من الوصول إلى لوحة الإدارة وميزاتها. ما لم تغيره، فإن اسم مستخدم تسجيل الدخول هو "admin" بدون الحاجة إلى كتابة كلمة المرور. على الرغم من أنك تستطيع الآن تشغيل البرنامج والوصول إلى محررات السجلات، لن تكون قادر على إضافة نسخة بند جديد item copies حتى تضع الموقع الافتراضي default location. لوضع الموقع الافتراضي:

1. أضف على الأقل موضع واحد من خلال وصلة "المواضع Locations" على لوحة الإدارة.

2. احصل على رقم المعرف ID لسجل CodeLocation الذي تريد أن يكون افتراضي.

تستطيع استخدام ميزات الاستعلام لمنصة إدارة سكول سرفر السريعة SQL Server Management Studio Express للوصول إلى السجلات في هذا الجدول. إذا كانت هذه المرة الأولى التي تصيف فيها سجلات إلى جدول CodeLocation، فإن البند الأول سيكون له قيمة ID رقم المعرف 1.

3. ارجع إلى برنامج المكتبة، حرر جدول SystemValue من خلال وصلة قيم النظام System Values على لوحة الإدارة.

4. أضف أو عدل قيمة نظام "DefaultLocation"، وضع قيمتها إلى رقم المعرف ID لسجل الموقع الافتراضي.

بشكل بديل، تستطيع تحديث سجل "الموقع الافتراضي DefaultLocation" في جدول قيم النظام SystemValue مباشرة باستخدام منصة إدارة سكول سرفر السريعة SQL Server Management Studio Express. إذا كان المعرف ID للموقع location المستخدم هو 1، استخدم عبارة سكول هذه لعمل التغيير:

```
Update (SystemValue)
SET ValueData = '1'
WHERE ValueName = 'DefaultLocation'
```

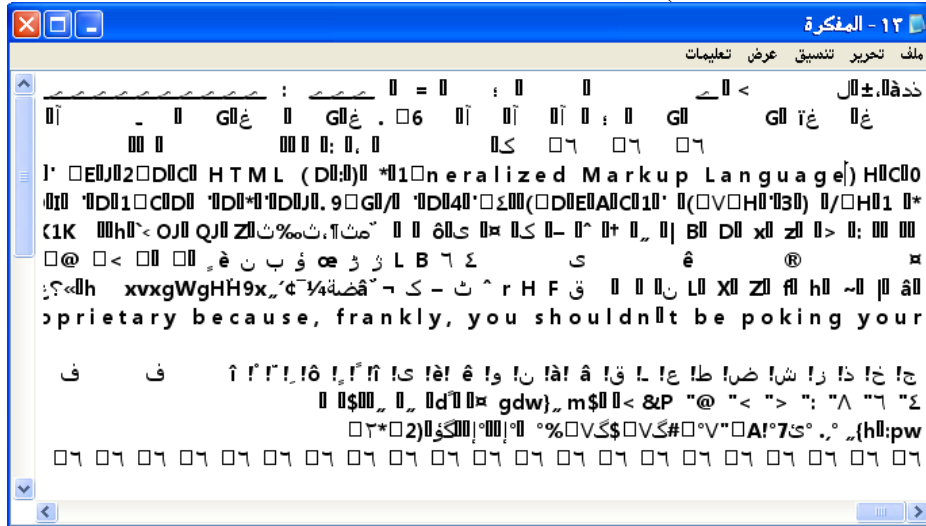
في فصل لاحق، سنعمل على إضافة طرق قريبة من المستخدم لتحديث هذا الموقع الافتراضي.

لغة الترميز القابلة للتوسع XML (Extensible Markup Language)

XML هي محاولة لتنظيم البيانات في تركيب قابل للاستخدام من قبل الناس والبرمجيات، وقد راجت في السنين الأخيرة، ولكن جذورها قديمة جداً. فقد تم اشتقاقها من SGML (لغة الترميز المعممة المعيارية Standard Generalized Markup Language) وكذلك HTML (لغة ترميز النصوص الفائقة) (الدمجة) (Hyper Text Markup Language) القريبة منها و SGML بدورها أتت من GML (لغة الترميز المعممة Generalized Markup Language)، وهي "لغة التوصيف metalanguage (لغة تصف لغة أخرى) تم تصميمها بواسطة IBM في عام 1960. تمثل أبجدية تقنيات معالجة البيانات، الأبجدية التي لديها سبع محاور Xs غريبة.

ماهي لغة الترميز القابلة للتوسع؟ What Is XML?

XML هي لا شيء أكثر من تنسيق بيانات بحيث تكون قابلة للقراءة من قبل الإنسان وقابلة للقراء من قبل الآلة. هل جربت فتح مستند ميكروسوفت ورد بواسطة المفكرة؟ شاهد الشكل التالي. على الرغم من أنك تستطيع عادةً التحقق sift out من النص الرئيسي للمستند، فمعظم ما تراه غير مفهوم (هراء). وهذا لأنه في تنسيق ثنائي خاص proprietary binary format. وهو خاص proprietary لأن، وبصراحة، ليس عليك حشر أصابعك فيه. وهذا ما عليه ميكروسوفت ورد. وهو ثنائي binary لأنك تستطيع تخزين الكثير من المعلومات بشكل مناسب في مساحة صغيرة من القرص. يمثل هذا الملف، أستطيع أن أأخذ بياناتي بأي طريقة أختارها في الحقيقة أستطيع تدوين بياناتي طوعاً أو كرهاً willy-nilly، وليس عليّ الحصول على ترخيص من أي واحد، لأنها تخصني، جميعها لي. (يبين الشكل التالي محاولة فتح مستند ورد لهذا الفصل بواسطة المفكرة)



الملفات الثنائية Binary files عظيمة من أجل تخزين أي نوع من البيانات: أعداد، سلاسل حرفية، صور مشفرة، جداول من الشبكات، أي شيء. المشكلة هي: إذا لم تعلم التركيب الدقيق الذي استخدمته لتدوينها، فهناك دائماً فرصة قليلة في استعادة البيانات. وهذا جيد إذا كان هدفك السرية secrecy، ولكن إذا احتجت إلى مشاركة البيانات دائماً مع أشخاص أو برامج أخرى، أو حتى أسوء، تصحيح مخرجات من برنامج منحرف، فإنك في مأزق. فإذا ما تم فقد بايت صغير، فمن المحتمل أن يصبح كامل الملف غير مفيد.

بالطبع يوجد طرق أخرى لتخزين بياناتك. من أجل الملفات التي تخزن سجلات بيانات، توفر ملفات محدّد الجدولة tab-delimited (محدد تنظيم الجدولة) و CSV (القيمة المفصولة بفاصلة comma-separated value) وسيلة نقل مريحة (مناسبة)، في تنسيق أكثر وداً للإنسان. على سبيل المثال، خذ هذه البيانات من عينة قاعدة بيانات ميكروسوفت التجارية "نورثويند Northwind"، فقد تم تخزينها كقيم مفصولة بفاصلة comma-separated values.

```
ProductID,ProductName,SupplierID,Category,UnitPrice,Available
"1","Chai","652","Beverages","$18.00","Yes"
"2","Chang","9874","Beverages","$19.00","No"
"3","Aniseed Syrup","9874","Condiments","On Sale","Yes"
```

الآن هذا أفضل. هذه البيانات سهلة الفهم نوعاً ما. كل قطعة من البيانات تم تجميعها بواسطة فواصل، والصف الأول يشير إلى ما يحتويه كل عمود. والجزء الأفضل أن العديد من البرامج تعرف سابقاً كيف تقرأ ملفات في هذا التنسيق. إذا حفظت هذه البيانات في ملف نصي بالامتداد .csv، وفتحتها باستخدام ميكروسوفت اكسل، فإن البيانات ستظهر بشكل آلي في أعمدة.

ولكن يمكن أن تكون أفضل. على سبيل المثال، إلى ما تشير هذه القيم "652" و "9874" وبأي طريقة؟ وهل صحيح أن سعر الوحدة لـ Aniseed Syrup هي "برسم البيع On Sale"؟

بالتأكيد أستطيع تحميل هذه البيانات في برنامجي، ولكن هل أستطيع فعل أي شيء بها؟ على الأقل هي سهلة القراءة لكل من الإنسان وبرامج الكمبيوتر، و أليس هذا ما قلته حول XML؟

حسناً، على الرغم من أن XML تتضمن قواعد وميزات تجعلها أكثر مرونة من ملف بيانات نصي عادي، وهذا ليس هو الفرق. من أجل جميع المبالغ، فإن XML هي مجرد طريقة في تخزين البيانات. أي سمة لـ XML تم مناقشتها في هذا الفصل يمكن أن يتم إنجازها بسهولة بواسطة البيانات المخزنة في نص أبسط simpler text أو تنسيقات خاصة ثنائية binary proprietary formats. في الحقيقة، على الأغلب من الأسرع وأكثر راحة للتطوير استخدام تنسيق خاص proprietary، لأن بياناتك ستحتوي على ما تحتاجه فقط وبدقة، دون أي خطأ.

يمكن القول أن XML تتضمن عدة سمات والتي تجعل منها منافس contender قوي عند اعتبار تنسيق البيانات:

إنها بسيطة بالنسبة للقراءة It's straightforward to read

يتضمن كل عنصر نوع بيانات عنوان، والعناوين الجيدة تجعل القراءة جيدة.

سهولة المعالجة It's easy to process

جميع البيانات تتضمن رسم بداية ونهاية، لذلك يمكن للبرنامج أن يعالج البيانات دون الكثير من الجهد. وعنصر واحد سيء لن يعمل بالضرورة على إتلاف ruin كامل الملف.

