

الباب السادس

تحليل برنامج MFC

مفتتم

فى هذا الباب سوف نتجول فى عمق البرنامج "MFCHello" الذى أنشأناه فى الباب السابق باستخدام ساحر MFC ، بهدف تتبع تسلسل الأحداث ، وفهم الدور الذى تلعبه كل فصيلة وكل دالة فى الكود الذى كتبته لنا الساحر. والكود الذى يكتبه الساحر ليس سطوراً متتابعة من عبارات اللغة ، ولكنه بناء متكامل من الفصائل التى تتصل ببعضها البعض من خلال علاقات سابقة الإعداد. وبفهم هذه الفصائل والعلاقات بينها وبين بعضها ، يمكننا أن نضيف إلى البناء ونوجهه إلى التطبيق المقصود.



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.

(٦-١) خصائص برامج MFC

قبل الدخول فى التفصيلات ، هناك ملاحظة هامة يجب الالتفات إليها عند قراءة برامج النوافذ عامة ، أو برامج MFC على وجه الخصوص. إن البرامج التى يكتبها الساحر فيها نوعين من الكود:

١. كود يختص بالعمليات التنظيمية التى تتكرر مع كل البرامج ، وتهدف إلى ربط برنامجك بنظام التشغيل وإكسابه الملامح العامة للنوافذ. وهذه الأجزاء تكتب غالباً باستخدام الماكرو (Macro) المميز بالحروف الكبيرة مثل `DECARE_MESSAGE_MAP`. وفى أغلب الأحوال فإن هذا الكود لا يجوز تعديله (كما توصى التعليمات بداخل بلوكات البرنامج).

٢. كود موجه إلى خدمة التطبيق نفسه وهو يتكون من الدوال التى ترى بداخلها التعليق "`// TODO`" غالباً ، حيث تستطيع أن تضيف إلى هذه الدوال الكود اللازم لخدمة التطبيق المعين.

ولهذا السبب ، فلا تتوقع أن نشرح البرنامج عبارة بعبارة كما تعودنا مع لغات دوس أو برامج الكونصول. بل سنهتم فقط بالأجزاء التى تخدم التطبيق أو تؤثر عليه بصورة مباشرة. أما الأجزاء التنظيمية فسوف نتخطاها ، أو نعلق عليها فى حينه ، عندما يستلزم الأمر أن نفهم شيئاً عن خلفية الأحداث الدائرة.

(٦-٢) مجرى الأحداث في البرنامج

أول ما نلاحظه في برنامج MFC أنه لا يحتوى على الدالة الرئيسية (**WinMain()**) الممثلة لقلب البرنامج النوافذ. وهذه الدالة موجودة ، وتعمل في الخلفية ، ولكنها محجوبة عن المبرمج. ونتوقع بذلك أن يكون تتابع الأحداث مختلفاً عن البرامج المكتوبة باستخدام دوال API مباشرة. وهذا هو السيناريو العام للأحداث في برامج MFC:

١. يتم استدعاء دالة بناء فصيلة التطبيق **CWinApp**.
٢. تأخذ الدالة (**WinMain()**) (التابعة للمؤسسة MFC) دفة التحكم.
٣. يتم خلق و شحن هدف التطبيق باستخدام الدالة (**InitInstance()**).
٤. تبدأ الدالة (**WinMain()**) الحلقة التكرارية لاستلام ومعالجة وتوزيع الرسائل.
٥. عندما يغلق المستخدم نافذة التطبيق ، ينتهى عمل الدالة (**WinMain()**) (يصطاح على ذلك بعودة الدالة) وينتهى البرنامج. وفيما يلي نعرض تفاصيل هذا السيناريو من خلال دوال البرنامج **MFCHelloApp** ، بادئين بفصيلة التطبيق **CMFHelloApp**.

The Application Object

(٦-٣) هدف التطبيق

يعتبر هدف التطبيق ممثلاً للتطبيق ككل ، وفصيلة التطبيق لماننا المطروح **CMFHelloApp** مشتقة من الفصيلة **CWinApp**. وهذا هو ملف الإعلان الذى يمنحك فكرة عامة عن أعضاء الفصيلة:

```
class CMFHelloApp : public CWinApp
{
```

```
public:
```

```
CMFCHelloApp();
```

دالة البناء

```
// Overrides
```

```
// ClassWizard generated virtual function overrides
```

```
//{{AFX_VIRTUAL(CMFCHelloApp)
```

```
public:
```

```
virtual BOOL InitInstance(); دالة عضوة
```

```
//}}AFX_VIRTUAL
```

```
// Implementation
```

```
//{{AFX_MSG(CMFCHelloApp)
```

```
afx_msg void OnAppAbout(); دالة عضوة
```

```
//}}AFX_MSG
```

```
DECLARE_MESSAGE_MAP()
```

```
};
```

دالة بناء فصيلة التطبيق CWinApp()

