الفصل الثالث عشر
التطبيقات أحادية الوثيقة

تحتوى لغة Visual C++ على إمكانية فصل بيانات التطبيق عن عروضه. وهذه العروض عبارة عن الشكل أو المظهر الذى تظهر به هذه البيانات على الشاشة بحيث عليك عرض نفس البيانات على الشاشة بأكثر من طريقة، وذلك باستخدام تركيب مترابط يسمى تركيب الوثيقة والعرض.

بانتهاء هذا الفصل ستتعرف على:

- إنشاء التطبيق أحادى الوثيقة
- تطوير تطبيق يحتوى على تركيب الوثيقة والعرض

تحتوى معظم تطبيقات النوافذ على شكل ثابت يتمثل فى وجود شريط العنوان أعلى النافذة ثم شريط القوائم المنسدلة الذى يحتوى على قائمة أو أكثر، ثم صف أو أكثر من أشرطة الأدوات التى تعمل كمختصرات لتنفيذ الأوامر التى توجد غالباً داخل قائمة من القوائم، وبعد ذلك يظهر الجزء الرئيسى من النافذة والذى يتم فيه إظهار بيانات التطبيق، وأخيراً شريط المعلومات الذى يقبع فى أسفل النافذة. كما تظهر نفس خيارات القوائم ونفس أشرطة الأدوات فى معظم التطبيقات. لذا كان من الأولى أن يكون هناك شكل ثابت يتم التعديل فيه كى يتناسب مع التطبيق الذى تقوم بإنشائه، وهذا ما توفره لك لغة **Visual C++** بما تحويه من تصنيفات متعددة تساعدك باستخدام معالج التطبيقات **Application Wizard** على إنشاء تطبيقات تحتوى على قائمة وشريط أدوات وشريط المعلومات وأدوات أخرى حسب احتياجاتك بحيث يمكنك التحكم فى خصائص كل هذه المكونات بسهولة تامة، حيث تعمل التصنيفات المستخدمة فى تركيب مترابط يسمى تركيب الوثيقة والعرض **Document/View architecture** والذى يعنى فصل البيانات الحقيقية عن البيانات التى يتم عرضها للمستخدم، وذلك بوضع البيانات الحقيقية داخل تصنيف يسمى تصنيف الوثيقة **Document Class** ووضع بيانات العرض داخل تصنيف يسمى تصنيف العرض **View Class**.

يوجد ثلاثة أنواع من تطبيقات الوثيقة والعرض، الأول هو تطبيق الواجهة أحادية الوثيقة **Single Document Interface (SDI)** والثانى هو تطبيق الواجهة متعددة الوثائق **Multiple Document Interface (MDI)** ، أما النوع الثالث والأخير فهو التطبيق متعدد الوثائق ذات المستوى الواحد **Multiple top-level Documents** وهو نوع جديد أتى مع الإصدار السابق من **Visual C++** وفيه يكون لكل وثيقة أو مستند نافذته المستقلة عن بقية النوافذ الأخرى الموجودة بنفس التطبيق، وسيتم التركيز فى هذا الفصل بمشيئة الله على النوع الأول من هذه التطبيقات وهو تطبيق الواجهة أحادية الوثيقة **Single Document Interface (SDI)**.

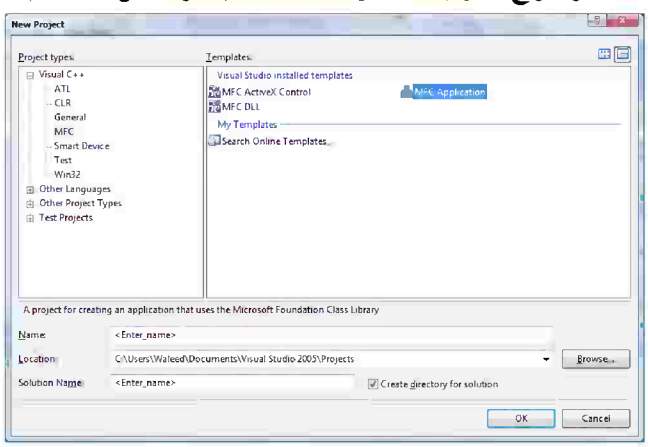
إنشاء تطبيق أحادى الوثيقة

تعد عملية إنشاء تطبيق يحتوى على السمات التى تظهر عادةً بالواجهة النوافذية ؛ مثل شريط الأدوات أو شريط المعلومات، أمراً غاية فى الصعوبة ما لم نستخدم معالج التطبيقات **Application Wizard** .حيث لا يعطينا المعالج السمات الأساسية فحسب، وإنما يمدنا بخيارات متعددة أخرى مثل إمكانية الاتصال بقاعدة بيانات أو العمل مع وظائف البريد الإلكترونى. وفيما يلى سنقوم بإنشاء تطبيق أحادى الوثيقة **SDI Application** وسنستخدم هذا التطبيق فى التعرف على التصنيفات المستخدمة لعمل تركيب الوثيقة والعرض **Document\View Architecture** .

لإنشاء تطبيق أحادى الوثيقة باستخدام معالج التطبيقات **Application Wizard**، تابع معنا الخطوات الآتية:

١. تأكد أنك داخل بيئة التطوير المتكاملة وإلا تابع الخطوات السابق شرحها لفتح بيئة التطوير من قائمة **Start**.

٢. من صفحة البدء، انقر الارتباط **Project** الموجود بجوار كلمة **Create** أو افتح قائمة **File** من شريط القوائم واختر **New Project** من القائمة المنسدلة، وفى الحالتين يظهر المربع الحوارى **New Project** (انظر شكل ١٣-١).



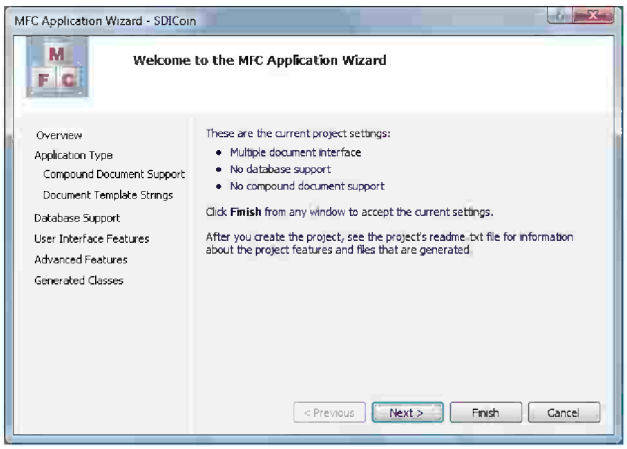
شكل ١٣-١ المربع الحوارى **New Project** المستخدم فى إنشاء مشروع جديد

٣. يحتوى المربع **Project Types** الموجود بالجزء الأيسر من المربع الحوارى **New Project** على الأدوات التى يمكنك استخدامها فى إنشاء مشروعك. تأكد من اختيار **MFC** داخل المجلد **Visual C++** لأننا نرغب فى إنشاء مشروع **MFC Application** باستخدام لغة **Visual C++ 2005**.

