Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

③

Hostování ActiveX prvků ve vlastní Win32 aplikaci

Knihovna ATL nám kromě silného nástroje pro vytváření vlastních ActiveX prvků také výrazným způsobem usnadňuje použití hotových ActiveX prvků a obecných COM objektů v běžné Win32 aplikaci. V této tématické části si ukážeme použití (takzvané hostování) ActiveX prvků dvěma způsoby

- ActiveX umístěný na dialogovém okně.
- ActiveX jehož instanci vytvoříme za běhu programu.

První způsob si ukážeme na příkladu vytvoření jednoduchého internetového prohlížeče využívajícího ActiveX prvek nazvaný Webový prohlížeč společnosti Microsoft. V dalším příkladu si vytvoříme jednoduchý přijímač a přehrávač vysílání streamového zvuku, videa nebo multimediálního souboru na disku.

3.1 ActiveX prvek na dialogovém okně

Pokud máte praktické zkušenosti s vývojem aplikací v MFC nebo WinForms, pravděpodobně víte, že použití hotových ActiveX prvků umístěných na dialogu nebo „formuláři" (jak se říká oknům ve WinForms) je relativně jednoduché. Naopak pokud vytvoříte aplikaci jako Win32 projekt a budete psát v čistém Win32 API, bude realizace hostování ActiveX prvku se vším všudy – tedy včetně volání metod a zachytávání jeho událostí – pro většinu programátorů frustrující záležitostí. ATL knihovna nám pro případ hostování ActiveX prvků na dialogovém okně nabízí třídu *CAxDialogImpl* a pro snadné zachytávání událostí (*events*) *IDispEventImpl*.

3.1.1 Vlastní internetový prohlížeč

Na následujících stránkách si vytvoříme projekt jednoduchého internetového prohlížeče, na kterém si (kromě realizace vlastního hostování ActiveX prvků) ukážeme také některé další techniky programování ve Win32 a ATL, které se budou jistě hodit zejména začínajícím programátorům.

Jak bude vypadat výsledek? Aplikace bude mít běžné hlavní okno s hlavní nabídkou, panelem nástrojů a stavovým řádkem. V klientské oblasti bude umístěn dialog s jedním editačním polem, tlačítkem a ActiveX prvkem *Webový prohlížeč společnosti Microsoft*. Do editačního pole budeme zadávat požadovanou URL adresu, kterou po stisknutí tlačítka otevřeme v prohlížeči. Dále si ukážeme, jak zjistit informace o prohlížené webové stránce. Volbou z hlavní nabídky nebo panelu nástrojů zobrazíme dialog s informací o velikosti právě zobrazené stránky a seznamem obrázků na této stránce, s informací o velikosti

každého z nich. Vybraný obrázek bude možno zobrazit ve výchozím editoru obrázků tak, že ho uložíme na disk do dočasného souboru a ten pak otevřeme ve výchozím editoru přidruženém příponě tohoto souboru.