يحتوى الملف MFCHello.cpp على الدوال الثلاثة المستخدمة فى بناء وشحن التطبيق:

- CMFCHelloApp()
- InitInstance()
- OnAppAbout()

وتقوم الدالة CMFCHelloApp() ببناء هدف التطبيق باستخدام دالة البناء سابقة التعريف لفصيلة الأساس CWinApp() (ولذلك جاء جسم الدالة خاليا من المحتويات). يلى ذلك إعلان هدف واحد فقط للتطبيق بالاسم theApp وهو عبارة عن هدف من الفصيلة CMFCHelloApp ، كما نرى فى شريحة البرنامج التالى:

```
CMFCHelloApp::CMFCHelloApp()  
{
```

```

// TODO: add construction code here,
// Place all significant initialization in InitInstance
دالة البناء خالية من المحتويات ، حيث تستخدم دالة البناء للفصيلة الأم.
}
// The one and only CMFCHelloApp object
CMFCHelloApp theApp;          إعلان هدف التطبيق الأوحد

```

وكما نرى أن هدف التطبيق قد تم الإعلان عنه فى النطاق العام للملف بحيث يصبح هدفا عاما (Global) يمكن التوصل إليه من أى دالة من دوال MFC.

ملاحظة:

إن معظم دوال الفصيلة CWinApp دوال افتراضية يمكن ركوبها من داخل التطبيق.

ملاحظة:

دالة شحن الهدف InitInstance()

بالرغم من أن الدالة الرئيسية (**WinMain()**) محجوبة عن المبرمج ، لكنه يتم ربطها بالبرنامج أثناء عمليتي البناء. وهى تقوم باستدعاء الدالة (**InitInstance()**) من فصيلة التطبيق لشحن هدف التطبيق بالقيم الابتدائية ، وبدء حلقة الرسائل ، وتستمر الدالة فى العمل حتى ينتهى التطبيق. وفيما يلى نص الدالة (**InitInstance()**):

```

BOOL CMFCHelloApp::InitInstance()
{
// إعداد عناصر التحكم بالنافذة
AfxEnableControlContainer();
#ifdef _AFXDLL

```

```

        Enable3dControls();
    #else
        Enable3dControlsStatic();
    #endif
    // إجراءات التسجيل
        SetRegistryKey(_T("Local AppWizard-Generated
Applications"));
        LoadStdProfileSettings();
    //
        خلق هدف الوثيقة
        CSingleDocTemplate* pDocTemplate;
        pDocTemplate = new CSingleDocTemplate(
            IDR_MAINFRAME,
            RUNTIME_CLASS(CMFCHelloDoc),
            RUNTIME_CLASS(CMainFrame),
            RUNTIME_CLASS(CMFCHelloView));
        AddDocTemplate(pDocTemplate);
    // إجراءات خط الأوامر
        CCommandLineInfo cmdInfo;
        ParseCommandLine(cmdInfo);
        if (!ProcessShellCommand(cmdInfo))
            return FALSE;
    //
        عرض وتحديث النافذة
        m_pMainWnd->ShowWindow(SW_SHOW);
        m_pMainWnd->UpdateWindow();
        return TRUE;
}

```

وكما نرى في شريحة الكود السابقة أن الدالة `InitInstance()` تحتوى على بعض الإجراءات التحضيرية والتي أسبقناها بتعليقات مختصرة ، أما سطور الكود التي تمثل قلب الدالة فقد ميزناها بالبنط الثقيل ، وهي تنقسم إلى جزئين:

١. خلق نموذج الوثيقة (Document Template).

٢. عرض وتحديث نافذة التطبيق.

نموذج الوثيقة

أما نموذج الوثيقة فهو عبارة عن هدف يخزن معلومات هامة عن الفصائل المختلفة والموارد التي تستخدم في خلق كل من أهداف الوثيقة والمشهد والإطار العام. ويتم إعلان نموذج الوثيقة كمؤشر إلى هدف من الفصيلة **CSingleDocTemplate** كالآتي:

```
CSingleDocTemplate* pDocTemplate
```

ثم يتم تخصيص حيز من الذاكرة لهذا الهدف بالاستعانة بالمؤثر **new** ودالة بناء الفصيلة **CSingleDocTemplate** التي تستخدم أربعة بارامترات كالآتي:

```
pDocTemplate = new CSingleDocTemplate  
(  
    IDR_MAINFRAME,  
    RUNTIME_CLASS(CMFCHelloDoc),  
    RUNTIME_CLASS(CMainFrame),  
    RUNTIME_CLASS(CMFCHelloView)  
);
```

أما البارامتر الأول **IDR_MAINFRAME** فهو عبارة عن رقم التعارف (أو الهوية) لموارد البرنامج. وأما البارامترات الثلاثة التالية فتحتوي على معلومات فصيلة الوثيقة (**CMFCHelloDoc**) والإطار (**CMainFrame**) والمشهد (**CMFCHelloView**). ويتم الحصول على المعلومات عن هذه الفصائل بالاستعانة بالماكرو **RUNTIME_CLASS** الذي يستخدم اسم الفصيلة كبارامتر.