٤. يحتوى المربع **Templates** الموجود بالجزء الأيمن من المربع الحوارى **New Project** على العديد من أنواع المشروعات التى يمكنك الاختيار من بينها، حيث يعتمد اختيارك بالطبع على نوع التطبيق الذى ترغب فى إنشائه. ولأننا نرغب فى إنشاء تطبيق نوافذى، اختر رمز **MFC Application** من المربع **Templates**.

٥. يحتاج كل مشروع إلى اسم يميزه عن باقى المشروعات الأخرى. يتم تحديد هذا الاسم فى مربع النص **Name** بالمربع الحوارى **New Project**. اكتب اسماً مناسباً للمشروع الجديد وليكن **SDICoin** فى مربع النص **Name**. قم أيضاً بتحديد المجلد الذى سيحتوى على جميع ملفات المشروع داخل مربع النص **Location**.

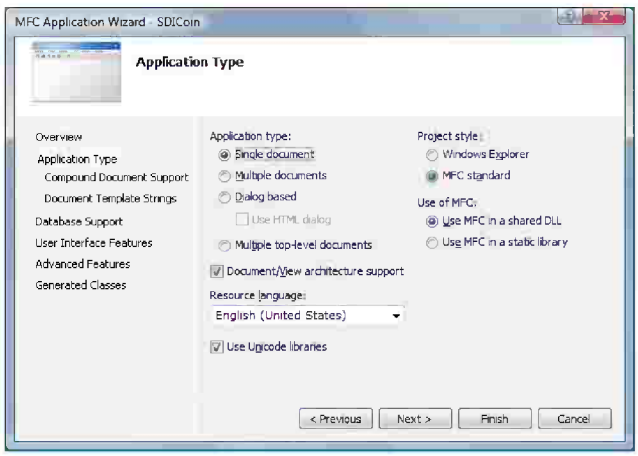
٦. انقر زر **Ok** ليبدأ معالج التطبيقات **Application Wizard** فى إنشاء هيكل المشروع باستخدام مكتبة **MFC** وكتابة الكود المطلوب نيابةً عنك وذلك بعد أن تحدد له نوع البرنامج المطلوب إنشاؤه حيث تظهر نافذة المعالج التى تحتوى على عدة تبويبات لاستخدامها فى تعيين خصائص المشروع الجديد، كما تحتوى أيضاً على الإعدادات الافتراضية للمعالج داخل التبويب النشط **Overview** (انظر شكل ١٣-٢). سنقوم فيما يلى بالتعرف على الإعدادات المرتبطة بالتطبيقات أحادية الوثيقة داخل كل تبويب من هذه التبويبات.



شكل ١٣-٢ نافذة معالج التطبيقات Application Wizard

التبويب *Application Type*

من نافذة معالج التطبيقات، انقر التبويب **Application Type** لتنشيطه حيث يمكنك من خلال هذا التطبيق اختيار نوع التطبيق حيث يوجد أربعة أنواع من واجهات التطبيق كما ذكرنا من قبل (انظر شكل ١٣-٣):



شكل ١٣-٣ محتويات التبويب Application Type

ومن هذه الأنواع التطبيق أحادى الوثيقـــة **Single Document Interface (SDI)** ويتم من خلاله فتح مستند واحد فى الوقت الواحد مثل برنامج **Notepad** الموجود داخل

Windows. فحينما تختار Open من قائمة File، يتم إغلاق الملف المفتوح حالياً قبل فتح الملف الجديد. ولأننا نرغب فى إنشاء تطبيق أحادى الوثيقة فى المثال الذى بين أيدينا، انقر زر الخيار Single document وفى هذه الحالة تظهر صورة صغيرة بالركن الأيسر العلوى من النافذة تحتوى على مظهر التطبيق المختار.

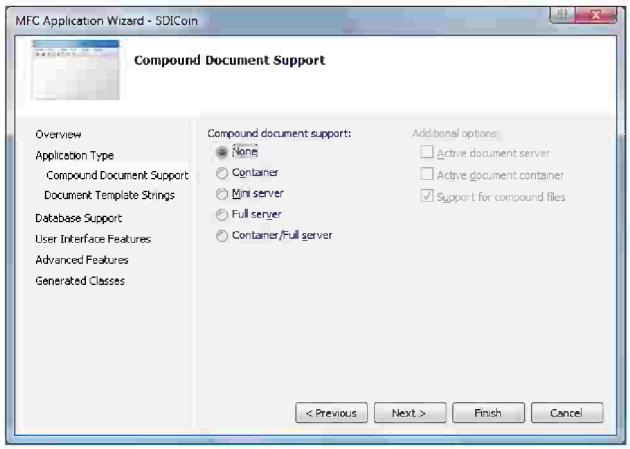
يحتوى التبويب Application Type على العديد من الخيارات الأخرى التى نوضحها فيما يلى:

- نشط مربع الاختيار Document/View architecture Support إذا أردت تدعيم تركيب العرض والوثيقة الذى نحن بصدده فى هذا الفصل.
- يمكنك تغيير اللغة المستخدمة لتسمية الموارد من القائمة المنسدلة Resource Language ومن الأفضل الإبقاء على اللغة الافتراضية (اللغة الإنجليزية).
- يمكنك من خلال المجموعة Project Style تعيين مظهر التطبيق سواءً فى صورة مستكشف Windows (Windows Explorer) أو فى الصورة التقليدية MFC Standard.
- يمكنك من خلال المجموعة Use of MFC تحديد إذا ما كنت ترغب فى استخدام مكتبة MFC فى صورة ملفات DLL مشتركة أم ربطها بالملفات التنفيذية. فإذا توقعت أن يكون لدى مستخدمى تطبيقك ملفات MFC DLLs لأنهم مطورون أو لديهم تطبيقات أخرى تستخدم نفس الملفات أو أنهم لن يمانعوا فى تثبيت هذه الملفات تماماً كما يقومون بتثبيت الملف التنفيذى للتطبيق، فى جميع هذه الأحوال يكون من الأفضل تنشيط زر الاختيار Use MFC in a shared DLL وهذا من شأنه تقليل حجم الملف التنفيذى وزيادة سرعة الاتصال مع الملفات الأخرى. وعدا ذلك، قم بتنشيط زر الاختيار Use MFC in a static library.
- لاستخدام مكتبات الكود الموحد Unicode Libraries، يجب أن تقوم بتثبيت ملفات هذه المكتبات بنفسك أثناء تثبيت Visual Studio 2005 أو بعد ذلك من خلال إضافة/إزالة مكونات البرنامج وإلا لن تتمكن من بناء وتنفيذ التطبيقات

التى تقوم بإنشائها. ففى حالة عدم تثبيت هذه الملفات، قم بتعطيل مربع الاختيـــار **Use Unicode libraries**.

