الباب التاسع

استخدام فصيلة الوثيقة

مفتتح

فى هذا الباب سوف نستخدم إمكانات فصيلة الوثيقة فى حفظ بيانات برنامج الرسم الذى بدأناه فى الباب السابع. وباستخدام هدف الوثيقة سوف نمنح البرنامج الخصائص الآتية:

١. إمكانية إعادة طلاء نافذة البرنامج بمحتوياتها عند الحاجة.

٢. إمكانية مسح الرسم الموجود بالنافذة.

٣. إمكانية الرجوع فى آخر خطوة من خطوات الرسم.

هذا علاوة على مجموعة كبيرة من فصائل ودوال المؤسسة MFC التى سوف نتعرف بها فى هذا الباب.

تذكر:

١. للاطلاع على الصيغة التفصيلية لإحدى الدوال أو الفصائل ، ضع مؤشر الفأر فوقها ثم اضغط زر اللوحة F1.

٢. لعرض الخريطة الكاملة لشجرة فصائل المؤسسة MFC ابحث فى شاشة النجدة (باستخدام البطاقة Search) عن "Hierarchy Chart".

٣. للاطلاع على الكود الكامل لأحد المشروعات ، افتح المشروع من القرص المصاحب للكتاب.

(٩-١) برنامج رسم ــ الطراز ٢

فى برنامج الرسم الذى أعددناه من قبل ، كان فى إمكاننا أن نرسم على صفحة النافذة بعض الخطوط ، لكن هذه الصفحة غير قابلة لإعادة الطلاء. بمعنى أنك لو أتلفت المحتويات بتصغير النافذة مثلا (أى بتحويلها إلى أيقونة على المهام) ، فإنك عندما تعرض النافذة مرة أخرى ، سوف تلاحظ أنها تفقد محتوياتها. ويحدث نفس الشىء لو أن نافذة أخرى حجبت محتويات النافذة كلها أو بعضها. والطريقة التى نحفظ بها محتويات النافذة هى تخزين معلوماتها فى فصيلة الوثيقة بحيث يمكن طلاء النافذة بمحتوياتها عندما يتطلب الأمر. وفى هذا الباب سوف نستخدم برنامج الرسم الذى أنشأناه فى الباب السابع ، ونقوم بتطويره لإضافة الملامح التالية:

١.خاصية إعادة الطلاء.

٢.أوامر قائمة التحرير الآتية:

- Undo: للرجوع فى آخر خطوة من خطوات الرسم.
- Clear All: لمسح محتويات النافذة.

خطة عمل

لن نبدأ مشروعا جديدا فى هذا الباب وإنما سنقوم بتطوير البرنامج MyPaintBrush (الذى أعددناه فى الباب السابع). ولذلك انسخ دوسيه هذا المشروع إلى دوسيه جديد (مع منح الدوسيه اسما جديدا بالطبع ــ أما المشروع نفسه فلا حاجة بنا إلى تغيير اسمه) تمهيدا لتطويره

فى هذا الباب. كما يمكنك نسخ المشروع من القرص المصاحب للكتاب (الدوسيه Ch07).

(٩-٢) استخدام فصيلة الوثيقة

افتح المشروع MyPaintBrush.dsw إما باستخدام أمر قائمة الملفات: Open WorkSpace أو بضغطة مزدوجة على اسم الملف فى نافذة الكشاف. وفيما يلى سوف نعرض عملية تطوير البرنامج فى صورة خطوات تبدأ من الخطوة ١ ، وتنتهى بالخطوة ١٠ فى نهاية الباب.

الخطوة ١: إضافة فصيلة لحفظ الإحداثيات

١. اعرض مشهد الفصائل (Class View).

٢. افتح ملف العناوين لفصيلة الوثيقة (MyPaintBrushDoc.h) بضغطة مزدوجة على اسم الفصيلة.

٣. أضف الفصيلة الجديدة MyPaintClass إلى ملف العناوين CMyPaintBrushDoc.h لاستخدامها فى حفظ إحداثيات الخط. وفى شريحة البرنامج الموضحة بعد ، قد ميزنا السطور المطلوب إضافتها بالبنط الثقيل كالعادة. (لاحظ أن الفصيلة يتم إضافتها قبل فصيلة الوثيقة CMyPaintBrushDoc مباشرة).

الكود الجديد

```
class CMyPaintClass: public CObject
{
protected:
        int m_x1, m_x2, m_y1, m_y2;
```

```
public:
        CMyPaintClass(int x1, int y1, int x2, int y2)
{
                m_x1 = x1;
                m_x2 = x2;
                m_y1 = y1;
                m_y2 = y2;
}
        void DrawIt(CDC *PDC);
};                                                  نهاية الكود الجديد
class CMyPaintBrushDoc : public CDocument          بداية فصيلة الوثيقة
```

وتحتوى الفصيلة CMyPaintClass على الأعضاء الآتية:

١. إحداثيات بداية الخط m_x1, m_y1

٢. إحداثيات نهاية الخط m_x2, m_y2

٣. دالة بناء خطية CMyPaintClass() ذات بارامترات

٤. الدالة DrawIt() التى سوف نستخدمها فى رسم الخط

والفصيلة الجديدة ترث الفصيلة **CObject** (الفصيلة الأم لفصائل المؤسسة MFC).

الخطوة ٢: استخدام نموذج الفصيلة CTypedPtrArray

لتخزين الخطوط المرسومة فى مصفوفة فإننا سوف نضيف عضوا جديدا إلى فصيلة الوثيقة وهو عبارة عن مصفوفة مبنية على نموذج الفصيلة **CTypedPtrArray** (راجع نماذج الفصائل بالباب الثالث). لتحقيق ذلك أضف الكود الموضح بالبنط الثقيل فى شريحة البرنامج التالية ، إلى إعلان فصيلة الوثيقة CMyPaintBrushDoc بملف العناوين CMyPaintBrushDoc.h.

```
class CMyPaintBrushDoc : public CDocument
{
protected:                                                    الكود الجديد
  CTypedPtrArray<CObArray, CMyPaintClass*> m_MyPaintArray;
public:
  void PutLine(int x1, int y1, int x2, int y2);
  CMyPaintClass *GetLine(int i);
  int GetLines();
```

