Formulář WinBase602 jako ActiveX

Jednou z možností, jak zpřístupnit uživateli data spravovaná databázovým systémem WinBase602, **je použití formulářů jako tzv.** ActiveX **komponent.**

Vývojář může např. do aplikace ve ***Visual Basicu***, ***Visual C/C++***, ***Borland Delphi*** nebo do HTML stránky na WWW umístit objekt **ActiveX** typu **WinBase602 PohledX** a tím začlenit plnohodnotný formulář **WinBase602** do cizího prostředí. Specifikováním vlastností tohoto objektu (jméno serveru, jméno aplikace, jméno formuláře, ...) určí vzhled formuláře a zdroj dat. Po spuštění aplikace nebo aktivaci HTML stránky se komponenta automaticky přihlásí k zadanému serveru a umožní editovat data jako běžný formulář.

Mechanismus **ActiveX** je v současné době nejužívanějším nástrojem k vytváření distribuovaných objektů a k i integraci nezávisle implementovaných objektů do aplikací.

Starší označení pro **ActiveX** je **OCX**.

WinBase602 PohledX má toto CLSID:

```
32F34910-9288-101B-96B8-04021C007002
```

Šíření WinBase602 PohledX po Internetu

Jednou z výhod použití **ActiveX** komponent na WWW stránkách je i možnost jejich automatické instalace. Jestliže WWW prohlížeč zjistí, že uživatel otevřel stránku, obsahující **ActiveX** komponentu, která na daném počítači není nainstalovaná, případně, že komponenta je starší verze, nabídne instalaci nejnovější verze. Vývojář WWW stránky ovšem musí v popisu objektu specifikovat zdrojovou URL adresu instalační sady. Všechny moduly potřebné pro činnost komponenty **WinBase602 PohledX** jsou spakovány do souboru **WBVIEW.CAB**, který je součástí **WinBase602 SDK** nebo je k dispozici na WWW serveru **Software602.cz** (odkaz na aktuální verzi najdete pod domovskou stránkou **http://www.software602.cz**). Zdrojová adresa instalační sady se pak v popisu objektu zadává parametrem **CODEBASE** např:

```
<OBJECT
CLASSID=...
.
.
.
CODEBASE="http://www.mujserver.cz/PohledX/wbview.cab">
<PARAM NAME=...>
.
.
```

```
.
</OBJECT>
```

Vlastnosti komponenty "WinBase602 PohledX"

Vlastnosti (properties) představují základní charakteristiky komponenty. **WinBase602 PohledX** má následující vlastnosti:

Server	Jméno databázového serveru
ServerIP	IP adresa databázového serveru
UserName	Jméno uživatele
Password	Heslo uživatele
Application	Jméno databázové aplikace
Project	Jméno projektu (programu), v jehož kontextu bude objekt pracovat
ViewSrc	Jméno formuláře uloženého v databázi nebo zdrojový text popisu formuláře
ViewSrcCateg	Rozlišuje variantu `ViewSrc`
DataSrc	Jméno tabulky, dotazu uloženého v databázi nebo zdrojový text dotazu
DataSrcCateg	Rozlišuje variantu `DataSrc`
TopRec	Číslo záznamu, na kterém se formulář otevře

Detailní popis je uveden v nápovědě. Vlastnosti mají význam pouze při vytváření objektu, vzhled a chování aktivního formuláře už nemohou ovlivnit. K tomu, aby bylo možné **WinBase602 PohledX,** aktivovat je třeba specifikovat minimálně tyto vlastnosti: **Server**, **Application** a **ViewSrc** nebo **DataSrc**.

Programové ovládání vlastností

V prostředích, která umožňují zahrnout **ActiveX** (OCX) komponenty do své nabídky konstrukčních prvků, jako je např. ***Microsoft Visual Basic*** nebo ***Borland Delphi***, jsou v době návrhu aplikace vlastnosti zpravidla začleněny do panelu vlastností mezi vlastnosti ostatních objektů aplikace. Za běhu aplikace lze pak vlastnosti nastavovat běžným přiřazovacím příkazem např.:

```
Formular.Server := "MujServer";
```

V HTML se vlastnosti nastavují pomocí příkazu **<PARAM>**. Implicitní formulář do tabulky `"MojeTabulka"` v aplikaci `"MojeAppl"` na serveru `"MujServer"` můžeme vytvořit např. takto:

```
<OBJECT
```

```
CLASSID="clsid:32F34910-9288-101B-96B8-04021C007002"
ALIGN=CENTER WIDTH=275 HEIGHT=156 BORDER=1 HSPACE=5
NAME=Pohled
>
<PARAM NAME="Server"       VALUE="MujServer">
<PARAM NAME="Application"  VALUE="MojeAppl">
<PARAM NAME="DataSrc"      VALUE="MojeTabulka">
<PARAM NAME="DataSrcCateg" VALUE=1>
</OBJECT>
```