التبويب Compound Document Support

يستخدم هذا التبويب لتعيين مقدار تدعيم تطبيقك لمفهوم الوثيقة المركبة **Compound Document** وهو مصطلح بديل لمفهوم الربط والتضمين **OLE** (انظر شكل ١٣-٤).



شكل ١٣-٤ محتويات التبويب **Compound Document Support**

يحتوى هذا التبويب على خمسة خيارات يمكن بيانها كما يلى:

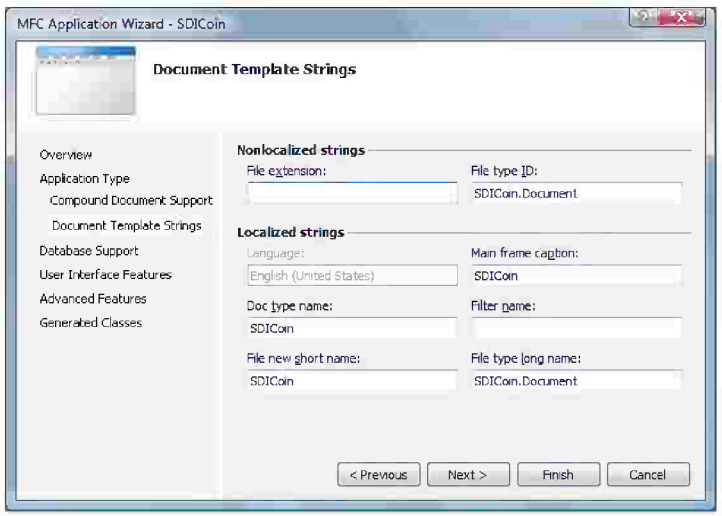
- نشط زر الاختيار **None** فى حالة عدم احتواء الوثائق الموجودة بتطبيقـــك علـــى كائنات أخرى أو لا يقوم بإنشاء كائنات لاستخدمها من قِبَل التطبيقات الأخرى.
- نشط زر الاختيار **Container** إذا أردت أن يحتوى تطبيقك على كائنات مضمنة أو مرتبطة مثل مستندات **Word** أو ورق عمل **Excel**.
- نشط زر الاختيار **Mini-server** إذا كان تطبيقك يقوم بإعداد الكائنات التى يتم تضمينها أو ربطها بالتطبيقات الأخرى إلا أنها لا تحتاج إلى تشغيل تطبيقات مستقلة.
- نشط زر الاختيار **Full-server** إذا كان تطبيقك يقوم بإعداد الكائنات التى يتم تضمينها أو ربطها بالتطبيقات الأخرى وتقوم بتشغيل تطبيقات مستقلة.

- نشط زر الاختيار **Container/Full-server** إذا كان تطبيقك يقـــوم بإعـــداد الكائنات التى يتم تضمينها أو ربطها بالتطبيقات الأخرى كما يقوم بإعداد الكائنات التى تعمل مع التطبيقات الأخرى.

فى المثال الذى بين أيدينا، ابق على الخيار **None** كما هو.

التبويب Document Template Strings

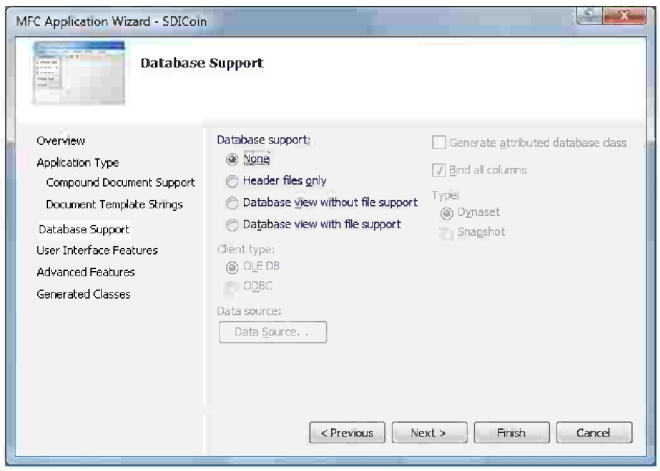
عند إنشاء تطبيقاتك باستخدام معالج التطبيقات، يقوم المعالج بإنشاء العديد من الأسمـــاء. يمكنك التحكم فى هذه الأسماء من خلال محتويات التبويـــب **Document Template Strings** (انظر شكل ١٣-٥). يمكنك تغيير هذه الأسماء بالإضافة إلى تعيين امتـــدادات الملفات التى يقوم المعالج بإنشائها ولكن بعد أن تزداد خبرتك بعناصر التطبيـــق أحـــادى الوثيقة كما سنعرف من خلال هذا الفصل.



شكل ١٣-٥ التبويب **Document Template Strings**

التبويب Database Support

يمكنك من خلال هذا التبويب تحديد مدى تدعيم تطبيقك لقواعد البيانات، حيث يحتـــوى هذا التبويب على أربعة خيارات يمكن بيانها كما يلى (انظر شكل ١٣-٦):



شكل ١٣-٦ التبويب Database Support

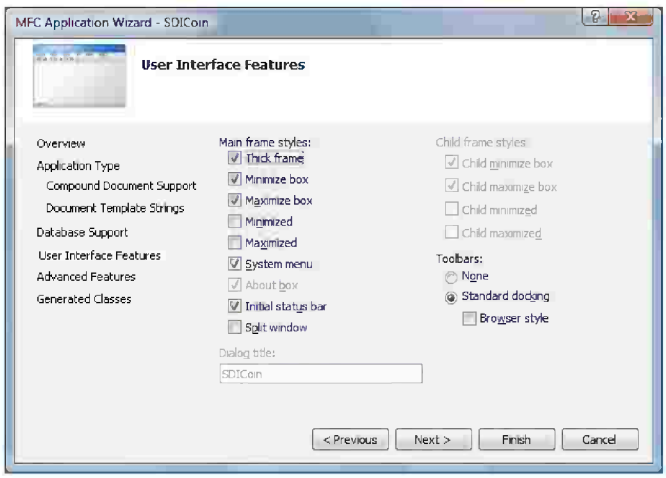
- نشط زر الاختيار **None** إذا كنت لا ترغب فى تدعيم تطبيقك لقواعد البيانات.
- نشط زر الاختيار **Header files only** إذا كنت ترغب فى تدعيم تطبيقك لقواعد البيانات إلا أنك لا ترغب فى أن يقوم المعالج بإنشاء عرض لقاعدة البيانات أو القائمة التى يمكنك استخدامها فى الوصول إلى قاعدة البيانات.
- نشط زر الاختيار **Database view without file support** إذا كنت ترغب فى تدعيم تطبيقك لقواعد البيانات على أن يقوم المعالج بإنشاء عرض لقاعدة البيانات باستخدام التصنيف **CRecordView** وقائمة للسجلات إلا أنك لا ترغب فى تخزين أى بيانات على القرص الصلب.
- نشط زر الاختيار **Database view with file support** إذا كنت ترغب فى تدعيم تطبيقك لقواعد البيانات على أن يقوم المعالج بإنشاء عرض لقاعدة البيانات باستخدام التصنيف **CRecordView** وقائمة للسجلات وكذلك ترغب فى تخزين البيانات على القرص الصلب.