ونموذج الفصيلة **CTypedPtrArray** يستخدم فى توليد عائلة من الفصائل ذات الأغراض المختلفة. ويحدد البارامتر الأول (**CObArray** فى هذا المثال) نوع فصيلة الأساس التى تشتق منها الفصائل ، أما البارامتر الثانى CMyPaintClass فيحدد نمط البيانات المطلوب تخزينها ، وهو نمط فصيلة الخطوط التى أضفناها إلى ملف العناوين فى الفقرة السابقة. أما الفصيلة **CObArray** فهى تنتمى إلى نوعية من الفصائل تسمى بفصائل المصفوفات (أو فصائل التجميع Collection Classes). ويختزن هدف الفصيلة **CObArray** مجموعة من المؤشرات تشير إلى الفصيلة الأم **CObject** ، أو إلى أى فصيلة مشتقة منها ، فى صورة مصفوفة. أى أن العضو m_MyPaintArray عبارة عن هدف من فصيلة مشتقة من **CObArray** يمكنه اختزان مجموعة من المؤشرات إلى أهداف الفصيلة CMyPaintClass (وهى الخطوط المرسومة فى النافذة!). كما أضفنا أيضا ثلاثة من الدوال الأعضاء للتعامل مع الخطوط وهى:

- PutLine()
- GetLine()
- GetLines()

وسوف يلى تعريف هذه الدوال فى الفقرة التالية.

ملاحظة:

تستخدم الفصيلة **CTypedPtrArray** كغلاف (Wrapper) لكل من الفصائل **CPtrArray** و **CObArray** ، بمعنى أنه يمكن استخدامها بدلا من استخدام هذه الفصائل مباشرة لأنها تحتوى على تسهيلات داخلية ولا سيما فى عملية اختبار الأخطاء وتحويل أنماط المتغيرات (Casting).

ملاحظة:

وعند استخدام نماذج فصائل المؤسسة MFC ، فإنه يلزم تضمين ملف العناوين afxtempl.h بداخل الملف StdAfx.h (أحد ملفات العناوين التى يضيفها الساحر إلى المشروع). وفيما يلى جزء من ملف العناوين وقد أضفنا إليه الملف afxtempl.h بالبنط الثقيل.

```
#define VC_EXTRALEAN        // Exclude rarely-used stuff from ...
#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>       // MFC support for Internet Explorer 4 Common
Controls
#include <afxtempl.h>       // أضف هذا السطر
#ifndef _AFX_NO_AFXCMN_SUPPORT
```

ملاحظة:

توجد عبارة تضمين ملف العناوين StdAfx.h فى بداية جميع ملفات المصدر التى ينشئها الساحر ، وينتمى هذا الملف إلى فئة الملفات سابقة الترجمة (precompiled) ، بمعنى أنها لا تترجم مع ملفات المشروع كل مرة ، فهى عبارة عن بلوكات جاهزة.

ملاحظة:

الخطوة ٣: تطبيق الفصيلة

فى هذه الفقرة نضيف تعريف الدوال الأعضاء بالفصيلة الجديدة CMyPaintClass. ولنبدأ بإضافة تعريف الدالة DrawIt() بإضافتها إلى نهاية الملف المصدر للوثيقة MyPaintBrushDoc.cpp. والدالة DrawIt() تستخدم مؤشرا إلى فصيلة منطقة العرض **CDC** وتحتوى على الدالتين **CDC::MoveTo()** و **CDC::LineTo()** التى تستخدم فى الرسم. وفى شريحة الكود التالية قد وضحنا الكود الجديد بالبنط الثقيل.

```
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMyPaintBrushDoc commands

void CMyPaintClass::DrawIt(CDC *PDC)          أضف هذا الكود
{
        PDC -> MoveTo(m_x1, m_y1);
        PDC -> LineTo(m_x2, m_y2);
}
```

وكما نرى فى شريحة الكود السابقة ، أن الدالة **CDC::MoveTo()** تؤدى إلى نقل الإحداثيات إلى الموقع (m_x1, m_y1). أما الدالة LineTo() فهى ترسم الخط من هذا الإحداثى إلى الموقع الجديد (m_x2, m_y2).

وفى الفقرة التالية فإننا سوف نقوم بتعريف ثلاثة من الدوال الأعضاء بداخل نفس الملف (MyPaintBrushDoc.cpp). أكتب

التعريفات الآتية فى نهاية الملف بعد تعريف الدالة DrawIt() مباشرة. وجميع السطور فى الأشكال الثلاثة التالية سطور جديدة يلزم كتابتها وهى تظهر بالبنط الثقيل كالمعتاد.

الدالة PutLine()

```
void CMyPaintBrushDoc::PutLine(int x1, int y1, int x2, int y2)
{
  CMyPaintClass *ptrToLine = new CMyPaintClass(x1, y1, x2, y2);
  m_MyPaintArray.Add(ptrToLine);
}
```

أما الدالة PutLine() فكما هو واضح من اسمها تستخدم فى إضافة خط جديد إلى مجموعة الخطوط. والمنطق المستخدم فى هذه الدالة كالآتى:

١. يتم خلق وشحن هدف جديد من الفصيلة CMyPaintBrush وتخصصيه إلى المؤشر ptrToLine:

```
CMyPaintClass *ptrToLine = new CMyPaintClass(x1, y1, x2, y2);
```

٢. يلى ذلك استخدام الدالة **Add()** لإضافة الهدف الجديد إلى نهاية المصفوفة:

```
m_MyPaintArray.Add(ptrToLine);
```

وكما نرى ، أننا استخدمنا هنا الهدف m_MyPaintArray لاستدعاء أحد دوال الفصيلة **CObArray**. أما العنصر المطلوب إضافته فهو المؤشر إلى الخط ptrToLine الذى جاء كبارامتر للدالة **Add()**.