مرنة *It's flexible*

تستطيع تخزين أي نوع من البيانات في XML. بالنتيجة فهو مجرد ملف نصي. إذا كان لديك تنسيق ملف XML معين مستخدم في النسخة رقم 1 من برنامجك، وأضفت ميزات له في النسخة رقم 2، تستطيع عمله بطريقة بحيث يبقى متاح لبرامج النسخة 1 استخدام ملفات النسخة 2 بدون خرق.

تصف نفسها *It's self-describing*

تتضمن XML العديد من الميزات والتي تتيح لك وصف محتوى ملف XML. الأكثر شيوعاً اثنان هما DTD (تعريف نوع المستند Document Type Definition) و XSD (تعريف تصميم بيانات XML Schema Definition). إنك تستخدم هذه الأدوات للإشارة إلى ما تتوقع أن يحويه بالضبط ملف بياناتك. بالإضافة لذلك، تسمح XML لك تضمين تعليقات في المحتوى دون التأثير على البيانات الفعلية.

ذاتية التحقق *It's self-verifying*

الأدوات المتاحة من ضمنها الأدوات في الدوت نت، والتي تستطيع التشديد على تكامل وتنسيق ملف XML بمقارنة المحتوى لـ DTD و XSD المصاحب. وهذا يتيح لك التحقق من ملف قبل حتى معالجته.

إنها معيار مفتوح *It's an open standard*

لقد حصلت XML على قبول واسع الانتشار widespread acceptance، حتى عبر منصات الكمبيوتر المتشعبة divergent computer platforms.

إنها مبنية (جاهزة) في الدوت نت *It's built into .NET*

هذا سيكون السبب الأعظم لاستخدامها في الحقيقة، لن تكون قادر على الابتعاد عن XML في الدوت نت، حتى لو حاولت فهي في كل مكان.

ولكن توجد أخبار سيئة أيضاً:

إنها ضخمة (كبيرة الحجم) *It's bulky*

يحتوي محتوى XML على الكثير من المعلومات التركيبية المتكررة، وبشكل عام الكثير من الفراغات البيضاء. تستطيع اختصار abbreviate العديد من عناصر التركيب، وإزالة جميع المساحة البيضاء whitespace (لا تحتاجها XML)، ولكن هذا سيزيل سمات قابلية القراءة بالنسبة للإنسان من البيانات. بعض منصات العمل platforms، مثل مستعرضات الهواتف الخلوية cell phone browser، تحب الاحتفاظ بالبيانات صغيرة. XML هي أي شيء ولكن صغيرة.

إنها نص *It's text*

انتظر هذا شيء جيد في معظم الأوقات. بعض الأحيان تريد تخزين بيانات ثنائية binary فقط، مثل الصور في الواقع لا تستطيع تخزين بيانات ثنائية حقيقية في ملف XML دون كسر واحد من القواعد الأساسية لـ XML تقريباً: نصوص فقط! على الأغلب، البيانات الثنائية يتم تشفيرها في تنسيق يشبه النص، مثل القاعدة-64 (والتي تستخدم الحروف القابلة للقراءة لتخزين البيانات الثنائية).

إنها غير فعالة *It's inefficient*

وهذا يأتي من كون لديها بيانات في تنسيق شبه قابل للقراءة من قبل الإنسان كثير الحشو (مسهب)، بدلاً من المصقول (الوجيز)، وضغط النموذج الثنائي ببساطة تستغرق وقت أطول بالنسبة للكمبيوتر لعمل مسح نص في البحث عن الأقواس المحدبة المتطابقة من عمل نقل عدة بايتات مباشرة من كتلة بيانات ثنائية ضمن موقع في الذاكرة.

إنها قابلة للقراءة من قبل الإنسان *It's human-readable*

لا يوجد الكثير من الأسرار في ملف XML. على الرغم من أنك تستطيع تشفير عناصر البيانات في الملف، أو كامل الملف لسبب ما، هذا سيؤدي نوعاً ما إلى إحباط الهدف من استخدام XML.

إنها قابلة للقراءة من قبل الآلة. *It's machine-readable.*

XML ليست مناسبة لكل نوع ملف بيانات.

ليست معفية (منيعه) من الأخطاء. *It's not immune to errors.*

كما أكرر، إن XML مجرد ملف نصي. فهي ليست الدواء العام، مجرد تنسيق ملف مفيد.

مبدأ لغة الترميز القابلة للتوسع. *The XML Rule.*

قبل أن نلقي نظرة على XML الحقيقي، تحتاج إلى معرفة المبدأ. عليك أن تطيع obey القاعدة في كل قطعة من نص XML الذي تكتبه.

القاعدة THE RULE

إذا فتحتها، أغلقها. If you open it, close it.

هذه هي لانتسها. وأشرح ما تعني فيما بعد.

محتوى XML. XML Content.

إذا لم تستخدم XML، ولكن قد كتبت بعض HTML، فإن ما أكتبه سيكون معروف نوعاً ما.

بعض قواعد XML

إليك قطعة من XML لتستمع بها.

```
<?xml version="1.0"?>
<hello>
  <there>
    <!-- Finally, real data here. -->
    <world target="everyone">think XML</world>
    <totalCount>694.34</totalCount>
    <goodbye />
```

</there>

</hello>

إنها بيانات وبيانات مفيدة واليك الأسباب:

إنها XML بوضوح. It's obviously XML.

وهذا واضح من السطر الأول، والذي دائماً يبدأ بـ...<?xml...>، هذا السطر يشير أيضاً إلى رقم نسخة XML، والتي تخبر روتين معالجة XML (المعربات parsers) لضبط السلوك عند الحاجة.

إنها هيكلية It's structured

إن XML هو تركيب بيانات هرمي. تستطيع الحصول على عناصر البيانات المضمنة داخل عناصر بيانات أخرى لأي عمق تريد. كل عنصر محصور بمجموعة من الوسوم، في هذه العينة، الوسوم هي hello، there، world، totalCount، و goodbye. تظهر الوسوم دائماً ضمن <angle brackets>، ودائماً تظهر بشكل زوجي، كما في <hello>...</hello>. (ومن هنا أتت القاعدة "إذا فتحته، أغلقه") لا تنس " / " قبل اسم الوسم في قوس الإغلاق. هذا البناء يتيح لك تنظيم بياناتك في وحدات مسماة منظمة بشكل خاص. من أجل الوسوم التي ليس لديها شيء بين البداية والنهاية، تستطيع استخدام البناء المختزل (المختصر shortened syntax) <tagname />، كما فعلت مع الوسم goodbye. على فكرة By the way، وسوم XML حساسة لحالة الأحرف case-sensitive، لذلك اكتب بحذر.

إنها قابلة للقراءة It's readable

إنها قابلة للقراءة من قبل الإنسان، بفضل جميع الفراغات البيضاء، على الرغم من أنك تستطيع إزالة هذه الفراغات وسيبقى لديك XML. وهي أيضاً قابلة للقراءة من قبل الكمبيوتر بسبب الاستخدام الراسخ consistent للوسوم.

إنها وحدة مفردة من البيانات It's a single unit of data

جميع ملفات XML لديها عنصر جذري root element مفرد والذي يجب أن تظهر فيه جميع العناصر الأخرى. في العينة، <hello> هو العنصر الجذري (الإقلاعي). حالما يتم إغلاق ذلك العنصر (من خلال وسم النهاية له) لا تستطيع إضافة أي عناصر إضافية.

إنها تحوي التعليقات It's got comments

شاهد السطر <!-- Finally, real data here. -->، إنه تعليق. تستطيع لصق التعليقات هنا وكأنها وسوم حرة الحركة free-floating tags.

إنها تحوي (تحصل على) المواصفات. It's got attributes.

تدعم XML تنوعين varieties من البيانات: real data: البيانات الحقيقية والمواصفات attributes. قيم البيانات الحقيقية تأتي بين صلب innermost أزواج الوسوم، كما مع think XML و 694.34 في العينة. توفر المواصفات معلومات موسعة حول الوسوم نفسها. لقد ضمنت مواصفة attribute مسماة target في العنصر world. إن المحتوى لجميع المواصفات يجب أن يكون ضمن علامتي اقتباس (""). كان باستطاعتي جعل هذه المواصفة عنصر فرعي subelement بدلاً عن ذلك، والكثير من الناس يعملون ذلك. يوجد عدم اتفاق disagreement بين المبرمجين بخصوص متى يجب أن تكون البيانات عنصر أو مواصفة دع ضميرك conscience (قلبك) يقودك (يدلك).

بعض القواعد- والأهمية لـ XML. Some Basic—and Meaningful—XML

دعنا نرى كيف تبدو البيانات المفصلة بفاصلة من قاعدة بيانات "Northwind" التجارية التي ذكرناها سابقاً في XML.

```
<?xml version="1.0"?>
<productList>
  <supplier ID="652" fullName="Beverages R Us">
    <product ID="1" available="Yes">
      <productName>Chai</productName>
      <category>Beverages</category>
      <unitPrice>18.00</unitPrice>
    </product>
  </supplier>
  <supplier ID="9874" fullName="We Sell Food">
    <product ID="2" available="No">
      <productName>Chang</productName>
      <category>Beverages</category>
      <unitPrice>19.00</unitPrice>
    </product>
    <product ID="3" available="Yes" onSale="true">
      <productName>Aniseed Syrup</productName>
      <category>Condiments</category>
      <unitPrice>12.00</unitPrice>
    </product>
  </supplier>
</productList>
```

نقل البيانات إلى XML قد زاد بشكل كبير حجم المحتوى. ولكن مع الزيادة في الحجم تأتي الزيادة في القيمة المعالجة. لقد كان باستطاعتي بشكل مباشر الحصول على بعض الفائدة من تركيب XML الهيكلي. في البيانات الأصلية، المزود supplier كان عمود آخر فقط، ولكن في نسخة XML، جميع البيانات تم تجميعها الآن ضمن مقاطع المزود supplier، والتي تكون مفهومة (على الأقل، إذا ما كانت هذه هي الكيفية التي عملت على تخطيطها لاستخدام البيانات). ويمكنك أن ترى أنني قد اتبعت القاعدة، كل سمة مفتوحة لديها سمة إغلاق مطابقة. مهما تعمل لا تنس القاعدة.