فى حالة اختيار أى من الخيارات الثلاثة الأخيرة، يجب أن تقوم بتحديد نوع قاعدة البيانات من الجزء **Client Type**، كما يجب أن تقوم بتحديد مصدر البيانات بنقر زر **Data**

Source فى حالة اختيارك لأيٍ من الخيارين الأخيرين.

التبويب *User Interface Features*

يمكنك من خلال التبويب User Interface Features تعيين الخيارات التى تتحكم فى مظهر التطبيق وذلك كما يلى (انظر شكل ١٣-٧):



شكل ١٣-٧ محتويات التبويب User Interface Features

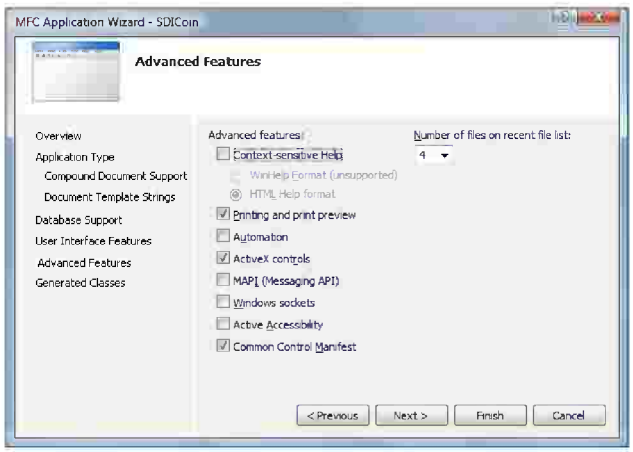
- نشط مربع الاختيار Thick Frame إذا أردت وضع إطار حول النافذة كى تتمكن من تغيير حجم هذه النافذة من خلاله.
- نشط مربع الاختيار Minimize box إذا أردت إظهار زر التقليص (التصغير) بشريط عنوان النافذة.
- نشط مربع الاختيار Maximize box إذا أردت إظهار زر التكبير بشريط عنوان النافذة.
- نشط مربع الاختيار Minimized إذا أردت أن تظهر النافذة مصغرة بمجرد فتحها.
- نشط مربع الاختيار Maximized إذا أردت أن تظهر النافذة مكبرةً بمجرد فتحها. ويلاحظ أن تنشيط هذا المربع، يتسبب فى تعطيل مربع الاختيار السابق والعكس

صحيح.

- نشط مربع الاختيار **System menu** إذا أردت إظهار قائمة التحكم بالركن الأيسر العلوى من النافذة.
- نشط مربع الاختيار **Initial Status bar** إذا أردت إظهار شريط معلومات بنافذة التطبيق.
- نشط مربع الاختيار **Split window** إذا أردت تدعيم تطبيقك لعملية تقسيم النوافذ.
- من الجزء **Toolbars** قم بتنشيط زر الاختيار **Standard docking** إذا كنت ترغب فى تضمين شريط أدوات بنافذة التطبيق وإلا قم بتنشيط زر الاختيار **None**.

التبويب Advanced Features

يحتوى التبويب **Advanced Features** على بعض الخيارات الإضافية التى يمكنك تضمينها داخل تطبيقك وذلك كما يلى (انظر شكل ١٣-٨):



شكل ١٣-٨ التحكم فى سمات وخيارات المشروع

- نشط مربع الاختيار **Context-sensitive Help** إذا أردت إضافة وسيلة المساعدة إلى نافذة التطبيق. وفى هذه الحالة يمكنك تحديد أسلوب عرض المساعدة

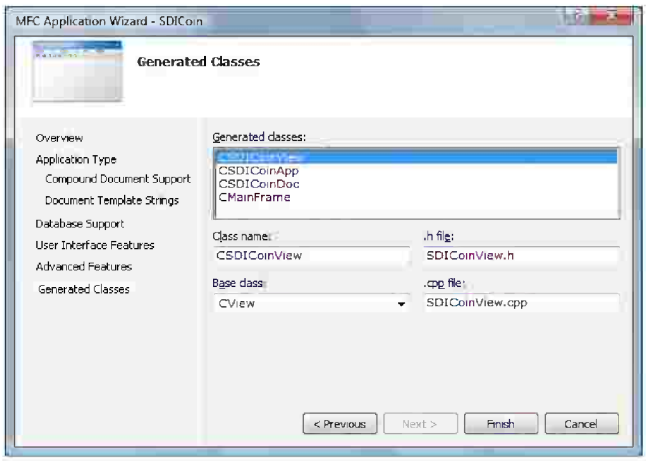
سواءً فى الصورة النصية المعتادة (بتنشيط زر الاختيار **WinHelp Format**) أو فى صورة **HTML** (بتنشيط زر الاختيار **HTML Help Format**).

- نشط مربع الاختيار **Printing and print preview** إذا أردت تضمين سمات الطباعة ومعاينتها إلى تطبيقك.
- نشط مربع الاختيار **Automation** إذا أردت الإذعان الكامل ما بين تطبيقك والتطبيقات الأخرى.
- نشط مربع الاختيار **ActiveX Controls** إذا أردت تدعيم تطبيقك لكائنات **ActiveX**.
- نشط مربع الاختيار **Windows sockets** إذا كنت تنوى استخدام هذا التطبيق من خلال الإنترنت من خلال التجويفات **Sockets** كما سنعرف فيما بعد.
- نشط مربع الاختيار **Active Accessibility** إذا أردت تدعيم تطبيقك لمفهوم الوصول النشط.
- نشط مربع الاختيار **Common Control Manifest** إذا أردت إنشاء كشف مصاحب للتطبيق بأدوات التحكم الجديدة المصاحبة لنظام التشغيل **Windows XP**.