بعد ذلك يتم تخزين نموذج الوثيقة بداخل هدف الوثيقة بحيث يصبح النموذج متاحا عند فتح الوثيقة. ويتم ذلك باستخدام الدالة **AddDocTemplate()** أحد أعضاء فصيلة التطبيق **CWinApp** كالآتي:

وظائف أخرى لدالة شحن الهدف

تعتبر الدالة `InitInstance()` هي الدالة المناسبة لإعلان المتغيرات العامة في برنامجك ، كما أنها قابلة للتعديل في حالة إذا ما أردت الاستغناء عن الوثيقة والمشهد سابقى التعريف ، وفضلت استخدام وثيقة أو مشهدا خاصا ببرنامجك.

وفيما يلي نلخص الإجراءات التنظيمية التي تحتوى عليها الدالة.

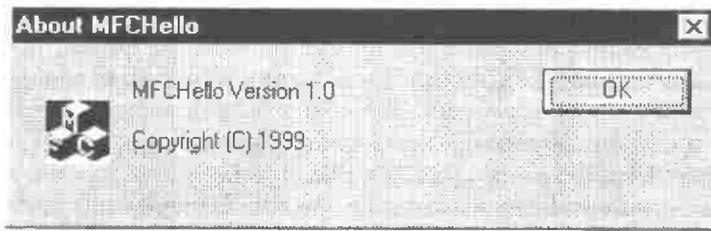
الوظيفة	الدالة/الدوال
تمنح النوافذ والأزرار مظهرا مجسما وهي غير ضرورية مع النوافذ التالية للطرز NT3.51	<code>Enable3dControls()</code> <code>Enable3dControlsStatic()</code>
لتسجيل اختيارات البرنامج (أو تحميلها) في سجل النوافذ (Windows Registry).	<code>SetRegistryKey()</code> <code>LoadStdProfileSettings()</code>
تستخدم الدالة الأولى في تحليل الأمر في حالة تشغيل البرنامج في بيئة خط الأوامر (دوس). وتستخدم الدالة الثانية في تنفيذ الأمر.	<code>ParseCommandLine()</code> <code>ProcessShellCommand()</code>

جدول (٦-١) دالة شحن الهدف `InitInstance()`

دالة المعلومات عن التطبيق `OnAppAbout`

تؤدي هذه الدالة دورا بسيطا ولكنها تستحق وقفة. إن هذه الدالة تؤدي إلى ظهور النافذة الموضحة بالشكل التالي عندما تستخدم أمر

القائمة Help - About MFCHello ، أو تضغط على الزر المميز بعلامة الاستفهام (?) بسطر الأدوات.



شكل (٦-٣) صندوق المعلومات عن التطبيق (About)

وهذه هي الدالة نفسها:

```
void CMFCHelloApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}
```

إن الدالة في مجملها تمثل مقبضا لرسالة ، فهي عبارة عن استجابة لأحد اختيارات القائمة. وهي تحتوى على عبارتين ، الأولى لإعلان هدف صندوق حوار من الفصيلة **CAboutDlg**:

```
CAboutDlg aboutDlg;
```

والثانية لاستدعاء الدالة **DoModal()** العضوة بالفصيلة الأساسية **CDialog**:

```
aboutDlg.DoModal();
```

أما وظيفة الدالة **DoModal()** فهي تنحصر في عرض الصندوق والاستجابة لرد فعل المستخدم وهو الضغط على الزر OK في هذه الحالة. والصندوق المستخدم هنا يسمى بالصندوق النمطي (Modal).



يوجد نوعان من صناديق الحوار ، الصندوق النمطي (Modal) والصندوق غير النمطي (Modeless). والنوع الأول يحتل بؤرة التركيز بحيث لا يستطيع المستخدم أن ينتقل إلى نشاط آخر فى التطبيق قبل الاستجابة للصندوق وإغلاقه. أما النوع غير النمطي فيمكنك تركه مفتوحا والانتقال إلى نافذة التطبيق الرئيسية.