ملاحظة:

تحتوى الفصيلة **CObArray** على مجموعة من الدوال الأعضاء التى سوف نستخدمها فى الفقرات التالية و هذه الدوال هى:

- **()CObArray**: دالة البناء ، وهى تستخدم فى بناء مصفوفة خالية لاختزان مؤشرات إلى الفصيلة **CObject**
- **()Add**: إضافة عنصر إلى آخر المصفوفة
- **()GetAt**: ترجع قيمة العنصر المناظر لدليل معين (index)
- **()ElementAt**: ترجع مؤشرا إلى العنصر الموجود عند الدليل المعين بداخل المصفوفة
- **()RemoveAt**: حذف العنصر المناظر لدليل معين
- **()GetSize**: ترجع عدد عناصر المصفوفة
- **()GetUpperBound**: ترجع أقصى قيمة لدليل المصفوفة
- **()RemoveAll**: لحذف جميع عناصر المصفوفة

كما أن فصيلة النموذج **CTypedPtrArray** ــ وهى ترث الفصيلة **CObArray** ــ تحتوى على بعض الدوال التى تركب الدوال الافتراضية الآتية:

- **()Add**
- **()GetAt**
- **()ElementAt**
- **()RemoveAt**

ولذلك فإن أى استدعاء لهذه الدوال خلال البرنامج سوف يؤدى إلى استخدام الدالة الموجودة بالفصيلة الوارثة.

ملاحظة:

الدالة GetLine()

```
CMyPaintClass *CMyPaintBrushDoc::GetLine(int i)
{
  if (i < 0 | | i > m_MyPaintArray.GetUpperBound() )
        return 0;
  return m_MyPaintArray.GetAt(i);
}
```

إن العناصر المستخدمة فـــى هـــدف المصفوفـــة m_MyPaintArray عناصر ذات أدلة ، تبدأ من الدليل 0 . أما أكبر قيمة للدليل فيمكن الحصول عليها باستخدام الدالـــــة **CObArray::GetUpperBound()**. وتقوم الدالة GetLine() بإرجاع قيمة المؤشر المختزن عند الدليـــل المعين الممر إليها كبارامتر (البارامتر i). وتبدأ الدالـــــة عملـــها باختبار قيمة البارامتر i للتأكد من أنه يقع فى الحدود المقبولـــــة للدليل ، فإذا كانت قيمته أقل من الصفر أو أكبر من الحد الأعلى للدليل ، ترجع الدالة القيمة صفر:

```
if (i < 0 || i > m_MyPaintArray.GetUpperBound() )
        return 0;
```

وإلاّ فإنها ترجع قيمة المؤشر باستخدام الدالة **GetAt()**:

```
return m_MyPaintArray.GetAt(i);
```

الدالة GetLines()

```
int CMyPaintBrushDoc::GetLines()
{
  return m_MyPaintArray.GetSize();
}
```

ترجع هذه الدالة عدد المؤشرات المختزنة في المصفوفة m_MyPaintArray. وهي تستخدم الدالة **CObArray::GetSize()** للحصول على هذه القيمة.

وفى الفقرات القادمة سوف تلاحظ أن هذه الدوال الثلاث سوف تستدعى من داخل فصيلة المشهد.

الخطوة ٤: تحديث البيانات بفصيلة الوثيقة

عندما تطلق الزر الأيسر للفأر ، فإن الرسم يستقر على الشاشة. هذا ما تمت برمجته من قبل فى فصيلة المشهد. بقى أن نقوم بتحديث فصيلة الوثيقة لحفظ بيانات الخط المرسوم. يتم ذلك باستدعاء الدالة **CView::GetDocument()** وتسجيل الخط المحدّث باستخدام الدالة PutLine(). وهذا هو الكود المطلوب إضافته إلى الدالة **OnLButtonUp()** بالملف MyPaintBrushView.cpp (والأجزاء الجديدة مكتوبة بالبنط الثقيل كالعادة):

```
void CMyPaintBrushView::OnLButtonUp(UINT nFlags, CPoint point)
{
// TODO: Add your message handler code here and/or call default
// Add these lines:
  if (m_Moving)
  {
        m_Moving = 0;
        ::ReleaseCapture();
        ::ClipCursor(NULL);
        CClientDC ClientDC(this);
        ClientDC.SetROP2(R2_NOT);
```

```
        ClientDC.MoveTo(m_NewPoint);
        ClientDC.LineTo(m_PrevPoint);
        ClientDC.SetROP2(R2_COPYPEN);
        ClientDC.MoveTo(m_NewPoint);
        ClientDC.LineTo(point);
                                                أضف هذا الجزء
        CMyPaintBrushDoc * pDoc = GetDocument();
        pDoc ->PutLine(m_NewPoint.x, m_NewPoint.y,
                              point.x, point.y);
                                                نهاية الكود الجديد
    }
// End of code. The rest is added by MFC
        CView::OnLButtonUp(nFlags, point);
}
```

وترجع الدالة **GetDocument()** مؤشرا إلى هدف الوثيقة المرتبط بالمشهد ، وتستخدم فى استدعاء الدوال الأعضاء بفصيلة الوثيقة ، ولذلك فقد استخدمنا المؤشر pDoc فى استدعاء الدالة PutLine() من فصيلة الوثيقة.

الخطوة ٥: إعادة طلاء النافذة

عندما تتلف محتويات النافذة فإن النوافذ تمسح محتوياتها ، وتستدعى الدالة **CView::OnDraw()** لإعادة رسم المحتويات. ولذلك فإنه بعد تسجيل بيانات الرسم فى فصيلة الوثيقة ، علينا أن نضيف الكود اللازم لإعادة الطلاء إلى الدالة **CView::OnDraw()** بفصيلة المشهد. وهذه هى السطور الجديدة المطلوب إضافتها إلى الدالة بالملف MyPaintBrushView.cpp (موضحة بالبنط الثقيل):

```
// CMyPaintBrushView drawing
```

```
void CMyPaintBrushView::OnDraw(CDC* pDC)
{
        CMyPaintBrushDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        // TODO: add draw code for native data here

        int i = pDoc -> GetLines();                    أضف هذا الكود
        while (i--)
        pDoc ->GetLine(i) -> Draw(pDC);
}
```

وكما نرى فى هذا الكود ، أن الدالة GetLines() تستدعى من فصيلة الوثيقة باستخدام المؤشر pDoc ، ويتم تخزين القيمة المرتجعة منها فى المتغير i. ويمثل المتغير i عدد الخطوط المرسومة فى النافذة والتى سبق اختزانها فى هدف الوثيقة. ولرسم الخط فإن هناك مرحلتين: الأولى هى الحصول على مؤشر إلى هذا الخط باستخدام الدالة GetLine() ، والثانية هى رسم الخط باستخدام الدالة DrawIt(). ولأن الدالة GetLine() ترجع مؤشرا إلى الفصيلة الجديدة CMyPaintClass ، فإن هذا المؤشر يستخدم لاستدعاء الدالة DrawIt() العضوة بهذه الفصيلة.