التبويب Generated Classes

يحتوى هذا التبويب على التصنيفات التى سيقوم معالج التطبيقات بإنشائها بحيث يمكنك تغيير أسمائها إن أحببت (انظر شكل ١٣-٩).

يمكنك الآن معاينة خصائص التطبيق تبعاً للخيارات التى قمت بتحديدها بالتبويبات المختلفة وذلك من خلال التبويب **Overview** (راجع شكل ١٣-٢). انقر زر **Finish** كى يبدأ المعالج فى إنشاء التطبيق الجديد.



شكل ١٣-٩ التصنيفات الناتجة داخل التبويب **Generated Classes**

يقوم التطبيق **SDICoin** بعرض مجموعة من العملات المتراصة فوق بعضها **(Stacked)** والتى يمكنك إضافة أو حذف أيٍ منها باستخدام الخيارات المتاحة بشريط القوائم. سنقوم بتخزين البيانات المتمثلة فى عدد العملات داخل تصنيف الوثيقة **Document Class** كما سنقوم بعرض هذه البيانات للمستخدم باستخدام تصنيف العرض **View Class** وذلك بمساعدة دوال الوصول الموجودة بتصنيف الوثيقة. وعلى ذلك فإن الاختلاف فى طريقة العرض لا يؤثر بأى حالٍ من الأحوال على البيانات الأساسية المخَّزنة داخل تصنيف الوثيقة وهذا هو المراد من استخدام تركيب الوثيقة والعرض **Document\View Architecture**.

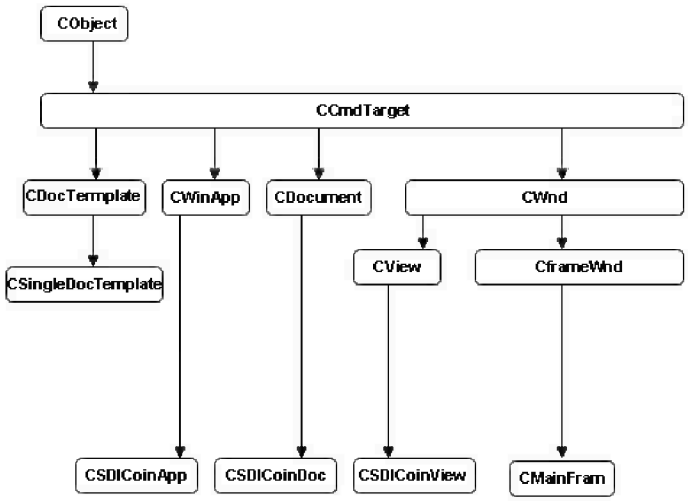
يشتمل التطبيق أحادى الوثيقة على تصنيف وثيقة واحد فقط منبثق من تصنيف الوثيقة الرئيسى **CDocument**، كما يحتوى على تصنيف عرض واحد فقط منبثق من أحد تصنيفات العرض الرئيسية الموضحة بجدول ١٣-١. وقد قام المعالج تلقائياً باشتقاق تصنيف العرض فى هذا المثال من التصنيف الافتراضى **CView**.

جدول ١٣-١ تصنيفات العرض الرئيسية داخل معالج التطبيقات

اسم التصنيف	الوصف
CView	التصنيف الافتراضى الذى يحتوى على معظم وظائف العرض بما فى ذلك التفاعل مع تصنيف الوثيقة
CScrollView	يتناسب مع المواقف التى يتطلب فيها وجود أشرطة تمرير
CListView	يستخدم مربع السرد كوسيلة للعرض
CTreeView	يستخدم شجرة البيانات كوسيلة للعرض
CEditView	يستخدم مربع النص متعدد السطور كوسيلة للعرض
CRichEditView	يستخدم مربع النص المنوع كوسيلة للعرض
CFormView	يستخدم قالب المربع الحوارى كوسيلة للعرض
CRecordView	يستخدم لتحقيق الاتصال بين العرض وقواعد البيانات
CHtmlView	يستخدم عرض مستعرض الإنترنت لتمكين إنشاء تطبيقات مستعرض الويب

فهم تصنيفات التطبيق أحادى الوثيقة

يختلف التركيب العام لتصنيفات التطبيق أحادى الوثيقة نوعاً ما عن مثيله فى التطبيق الحوارى. بالنظر إلى تبويب عرض التصنيفات **ClassView** من نافذة عمل المشروع **SDICoin** الذى أنشأناه منذ قليل أو شكل ١٣-٩ السابق، تلاحظ وجود أربعة تصنيفات يمكن بيانها كما يلى (انظر شكل ١٣-١٠):



شكل ١٣-١٠الشكل الهرمى لتصنيفات التطبيق أحادى الوثيقة

- تصنيف التطبيق **CSDICoinApp** المنبثق من التصنيف **CWinApp** والذى يختلف فى تمثيله عن التصنيف المستخدم بالمربعات الحوارية ومنوط به تنفيذ مهام متعددة. فهو يستخدم لإعطاء القيم الاستهلالية للتطبيق، كما أنه المسئول الرئيسى عن الربط بين تصنيفات الوثيقة والعرض والإطار، كما أنه أيضاً يستقبل رسائل نظام التشغيل ثم يقوم بتوجيهها إلى النافذة المناسبة.
- تصنيف الوثيقة **CSDICoinDoc** ومهمته الأساسية هى تخزين بيانات التطبيق. وعلى الرغم من بساطة البيانات التى سنستخدمها فى هذا المثال (عدد صحيح) ، إلا أنه غالباً ما نستخدم بيانات أكثر تعقيداً لتلبية الاحتياجات المنوط بالتطبيق تنفيذها. ففى تطبيقات معالجة النصوص، لا نحتاج لتخزين النص فقط وإنما نحتاج أيضاً لتخزين معلومات التنسيق اللازمة لعرض هذا النص.
- تصنيف العرض **CSDICoinView** وهو المسئول عن عرض بيانات الوثيقة الموجودة بالتصنيف **CSDICoinDoc** للمستخدم، كما يمكّن المستخدم من التفاعل مع التطبيق.