ماذا حول القسم القابل للقراءة من قبل الإنسان؟ What About the Human-Readable Part?

واحدة من الأدوات المستخدمة مع XML هي XSLT، والتي تحل محل Transformations تحويلات XSL (XSL) تحل محل لغة الصفيحة النمطية القابلة للتوسع (eXtensible Stylesheet Language) إن من الصعب على XSLT استخدام لغة الصياغة scripting language والتي تتيح لك تحويل بعض بيانات XML إلى أي نوع بيانات آخر، أو تنسيق مخرجات تريده أنت. فهي مجرد مقدار قليل من اللغات المتعلقة بـ XSL المنشئة لمعالجة بيانات XML بطرق معقدة. لنرى عمل XSL، خذ القطعة المفيدة من XML المجدولة سابقاً (العينة <productList>) وبديل السطر الأول <?xml> بالأسطر التالية:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="hello.xsl"?>
```

احفظ نص إلى ملف على سطح مكتبك كـ hello.xml ما يليه: ضع صيغة XSLT التالية في ملف آخر على سطح مكتبك سمه hello.xsl.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <xsl:text>
      ProductID,ProductName,SupplierID,Category,UnitPrice,Available
    </xsl:text>
    <BR/>
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="supplier">
    <xsl:variable name="supID" select="@ID"/>
    <xsl:for-each select="product">
      <xsl:value-of select="@ID"/> ", "
      <xsl:value-of select="productName"/> ", "
      <xsl:value-of select="$supID"/> ", "
      <xsl:value-of select="category"/> ", "
      <xsl:choose>
        <xsl:when test="@onSale='true'">On Sale</xsl:when>
        <xsl:otherwise>$
          <xsl:value-of select="unitPrice"/>
        </xsl:otherwise>
      </xsl:choose> ", "
      <xsl:value-of select="@available"/> "
      <BR/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

لقد أخبرتك أنه صعب الاستخدام، وحتى الأصعب النظر إليه. والآن من أجل العرض. لدي مستكشف انترنت Internet Explorer منصب على نظامي، ولكن هذا سيعمل مع أغلب المستعرضات الحديثة. افتح ملف hello.xml في مستعرضك، سيظهر النص المنسق الجميل التالي:

```
ProductID,ProductName,SupplierID,Category, UnitPrice,Available
"1", "Chai", "652", "Beverages", " $18.00", "Yes"
"2", "Chang", "9874", "Beverages", " $19.00", "No"
"3", "Aniseed Syrup", "9874", "Condiments", "On Sale", "Yes"
```

والآن، هذا محبب أكثر، حيث اجتمعت كل من XML و XSLT مع بعضهما وصنعا هذا التقدم في تقنية البيانات الممكنة. (لقد عملت خدعة صغيرة في هذا المثال، ستلاحظ المدخلات
 في صيغة XSLT والتي لا تظهر في المخرجات النهائية. وقد عملت هذا لتبدو مناسبة في مستعرضك)، ولكن بشكل جدي، على الرغم من أنني كنت قادر على توليد مجموعة بيانات مفصلة بفاصلة comma-separated data set من خلال XSLT، فإن مهمات أكثر شبيهاً لـ XSLT تتضمن توليد HTML منسقة بشكل جميل بالاعتماد على بيانات XML، أو توليد مستند XML ذو عرض بديل خاص للبيانات الأصلية. كيف يعمل؟ بشكل أساسي، تخبر العناصر <xsl:template> المفسر parser للبحث عن وسوم في مستند XML والذي يطابق نموذج (عينة) ما pattern (مثل supplier). عندما يجد مطابقة، فإنه يقدم كل شيء داخل الوسوم <xsl:template> إلى وسم XML المطابق ومحتوياته. العينة (النموذج) المخصص في المواصفات match تستخدم تقنية XML تدعى XPath: نظام للبحث بشكل عام عن الوسوم المتطابقة ضمن مستند XML. من الظاهر أنه مريب! هو كذلك.

مخططات XML .XML Schemas

تتيح لك XSD (تعريف تركيب XML، XML Structure Definitions) تعريف مخطط إما "لغة language" أو "مفردات vocabulary" لمستند XML خاص بك، تذكر، أن XML معيار شامل مفتوح وواسع، تستطيع تعريف الوسوم بأي طريقة تريد ولن يابه أحد، على الأقل حتى يصبح من الواجب عليك معالجة الوسوم ضمن برمجياتك. فإذا لم تكن صحيحة، فمن المحتمل أن تفشل معالجتك. تتيح لك XSD تعريف القواعد التي يجب أن يتبعها مستندك XML إذا كان يجب اعتباره مستند صحيح من أجل أهدافك.

(DTD تعريف نوع مستند Document Type Definition، مشابه لـ XSD، مع أنه تقنية أقدم. وهو مدعوم بشكل واسع من قبل أدوات XML، ولكنه ليس بتلك المرونة التي عليها XSD. يوجد أيضاً لغات تعريف مخطط أخرى مشابهة لـ XSD، ولكن بما أن XSD مبني داخل الدوت نت فإننا سنركز عليه.) مخططات XSD هي كل بت مقدر (مثنى endearing) كما هو الحال في صيغ XSLT. لنعمل على إنشاء XSD من أجل مثالنا XML <productList> الذي ذكرناه فيما سبق. أولاً، نحتاج إلى تغيير أعلى XML لنسمح له معرفة أن ملف XSD هو المتاح. غير التالي:

```
<?xml version="1.0"?>
<productList>
```

إلى:

```
<?xml version="1.0"?>
```

```
"productList xmlns="SimpleProductList"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
<"xsi:noNamespaceSchemaLocation="hello.xsd
```

هذه التعليمات (التوجيهات directives) تخبر مفسر XML للبحث في مخطط hello.xsd. وهي تعرف أيضاً فضاء أسماء، فيما بعد. يحتوي ملف hello.xsd المخطط التالي:

```
"xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
<"targetNamespace="SimpleProductList
</"xs:element name="productList" type="ProductListType"
<"xs:complexType name="ProductListType"
<xs:sequence>
"xs:element name="supplier" type="SupplierType"
<"maxOccurs="unbounded
<xs:sequence/>
<xs:complexType/>
<"xs:complexType name="SupplierType"
<xs:sequence>
"xs:element name="product" type="ProductType"
<"maxOccurs="unbounded
<xs:sequence/>
<"xs:attribute name="ID" type="xs:integer"
<"xs:attribute name="fullName" type="xs:string"
<xs:complexType/>
<"xs:complexType name="ProductType"
<xs:sequence>
</"xs:element name="productName" type="xs:string"
</"xs:element name="category" type="xs:string"
</"xs:element name="unitPrice" type="xs:decimal"
<xs:sequence/>
<"xs:attribute name="ID" type="xs:integer"
<"xs:attribute name="available" type="YesOrNoType"
<"xs:attribute name="onSale" type="xs:boolean"
<xs:complexType/>
<"xs:simpleType name="YesOrNoType"
<"xs:restriction base="xs:string"
</"xs:enumeration value="Yes"
</"xs:enumeration value="No"
<xs:restriction/>
<xs:simpleType/>
<xs:schema/>
```

يبدو أنه متداخل، أليس كذلك؟ عملياً إنه أكثر بساطة من XSLT. بشكل أساسي، يقول المخطط أنه من أجل كل عنصر (أو "tag" أو "node" في مستند XML، هنا توجد العناصر الفرعية subelements والمواصفات التي تحويها attributes، ونوع البيانات لكل منهم. تستطيع حتى إنشاء أنواع بيانات مستعارة (أو زائفة pseudodata) خاصة بك (عملياً، العوامل المقيدة limiting factors على أنواع البيانات المتواجدة)، كما فعلت مع نوع البيانات YesOrNoType، والذي يقيد القيمة المتعلقة إلى نصوص نعم Yes ولا No.

تستطيع النظر إلى ملف XML مع مخطط XSD الملحق (المرفق) في مستعرضك. ولكنه لن يكون بكل تلك الأهمية إنه يريك فقط الـ XML. ولكن المخططات ستكون مفيدة عندما تحتاج تقييم (تعيين أهمية) النوعية بالنسبة لبيانات XML التي تأتي ضمن تطبيقات برمجياتك من مصادر خارجية.

فضاءات أسماء XML. XML Namespaces

يمكن أن تختلف قائمة المنتج product list لـ XML التي عرضتها سابقاً، فقد تكون ذات اسم مختلف وقواعد تنسيق مختلفة أيضاً، على سبيل المثال، يمكن أن يتم إنشاء مستند بحيث يبدو كالتالي:

```
<"xml version="1.0"
<allProducts>
<"vendor ID="652" vendorName="Beverages R Us"
<"item ID="1" available="Yes"
<itemName>Chai</itemName>
<group>Beverages</group>
<priceEach>18.00</priceEach>
<item/>
```

```
<vendor/>
<allProducts/>
```

جميع البيانات نفسها، ولكن الوسوم tags مختلفة. مستند مثل هذا سيكون متضارب incompatible مع برمجيات مكتوبة للعمل مع مستندنا الأصلي. فتشغيل المستند من خلال XSD الخاص بنا سوف يخبرنا بسرة أن لدينا أخطاء في مجموعة البيانات bogus data set، ولكنه سيكون من الجميل لو أخبرنا شيء ما هذا من البداية. أدخل فضاءات الأسماء namespaces. توفر فضاءات الأسماء namespaces طريقة مريحة لقول "هذا الوسم الخاص في مستند XML يستخدم لغة التعريف XSD-defined language هذه." لاحظ بداية مخطط XSD المبين سابقاً:

```
"xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

هذا السطر يثبت فضاء الأسماء المسمى xs باستخدام المواصفة xmlns. (الجزء xs: يخبر XML ماذا تريد أن تسمى فضاء أسماءك.) قيمة المواصفة هي معرف (محدد) مورد موحد (Uniform Resource Identifier (URI)، مجرد قيمة مفردة ومميزة بحيث تكون واثق من أنه لن يستخدمها شخص آخر. نموذجياً، تستخدم عنوان موقع ويب من أجل شركتك، ليس من الواجب أن يكون الموقع موجود. تستطيع حتى وضع رقم هاتفك هناك، مجرد كونها قيمة مفردة ومميزة.