(٩-٣) إضافة وتوظيف أوامر القائمة

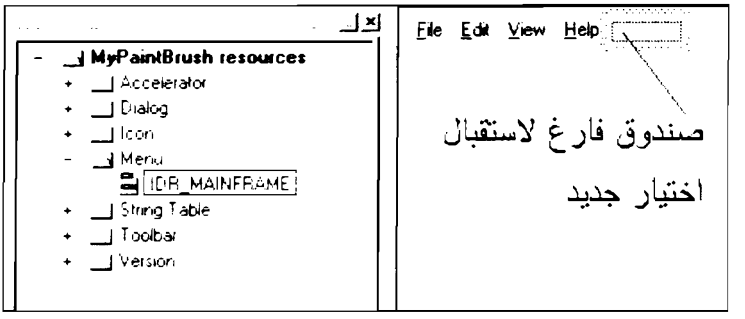
فى الفقرات التالية سوف نعد أمرين من أوامر القائمة: الأول هو الأمر Undo المستخدم فى إلغاء الخطوة الأخيرة (آخر خط تم رسمه) والثانى هو Clear All المستخدم فى تنظيف النافذة تماما. ومن

البديهى أن هذه الأوامر تستمد معلوماتها من فصيلة الوثيقة. ولكى تبدأ العمل فى إضافة أو توظيف أوامر القائمة اتبع الآتى:

١.من القسم الأيسر من نافذة الأستوديو ، اختر بطاقة الموارد (Resource View).

٢.اضغط ضغطة مزدوجة على دوسيه القائمة (Menu).

٣.اضغط ضغطة مزدوجة على الملف IDR_MAINFRAME فتشاهد القائمة الموضحة بالشكل التالى. هذا هو محرر القائمة (Menu Editor).



شكل (٩-١) القائمة فى مشهد الموارد

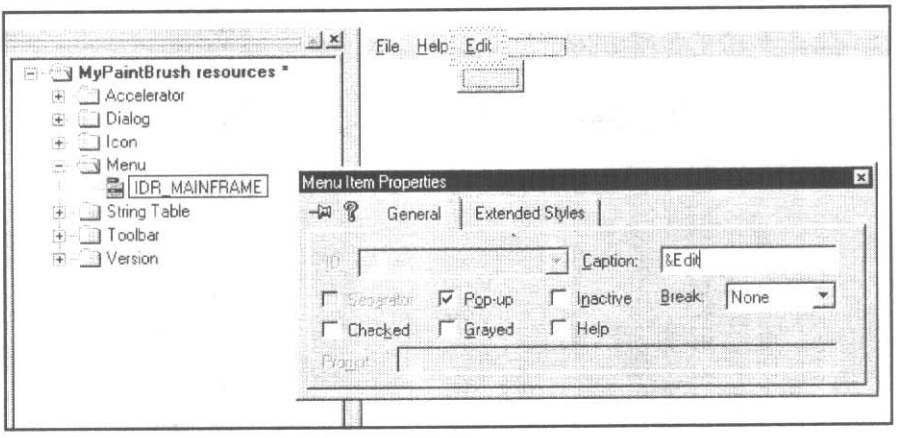
٤.امسح اختيارات القائمة التى لن نستخدمها فى هذا البرنامج وهى View و Edit (سوف نعيد إنشاء الاختيار Edit من البداية). ويتم المسح باختيار العنصر ثم الضغط على زر اللوحة Delete.

الخطوة ٦: أمر إلغاء الخطوة السابقة Undo

١.اضغط ضغطة مزدوجة على الصندوق الفارغ الموجود فى أقصى اليمين بسطر القائمة (أنظر الشكل السابق) ، فيظهر

صندوق حوار الخصائص الذى يحمـــل العنـــوان Menu Item Properties.

٢.اكتب الكلمة &Edit فى الصندوق Caption ، فيظهر اختيار القائمة Edit بسطر القائمة ، أنظر الشكل التالى.

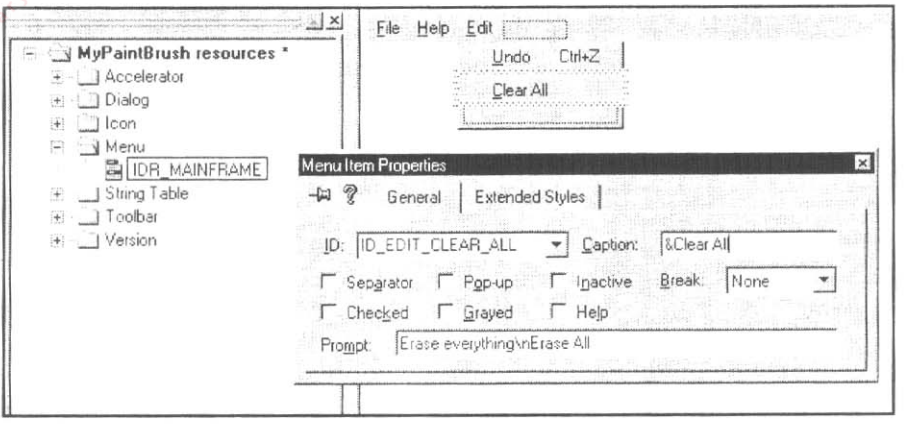


شكل (٩-٢) إنشاء الاختيار Edit بالقائمة

٣.اضغط بالفأر على اختيار القائمة Edit فيتدلى منه صندوق فـارغ ، ثم اضغط ضغطة مزدوجة على الصندوق الفـــارغ فيظـــهر صندوق حوار الخصائص من جديد.

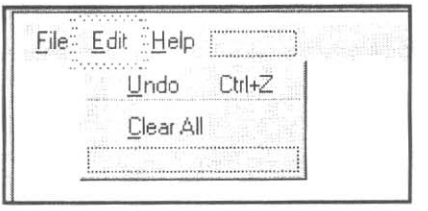
٤.من الصندوق ID ، اختر ثابت التعارف ID_EDIT_UNDO.

٥.فى الصندوق Caption ، اكتب العبـــارة &Undo\tCtrl+Z. أنظـــر الشكل التالى. وبـــالضغط علـــى أى نقطـــة خـــارج صنـــدوق الخصائص فإنه يختفى.



شكل (٩-٤) إضافة الاختيار Clear All بصندوق الخصائص

٥.اسحب الاختيار Edit بالفأر وضعه ما بين الاختيارين Help و File بحيث يكون شكل القائمة النهائى كما هو موضح بالشكل التالى.