V ostatních jazycích jsou vlastnosti programově přístupné pomocí metody **`Invoke`** z rozhraní **`IDispatch`**. Identifikátory jednotlivých charakteristik jsou uvedeny v souborech definic **WBPREZEN.H** (C/C++), **WINBASE.PAS** (Pascal) resp. **WBPREZEN.TXT** (Visual Basic). Hodnoty vlastností **`ViewSrcCateg`** a **`DataSrcCateg`** lze předávat jako 16- i 32-bitová čísla. Řetězcové hodnoty se předávají jako typ BSTR tj. řetězec 16-bitových znaků, kterému předchází jeho délka. Kvůli sdílení řetězců mezi aplikacemi je třeba paměť pro řetězce alokovat pomocí funkce **`SysAllocString`** a uvolňovat funkcí **`SysFreeString`**. Metoda **`Invoke`** umožňuje nastavení vlastností jak jednotlivé, tak po celých skupinách.

Příklady nastavování vlastností v C++

Nastavení vlastnosti ViewSrc a ViewSrcCateg jednotlivě v C++ demonstruje následující příklad:

```
DEFINE_GUID(CLSID_VIEW, 0x32F34910L, 0x9288, 0x101B, 0x96,
             0xB8, 0x04, 0x02, 0x1C, 0x00, 0x70, 0x02);

if (CoCreateInstance(CLSID_VIEW,NULL,CLSCTX_INPROC_HANDLER,
                IID_IUNKNOWN, &m_lpObject) != NOERROR)
    goto error;

LPDISPATCH lpDispatch;
if (m_lpObject->QueryInterface(IID_IDispatch,
                    (LPVOID *)&lpDispatch) != NOERROR)
    goto error;

VARIANT Var;
DISPPARAMS DispParams =
{
    &Var,                          // Ukazatel na hodnotu
                                   // parametru
    NULL,                          // Ukazatel na pole ID
                                   // parametrů
    1,                             // 1 parametr
    0                              // Žádný pojmenovaný parametr
```

```
};

VariantInit(&Var);              // Vynulování Var

Var.vt      = VT_BSTR;          // ViewSrc je BSTR
Var.bstrVal = SysAllocString(L"MujPohled");
lpDispatch->Invoke
(
    DID_VIEWSRC,                // ID vlastnosti
    IID_NULL,                   // Rezervováno
    LOCALE_SYSTEM_DEFAULT,      // Jazyk implicitní pro daný
                                // systém
    DISPATCH_PROPERTYPUT,       // Nastavení vlastnosti
    &DispParams,                // Seznam parametrů
    NULL,                       // Výsledek není zajímavý
    NULL,                       // Výjimky a označení
    NULL                        // chybného argumentu také ne
);

Var.vt   = VT_I2;               // ViewSrcCateg je číslo (může
                                // být i VT_I4)
Var.iVal = CTG_VIEWNAME;        // ViewSrc je jméno pohledu
                                // uloženého v databázi
                                // (v případě VT_I4 tu bude
                                // Var.lVal)
lpDispatch->Invoke
(
    DID_VIEWSRCTG,              // ID vlastnosti
    IID_NULL,                   // Rezervováno
    LOCALE_SYSTEM_DEFAULT,      // Jazyk implicitná pro daný
                                // systém
    DISPATCH_PROPERTYPUT,       // Nastavení vlastnosti
    &DispParams,                // Seznam parametrů
    NULL,                       // Výsledek není zajímavý
    NULL,                       // Výjimky a označení
    NULL                        // chybného argumentu také ne
);
```

Další příklad ukazuje nastavení několika vlastností jedním příkazem:

```
VARIANT Var[4];
DISPID  Ids[4];

DISPPARAMS DispParams =
{
    Var,            // Ukazatel na seznam hodnot parametrů
    Ids,            // Ukazatel na pole ID  parametrů
```

```
    4,              // 4 parametry
    4               // 4 pojmenované parametry
};

for  (VARIANT *v = Var, v < Var + 4, v++);
       VariantInit(v);           // Vynulování Var

Var[0].vt      = VT_BSTR;        // Server je BSTR
Var[0].bstrVal = SysAllocString(L"MujServer");
Var[1].vt      = VT_BSTR;        // Application je BSTR
Var[1].bstrVal = SysAllocString(L"MojeAppl");
Var[2].vt      = VT_BSTR;        // ViewSrc je BSTR
Var[2].bstrVal = SysAllocString(L"MujPohled");
Var[3].vt      = VT_I2;          // ViewSrcCateg je číslo
Var[3].iVal    = CAT_VIEWNAME;

lpDispatch->Invoke
(
    DISPID_UNKNOWN,              // ID vlastnosti zadáno
                                 // v DispParams
    IID_NULL,                    // Rezervováno
    LOCALE_SYSTEM_DEFAULT,       // Jazyk implicitní pro daný
                                 // systém
    DISPATCH_PROPERTYPUT,        // Nastavení vlastnosti
    &DispParams,                 // Seznam parametrů
    NULL,                        // Výsledek není zajímavý
    NULL,                        // Výjimky a označení
    NULL                         // chybného argumentu také ne
);
```