ويوجد إعلان الفصيلة CAboutDlg بملف الكود MFCHello.cpp وهو موضع بشريحة البرنامج التالية:

```
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
   //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    }}AFX_DATA
    // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CAboutDlg)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);
    }}AFX_VIRTUAL
// Implementation
protected:
    {{{AFX_MSG(CAboutDlg)
        // No message handlers
    }}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

والأجزاء الهامة هنا هي:

١. استدعاء الدالة DoModal() التابعة للفصيلة CDialog.

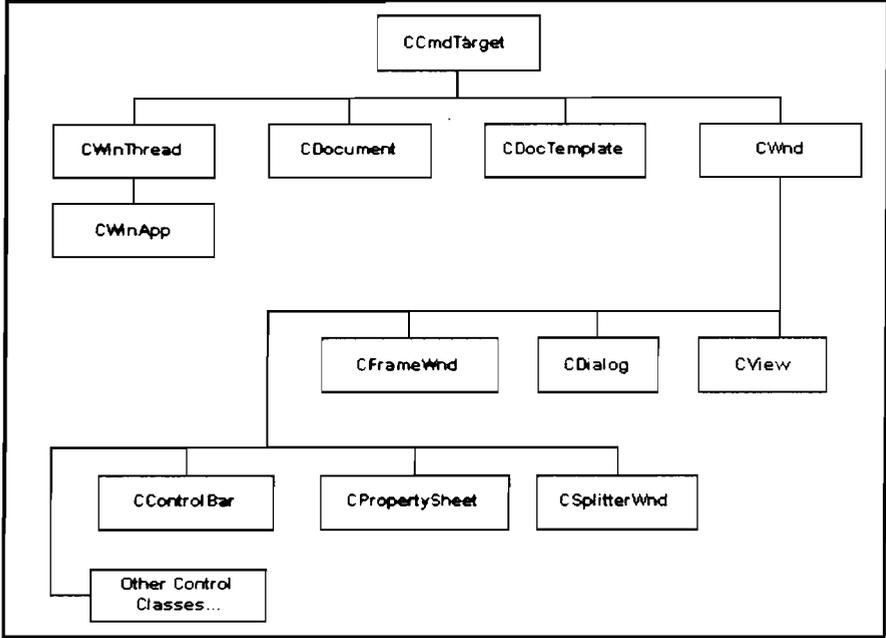
٢. ركوب الدالة الافتراضية (**DoDataExchange()** سيلي الحديث عن الدور الذي تؤديه هذه الدالة في الفقرات القادمة).

خريطة الرسائل (Message Map)

سنناقش في هذه الفقرة الأحداث المحركة لعملية الاستجابة لهذه الرسالة والتي تؤدي إلى عرض نافذة المعلومات About.

١. عندما تضغط على اختيار القائمة Help-About فإن النوافذ ترسل إلى التطبيق الرسالة **WM_CMMAND** التي تحدد الأمر الذي تم اختياره بالاستعانة برقم التعارف (ID). وفي هذا المثال فإن رقم التعارف هو **ID_APP_ABOUT** (وهو موجود بجدول الحرفيات). وعلى هذا الأساس يتم استلام وتوزيع الرسالة بداخل التطبيق.

٢. والرسالة **WM_CMMAND** (وتسمى رسالة الأوامر) تتميز بأهمية خاصة ، لأنها بخلاف الرسائل الأخرى ، لا يتم توزيعها على النوافذ فقط ، بل على هدف التطبيق وما يتبعه من أهداف النوافذ والمشهد والوثيقة. ويطلق على هذه المجموعة من الأهداف اسم أهداف الأوامر (Command Targets) ، وهي تنحدر جميعا من الفصيلة **CCmdTarget**. وفي الشكل التالي نعرض جانبا هاما من شجرة فصول المؤسسة MFC وهو الجانب المحتوى على الفصيلة **CCmdTarget** والفصول التي تتبعها بما في ذلك الفصيلة **CWnd** التي تنحدر منها النوافذ ومشتقاتها.



شكل (٤-٦) الفصائل المشتقة من CCmdTarget

٣. عندما يتلقى أحد الأهداف رسالة أوامر (WM_COMMAND) يتولى مقبض الرسالة (الدالة OnCmdMsg الموروثة من الفصيلة CCmdTarget) فحص الرسالة لمعرفة إذا ما كان جاهزا للاستجابة لها أم لا. فإن لم يكن ، فيتم إرسالها إلى هدف آخر من أهداف الأوامر وهكذا. أما كيفية فحص الرسالة فيتم باستخدام جدول خاص بمؤسسة الفصائل MFC ، يسمى خريطة الرسائل (Message Map) ، ويربط ما بين كل رسالة وما بين الدالة المخصصة لمعالجتها (المقبض). ويستخدم هذا الجدول في استدعاء الدالة المناسبة لكل رسالة.

٤. فى شريحة الكود التالية نرى خريطة الرسائل لهدف التطبيق ،
والواردة فى بداية الملف MFCHello.cpp. ولأن هدفى التطبيق
والوثيقة ليسا بنوافذ ، فهما يستجيبان فقط لرسائل الأوامر
(WM_COMMAND).