شكل (٩-٥) اختيارات القائمة

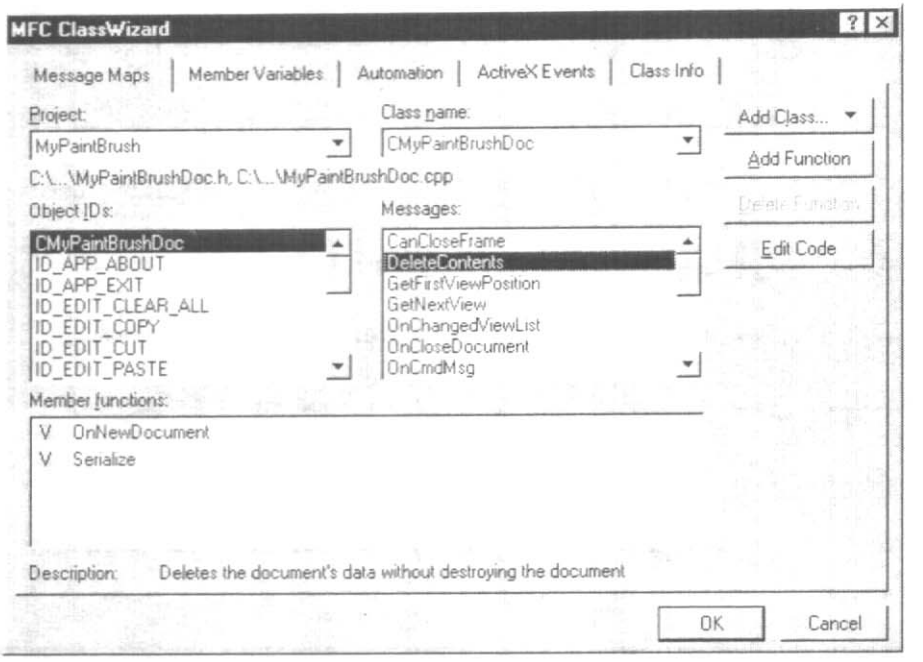
الخطوة ٨: مسح بيانات الوثيقة

بصورة سابقة التعريف ، فإن أمر القائمة File-New يؤدى إلى تنظيف أية نافذة من محتوياتها حيث تستدعى الدالة الافتراضية **CDocument::DeleteContents()**.

فى هذه الخطوة سوف نربط اختيار القائمة Clear All بالدالة **DeleteContents()** ، ولكننا أيضا سوف نركب هذه الدالة

الافتراضية بكتابة الكود اللازم بهذه الدالة لمسح محتويات الوثيقة والمشهد.

١. اختر أمر القائمة View - Class Wizard. فتظهر نافذة ساحر الفصائل (MFC Class Wizard) الموضحة بالشكل التالى.



شكل (٩-٦) نافذة ساحر الفصائل (Class Wizard)

٢. اختر البطاقة Message Maps إذا لم تكن هى البطاقة المختارة حاليا.

٣. من صندوق القائمة Class name ، اختر اسم الفصيلة CMyPaintBrushDoc.

٤. من الصندوق Object IDs ، اختر العنصر CMyPaintBrushDoc.

إن الكود الذى أضفناه هنا يؤدى العمليات الآتية:

- استدعاء الدالة **CObArray::GetSize()** للحصول على عدد الخطوط المرسومة (المؤشرات) والمختزنة بالهدف m_MyPaintArray. يختزن هذا العدد فى متغير الدليل i:

```
int i = m_MyPaintArray.GetSize();
```

- من خلال حلقة تكرارية تنازلية ، تبدأ من أعلى قيمة للدليل i ، يتم الحصول على قيمة المؤشر بالدالة **GetAt()** ، ثم إتاحة الذاكرة المحجوزة للمؤشر باستخدام المؤثر **delete**. (تذكر أن استخدام المؤثر **delete** هو المقابل لاستخدام المؤثر **new**):

```
while (i--)
        delete m_MyPaintArray.GetAt(i);
```

- يلى ذلك مسح جميع محتويات هدف الوثيقة باستخدام الدالة **CObArray::RemoveAll()** كالآتى:

```
m_MyPaintArray.RemoveAll();
```

بهذه الخطوة فإن استدعاء الدالة **DeleteContents()** يؤدى إلى مسح بيانات الوثيقة علاوة على محتويات النافذة ، بحيث أنك لو أعدت طلاء النافذة ــ فى هذه الحالة ــ فإن هدف الوثيقة سوف يكون فارغا من البيانات التى تغذى عملية إعادة الطلاء.

الخطوة ٩: توظيف أمر القائمة Clear All

حتى الآن فإن أوامر القائمة غير عاملة فهى لم يتم ربطها بأى كـود فى البرنامج ، وهذا هو موضوع هذه الفقرة. وسوف نبـــدأ بـــأمر القائمة Edit – Clear All.

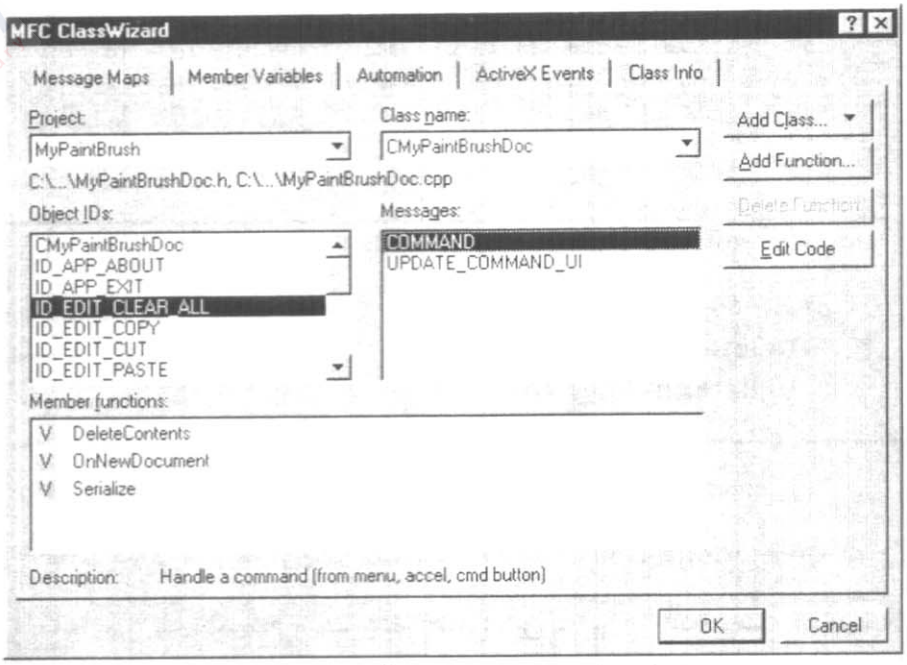
١. افتح صندوق حوار ساحر الفصائل باستخدام أمر القائمة View – Class Wizard.