الطريقة الأكثر شيوعاً في استخدام فضاء أسماء هي سبق الوسوم ذات الصلة في مستند XML باسم فضاء الأسماء الجديد، كما في "xs:schema" عوضاً عن "schema" فقط. وهذا يقول للمفسر "إذا كنت تختبر بناءي على مخطط XSD، استخدم الاسم الذي حددته لفضاء الأسماء xs." تستطيع أيضاً استخدام فضاء الأسماء " الافتراضي default " أيضاً من أجل عنصر معطى وجميع ما ينحدر منه descendants يتضمن المواصفة xmlns في عنصر خارجي. ومن ثم جميع العناصر ضمن ذلك العنصر الخارجي سيستخدم فضاء الأسماء المخصص. ولقد استخدمت هذه الطريقة في واحد من الأمثلة السابقة:

```
<productList xmlns="SimpleProductList"...
```

من أجل ملفات XML القاعدية والتي سيتم استخدامها فقط بواسطة برنامجك، من المحتمل أن لا تحتاج التعب بفضاءات الأسماء في الحقيقة تأتي في المتناول عندما تعمل على إنشاء بيانات XML والتي تستخدم معيار توزيع عمومي publicly published standard نوعاً ما يوجد أيضاً حالات (نسخ) بحيث ومن المحتمل أن يحتوي ملف XML مفرد بيانات ذات صلة باستخدامات متميزة لاثنتان أو أكثر من XML. في هذه الحالة، الأجزاء المختلفة من ملف XML يمكن أن تشير إلى فضاءات أسماء مختلفة namespaces.

بالنسبة للأجزاء الأخرى من عالم XML، XSD وفضاءات الأسماء ليست سهلة الاستخدام، ولكنها مرنة flexible وقوية powerful. وكالعادة، توجد أدوات tools، أدوات مضمنة في الفيجوال أستوديو، والتي تتيح لك بناءها جميعاً دون أن يكون عليك التفكير بخصوص التفاصيل. كما أقول دائماً، إن XML هي مجرد بيانات، XML وتقنياتها ذات الصلة توفر طريقة للمساعدة في ضمان ensure أن بياناتك جاهزة للاستخدام في تطبيقك.

استخدام XML في الدوت نت: الطريقة القديمة Using XML in .NET: The Old Way

تتضمن الفيجوال بيسك طريقتين للعمل مع محتوى XML: الطريقة القديمة والطريقة الجديدة. تستخدم الطريقة القديمة فئات من فضاء الأسماء System.Xml، وتوفر وصول تقليدي معتمد على كائن object-based إلى وسوم XML، المواصفات attributes، والبيانات data. الطريقة الجديدة، تم تقديمها في الإصدار 2008، تستخدم الفئات في فضاء الأسماء System.Xml.Linq وتوفر وصول إلى محتوى XML مباشرة ضمن بناء الكود المصدري للفيجوال بيسك. سأناقش كل من الطريقتين في هذا الفصل.

بما أن جودة إدارة XML ليست بقدر ضخامة مقطع نصه، تتضمن الدوت نت العديد من الفئات التي تدير بيانات XML. جميع هذه الأدوات تظهر في فضاء الأسماء System.Xml وفضاءات الأسماء التابعة له:

System.Xml

التجميع الرئيسي للفئات ذات الصلة ب XML في الطريقة القديمة old-way XML-related classes.

System.Xml.Linq

الفئات التي تكامل (تدمج integrate) مع تقنيات لينكو LINQ. وهذه هي الطريقة الجديدة والتي سأحدث عنها لاحقاً.

System.Xml.Schema

الفئات التي تنشئ وتستخدم مخططات XSD.

System.Xml.Serialization

الفئات التي تقرأ وتكتب مستندات XML بواسطة جدول دوت نت القياسية .NET stream standard.

System.Xml.XPath

الفئات التي تنفذ تقنية XPath المستخدمة لبحث مستندات XML.

System.Xml.Xsl

الفئات التي تمكن تحويلات XSL.

الميزات المضمنة في كل فئة ترتبط بإحكام إلى تركيب XML والتقنيات ذات الصلة مثل XSLT و XSD.

فئات XML القاعدية، قواعدياً. The Basic XML Classes, Basically.

يتضمن فضاء الأسماء System.Xml The أغلب الفئات القاعدية التي ستستخدمها لإدارة بيانات XML. إن الكائن XmlDocument هو العرض في الذاكرة لمستندك XML الفعلي:

```
Dim myData As New System.Xml.XmlDocument
```

إن مستندك مركب من التصريحات declarations (ذلك الشيء <?xml...?> في الأعلى)، عناصر البيانات data elements (جميع الوسوم tags النوعية في مستندك)، المواصفات attributes (داخل كل بداية وسم عنصر)، والتعليقات comments. وهذا كله يتم تمثيله بواسطة الفئات XmlDocument، XmlDeclaration، XmlComment، XmlAttribute، XmlElement على الترتيب. مع بعضها هذه الوحدات الأربع لمستندك تدعى "العقد nodes"، ممثلة بشكل عام بواسطة الفئة. (جميع الفئات الأربع ترث من الفئة القاعدية XmlNode الأكبر). عندما تبني مستند XML يدوياً في الذاكرة، فإنك تستخدم فئات مستقلة مثل XmlDocument، بعدئذٍ، عندما تحتاج لعمل مسح خلال مستند موجود، من الأسهل استخدام الفئة XmlNode الشاملة. دعنا نبني مجموعة جزئية من مثال بيانات المنتج product data لـ XML.

```
<?xml version="1.0"??>
```

```
<productList>
```

```
<!-- We currently sell these items. -->
```

```
<supplier ID="652" fullName="Beverages R Us">
<product ID="1" available="Yes">
<productName>Chai</productName>
<category>Beverages</category>
<unitPrice>18.00</unitPrice>
</product>
</supplier>
</productList>
```

صرح عن جميع المتغيرات التي ستستخدمها، ومن ثم استخدمها.

```
Dim products As Xml.XmlDocument
Dim prodDeclare As Xml.XmlDeclaration
Dim rootSet As Xml.XmlElement
Dim supplier As Xml.XmlElement
Dim product As Xml.XmlElement
Dim productValue As Xml.XmlElement
Dim comment As Xml.XmlComment

' ----- Create the document with a valid declaration.
products = New Xml.XmlDocument
prodDeclare = products.CreateXmlDeclaration("1.0", _
Nothing, String.Empty)
products.InsertBefore(prodDeclare, products.DocumentElement)
' ----- Create the root element, <productList>.
rootSet = products.CreateElement("productList")
products.InsertAfter(rootSet, prodDeclare)
' ----- Add a nice comment.
comment = products.CreateComment(
" We currently sell these items. ")
rootSet.AppendChild(comment)
' ----- Create the supplier element, <supplier>.
' Include the attributes.
supplier = products.CreateElement("supplier")
supplier.SetAttribute("ID", "652")
supplier.SetAttribute("fullName", "Beverages R Us")
rootSet.AppendChild(supplier)
' ----- Create the product element, <product>, with the
' subordinate data values.
product = products.CreateElement("product")
product.SetAttribute("ID", "1")
product.SetAttribute("available", "yes")
supplier.AppendChild(product)
productValue = products.CreateElement("productName")
productValue.InnerText = "Chai"
product.AppendChild(productValue)
productValue = products.CreateElement("category")
productValue.InnerText = "Beverages"
product.AppendChild(productValue)
productValue = products.CreateElement("unitPrice")
productValue.InnerText = "18.00"
product.AppendChild(productValue)
```

إنه يعمل حقاً، ليرهان ذلك، ضع هذا الكود في حدث النقر لزر ما، وأنهيه بالسطر التالي (أي أضف السطر التالي لنهاية الكود السابق).

```
products.Save("c:\products.xml")
```

شغل البرنامج و اعرض الملف `c:\products.xml` لرؤية XML product data. توجد عدة طرق مختلفة لاستخدام فئات XML لإنشاء مستند XML في الذاكرة. على سبيل المثال، على الرغم من أنني استخدمت الطريقة `SetAttribute` لإضافة مواصفات إلى عقدة المزود `supplier` والمنتج `product`، فقد أستطيع إنشاء كائنات مواصفات منفصلة، وتزليهم على هذه العقدة، تماماً مثلما فعلت من أجل العنصر الرئيسي.

```
Dim attrData As XmlAttribute
attrData = products.CreateAttribute("ID")
attrData.Value = "652"
supplier.SetAttributeNode(attrData)
```

إذاً، هذا جميل، ولكن ماذا بخصوص إذا ما كان لديك بعض XML في ملف ما، وتريد تحميلها في كائن `XmlDocument` ببساطة استخدم الطريقة `Load` للكائن `XmlDocument`.

```
Dim products As XmlDocument
products = New XmlDocument
products.Load("c:\products.xml")
```

من أجل تلك النسخ (الحالات) التي تريد فقط القراءة أو كتابة بعض XML من أو إلى ملف ما، ولا تأبه كثيراً بخصوص معالجتها في الذاكرة، تتيح لك الفئة XmlTextReader والفئة XmlTextWriter قراءة وكتابة بيانات XML بسرعة بواسطة جدول النص text stream. ولكن إذا كنت ذاهب لعمل أشياء ببيانات XML في برنامجك، فإن الطرق Load و Save لكائن XmlDocument هي الخيار الأفضل.