```
// CMFCHelloApp
BEGIN_MESSAGE_MAP(CMFCHelloApp, CWinApp)
  {{{AFX_MSG_MAP(CMFCHelloApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
      // NOTE - the ClassWizard will add and remove mapping macros here.
      // DO NOT EDIT what you see in these blocks of generated code!
  }}}AFX_MSG_MAP
  // Standard file based document commands
  ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
  ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
  // Standard print setup command
  ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()
```

وتبدأ خريطة الرسائل بالعبارة BEGIN_MESSAGE_MAP وتنتهى بالعبارة
END_MESSAGE_MAP. وتحتوى بداخلها على أوامر ماكرو تبدأ بالكلمة
ON_COMMAND (وهو الماكرو المختص بالرسائل WM_COMMAND).
ويختص كل ماكرو برسالة معينة ، ويحدد الدالة التى تستجيب لها
كالمثال الآتى:

```
ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
```

إن هذا الماكرو يحدد التناظر بين الرسالة ذات رقم التعارف
ID_APP_ABOUT وبين المقبض OnAppAbout.

(٦-٤) هدف نافذة الإطار MainFrame Window Object

إن هذا الهدف يمثل الإطار الخارجى لنافذة التطبيق شاملا سطر العنوان (Title bar) والقائمة (Menu bar) وسطر الأدوات (Toolbar) وسطر الحالة (Status bar). أما نافذة المشهد التى تحتوى على العبارة "Hello my MFC friends" (منطقة العميل) فهى نافذة ابنة لنافذة الإطار.

وعندما تبدأ من شجرة الفصائل ، وتضغط على اسم الفصيلة CMainFrame فإنك تشاهد ملف الإعلانات للفصيلة "MainFrm.h" ، وترى به لإعلان الفصيلة CMainFrame الموروثة من الفصيلة CFrameWnd. ولو تتبععت شجرة الفصائل الموضحة بالشكل السابق ، فإنك ترى أن هذه الفصيلة ترث العديد من فصائل المؤسسة ، فالفصيلة CFrameWnd ترث CWnd التى ترث بدورها CCmdTarget التى ترث الفصيلة الأم CObject وهى الفصيلة التى تأتى منها أغلب فصائل المؤسسة. ومعنى ذلك أن هذه الفصيلة غنية بالبيانات والدوال الأعضاء الموروثة من جميع هذه الفصائل.

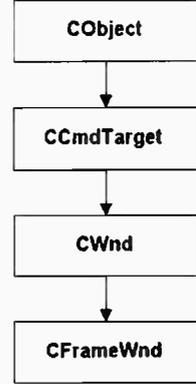
والفارق الأساسى بين التطبيق متعدد الوثائق (MDI) وبين هذا التطبيق (SDI) أن فصيلة نافذة الإطار بالنوع الأول ترث من الفصيلة CMDIFrameWnd وليس من CFrameWnd.

من الأجزاء الهامة بهذا الملف أيضا إعلان الأعضاء المحمية الممثلة لسطر الحالة (Status bar) وسطر الأدوات (Toolbar) كالآتى:

```
protected: // control bar embedded members
    CStatusBar m_wndStatusBar;
    CToolBar m_wndToolBar;
```

شجرة الوراثة

يوضح الشكل التالي الفصيلة CFrameWnd وموقعها في شجرة الوراثة.



شكل (٥-٦) فصيلة نافذة الإطار بشجرة الوراثة

خريطة الرسائل

بفتح ملف التطبيق "MainFrm.cpp" فإننا نرى في بدايته خريطة الرسائل التي تعتمد على ماكرو واحد ON_WM_CREATE ونصها كالآتي:

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
//{{AFX_MSG_MAP(CMainFrame)
// NOTE - the ClassWizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code !
ON_WM_CREATE()
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

والرسالة التي يتم معالجتها هنا هي رسالة النوافذ WM_CREATE ويتم التعامل معها بواسطة مقبض الدالة **OnCreate()** الموروثة من الفصيلة **CWnd**.

الدالة **OnCreate()**

توجد الدالة بملف التطبيق "MainFrm.cpp" أيضا ، وهي تمثل قلب فصيلة نافذة الإطار . وهذا هو نصها:

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
    سطر الأدوات
    if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD
        | WS_VISIBLE | CBRS_TOP | CBRS_GRIPPER
        | CBRS_TOOLTIPS
        | CBRS_FLYBY | CBRS_SIZE_DYNAMIC)
        || !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;    // fail to create
    }
    سطر الحالة
    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
        sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar\n");
        return -1;    // fail to create
    }
    خاصية الحوض لسطر الأدوات
    // TODO: Delete these three lines if you don't want the toolbar to
    // be dockable
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);

    return 0;
}
```

وكما نرى فى الدالة أن هناك بعض السطور الحاكمة وقد كتبناها باستخدام البنط الثقيل. أما الخطوط الرئيسية لمنطق الدالة فهى:

١. لو أنك حذفته جميع محتويات هذه الدالة فيما عدا العبارة "return 0" الموجودة فى نهاية الدالة ، فإن الترجمة سوف تتم بنجاح وسوف تحصل على نافذة التطبيق فيما عدا سطر الأدوات وسطر الحالة وبالطبع فإنك لن ترى العبارة "Hello my MFC friends"! معنى هذا أن الوراثة تلعب الدور الرئيسى هنا.