3.1.2 Třída pro stavový řádek

Nejdříve si rozšíříme naši ukázkovou knihovnu tříd o dvě třídy, které nám usnadní realizaci zmíněného panelu nástrojů a stavového řádku hlavního okna. Podívejme se nejprve na výpis kódu jednoduché třídy zapouzdřující stavový řádek:

```
/*************************************************
Třída StatusBar
Soubor: AtlDemo.h
(C) 2005 Radek Chalupa - www.radekchalupa.cz
*************************************************/

class StatusBar : public CWindow
{
public:
      StatusBar()
      {
      }

      bool Vytvorit(HWND hParent, UINT id = 1,
        DWORD plusStyl = 0)
      {
        RECT rect = {0, 0, 0, 0};
        plusStyl |= (WS_CHILD | WS_VISIBLE);
        if (!Create(STATUSCLASSNAMEW,
          hParent, rect, L"", plusStyl, 0, id))
          AtlThrowLastWin32();
      }

      ~StatusBar()
      {
        if (IsWindow())
          if (!DestroyWindow())
            AtlThrowLastWin32();
        m_hWnd = NULL;
      }

      void SetJednoduchy(bool jednoduchy)
      {
        SendMessage(SB_SIMPLE, (WPARAM)jednoduchy);
      }

      void SetCasti(int* pozice, int pocet)
      {
        if (!SendMessage(SB_SETPARTS,
          (WPARAM)pocet, (LPARAM)pozice))
```

```
            AtlThrowLastWin32();
      }

      void SetIkona(HICON hIcon, int cast)
      {
        if (!SendMessage(SB_SETICON,
          (WPARAM)cast, (LPARAM)hIcon))
          AtlThrowLastWin32();
      }

      void SetText(const wchar_t* text, int cast)
      {
        if (cast >= 0)
        {
          if (!SendMessage(SB_SETTEXT,
            (WPARAM)cast, (LPARAM)text))
            AtlThrowLastWin32();
        }
        else
        {
          if (!SendMessage(SB_SETTEXT,
            (WPARAM)SB_SIMPLEID, (LPARAM)text))
            AtlThrowLastWin32();
        }
      }
};
```

Třída je odvozená od CWindow. Kromě funkce pro vytvoření nám usnadňuje realizovat některé obvyklé funkce a nastavení stavového řádku pomocí funkcí jejichž použití je (zejména pro programátory méně zkušené ve Win32 API) intuitivnější než pokud bychom měli pouze handle okna a příslušné zprávy používat přímo a předtím je (alespoň někteří čtenáři) museli hledat a studovat v dokumentaci.

3.1.3 Třída pro panel nástrojů

Další jednoduchou třídou, kterou použijeme, budeme zapouzdřovat standardní panel nástrojů (*ToolBar*) systému Windows. Zdrojový kód implementující jeho nejběžnější funkce vypadá takto:

```
/*************************************************
Třída ToolBar
Soubor: AtlDemo.h
(C) 2005 Radek Chalupa - www.radekchalupa.cz
*************************************************/

enum ToolBarStrana
{
      toolBarNahore,
      toolBarVpravo,
      toolBarDole,
```

```
      toolBarVlevo
};

class ToolBar : public CWindow
{
public:
      ~ToolBar()
      {
        if (IsWindow())
          if (!DestroyWindow())
            AtlThrowLastWin32();
      }

      bool PridatTlacitko(int nPrikaz,
        int nBmpIndex,
        const wchar_t* text = NULL,
        BYTE bState = (BYTE)TBSTATE_ENABLED,
        BYTE bStyle = (BYTE)(TBSTYLE_BUTTON |
          TBSTYLE_TOOLTIPS | TBSTYLE_WRAPABLE),
        DWORD_PTR dwData = 0)
      {
        TBBUTTON tbb;
        ZeroMemory(&tbb, sizeof(TBBUTTON));
        tbb.iBitmap = nBmpIndex;
        tbb.idCommand = nPrikaz;
        tbb.fsState = bState;
        tbb.fsStyle = bStyle;
        tbb.dwData = dwData;
        tbb.iString = (INT_PTR)text;
        return (SendMessage(TB_INSERTBUTTON,
          -1, (LPARAM)&tbb) != 0);
      }

      void PovolitTlacitko(UINT uID, bool povolit)
      {
        SendMessage(TB_ENABLEBUTTON,
          uID, (LPARAM)povolit);
      }

      bool PridatOddelovac()
      {
        TBBUTTON tbb;
        ZeroMemory(&tbb, sizeof(TBBUTTON));
        tbb.iBitmap = 0;
        tbb.idCommand = 0;
        tbb.fsState = (BYTE)
          (TBSTATE_ENABLED | TBSTYLE_WRAPABLE);
        tbb.fsStyle = (BYTE)(TBSTYLE_SEP);
        tbb.dwData = 0;
        tbb.iString = (INT_PTR)L"";
```