إيجاد الإبر وكومات القش Finding Needles and Haystacks

في مثال البيانات خاصتنا، جميع المنتجات products تظهر في مجموعات المزود. إذا كنا نريد فقط قائمة بالمنتجات products، بغض النظر عن المزود supplier، نطلب من الكائن XmlDocument تزويد تلك البيانات بواسطة الكائن XmlNodeList.

```
Dim justProducts As XmlNodeList
Dim oneProduct As XmlNode
' ----- First, get the list.
justProducts = products.GetElementsByTagName("product")
' ----- Then do something with them.
For Each oneProduct In justProducts
' ----- Put interesting code here.
Next oneProduct
MsgBox("Processed " & justProducts.Count.ToString() & " product(s).")
```

من أجل اختيار أكثر تعقيداً للعقد ضمن المستند، ينفذ فضاء الأسماء System.Xml.XPath لغة بحث XPath، والتي تمنحك مرونة زائدة في البوند المستكشفة. تصف مجموعة مستندات الفيجوال أستوديو بناء الطرق والبحث المستخدم مع هذه الفئات.

مخطط التحقق (الإثبات) Schema Verification.

يستطيع الكائن XmlDocument الاحتفاظ بأي نوع من محتوى XML المحقق والعشوائي، ولكن تستطيع أيضاً التحقق من المستند على مخطط XSD. إذا كان مستندك XML يشير إلى مخطط XSD، أو يتضمن DTD، أو يستخدم XDR (مخططات بيانات XML المختزلة XML Data Reduced schemas، مشابهة لـ XSD)، أو XmlReader، عند تركيبه بواسطة إعدادات قارئ XmlReaderSettings مناسب، سيقارن بدقة بيانات XML على القواعد المعرفة، ويرمي باستثناء إذا كانت هناك مشكلة.

```
Dim products As New XmlDocument
Dim xmlRead As XmlTextReader
Dim withVerify As New XmlReaderSettings
Dim xmlReadGood As XmlReader
' ----- Open the XML file and process schemas
' referenced within the content.
withVerify.ValidationType = ValidationType.Schema
xmlRead = New XmlTextReader("c:\temp\products.xml")
xmlReadGood = XmlReader.Create(xmlRead, withVerify)
' ----- Load content, or throw exception on
' validation failure.
products.Load(xmlReadGood)
' ----- Clean up.
xmlReadGood.Close()
xmlRead.Close()
```

تحويلات XML .XML Transformations

تحويلات XSL ليست أكثر صعوبة من أية معالجات أخرى لـ XML. كما يوجد العديد من الطرق للحصول على بيانات مصدرية لـ XML (من ملف، بناءه بشكل يدوي بواسطة XmlDocument، إلخ)، يوجد أيضاً عدة طرق لتحويل البيانات. إذا كنت تريد فقط التنقل من ملف مدخلات input file إلى ملف مخرجات output file، فإن الكود التالي يوفر طريقة فعالة وسريعة. إنه يستخدم حالة System.Xml.Xsl.XslCompiledTransform لإتمام العمل.

```
' ----- Above: Imports System.Xml.Xsl
Dim xslTrans As XslCompiledTransform
' ----- Open the XSL file as a transformation.
xslTrans = New XslCompiledTransform()
xslTrans.Load("c:\convert.xsl")
' ----- Convert and save the output.
xslTrans.Transform("c:\input.xml", "c:\output.txt")
```

استخدام XML في الدوت نت: الطريقة الجديدة. Using XML in .NET: The New Way.

عندما ظهرت الفيجوال بيسك للمرة الأولى، لم يكن أحد قد سمع بـ XML. ولكن الآن إنها في كل مكان. والآن في الفيجوال بيسك 2008 أصبحت جزء من بناء اللغة نفسها. أين سينتهي بها المطاف؟

إن إخراج (إزالة) ما يجعل XML جزء من اللغة بعيد نوعاً ما في مقطع الطريقة القديمة، أربنتك بعض الكود الذي ينشئ XML لقائمة المنتج product list من أجل "Chai" محتوى XML كان بطول 11 سطر، ولكنه يأخذ تقريباً 50 سطر من الكود المصدري لإنتاجه. ولكن تستطيع بناء نفس محتوى XML باستخدام الطريقة الجديدة بحوالي 11 سطر تقريباً.

```
Dim chaiItem As System.Xml.Linq.XDocument = <?xml version="1.0"?>
<productList>
<!-- We currently sell these items. -->
<supplier ID="652" fullName="Beverages R Us">
<product ID="1" available="Yes">
```



```
<productName>Chai</productName>
<category>Beverages</category>
<unitPrice>18.00</unitPrice>
</product>
</supplier>
</productList>
```

ما عدا سطر التصريح الأول، المحتوى مطابق ل XML النهائي. إن ميزة XML الحرفية الجديدة *new XML Literals feature* تجعل بناء مستندات XML موجز ومحكم. يتم تخزين المحتوى في كائن XDocument جديد، جزء من فضاء الأسماء System.Xml.Linq. إذا أردت تخزين فقط مقطع XML بدلاً من كامل المستند، استخدم الفئة XElement بدلاً من ذلك.

```
Dim productSection As System.Xml.Linq.XElement = _
<product ID="1" available="Yes">
  <productName>Chai</productName>
  <category>Beverages</category>
  <unitPrice>18.00</unitPrice>
</product>
```

إذا كتبت تمكين الاستنتاج (الاستدلال) في برنامجك (خيار الاستنتاج فعال Option Infer On)، فلا تحتاج حتى لإخبار الفيجوال بيسك فيما إذا كان XElement أو XDocument.

```
Dim productSection = _
<product ID="1" available="Yes">
  <productName>Chai</productName>
  <category>Beverages</category>
  <unitPrice>18.00</unitPrice>
</product>
```

فبمجرد تشابهه مع فئة XmlDocument، تتضمن الفئة XDocument الطرق "تحميل Load" و "حفظ Save" لإدارة ملف معتمد على XML.

تعبير XML المدمجة. Embedded XML Expressions.

تضمن XML في كودك المصدري مادة مدهشة، ولكن ستبقى مدهشة لو اشتريت دائماً "الشاي Chai" بـ "18.00\$" لكل وحدة فقط. محتوى XML الحقيقي يأتي عادة من بيانات متغيرة. وعلى الرغم من أن XML اسم "حرفي Literal"، يمكن لحرفيات XML أن تتضمن محتوى غير حرفي من خلال تعبير XML المدمجة *embedded XML expressions*. حيثما تريد إلى إضافة بيانات من متغير أو تعبير لنص، تستخدم الرموز %> و %< لتعويض بياناتك المخصصة.

```
Dim productName As String = "Chai"
Dim productCategory As String = "Beverage"
Dim productPrice As Decimal = 18@
Dim isAvailable As Boolean = True
Dim productSection = _
<product ID=<%= productID %>
  available=<%= Format(isAvailable, "Yes/No") %>>
  <productName><%= productName %></productName>
  <category><%= productCategory %></category>
  <unitPrice><%= productPrice %></unitPrice>
</product>
```

بالطبع، من أجل إنتاج تصنيف المنتجات products كامل، سيكون عليك عمل الكثير من الكتابة. في الفصل 17، سأقدم بعض الطرق الإضافية لتضمين تعبير XML مع كامل جداول البيانات.