٢. لو أنك أبقيت على العبارة الأولى فقط فى الدالة ، وهى:
`return -1; if (CFrameWnd::OnCreate(lpCreateStruct) == -1)`
فإنك لا تزال ترى العبارة "Hello my MFC friends" على الشاشة. إن هذه العبارة هى المسئولة عن محتويات النافذة.

٣. أما بقية عبارات الدالة فهى مخصصة لخلق سطر الأدوات وسطر الحالة باستخدام الأعضاء `m_wndStatusBar` (هدف من الفصيلة `CStatusBar`) و `m_wndToolBar` (هدف من الفصيلة `CToolBar`) ، وهذه الأهداف معلنة فى ملف العناوين كما قدمنا. وكما نرى فى هذه العبارات أن الماكرو `TRACE0` قد استخدم هنا لتشخيص الأعطال ، حيث يطبع الرسالة الموجودة بين القوسين فى بطاقة الصيانة (Debug Tab) عند تحقق العبارة الشرطية.

٤. أما عملية خلق سطر الأدوات فهى تتم باستخدام الدالة `CreateEx()` ، ويتم تحميله باستخدام الدالة `LoadToolBar()` ، وهما من دوال الفصيلة `CToolBar` ، كالاتى (حذفنا بعض الشروط للتبسيط):

```

if (!m_wndToolBar.CreateEx(
....
|| !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
{ ...

```

أما الثوابت المفصولة بالمؤثرات | أو || فهي تحدد خصائص سطر الأدوات (Styles) ، وسوف يلي شرحها في الأبواب القادمة. ٥. تتم عملية خلق وتحميل سطر الحالة بطريقة مماثلة ، حيث تستخدم الدالتان **Create()** و **SetIndicators()** ، وهما من دوال

الفصيلة **CStatusBar**:

```

if (!m_wndStatusBar.Create(this) ||
!m_wndStatusBar.SetIndicators(...

```

خاصية الحوض لسطر الأدوات (Toolbar Docking)

أما السطور الثلاثة الأخيرة في الدالة **OnCreate()** فهي تؤدي إلى إكساب سطر الأدوات خاصية الحوض (Docking) بحيث يمكنك نقل السطر بما يحتويه – بالسحب و الإسقاط – إلى أي موقع على الشاشة:

```

m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndToolBar);

```

كما يمكنك تعديل أبعاد الحوض الذي يحتوى على الأدوات كما بالشكل التالي. فإذا حذفت هذه السطور الثلاثة ، أصبح سطر الأدوات أفقياً في مكانه المعروف ويسمى في هذه الحالة بالسطر الطافي (Floating). ونلاحظ استخدام الدالة **EnableDocking()** مرتين ، الأولى مع سطر الأدوات (وفي هذه الحالة تستدعى من الفصيلة **CControlBar**) والثانية مع نافذة الإطار (وفي هذه الحالة

يحتوى سطر الحالة على ثلاثة صناديق صغيرة على اليمين تسمى المبيئات (Indicators) وهى تعكس حالة الأزرار الثلاثة Caps Lock و Scroll Lock و Num Lock كما هو كموضح بالشكل السابق. وتحفظ أرقام التعارف لهذه الأزرار فى المصفوفة Indicators كالاتى:

```
static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
```

والنمط المستخدم لعناصر هذه المصفوفة هو النمط **UINT** وهو عبارة عن نمط صحيح طويل (٣٢ بت). وتقع هذه العبارة تالية لخريطة الرسائل مباشرة بالبرنامج MainFrm.cpp. وتستخدم هذه المجموعة من أرقام التعارف بواسطة الدالة **SetIndicators()** عند إنشاء سطر الحالة ، كما نرى فى الدالة **OnCreate()**.

دوال أخرى لنافذة الإطار

أغفلنا فى هذه المناقشة بعض الدوال التى يكتبها الساحر ولا تؤثر مباشرة على التطبيق. وهذا هو ملخص سريع بوظائفها:

الوظيفة	الدالة
لتغيير خصائص نافذة الإطار حسب الطلب. وهى دالة افتراضية من دوال الفصيلة CWnd .	PreCreateWindow()

دالة صيانة ، يجوز توظيفها للتحذير من أخطاء معينة عند بناء هدف النافذة. وهي دالة افتراضية تابعة للفصيلة الأم CObject .	AssertValid()
دالة صيانة تستخدم في طباعة معلومات عن الأهداف في بطاقة الصيانة (Debug Tab) ، وهي دالة افتراضية من الفصيلة الأم CObject .	Dump()