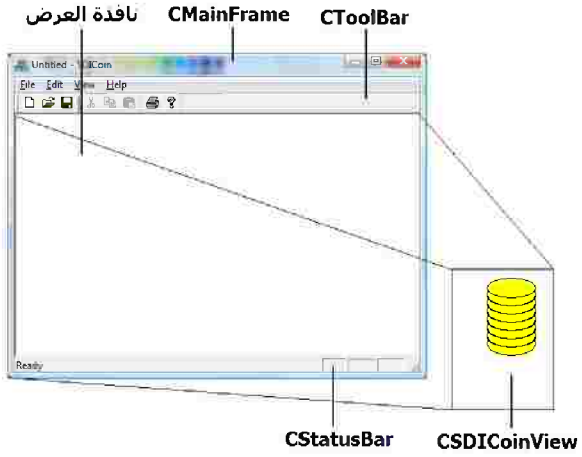
- تصنيف الإطار **CMainFrame** وهو المسئول عن إنشاء النافذة التى سيستخدمها التطبيق وينبثق من التصنيف **CFrameWnd** المستخدم لإنشاء نافذة بسيطة تحتوى على شريط عنوان به أزرار التكبير والتصغير والإغلاق وقائمة النظام، كما يمكن تحريك النافذة وإعادة تحجيمها، كما يتم أيضاً إظهار اسم الوثيقة تلقائياً على شريط العنوان.

العناصر المرئية بالتطبيق أحادى الوثيقة

إذا قبلت مجموعة الخيارات الافتراضية بالتبويب **User Interface Features** لمعالج التطبيقات **Application Wizard**، ستحصل على تطبيق يحتوى على شريط عنوان به أزرار التكبير والتصغير والإغلاق وقائمة النظام وشريط للأدوات وآخر لشريط المعلومات، إلا أن بعض هذه السمات اختيارى يمكنك تنشيطه أو تعطيله تبعاً لمتطلبات تطبيقك (راجع شكل ١٣-٧).

بالنظر إلى نافذة التطبيق أحادى الوثيقة (انظر شكل ١٣-١١) تجد أنها تحتوى على العناصر التالية:

- إطار التصنيف ويتم تمثيله بواسطة التصنيف **CMiainFrame** المنبثق من التصنيف **CFrameWnd**.
- أشرطة الأدوات المرتبطة بإطار التطبيق ويتم تمثيلها بواسطة التصنيف **CToolbar**.
- العرض كجزء من نافذة الإطار ويتم تمثيله بواسطة التصنيف **CSDICoinView**.
- شريط المعلومات المتصل بنافذة الإطار ويتم تمثيله بواسطة التصنيف **CStatusBar**.
- نافذة العرض والتى تعد جزءاً من نافذة إطار التطبيق.



شكل ١٣-١١ العناصر المرئية بالتطبيق أحادى الوثيقة

استخدام الوثائق والعروض معاً

تتيح لك عملية فصل البيانات الموجودة بالوثيقة عن الشكل الذى ستظهر عليه هذه البيانات للمستخدم التغيير والتعديل فى التطبيق بسهولة تامة. فإذا أردت مثلاً تمثيل البيانات الحالية بشكل جديد، فكل ما عليك أن تقوم بإنشاء تصنيف يحتوى على نمط العرض الجديد دون أن تقوم بأى تغيير فى البيانات نفسها.

ولعل تمثيل التفاعل بين الوثيقة والعروض المختلفة أمرٌ شديد الأهمية، حيث تكمن الصعوبة فى تحديد المعلومات التى تخص الوثيقة والأخرى التى تخص العرض لأن ذلك يعتمد بشكل كبير على نوع التطبيق نفسه. فغالباً لا ينبغى أن يحتوى تصنيف العرض على البيانات الحقيقية التى يجب أن تكون داخل تصنيف الوثيقة، إلا أن ذلك لا يكون مناسباً فى بعض التطبيقات مثل معالج النصوص الذى يقوم المستخدم فيه بتعديل البيانات مع كل ضغطة مفتاح من لوحة المفاتيح. ففى هذه الحالة إذا احتفظنا بالبيانات داخل تصنيف الوثيقة، سيتسبب ذلك فى بطئ التطبيق بشكل كبير.

إعطاء القيم الابتدائية لبيانات الوثيقة

يتم دائماً اشتقاق تصنيف الوثيقة من التصنيف الأساسى للوثائق **CDocument** والذى يحتوى بدوره على دوال ومتغيرات لتخزين بيانات التطبيق المختلفة. ولأن دالة **DeleteContents()** غالباً ما يتم استدعاؤها عند تنفيذ العمليات المختلفة، فيحبذ وضع القيم الابتدائية فيها.

إضافة المتغيرات إلى الوثيقة

يمكنك إضافة المتغيرات إلى تصنيف الوثيقة بنفس الطريقة المتبعة مع التصنيفات الأخرى باختيار **Add Variable** من القائمة الموضعية. وللحفاظ على سرية البيانات، ينبغى أن تكون البيانات من النوع المحمى **Protected** حتى يمكن تغييرها فقط من خلال الدوال الأعضاء فى التصنيف. ولتمكين العروض من الإطلاع على بيانات الوثيقة، يجب أن يحتوى تصنيف الوثيقة على دوال الوصول التى تحتوى بدورها على مراجع للمتغيرات.

والآن وبعد أن تعرفت على السمات العامة لتركيب الوثيقة والعرض، قد تكون بعض النقاط غامضة إلى حدٍ ما، لذا من الضرورى توضيح هذه النقاط من خلال التطبيق **SDICoin** الذى بدأناه فى أول الفصل. تابع معنا الخطوات الآتية:

١. من نافذة عرض التصنيفات **Class View**، انقر تصنيف الوثيقة **CSDICoinDoc** بزر الفأرة الأيمن ثم اختر **Add**⇐**Add Variable** من القائمة الموضعية، تظهر نافذة معالج إضافة متغير جديد.

٢. اختر **int** من مربع السرد والتحرير فى مربع النص **Variable Type** لتعريف نوع المتغير ثم اكتب اسم المتغير فى مربع النص **Variable Name** وليكن **m_nCoins** ثم اختر **Protected** من مربع السرد والتحرير **Access**.

٣. انقر زر **Finish** لإغلاق نافذة المعالج وإضافة المتغير الجديد.

٤. من نافذة عرض التصنيفات Class View، انقر تصنيف الوثيقة CSDICoinDoc بزر الفأرة الأيمن مرةً أخرى ثم اختر Add>Add Function من القائمة الموضعية، تظهر نافذة معالج إضافة دالة جديدة.

٥. اختر int فى مربع السرد Return Type لتعريف نوع الدالة ثم اكتب اسم الدالة فى مربع النص Function name وليكن GetCoinCount ثم اختر Public من مربع السرد Access.

٦. انقر زر Finish لإغلاق نافذة المعالج وإضافة الدالة الجديدة إلى التصنيف.

٧. قم بإدخال السطر التالى إلى الدالة الجديدة بدلاً من العبارة الافتراضية الموجودة بالدالة والتى تقوم بإرجاع القيمة 0:

```
return m_nCoins;
```

٨. تأكد من ظهور مربع الخصائص ثم قم بتنشيط التصنيف CSDICoinDoc داخل نافذة عرض التصنيفات.