الخصائص المحورية لـ XML. XML Axis Properties

سابقاً في هذا الفصل، في مقطع "إيجاد الإبر وكومات القش Finding Needles and Haystacks"، بينت لك كيفية الوصول إلى مقاطع نوعية لمستندات XML القديمة النمط. تتضمن كائنات XML الحديثة النمط أيضاً طرق لعمل مسح والوصول إلى أقسام XML الشجرية. وهذه تدعى الخصائص المحورية لـ XML، وهي تأتي محزمة في ثلاث تنوعات بنائية محببة:

محور العضو الابن Child-member axis

تستطيع الوصول إلى وسم فرعي (ابن) مباشرة لأي XElement باستخدام اسم الابن كعضو لكائن الأب، تغليف اسم الفرع في مجموعة أقواس محدبة:

```
childElement = parentElement.<childName>
```

محور العضو المنحدر Descendent-member axis

تنوع عن بناء محور العضو الابن يتيح لك الوصول إلى الأعضاء المسماة عند أي عمق ضمن عنصر الأب بدلاً من استخدام فقط نقطة مفردة (.). بين أسماء الأب والابن، استخدم ثلاث نقاط (.):

```
setOfElements = parentElement...<descendentName>
```

محور الموصفة Attribute axis

الوصول لأي موصفة لعنصر بمعاملة اسم الموصفة كاسم عضو، اسبق اسم الموصفة بالرمز @.

```
attributeValue = parentElement.@attributeName
```

المقطع التالي من الكود يعمل مسح على قائمة المنتج الذي صممه سابقاً في الفصل، يعرض رقم المعرف ID واسم كل منتج على لوحة console.

```
For Each oneProduct In allProducts...<product>
```

```
Console.WriteLine(oneProduct.@ID & " : " &
oneProduct.<productName>.Value)
Next oneProduct
```

يستخدم هذا الكود ثلاث أنماط محورية axis styles. تعمل الحلقة For Each...Next مسح على مدخلات <product> المتطابقة باستخدام محور العضو المنحدر descendent-member axis، في كل عنصر منتج متطابق، يصل الكود مواصفة ID باستخدام محور المواصفة attribute axis، ويحصل على اسم المنتج باستخدام محور العضو الابن child-member axis، على طول مع خاصية القيمة Value لعنصر الابن المعادة. تبدو المخرجات كالتالي:

```
1: Chai
2: Chang
3: Aniseed Syrup
```

فضاءات الأسماء والمخططات لمحارف XML. Namespaces and Schemas for XML Literals

كما مع الطريقة القديمة في إدارة XML، تسمح لك الطريقة الحديثة تضمين فضاءات أسماء في محتوى XML. لإضافة سابقة فضاء أسماء XML، ببساطة ضمنه في المحتوى كما تعمل مع أي سيناريو آخر.

```
Dim foodItems =
<?xml version="1.0"?>
<menu:items xmlns:menu="http://www.timaki.com/menu">
  <menu:item>
    <menu:name>Pizza</menu:name>
    <menu:price>12.99</menu:price>
  </menu:item>
  <menu:item>
    <menu:name>Breadsticks</menu:name>
    <menu:price>1.99</menu:price>
  </menu:item>
</menu:items>
```

تستطيع أيضاً تعريف فضاء الأسماء، الجزء xmlns، باستخدام تباين (variation) Imports عبارة الفيچوال بيسك .

```
Imports <xmlns:menu="http://www.timaki.com/menu">
'...later...
Dim foodItems =
<?xml version="1.0"?>
<menu:items>
...and so on...
```

ستحافظ الفيچوال بيسك على إدراج تعريف xmlns في المكان المناسب في محتوى XML. يتم تخزينه عملياً ككائن XElement متميز ضمن XDocument أو XElement. لإنتاج كائن XElement ما لاستخدامك الخاص، تتضمن الفيچوال بيسك الوظيفة (الدالة) GetXmlNamespace الجديدة.

```
Dim justTheNamespace = GetXmlNamespace(menu)
```

مشروع. Project

يريد مدير نظام المكتبة رؤية الإحصائيات والمعلومات بلحمة سريعة، أو تشغيل التقارير المتنوعة التي توفر ملخص هام أو عرض تفاصيل بيانات النظام. على الرغم من أنني وكمبرمج أستطيع تجريب إضافة كل نوع محتمل الحدوث conceivable من التقارير التي يحتاجها المستخدم، فقد تعلمت من خبرتي أن هذا غير ممكن. فدايماً يريد المستخدم القمر (الزبدة)، عادةً في نموذج مقتصر على تقرير والذي وعلى حد علمي سيتم استخدامه مرة واحدة وبعدها لن يُنظر إليه مرة أخرى. إنني لا أحب إعادة تجميع وتحرير كامل التطبيق كلما أراد مستخدم تقرير جديد بدلاً من ذلك، إنني أحفظ التقارير خارج نطاق التطبيق، أخرجها كبرامج منفصلة. ومن ثم، من نموذج واحد في التطبيق الرئيسي، أجعل جميع هذه التقارير الخارجية متاحة في قائمة مناسبة مريحة. لتنفيذ هذه الميزة الشاملة، استخدم ملف تركيب تقرير، ملف XML بسيط يحتوي على معلومات عن التقارير المتاحة، وكيفية تشغيلها. أريد قائمة الاختيار الخاصة بي أن تتضمن بنود فارغة بحيث أستطيع تجميع التقارير بشكل مرئي وبوسائل مريحة. لعمل هذا، سأعمل ملف XML ضمن تسلسل هرمي بعمق غير محدد، مع تمثيل كل مستوى بأبعد مستوى من الفراغ indent المعروف. على سبيل المثال، لنقول أنني أريد الخطوط الرئيسية التالية من التقارير (مع عناوين مجموعة التقرير بخط غامق).

تقارير التفاصيل Detail Reports

التقارير اليومية Daily Report

التقارير الشهرية Monthly Reports

القيم الشهرية Monthly Value

الجرد الشهري Monthly Inventory

تقارير الإيجاز (التلخيص) Summary Reports

ملخص الجرد Inventory Summary

إعداد XML سيتبع التركيب التالي:

```
<Group name="Detail Reports">
  <Item name="Daily Report"/>
  <Group name="Monthly Reports">
    <Item name="Monthly Value"/>
    <Item name="Monthly Inventory"/>
  </Group>
</Group>
```

```
<Group name="SummaryReports">
  <Item name="Inventory Summary"/>
</Group>
```

بالطبع هذا XML بسيط جداً (بغض النظر عن كونه غير ميسر noncompliant) بالإضافة إلى الهرمية، أريد أيضاً تضمين دعم لطرق التقرير المتوقعة. لحفظ الأشياء بسيطة، سيضمن مشروع المكتبة ثلاث أنواع من التقارير:

التقارير الجاهزة (المبنية داخلياً) Built-in reports.

يتضمن التطبيق عدد محدد من التقارير التي تبني بشكل دائم في التطبيق الرئيسي (المجمع assembly). تم ترقيم التقارير بدءاً من 1، وعند هذه المرحلة لدي خمسة تقارير في مخطتي. يمكن أن يختار مصمم ملف إعدادات XML تضمينها في عرض التقارير أو لا، بكل بساطة بتضمينها أو عدم تضمينها في الملف في حال غياب absence ملف الإعدادات، ستظهر هذه التقارير في القائمة بشكل افتراضي. بالإضافة إلى عدد التقارير (من 1 إلى 5)، كل مدخلة لديها نص عرض display text ووصف طويل long description.

تقارير التطبيق Application reports.

تكون هذه التقارير ملفات تنفيذية EXE متميزة ومفصلة وتبدأ بواسطة طرق تمهيد initiation التطبيق القياسية. كل مدخلة تتضمن نص العرض display text، المسار الكامل للتطبيق، معاملات نسبية اختيارية، علامة لتمرير هوية المستخدم identity of the user التي تمهد (تسند قيمة أولية) للتقرير، وشرح طويل.

تقارير عناوين الإنترنت: URL reports

هذه التقارير هي استدعاء بسيط لصفحات ويب، أو أي معرف إنترنت URL. كل مدخلة تتضمن نص العرض، معرف الإنترنت URL نفسه، ووصف طويل. نشاطات المشروع في هذا الفصل تتضمن كل من كتابة الكود وتوثيق مورد خارجي جديد (تنسيق ملف XML).

تحديث التوثيق التقني. Update Technical Documentation.

أولاً، دعنا نضيف توثيق واضح على تركيب ملف هيئة XML. لا توجد طريقة سهلة لوصول تركيب ملف XML إلى مستخدم اعتيادي، على الرغم من أن مثل هذا التوثيق مطلوب، على أمل أن يتضمن التطبيق أيضاً أداة تتيح للمدير بناء ملف التركيب (الإعداد).

ملف تركيب التقرير. Report Configuration File.

يمكن أن يتم إعداد تطبيق المكتبة ليشتغل أي عدد من التقارير من خلال نموذج التقارير. يتم إدارة قائمة التقارير المتاحة من خلال ملف تركيب تقرير XML. يحتوي الملف "مجموعات groups" و"بنود items". جميع البنود هي تقارير، وتظهر ضمن مجموعة. تستطيع إدخال مجموعة ضمن مجموعة أخرى لأى عمق، وقائمة التقارير المعروضة في برنامج المكتبة ستترك فراغ لكل مجموعة تابعة (ثانوية) للمساعدة المستخدم على رؤية تنظيم التقارير. لا يوجد حد لتداخل المجموعات.

العنصر الجذري لملف XML يجب أن يتم تسميته <reportList>، ويمكن أن يحتوي على أي عدد من عناصر البيانات <reportGroup> و<reportItem>: <reportItem>: يمثل مدخلة تقرير مفرد. وهذه المدخلة لديها مواصفة attribute واحدة مطلوبة، وتصل إلى خمسة عناصر بيانات تابعة (ثانوية subordinate) وتعتمد على إعدادات المواصفة.

— نوع type (المواصفة attribute): تعمل الإعداد إلى واحد من القيم التالية:

جاهز (مبني داخلياً) built-in: تشغيل واحد من البرامج الجاهزة (المبنية داخلياً). هذا النوع من التقارير يستخدم عناصر البيانات <displayText>، <reportPath> و<description>.

برنامج program: تشغيل برنامج تنفيذي EXE منفصل. هذا النوع من التقارير يستخدم عناصر البيانات <displayText>، <reportPath>، <reportArgs>، <reportFlags>، و<description>.

معرف ويب url: يشغل URL (محدد مواضع الموارد الموحد Uniform Resource Locator)، مثل صفحة ويب أو "بروتوكول البريدmailto" "إيميل إلى عنوان مستقل. هذا النوع من التقارير يستخدم عناصر البيانات <displayText>، <reportPath>، و<description>.