٢. من الصندوق Class name ، اختـــر CMyPaintBrushDoc. وهـــذا يعنى أننا سوف نربط الأمر Clear All بفصيلة الوثيقة.

٣. من الصندوق Object IDs ، اختر الثابت ID_EDIT_CLEAR_ALL ، فيظهر بصندوق الرسائل (Messages) أسماء الرسائل التى يمكن ربطها بهذا الثابت وهى:

- **COMMAND**: الرسالة التى ترسلها النوافذ عنـــد اســـتخدام اختيـــار القائمة.
- **UPDATE_COMMAND_UI**: الرسالة التى ترسلها النوافذ عنـــد فتـــح نافذة قائمة التحرير (Edit).

أنظر الشكل التالى.



شكل (٩-٩) الرسائل المناظرة لأمر القائمة Clear All

مقبض الرسالة COMMAND

١. اختر الرسالة **COMMAND**.

٢. اضغط الزر Add Function لإضافة دالة مقبض الرسالة إلى فصيلة الوثيقة ، فتظهر فى القسم السفلى من النافذة الدالة **OnEditClearAll()**. اضغط الزر OK فيتم خلق الدالة وإضافتها إلى الوثيقة. ومع ذلك فإنها حتى الآن دالة فارغة من المحتويات.

٣. ضع مؤشر الفأر على اسم الدالة **OnEditClearAll()** ثم اضغط الزر Edit Code ، فتنتقل إلى جسم الدالة بالملف

MyPaintBrushDoc.cpp. أدخل الكود الموضح بالشكل التالى والمميز بالبنط الثقيل.

```
void CMyPaintBrushDoc::OnEditClearAll()
{
// TODO: Add your command handler code here
        DeleteContents();                    أضف هذا الكود
        UpdateAllViews(0);
}
```

إن العبارة الأولى تؤدى إلى استدعاء الدالة **DeleteContents()** التى ركبناها من قبل لمسح محتويات هدف الوثيقة. أما الدالة الثانية **CDocument:: UpdateAllViews()** فهى تؤدى إلى تحديث المشهد بالأوضاع الجديدة.

مقبض الرسالة UPDATE_COMMAND_UI

بقى أن ننشئ دالة مقبض لمعالجة الرسالة **UPDATE_COMMAND_UI** ، وهى تستقبل عند فتح نافذة القائمة المحتوية على الأمر Clear All (أى قائمة التحرير). والغرض من معالجة هذه الرسالة هو تمكين أو عدم تمكين الأمر Clear All ، بحسب ظروف البرنامج. فلو كانت الصفحة خالية تماما من الرسم ، فإننا لا نحتاج هذا الأمر ومن المناسب تبطيله. وفى الحالة الأخيرة يظهر الأمر فى القائمة معتما. وهذه هى خطوات العمل:

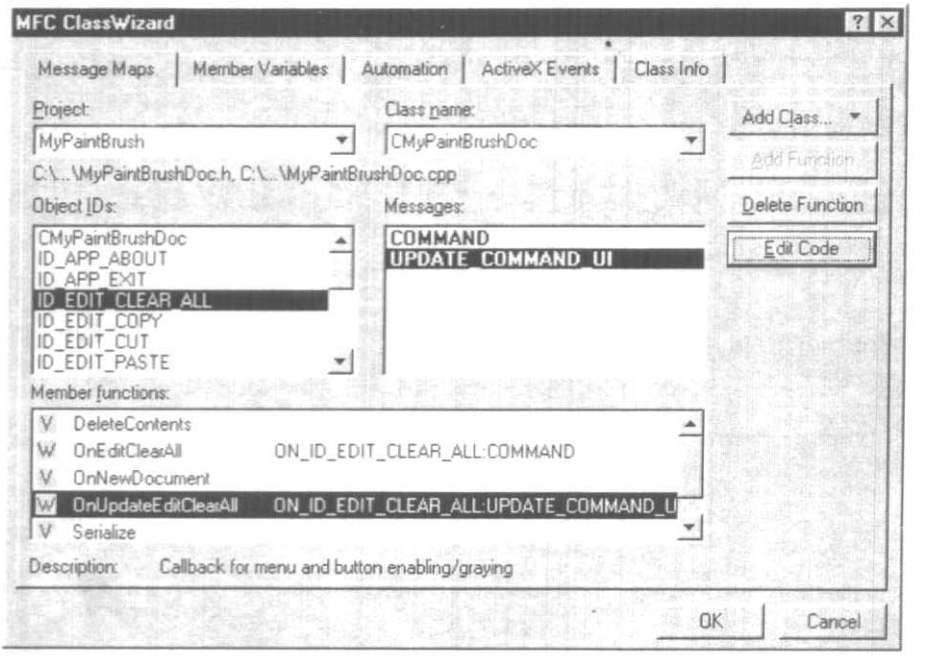
١. افتح صندوق حوار ساحر الفصائل ، باستخدام أمر القائمة:
View – Class Wizard.

٢. من الصندوق Class name ، اختر CMyPaintBrushDoc.

٣. من الصندوق Object IDs ، اختر الثابت ID_EDIT_CLEAR_ALL.

٤. من الصندوق Messages ، اختر الرسالة **UPDATE_COMMAND_UI**.

٥. اضغط الزر Add Function لإضافة دالة مقبض الرسالة إلى فصيلة الوثيقة ، فتظهر فى القسم السفلى من النافذة الدالة **OnUpdateEditClearAll()**. اضغط الزر OK. أنظر الشكل التالى.