٩. انقر الزر  من شريط أدوات مربع الخصائص وقم بالوصول إلى العنصر DeleteContents داخل القائمة الناتجة.

١٠. انقر مربع السرد والتحرير المجاور ثم اختر DeleteContents <Add>، يظهر قالب الدالة DeleteContents() داخل نافذة الكود.

١١. قم بإدخال السطر التالى إلى الدالة DeleteContents():

```
m_nCoins = 1;
```

وبذلك قمنا بتعريف متغير محمى باسم m_nCoins وكذا تخصيصه بقيمة افتراضية داخل الدالة DeleteContents(). كما قمنا بإضافة الدالة GetCoinCount() التى سيتم استخدامها من قِبَل تصنيف العرض للتعرف على قيمة المتغير m_nCoins.

إنشاء العرض من بيانات الوثيقة

كى يتمكن العرض من الحصول على بيانات الوثيقة، يجب أن يحقق أولاً عملية الوصول لعنصر الوثيقة، وهذا ما يمنحه لك المعالج تلقائياً عن طريق الدالة **GetDocument()** التى تقوم بإرجاع مؤشر لعنصر الوثيقة المرتبط بالعرض. يقوم المعالج فى الواقع بإنشاء دالتين وليس دالة واحدة، الأولى داخل ملف **SDICoinView.cpp** والأخرى داخل ملف **SDICoinView.h** كما يلى:

SDICoinView.cpp

```
#ifdef _DEBUG
CSDICoinDoc* CSDICoinView::GetDocument() // non-debug
                                          version is inline
{
        ASSERT(m_pDocument-> IsKindOf(RUNTIME_CLASS
        (CSDICoinDoc)));
      return (CSDICoinDoc*)m_pDocument;
}
#endif //_DEBUG
```

SDICoinView.h

```
#ifndef _DEBUG  // debug version in SDICoinView.cpp
inline CSDICoinDoc* CSDICoinView::GetDocument()
  { return (CSDICoinDoc*)m_pDocument; }
#endif
```

نلاحظ أن كلاً من الدالتين يؤدى تقريباً نفس المهمة، الفرق الوحيد أن الأولى تستخدم مع الإصدار **Debug** والثانية تستخدم مع الإصدار **Release**.

فى المثال الذى بين أيدينا، يعتبر التصنيف **CSDICoinView** هو المسئول الأول عن عرض مجموعة العملات باستخدام الدالة **OnDraw()** التى يتم استدعاؤها كلما تغير شكل العرض عند تحجيم النافذة مثلاً. قم بفتح الدالة **OnDraw()** داخل نافذة المحرر ثم قم بتعديل كود الدالة كما يلى:

```
1.    void CSDICoinView::OnDraw(CDC* pDC)
2.    {
3.      CSDICoinDoc* pDoc = GetDocument();
```

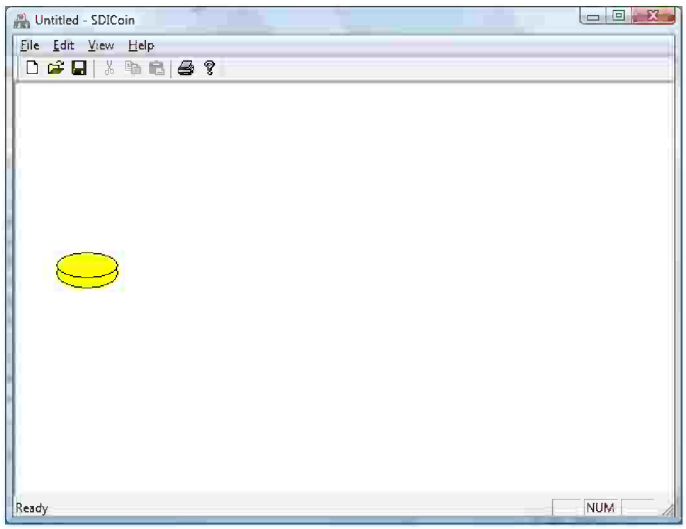
```
4.      ASSERT_VALID(pDoc);
5.      // TODO: add draw code for native data here
6.      CBrush* pOldBrush = pDC->GetCurrentBrush();
7.      CBrush br;
8.      br.CreateSolidBrush(RGB(255,255,0));
9.      pDC->SelectObject(&br);
10.     for(int nCount = 0;nCount<pDoc->
                                    GetCoinCount();nCount++)
11.     {
12.        int y = 200 - 10 *nCount;
13.        pDC->Ellipse(40,y,100,y-30);
14.        pDC->Ellipse(40,y-10,100,y-35);
15.     }
16.     pDC->SelectObject(pOldBrush);
17.  }
```

وعن هذا الكود، نوضح ما يلى:

- يتم إنشاء السطور رقم ٤ و٥ تلقائياً من قِبَل المعالج ، حيث تقوم الدالة GetDocument() بإرجاع مؤشر عنصر الوثيقة وتخزينه داخل المتغير pDoc.
- فى السطر رقم ٦ يتم تخزين مؤشر الفرشاة المستخدمة حالياً فى بيئة العنصر Device Context داخل المتغير pOldBrush حتى يمكننا استعادتها بعد ذلك.
- فى السطور من ٧ إلى ٩ قمنا بإنشاء فرشاة صفراء مصمتة وتخزينها داخل المتغير br ومن ثم داخل Device Context.
- فى السطور من ١٠ إلى ١٥ يتم استخدام الدوارة for بعدد العملات الموجود داخل المتغير nCoins لرسم عدد ٢ قطع ناقص لكل عملة حتى تظهر بشكل ثلاثى الأبعاد.
- فى السطر رقم ١٦ يتم استعادة الفرشاة القديمة من المتغير pOldBrush داخل Device Context مرة أخرى.

قم ببناء التطبيق وتنفيذه، تحصل على نتيجة مماثلة للشكل ١٣-١٢.

سنتعرف على كيفية إنشاء فرش الرسم واستخدامها داخل Device Context فيما بعد.



شكل ١٣-١٢ التطبيق أحادى الوثيقة

استخدام الموارد الرئيسية

كما ذكرنا من قبل، يتم إضافة الكثير من الموارد مثل القائمة وشريط الأدوات تلقائياً إلى التطبيق للحصول على العناصر المرئية للتطبيق أحادى الوثيقة. وعلى ذلك فجميع الموارد تعمل تلقائياً بشكل صحيح. فالخيار **New** داخل قائمة **File** مثلاً يتسبب فى استدعاء الدالة **CWinApp::OnFileNew()** لإنشاء وثيقة جديدة وكذلك عندما يقوم المستخدم بنقر زر **New** من شريط الأدوات. يمكنك تغيير الإعدادات الافتراضية لهذه الموارد وذلك بتحريرها داخل نافذة المحرر، إلا أننا ننصح بترك هذه الإعدادات كما هى ثم إنشاء إعدادات جديدة تتناسب مع تطبيقك إن أردت.