— عرض نص <displayText>: اسم مختص أو وصف لهذا التقرير، كما سيظهر في قائمة خيارات التقرير. هذا العنصر مطلوب لجميع أنواع التقارير. — مسار التقرير <reportPath>: المسار الكامل full path، محدد مواضع الموارد الموحد URL، أو عدد التقارير number of the report، بالاعتماد على نوع التقرير. من أجل تقارير برنامج (تنفيذي EXE)، هذا هو المسار UNC الكامل أو مسار المعتمد على حرف السواقة للتقرير، بدون معاملات نسبية إضافية. من أجل التقارير الجاهزة (المبنية داخلياً built-in) هذا هو رقم التقرير، من 1 إلى 5 (القيم ومعانيها تم جدولتها فيما بعد في هذا الفصل). من أجل تقارير URL، هذا هو محدد المواضع الفعلي، كما في "http://mysite.com/myreport.aspx" أو "mailto:helpdesk@mysite.com". وهذا العنصر مطلوب من أجل جميع أنواع التقارير.

— معاملات التقرير <reportArgs>: من أجل تقارير برنامج (تنفيذي EXE)، هذه المدخلة أي معاملات نسبية arguments لسطر أمر commandline يمكن تضمينه عند تشغيل البرنامج. هذا العنصر محقق فقط في تقارير البرنامج (التنفيذي)، وهو دائماً اختياري.

إشعارات التقرير <reportFlags>: من أجل تقارير برنامج (تنفيذي EXE)، هذه المدخلة هي الدلالات الاختيارية التي يجب إلحاقها إلى أمر التطبيق كمعاملات نسبية. عند هذه المرحلة الإشعار الوحيد هو U. عندما يتم وضع هذا العنصر إلى U، يتم إلحاق المعامل النسبي u-userid. إلى نص الأمر (حيث userid هو معرف تسجيل دخول المستخدم من حقل قاعدة البيانات UserName.LoginID). هذا العنصر متاح فقط في تقارير البرنامج التنفيذي EXE، وهو دائماً اختياري.

— الوصف <description>: هذا هو الأطول، وصف كثير الحشو verbose للتقرير، يصل إلى 200 حرف، والتي ستظهر على نموذج التقرير Report form عندما يختار المستخدم قائمة نموذج التقرير. سيساعد هذا الشرح المستخدم في اختيار التقرير المناسب. هذا العنصر متاح لجميع أنواع التقارير، ولكن دائماً اختياري.

• مجموعة التقرير <reportGroup>: تمثل مجموعة التصنيف category group، تستخدم للمجموعة المرئية وترك فراغ في بداية indent التقارير في قائمة العرض. يجب أن يحتوي هذا العنصر بالضبط العنصر <displayText> الوحيد، ولكن يمكن أن يحتوي أي عدد من العناصر <reportItem> أو <reportGroup> — <displayText> نص العرض: اسم مختصر أو وصف لهذه المجموعة، كما ستظهر في قائمة اختيار تقرير. هذا العنصر مطلوب. عند استخدام نوع التقرير "الجاهز" يتم وضع العنصر إلى واحد من قيم الانترغر التالية:

1. — تقرير مخرجات (مراجعة) البنود. Items Checked Out Report.

2. — تقرير متأخرات للبنود. Items Overdue Report.

3. — تقرير فقدان بنود. Items Missing Report.

4. مستحقات الغرامات بواسطة تقرير الزبائن. Fines Owed by Patrons Report.
5. تقرير إحصاءات قاعدة بيانات المكتبة. Library Database Statistics Report.
- هذا الوصف التقني يظهر في مستند مجموعة المصدر التقني، تم تطويره أصلاً في الفصل الرابع.

إنشاء فئة مدخلة تقرير. Create Report Entry Class

مع إمكانيات الدوت نت لتخزين جميع الكائنات كبنود صندوق قائمة ListBox items، نستطيع إنشاء فئة خاصة بحيث تحتوي جميع المعلومات المطلوبة لاختيار وتشغيل تقرير من قائمة التقارير. هذه الفئة بسيطة جداً، لا يوجد فيها شيء ماعدا حقول عامة أساسية، زائد إعادة قيادة لدالة " ToString "، والمستخدم بواسطة أداة صندوق القائمة لعرض كل بند قائمة بشكل مناسب.

في مشروع المكتبة، أضف ملف فئة جديدة وسمه ReportItem.vb من خلال مشروع Project << إضافة فئة Add Class. ومن ثم أضف العداد التالي لهذا الملف، ولكن أضفه خارج الحدود Class...End Class. يشير هذا العداد إلى ما يمثله نوع مدخلة كل بند قائمة كما يلي:

```
Public Class ReportItem
```

```
.  
. .  
. .
```

```
End Class
```

```
Public Enum ReportItemEnum
```

```
' نوع بند في قائمة اختيار تقرير
```

```
GroupLabel = 0
```

```
BuiltInCheckedOut = 1
```

```
BuiltInOverdue = 2
```

```
BuiltInMissing = 3
```

```
BuiltInFinesOwed = 4
```

```
BuiltInStatistics = 5
```

```
ExeProgram = 6
```

```
UrlProgram = 7
```

```
End Enum
```

أضف إلى نفس الملف، أعضاء الفئة ReportItem. تحتوي هذه الفئة جميع المعلومات التي نحتاجها لتشغيل التقارير المحملة من ملف التركيب configuration file.

```
' نسخة من بنود اختيار تقرير المستخدمة في نموذج
```

```
' ReportSelect
```

```
Public ItemType As ReportItemEnum
```

```
' مستوى الفراغ يبدأ بصفر
```

```
Public Indent As Integer
```

```
Public DisplayText As String
```

```
Public ReportPath As String ' ExeProgram / UrlProgram only
```

```
Public ReportArgs As String ' ExeProgram only
```

```
Public Description As String
```

```
Public Overrides Function ToString() As String
```

```
' بفراغ مسبق نص عرض
```

```
Return StrDup(Indent * 5, " ") & DisplayText
```

```
End Function
```

تصميم نموذج التقرير. Design the Report Form.

يستخدم أمناء المكتبة Librarians والمدراء administrators نموذج اختيار تقرير (شاهد الشكل التالي) لعرض التقارير. يتضمن النموذج أداة صندوق قائمة ListBox تعرض جميع التقارير ومجموعات التقارير، زر التشغيل Run الذي يشغل التقرير، وزر الإغلاق الذي يعود بالمستخدم إلى النموذج الرئيسي. تعرض أداة عنوانة label وصف كامل عن التقرير، عندما يكون متاح، تحت صندوق القائمة تماماً.



لإضافة هذا النموذج إلى مشروعك (راجع مشروع هذا الفصل).

على الرغم من أن المدير عمل على منح أسماء مفيدة لكل تقرير، الاسم المختصر لاسم كل تقرير من المحتمل أنه ما يزال غامض بالنسبة للمستخدم. يتضمن كل تقرير وصف كامل اختياري. عندما يختار المستخدم قائمة نموذج التقارير، يعمل معالج حدث على تحديث اللافتة " FullDescription " label التي تحت القائمة تماماً. أضف معالج الحدث هذا.

```
Private Sub AllReports_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs)
Handles AllReports.SelectedIndexChanged
    ' عرض وصف التقرير، إذا كان متاح
    Dim reportEntry As Library.ReportItem
    ' إزالة أي وصف سابق
    FullDescription.Text = "خيار تقرير لا يوجد"
    If (AllReports.SelectedIndex <> -1) Then
    ' إيجاد المحتوى وعرضه
        reportEntry = CType(AllReports.SelectedItem, Library.ReportItem)
        FullDescription.Text = reportEntry.Description
    End If
End Sub
```

ملئ التقارير من ملف التركيب. Populate Reports from Configuration File.

نحمل الطريقة RefreshReportList البيانات من ملف تركيب التقرير وتعالج النتائج في النهاية، سيتم تسجيل موقع هذا الملف في ملف التركيب للتطبيق، ولكن لن نعمل على إضافته حتى فصل لاحق.

أما الآن، دعنا ندخل موقع ملف الاختبار، ونعلمه من أجل التحديث اللاحق. فضلت اختيار كائنات XML للنمط القديم من أجل هذا الكود لأن ميزات XML للنمط الجديد التي نحتاجها لجعل الكود سهل الكتابة لن يتم تقديمها حتى فصل لاحق.

```
Private Sub RefreshReportList()
    ' التحميل بالقائمة التقارير المتاحة
    Dim configFile As String
    Dim configData As Xml.XmlDocument
    Dim reportEntry As ReportItem
    Dim counter As Integer
    On Error GoTo ErrorHandler
    ' تنظيف القائمة الموجودة
    AllReports.Items.Clear()
    ' الحصول على موقع ملف التركيب
    ' لعمل هذا: حمل هذا من تركيب التطبيق، حالياً علينا فقط كتابة كود القيمة
    configFile = "c:\ReportConfig.txt"
    ' تحميل ملف التركيب
    If (configFile <> "") Then
        If (System.IO.File.Exists(configFile)) Then
            ' تحميل الملف
            configData = New Xml.XmlDocument
            configData.Load(configFile)
            ' معالج ملف التركيب
            LoadReportGroup(configData.DocumentElement, 0)
        End If
    End If
    ' إذا كان لم ينتج عن ملف التركيب أي تقرير ظاهر في القائمة
    ' أضف التقارير الافتراضية
    If (AllReports.Items.Count = 0) Then
        For counter = 1 To CInt(ReportItemEnum.BuiltInStatistics)
            ' بناء مدخل التقرير
            reportEntry = New ReportItem
            reportEntry.Indent = 0
            reportEntry.ItemType = CType(counter, ReportItemEnum)
            Select Case reportEntry.ItemType
                Case ReportItemEnum.BuiltInCheckedOut
                    reportEntry.DisplayText = "إخراجها تم أو معارة البنود"
                Case ReportItemEnum.BuiltInOverdue
                    reportEntry.DisplayText = "متأخرة البنود"
                Case ReportItemEnum.BuiltInMissing
                    reportEntry.DisplayText = "مفقودة البنود"
                Case ReportItemEnum.BuiltInFinesOwed
                    reportEntry.DisplayText = "بغرامات مدين الزبون"
                Case ReportItemEnum.BuiltInStatistics
                    reportEntry.DisplayText = "البيانات قاعدة إحصاءات"
            End Select
            ' أضف مدخل التقرير إلى القائمة
            ' ----- Add the report entry to the list.
            AllReports.Items.Add(reportEntry)
        Next counter
    End If
End Sub
```

```

End If
Return
ErrorHandler:
GeneralError("ReportSelect.RefreshReportList", Err.GetException())
Resume Next
End Sub