شكل (٩-١٠) إضافة مقبض الرسالة UPDATE_COMMAND_UI

٦.ضع مؤشر الفأر على اسم الدالة **OnUpdateEditClearAll()** ثم اضغط الزر Edit Code ، فتنتقل إلى جسم الدالة بالملف MyPaintBrushDoc.cpp. أدخل الكود الموضح بالشكل التالى والمميز بالبنط الثقيل.

```
void CMyPaintBrushDoc::OnUpdateEditClearAll(CCmdUI* pCmdUI)
{
// TODO: Add your command update UI handler code here
```

اكتب هذا الكود:

```
    pCmdUI->Enable(m_MyPaintArray.GetSize());
}
```

إن العبارة التى أضفناها تستخدم مؤشرا إلى الفصيلة **CCmdUI** وهى الفصيلة المسئولة عن تشغيل أوامر القوائم (علاوة على وظائف أخرى تتعلق بالوصلة البينية للمستخدم بصفة عامة). وباستخدام المؤشر pCCmdUI تم استدعاء الدالة **CCmdUI::Enable()** لتمكين الأمر Clear All إذا تحقق الشرط الوارد بين قوسى الدالة ، وهو أن يكون عدد عناصر المصفوفة أكبر من صفر:

```
Enable(m_MyPaintArray.GetSize())
```

لاحظ أن الدالة ترجع عددا منطقيا (TRUE أو FALSE). فإذا كان محتويات قوس أكبر من الصفر كانت القيمة هى TRUE.

الخطوة ١٠: توظيف أمر القائمة Undo

سوف نتبع أسلوبا مماثلا لما اتبعناه من قبل لإضافة مقبض رسالة يرتبط بالأمر Undo.

مقبض الرسالة COMMAND

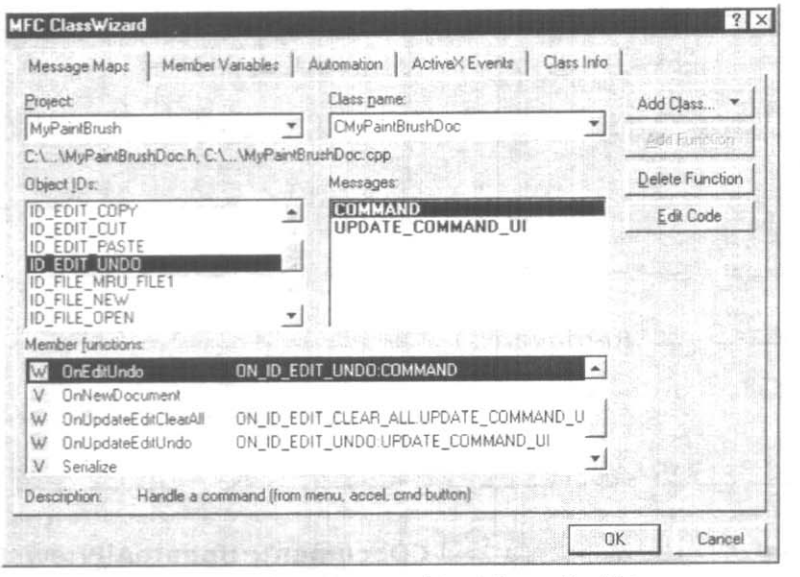
١. افتح صندوق حوار ساحر الفصائل ، باستخدام أمر القائمة: View – Class Wizard.

٢. من الصندوق Class name ، اختر CMyPaintBrushDoc.

٣. من الصندوق Object IDs ، اختر الثابت ID_EDIT_UNDO.

٤. من الصندوق Messages ، اختر الرسالة **COMMAND**.

٥. اضغط الزر Add Function ، فتظهر فى القسم السفلى من النافذة الدالة **OnEditUndo()** ثم اضغط الزر OK. أنظر الشكل التالى.



شكل (٩-١١) إضافة مقبض الرسالة COMMAND

٦.ضع مؤشر الفأر على اسم الدالة OnEditUndo() ثم اضغط الزر Edit Code ، فتنتقل إلى جسم الدالة بالملف MyPaintBrushDoc.cpp. أدخل الكود الموضح بالشكل التالى والمميز بالبنط الثقيل.

```
void CMyPaintBrushDoc::OnEditUndo()
{                                                    أدخل هذا الكود
        int i =  m_MyPaintArray.GetUpperBound();
        if (i > -1) {
                delete m_MyPaintArray.GetAt(i);
                m_MyPaintArray.RemoveAt(i);
        }
        UpdateAllViews(0);
}
```

إن شريحة الكود السابقة تؤدى المهام الآتية:

- باستخدام الدالة **CObArray::GetUpperBound()** يتم معرفة أقصى قيمة لدليل العناصر المخزنة فى المصفوفة ، وتختزن هذه القيمة فى المتغير i.
- إذا كانت قيمة الدليل i أكبر من 1- (بمعنى آخر ، إذا كانت صفرا أو أكثر. لاحظ أن قيمة دليل المصفوفة تبدأ من صفر) يتم مسح آخر عنصر وإتاحة الذاكرة التى كانت مخصصة له.
- تتم عملية إتاحة الذاكرة كالمعتاد باستخدام المؤشر delete ، كما تتم عملية المسح باستخدام الدالة**RemoveAt()** التى تستخدم لمسح العنصر المناظر للدليل i.
- تنتهى العملية باستدعاء الدالة **CDocument::UpdateAllViews()** لتحديث المشهد ، وذلك بمسح

محتويات النافذة وإعادة رسمها وفقا للمعلومات المحدثة بهدف الوثيقة.

ومن البديهى أن استخدام الأمر Undo مرة بعد مرة يؤدى إلى مسح الرسم كله خطا بخط.

مقبض الرسالة UPDATE_COMMAND_UI

كما هو الحال مع الأمر Clear All فإن الأمر Undo أيضا يحتاج إلى تبطيله وإعتامه عندما لا تكون صفحة الرسم فارغة. وهذا هو اختصاص الرسالة **UPDATE_COMMAND_UI**. لإنشاء مقبض هذه الرسالة اتبع الآتى:

١. افتح صندوق حوار ساحر الفصائل باستخدام أمر القائمة: View-Class Wizard.

٢. من الصندوق Class name ، اختر CMyPaintBrushDoc.

٣. من الصندوق Object IDs ، اختر الثابت ID_EDIT_UNDO.

٤. من الصندوق Messages ، اختر الرسالة **UPDATE_COMMAND_UI**.