جدول (٦-٢) دوال أخرى بنافذة الإطار

The Document Object

(٦-٥) هدف الوثيقة

استخدمنا من قبل فصيلة الوثيقة CMFCHelloDoc بالملف CMFCHelloDoc.cpp لتخزين المتغير الحرفي m_strHelloText وشحنه بالقيمة "Hello my MFC friends!". كما استخدمنا فصيلة المشهد CMFCHelloView لعرض هذا الحرفي على الشاشة. وتسمى هذه الطريقة بالفصل ما بين البيانات ومشهد البيانات. فالوثيقة تستخدم بصفة عامة في تخزين البيانات بصرف النظر عن طريقة عرضها. أما المشهد فيستمد بياناته من الوثيقة ويتولى عرضها على الشاشة بصرف النظر عن نوعها ، سواء كانت رسماً أو نصاً أو جولا إلكترونياً. وهذا الفصل ما بين المشهد والوثيقة يمكنك من عرض نفس البيانات بصور متعددة. وعلى سبيل المثال فإن برامج الجداول الإلكترونية (مثل إكسيل) يمكنك من عرض نفس البيانات في صورة جدول أو في صورة خرائط بيانية متنوعة.

أما البيانات التي خزناها في الوثيقة في المثال MFCHello فقد أضفناها إلى دالة البناء كالتالي:

```
CMFCHelloDoc::CMFCHelloDoc()
{
    // TODO: add one-time construction code here
    m_strHelloText = "Hello my MFC friends!";
}
```

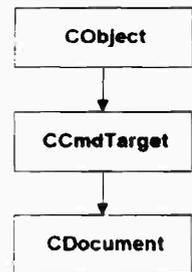
خريطة الرسائل

ونلاحظ في هذه الفصيلة أن خريطة الرسائل قد جاءت خالية من المحتويات. وسوف نضيف إليها بعض سطور الكود في الأمثلة القادمة.

```
BEGIN_MESSAGE_MAP(CMFCHelloDoc, CDocument)
//{{AFX_MSG_MAP(CMFCHelloDoc)
// NOTE - the ClassWizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

شجرة الوراثة

أما الشكل التالي فيوضح فصيلة الوثيقة في شجرة الوراثة من فصائل المؤسسة MFC.



شكل (٦-٧) فصيلة الوثيقة في شجرة الوراثة

دوال أخرى بفصيلة الوثيقة

أما بقية الدوال التي تحتوى عليها فصيلة المشهد فهي إما خالية (يمكن توظيفها فى تطبيقات أخرى قادمة) أو تختص بالإجراءات التنظيمية العامة ، وهذا ملخصها:

الوظيفة	الدالة/الدوال
البناء والهدم. لاحظ أن المتغير الحرفى m_strHelloText الذى أضناه إلى دالة البناء يظهر فى قسم الشجرة كعضو من أعضاء الفصيلة.	CMFCHelloDoc() ~CMFCHelloDoc()
دوال صيانة (راجع جدول فصيلة نافذة الإطار – جدول (٦-٢)).	AssertValid() Dump()
تستخدم هذه الدالة فى عملية القراءة أو الكتابة من/فى الملفات. وسيلى توظيفها فى الأمثلة القادمة.	Serialize()
يمكنك إضافة عبارات شحن الأهداف من النوع SDI بهذه الدالة.	OnNewDocument()

جدول (٦-٣) دوال أخرى بفصيلة الوثيقة

The View Object

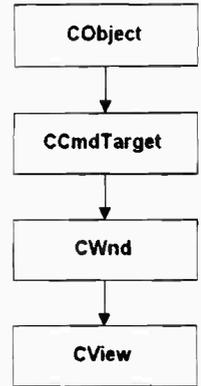
(٦-٦) هدف المشهد

كما ذكرنا من قبل أن المشهد عبارة عن نافذة تغطى منطقة العميل فى النافذة الرئيسية للإطار وتختص بعرض البيانات على الشاشة.

كما أنه من الجائز تقسيم نافذة المشهد إلى عدة إطارات يعرض كل منها مشهدا مختلفا مستمدا من نفس الوثيقة.

شجرة الوراثة

وفصيلة المشهد من أغنى الفصائل بالدوال والبيانات ، وهي تـرث الفصيلة CView و CWnd و CCmdTarget و CObject بالتسلسل. أنظر الشكل التالي.



شكل (٦-٨) فصيلة المشهد في شجرة الوراثة

ملف العناوين

توضح شريحة الكود التالية الجزء الأساسي من ملف العناوين MFCHelloView.h ونرى فيه مجموعة كبيرة من الدوال الافتراضية المستخدمة في فصيلة المشهد (تحت التعليق //Overrides). كما نرى في الجزء الأخير منها دالة خطية ترجع مؤشرا إلى هدف الوثيقة.