فى المثال الذى بين أيدينا، سنقوم بإضافة خيارين لشريط القوائم، أحدهما لإضافة عملة والآخر لحذف أخرى. تابع معنا الخطوات الآتية:

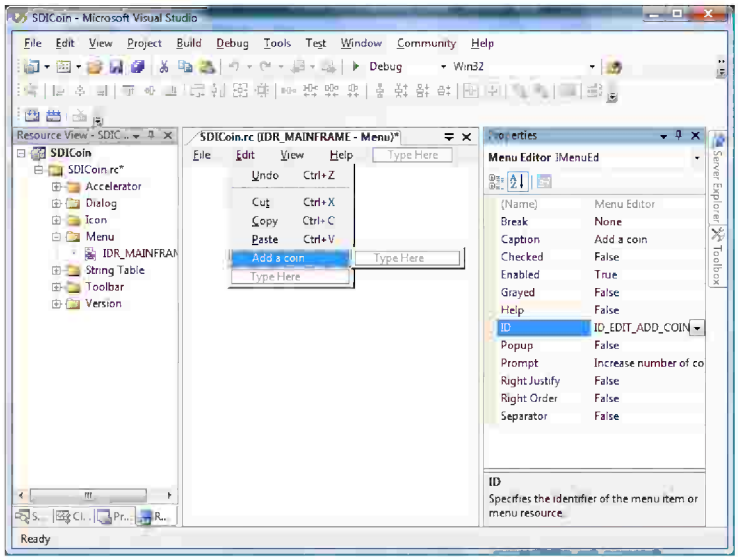
١. من تبويب عرض الموارد **Resource View**، قم بتوسيع المجلد **Menu** ثم انقر المورد **IDR_MAINFRAME** نقراً مزدوجاً، تظهر القائمة داخل نافذة محرر الموارد.

٢. انقر عنصر القائمة **Edit** من داخل محرر الموارد، تظهر قائمة **Edit** المنسدلة.

٣. انقر آخر عنصر بالقائمة لاختياره، يظهر مربع نص مكان العنصر تمهيداً لكتابة عنوان العنصر الجديد.

٤. اكتب **Add a coin** ومن مربع الخصائص قم بتغيير اسم العنصر إلى اسم مناسب داخل الخاصية **ID** وليكن **ID_EDIT_ADD_COIN** (انظر شكل ١٣-١٣).

٥. قم بإدخال تعليق للعنصر الجديد إلى الخاصية **Prompt** وليكن **Increase number of coins**.



شكل ١٣-١٣ إضافة عنصر قائمة جديد لقائمة **Edit**

٦. انقر آخر عنصر بالقائمة لاختياره، يظهر مربع نص مكان العنصر تمهيداً لكتابة عنوان العنصر الجديد.

٧. اكتب **Remove a coin** ومن مربع الخصائص قم بتغيير اسم العنصر إلى اسم مناسب داخل الخاصية **ID** وليكن **ID_EDIT_REMOVE_COIN**.

٨. قم بإدخال تعليق للعنصر الجديد إلى الخاصية **Prompt** وليكن **Decrease number of coins**.

٩. تأكد من ظهور مربع الخصائص ثم قم بتنشيط التصنيف CSDICoinDoc داخل نافذة عرض التصنيفات.

١٠. انقر زر الأحداث من شريط أدوات مربع الخصائص وقم بتوسيع العنصر ID_EDIT_ADD_COIN داخل المجموعة Menu Commands ثم اختر Command من القائمة المنسدلة الناتجة.

١١. انقر مربع السرد والتحرير المجاور ثم اختر OnEditAddCoin <Add>، يظهر قالب الدالة ()OnEditAddCoin داخل نافذة الكود.

١٢. قم بتكرار الخطوتين السابقتين ولكن مع العنصر ID_EDIT_REMOVE_COIN لإضافة هيكل الدالة ()OnEditRemoveCoin إلى التصنيف.

تحديث العرض

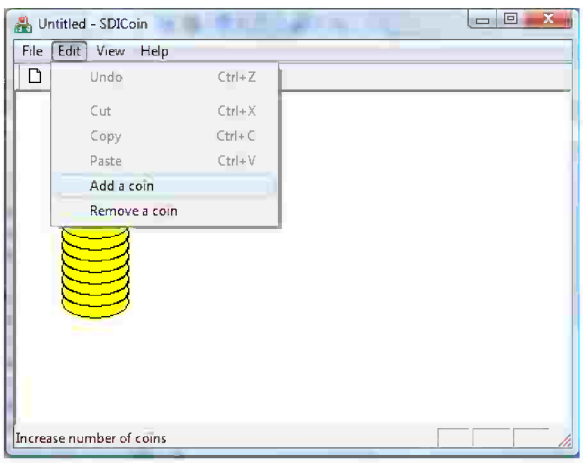
بعد إجراء أى تعديل على بيانات الوثيقة، يجب أن ينعكس هذا التعديل على شكل العرض. يتحقق ذلك باستدعاء الدوال التى تتسبب فى إعادة تركيب العرض. قم بتعديل كود الدالتين ()OnEditAddCoin و()OnEditRemoveCoin وذلك كما يلى:

```
void CSDICoinDoc::OnEditAddCoin()
{
	// TODO: Add your command handler code here
	m_nCoins ++;
	UpdateAllViews(NULL);
}
void CSDICoinDoc::OnEditRemoveCoin()
{
	// TODO: Add your command handler code here
	if(m_nCoins > 0)
		m_nCoins--;
	UpdateAllViews(NULL);
}
```

وكما ترى، تقوم كل دالة بزيادة أو إنقاص المتغير m_nCoins ثم استدعاء الدالة ()UpdateAllViews لتحديث العرض بتمرير القيمة NULL والتى تعنى تحديث كل

العروض المتصلة بالوثيقة، حيث يتم تحديث كل عرض باستدعاء الدالـــة **OnUpdate()** والتى تتسبب بدورها فى استدعاء الدالة **OnDraw()** لإعادة رسم العرض.

والآن قم ببناء التطبيق وتنفيذه بعد إجراء التغييرات الأخيرة. تلاحظ أنك كلما اختـــرت **Add a Coin** أو **Remove a Coin** يتغير شكل العرض بإضافة عملة جديدة أو حذف أخرى على الترتيب (انظر شكل ١٣-١٤).



شكل ١٣-١٤ زيادة عدد العملات باختيار **Add a Coin** من قائمة **Edit**