٥. اضغط الزر Add Function لإضافة دالة مقبض الرسالة إلى فصيلة الوثيقة ، فتظهر فى القسم السفلى من النافذة الدالة **OnUpdateEditUndo()**. اضغط الزر OK.

٦. ضع مؤشر الفأر على اسم الدالة OnEditClearAll() ثم اضغط الزر Edit Code ، فتنتقل إلى جسم الدالة بالملف

MyPaintBrushDoc.cpp. أضف الكود الموضح بعد والمميز بالبنط الثقيل.

```
void CMyPaintBrushDoc::OnUpdateEditUndo(CCmdUI* pCmdUI)
{
// TODO: Add your command update UI handler code here
                                                    أضف هذا الكود
        pCmdUI -> Enable(m_MyPaintArray.GetSize());
}
```

إن هذه الدالة تماثل الدالة التى أضفناها لشحن أمر القائمة Clear All حيث أنها تختبر وجود أية عناصر فى المصفوفة ، ثـــم تمكـــن أو تبطل مفعول اختيار القائمة تبعا لذلك.

تجربة البرنامج

جرب الآن تشغيل البرنامج ولاحظ الآتى:

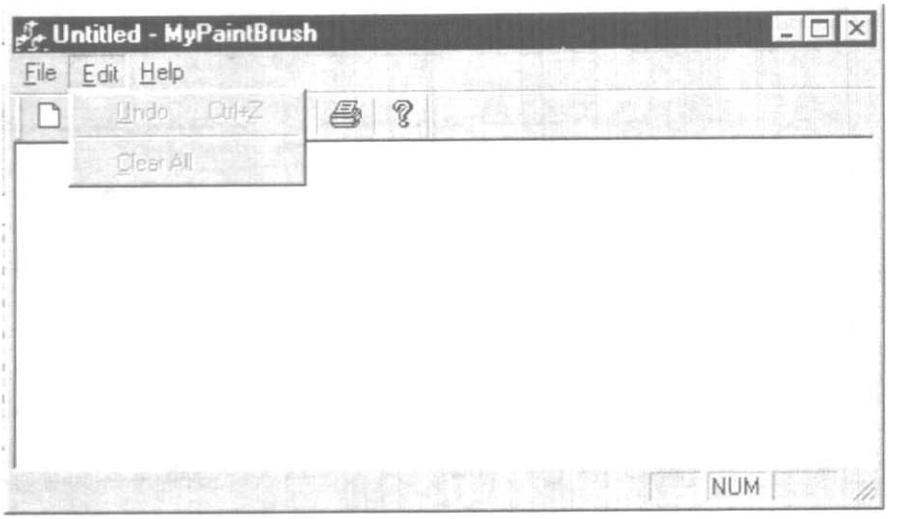
١. يمكنك مسح محتويات النافذة بالأمر Edit – Clear All.

٢. إذا كانت الصفحة خالية من المحتويات فإن الأمر Edit – Clear All يصبح معتما كما بالشكل التالى.

٣. يمكنك مسح الرسم خطا بخط باستخدام أمر القائمة Edit – Undo أو باستخدام الأزرار Ctrl+Z.

٤. إذا كانت الصفحة خالية من المحتويات فإن أمر القائمـــة Edit – Undo يصبح معتما كما بالشكل التالى.

٥.عند تصغير النافذة ، ثم تكبيرها ، تحتفظ الصفحة بمحتوياتـــها (قارن بالبرنامج السابق بالباب السابع).



شكل (٩-١٢) أوامر القائمة في حالة إعتام عندما تكون الصفحة خالية

الموجز

١.فى هذا الباب تعاملنا مع هدف الوثيقة وعرفنا كيفيـــة اختـــزان البيانات به بحيث نسترجعها عند الحاجة إليها.

٢.استخدمنا البيانات الموجودة بالوثيقة فى إعـــادة طـــلاء النـــافذة بالمحتويات عن طريق الدالة **()OnDraw** بفصيلة المشهد. كمـــا استخدمنا الوثيقة أيضا فى تحديث البيانات عند مسح المحتويـلت أو الرجوع فى آخر خطوة.

٣.تعرفنا بأحد الفصائل الهامة بمؤسســة الفصـــائل MFC ، وهـــى الفصيلة **CObArray** التى تستخدم فى تخزيـــن مجموعـــة مـــن المؤشرات إلى الأهداف فى صورة مصفوفة. وهـــذه الأهـــداف

لابد أن تكون تابعة لفصيلة مشتقة من الفصيلة الأم CObject بطريقة مباشرة أو غير مباشرة. كما رأينا أنه باستخدام نموذج الفصيلة **CTypedPtrArray** فإنك تستطيع أن ترث الفصيلة **CObArray** لكى تنشئ منها فصيلة جديدة يمكن تطويعها لتعمل مع نوعية معينة من الأهداف فى برنامجك. وكما رأينا فى البرنامج أننا قد استخدمنا العضو m_MyPaintArray (وهو هدف فصيلة مشتقة من CObArray) لتخزين المؤشرات التى تشير إلى الأهداف CMyPaintClass (الخطوط المرسومة) فى صورة مصفوفة.

٤. رأينا أيضا كيفية معالجة أوامر القائمة من خلال فصيلة الوثيقة. وكيفية استخدام ساحر الفصائل (Class Wizard) فى إنشاء مقابض رسائل القائمة.

٥. كما عرفنا أن هناك نوعين من رسائل القائمة ، هى **COMMAND_UPDATE_UI** و **COMMAND**. تستقبل الرسالة الأولى عند فتح نافذة القائمة ، وتستقبل الثانية عند اختيار أمر القائمة المعين.

٦. عرفنا أيضا الفصيلة CCmdUI التى تحتوى على دوال تشغيل أو تبطيل (إعتام) أوامر القائمة.

٧. عرفنا فى هذا الباب مجموعة جديدة من دوال المؤسسة MFC ، وهى:

نماذج الفصائل:

CTypedPtrArray

الفصائل:

CObArray
CPtrArray
CCmdUI

الدوال:

CObArray::Add()
CObArray::GetAt()
CObArray::RemoveAt()
CObArray::GetSize()
CObArray::GetUpperBound()
CObArray::RemoveAll()
CView::OnDraw()
CDocument::DeleteContents()
CDocument::UpdateAllViews()
CCmdUI::Enable()

الرسائل:

COMMAND_UPDATE_UI
COMMAND