```

لأن ملف تركيب التقرير يسمح بتداخل مجموعات التقرير لأي مستوى، نحتاج إلى استخدام روتين دوري recursive للنزول إلى كل مستوى متتابع (متتالي) مرة بعد مرة. يتم استدعاء الروتين LoadReportGroup بواسطة الروتين RefreshReportList، ويضيف جميع بنود التقارير ومجموعات التقارير ضمن مجموعة بدء التقرير. يتم استدعائه بشكل أولي (LoadReportGroup) من نقطة مرجعية للعنصر الجذري <reportList>. كل مرة يعمل على إيجاد العنصر الابن <reportGroup>، يستدعي نفسه مرة أخرى ولكن هذه المرة يبدأ من نقطة مرجعية reference point للعنصر الابن <reportGroup>.

```

Private Sub LoadReportGroup(ByVal groupNode As Xml.XmlNode, ByVal indentLevel As Integer)
    ' المستوى هذا وعند وينود مجموعات أضف
    ' .تحتاج ما واستدعي
    Dim scanNode As Xml.XmlNode
    Dim detailNode As Xml.XmlNode
    Dim reportEntry As ReportItem
    On Error GoTo ErrorHandler
    ' مجموعة أو بند كل عالج
    For Each scanNode In groupNode.ChildNodes
        ' القائمة في لتضمينه بند بناء
        reportEntry = New ReportItem
        reportEntry.Indent = indentLevel
        ' المعروض الاسم على احصل
        detailNode = scanNode.SelectSingleNode("displayText")
        If (detailNode Is Nothing) Then Continue For
        reportEntry.DisplayText = Trim(detailNode.InnerText)
        If (scanNode.Name = "reportGroup") Then
            ' جديدة عرض مجموعة ابدأ
            reportEntry.ItemType = ReportItemEnum.GroupLabel
            AllReports.Items.Add(reportEntry)
            ' الأبناء البنود إلى عد
            LoadReportGroup(scanNode, indentLevel + 1)
        ElseIf (scanNode.Name = "reportItem") Then
            ' موقعه سجل, بند هذا
            detailNode = scanNode.SelectSingleNode("reportPath")
            If Not (detailNode Is Nothing) Then reportEntry.ReportPath =
Trim(detailNode.InnerText)
            ' أمر لسطر نسي معامل أي على احصل
            detailNode = scanNode.SelectSingleNode("reportArgs")
            If Not (detailNode Is Nothing) Then reportEntry.ReportArgs =
Trim(detailNode.InnerText)
            ' معين بند علامات اية على احصل
            detailNode = scanNode.SelectSingleNode("reportFlags")
            If Not (detailNode Is Nothing) Then
                If (InStr(UCase(detailNode.InnerText), "U") > 0) And (LoggedInUserName <> "")
Then reportEntry.ReportArgs =
Trim(reportEntry.ReportArgs & " -u " & LoggedInUserName)
            End If
            detailNode = scanNode.SelectSingleNode("description")
            If Not (detailNode Is Nothing) Then reportEntry.Description =
Trim(detailNode.InnerText)
            If (scanNode.Attributes("type").Value = "built-in") Then
                If (IsNumeric(reportEntry.ReportPath) = False) Or (Val(reportEntry.ReportPath)
< 1) Or (Val(reportEntry.ReportPath) >
CInt(ReportItemEnum.BuiltInStatistics)) Then Continue For
                reportEntry.ItemType = CType(CInt(reportEntry.ReportPath), ReportItemEnum)
                AllReports.Items.Add(reportEntry)
            ElseIf (scanNode.Attributes("type").Value = "program") Then
                If (reportEntry.ReportPath = "") Then Continue For
                reportEntry.ItemType = ReportItemEnum.ExeProgram
                AllReports.Items.Add(reportEntry)
            ElseIf (scanNode.Attributes("type").Value = "url") Then
                If (reportEntry.ReportPath = "") Then Continue For
                reportEntry.ItemType = ReportItemEnum.UrlProgram
                AllReports.Items.Add(reportEntry)
            End If
        End If
        Next scanNode
    End Sub

```

Mhm76

```

Return
ErrorHandler:
    GeneralError("ReportSelect.LoadReportGroup", Err.GetException())
    Resume Next
End Sub

```

اعمل على إضافة حدث تحميل Load الفورم، والذي يحمل في محتوى الفورم ملف التركيب.

```

Private Sub ReportSelect_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    RefreshReportList()
End Sub

```

تشغيل التقارير. Running the Reports.

والآن، هذه جميع المجموعات والتقارير التي تظهر في القائمة، علينا تشغيل التقارير الفعلية بمعالج حدث زر "تشغيل ActRun" يعالج هذه المهمة حالياً سنعمل على إضافة فقط إطار العمل لدعم الاستدعاء بالنسبة لكل تقرير. أما تقارير "المبنية داخلياً (الجاهزة)" سيتم إضافتها في الفصل 21.

```

Private Sub ActRun_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ActRun.Click
    ' تشغيل التقرير المختار
    Dim reportEntry As Library.ReportItem
    On Error GoTo ErrorHandler
    ' التأكد من أن التقرير قد تم اختياره
    If (AllReports.SelectedIndex = -1) Then
        MsgBox("القائمة من تقرير اختر فضلك من", MsgBoxStyle.OkOnly Or MsgBoxStyle.Exclamation,
            ProgramTitle)
        Return
    End If
    ' كود مختلف من أجل كل نوع مدخلة
    reportEntry = CType(AllReports.SelectedItem, Library.ReportItem)
    Me.Cursor = Windows.Forms.Cursors.WaitCursor
    Select Case reportEntry.ItemType
        Case ReportItemEnum.GroupLabel
            ' لاتقرير من أجل مدخلات مجموعة
            MsgBox("القائمة من تقرير اختر فضلك من", MsgBoxStyle.OkOnly Or
                MsgBoxStyle.Exclamation, ProgramTitle)
        Case ReportItemEnum.BuiltInCheckedOut
            ' البنود تم استعارتها أو اخرجها
            ' للعمل: اكتب
            BasicReportCheckedOut()
        Case ReportItemEnum.BuiltInOverdue
            ' البنود متأخرة
            ' للعمل: اكتب
            BasicReportOverdue()
        Case ReportItemEnum.BuiltInMissing
            ' البنود مفقودة
            ' للعمل: اكتب
            BasicReportMissing()
        Case ReportItemEnum.BuiltInFinesOwed
            ' الغرامات المستحقة على الزبون
            ' للعمل: اكتب
            BasicReportFines()
        Case ReportItemEnum.BuiltInStatistics
            ' إحصاءات قاعدة بيانات المكتبة
            ' للعمل: اكتب التقرير
            BasicReportStatistics()
        Case ReportItemEnum.ExeProgram
            ' بدء البرنامج
            Process.Start(""" & reportEntry.ReportPath & "" " &
                reportEntry.ReportArgs)
        Case ReportItemEnum.UrlProgram
            ' بدء عناوين الانترنت
            ' ----- Start a URL.
            Process.Start(reportEntry.ReportPath)
    End Select
    Me.Cursor = Windows.Forms.Cursors.Default
    Return
ErrorHandler:
    GeneralError("ReportSelect.ActRun_Click", Err.GetException())
    Resume Next
End Sub

```

من أجل التقارير الخارجية، يستدعي معالج الحدث الطريقة `Process.Start`. وهي طريقة مدهشة تقبل إما تعبير سطر أمر قياسي، أو عنوان انترنت URL محقق، أو عنوان صفحة ويب.

الاتصال بنموذج اختيار تقرير. Connecting the Select Report Form.

لجعل التقارير متاحة بالنسبة للمستخدم، يجب علينا تمكين وصلة لنموذج التقرير من الفورم الرئيسي. فيما سبق عملنا على تضمين لوحة متميزة على تلك الفورم من أجل طباعة التقارير فقط. الزر "عمل التقارير ActDoReports" على تلك الفورم يعمل على إطلاق استدعاء هذا النموذج. أنشئ معالج حدث جديد للزر "عمل التقارير ActDoReports" وأضف الكود.

```
Private Sub ActDoReports_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ActDoReports.Click
    ReportSelect.ShowDialog()
End Sub
```

والآن، لدينا فهم راسخ (عميق) لعالم XML، سندع الفيچوال بيسك تعمل كل أعمال معالجة العمل الصعب من أجل عمليات إعداد التطبيق.