```
class CMFCHelloView : public CView
{
protected: // create from serialization only
    CMFCHelloView();
    DECLARE_DYNCREATE(CMFCHelloView)
```

```

// Attributes
public:
    CMFCHelloDoc* GetDocument();

// Operations
public:

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CMFCHelloView)
public:
    virtual void OnDraw(CDC* pDC); // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
//}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMFCHelloView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
...
...
};

#ifdef _DEBUG // debug version in MFCHelloView.cpp
inline CMFCHelloDoc* CMFCHelloView::GetDocument()
{ return (CMFCHelloDoc*)m_pDocument; }
#endif

```

خريطة الرسائل

تقع خريطة الرسائل كالعادة في بداية ملف الكود MFCHelloView.cpp وهي تحتوي على إمكانات مختلفة للطباعة (تتاظر اختيارات الطباعة بقائمة الملفات للتطبيق MFCHello) وتستخدم دوال الطباعة الموجودة بفصيلة الأساس CView. وهذا هو نصها:

```

BEGIN_MESSAGE_MAP(CMFCHelloView, CView)
//{{AFX_MSG_MAP(CMFCHelloView)
// NOTE - the ClassWizard will add and remove mapping macros here.

```

```
// DO NOT EDIT what you see in these blocks of generated code!
//}AFX_MSG_MAP
// Standard printing commands
ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()
```

الدالة OnDraw()

إن هذه الدالة هي محور فصيلة المشهد وقد استخدمناها في برنامجنا لطباعة العبارة على الشاشة كالآتي:

```
// CMFCHelloView drawing
void CMFCHelloView::OnDraw(CDC* pDC)
{
    CMFCHelloDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    pDC ->TextOut(100,100, pDoc->m_strHelloText);
}
```

علاوة على السطر الأخير الذي أضفناه إلى هذه الدالة ، فإن هناك بعض الأجزاء التي تستأهل الاهتمام. فالسطر الأول في هذه الدالة يستخدم الدالة **GetDocument()** (وهي عضوة بفصيلة الأساس **CView** للحصول على مؤشر إلى هدف الوثيقة **pDoc**:

```
CMFCHelloDoc* pDoc = GetDocument();
```

وهذا هو المؤشر الذي استخدمناه في التوصل إلى المتغير الحرفي **m_strHelloText** الذي اختزنناه من قبل في هدف الوثيقة. أما العبارة **ASSERT_VALID(pDoc)** فهي تستخدم لاختبار صلاحية المؤشر.

دوال أخرى بنافذة المشهد

فيما يلي ملخص بالدوال المتبقية في فصيلة المشهد.

الوظيفة	الدالة/الدوال
دالتا البناء والهدم.	CMFCHelloView() ~CMFCHelloView()
دوال صيانة (راجع جدول فصيلة نافذة الإطار – جدول (٦-٢)).	AssertValid() Dump()
دوال تحضير وبداية ونهاية الطباعة.	OnPreparePrinting() OnBeginPrinting() OnEndPrinting()
لتغيير خصائص نافذة الإطار حسب الطلب. وهى دالة افتراضية من دوال الفصيلة CWnd .	PreCreateWindow()

الموجز

١. تعرفنا فى هذا الباب بأحداث البرنامج MFC وبأهم الفصائل والدوال الأعضاء التى يتكون منها.

• عرفنا أيضا تفصيلات الأهداف الأربعة الأساسية التى يتكون منها أى تطبيق ، وهى أهداف التطبيق (Application) ، ونافذة الإطار الرئيسية (Main Frame Window) ، والمشهد (View) ، والوثيقة (Document) ، كما عرفنا العلاقة بينهم وبين بعضهم.

٢. تعرفنا أيضا بجزء رئيسى من شجرة فصائل المؤسسة MFC وعلاقة الوراثة بينها وبين بعضها.

٣. عرفنا أيضا هدف التطبيق الأوحد الذى يعلن كمتغير عام **theApp** ، كما عرفنا دالة شحن هدف التطبيق والدور الذى تؤديه فى التطبيق.

٤. تعرفنا بالتطبيق ذى الوثيقة المفردة (SDI) والتطبيق ذى الوثائق المتعددة (MDI) ، وعرفنا أن الأول ترث نافذته من الفصيحة CFrameWnd أما الثانى فيرث من CMDIFrameWnd .
٥. دوال MFC التى وردت فى هذا الباب:

CCmdTarget::OnCmdMsg()
CControlBar::EnableDocking()
CDC::TextOut()
CDialog::DoModal()
CFrameWnd::DockControlBar()
CFrameWnd::EnableDocking()
CObject::AssertValid()
CObject::Dump()
CObject::Serialize()
CStatusBar::Create()
CStatusBar::SetIndicators()
CToolBar::CreateEx()
CToolBar::LoadToolBar()
CView::GetDocument()
CView::OnBeginPrinting()
CView::OnDraw()
CView::OnEndPrinting()
CView::OnPreparePrinting()
CWinApp::AddDocTemplate()
CWinApp::InitInstance()
CWnd::DoDataExchange()
CWnd::OnCreate()
CWnd::PreCreateWindow()
CWnd::ShowWindow()
CWnd::UpdateWindow()

٦. البيانات الأعضاء التي وردت في هذا الباب:

m_pMainWnd
m_pDocument

٧. الرسائل التي وردت في هذا لباب:

WM_COMMAND
WM_CREATE

٨. الأنماط التي وردت في هذا الباب:

UINT
CString