Načtení vlastnosti Server **může vypadat např. takto:**

```
VARIANT Res;
VariantInit(&Res);               // Vynulování Res

lpDispatch->Invoke
(
    DID_SERVER,                  // ID vlastnosti
    IID_NULL,                    // Rezervováno
    LOCALE_SYSTEM_DEFAULT,       // Jazyk implicitní pro daný
                                 // systém
    DISPATCH_PROPERTYGET,        // Načtení vlastnosti
    NULL,                        // Parametry nejsou
    &Res,                        // Ukazatel na výsledek
    NULL,                        // Výjimky a označení chybného
    NULL                         // argumentu nejsou zajímavé
);
```

```
BSTR Server = Res.bstrVal;
.
.
.
SysFreeString(Server);
```

Metody komponenty "WinBase602 PohledX"

ActiveX komponenta **WinBase602 PohledX** neposkytuje žádné uživatelské rozhraní, metody komponenty slouží k napojení ovládání pohledu na uživatelské rozhraní cílové aplikace. K dispozici jsou tyto skupiny metod:

Pohyb mezi záznamy a složkami

FirstRec	Na první záznam
LastRec	Na poslední záznam
NextRec	Na další záznam
PrevRec	Na předchozí záznam
NextPage	Na další stránku
PrevPage	Na předchozí stránku
FirstItem	Na první složku v záznamu
LastItem	Na poslední složku v záznamu
NextTab	Na další složku v záznamu
PrevTab	Na předchozí složku v záznamu
UpItem	O složku výš
DownItem	O složku níž
GetCurItem	Vrátí identifikátor aktuální složky
GetCurPos	Vrátí číslo aktuálního záznamu
SetCurPos	Přejde do zadané složky v zadaném záznamu

Dotazy a uspořádání

OpenQBE	Přepne formulář do režimu zadávání QBE dotazu
OpenSort	Přepne formulář do režimu zadávání uspořádání
AcceptQuery	Akceptuje QBE dotaz nebo zadané uspořádání
CancelQuery	Zruší QBE dotaz nebo uspořádání
QBEState	Vrátí indikaci režimu zadávání QBE nebo uspořádání

Editace složky

Cut	Vystřihne vybranou část editované položky a uloží ji do clipboardu
Copy	Zkopíruje vybranou část editované položky do clipboardu

Paste	Vlepí do aktuální položky obsah clipboardu

Editace v textovém okně

EditFind	Najde výskyt řetězce v textu
EditFormat	Přeformátuje text
EditRefind	Najde další výskyt řetězce v textu
EditReplace	Otevře dialog pro náhradu řetězce
EditSave	Uloží editovaný text

Vkládání a rušení záznamů

Insert	Vloží nový záznam
DelAsk	Smaže všechny záznamy
DelRec	Smaže aktuální záznam
UnbindDel	Odváže zrušené záznamy

Čtení a zápis hodnoty složky

GetItemValue	Vrátí hodnotu sloupce, do kterého vede zadaná složka formuláře
SetItemValue	Zapíše zadanou hodnotu do složky ve formuláři i do příslušného sloupce

Synchronizace změn ve formuláři a v databázi

Reset	Uvede do souladu obsah formuláře s obsahem databáze
Commit	Zapíše změny provedené ve formuláři do databáze
RollBack	Zruší změny provedené ve formuláři

Tisky

Print	Vytiskne sestavu na tiskárnu
PrintOpt	Nastaví parametry generované sestavy
PrintOptDlg	Otevře dialog pro nastavení parametrů generované sestavy
PrintSel	Umožňuje programově zvolit tiskárnu a její parametry
PrintSelDlg	Otevře dialog pro volbu tiskárny

Pomocná okna

MultiAttrs	Otevře okno s hodnotami multiatributu
EditText	Otevře editor pro textovou složku
OpenPicture	Otevře okno s obrázkem
Help	Otevře okno s lokální nápovědou
Locks	Otevře dialog s nabídkou zamykání záznamů

Ostatní

ExecStatements	Provede posloupnost příkazů vnitřního programovacího jazyka

`PickApplication` Spustí aplikaci, které patří aktuální formulář, jako samostatného klienta **WinBase602** v samostatném hlavním okně

Volání metod komponenty "WinBase602 PohledX"

V prostředích, kde jsou **ActiveX** (OCX) komponenty součástí nabídky konstrukčních prvků, nebo ve scriptech jazyka HTML se metody volají obdobně jako kterékoli jiné procedury příslušného jazyka. Volání metody **`GetItemValue`** pro načtení hodnoty aktuální složky může ve ***Visual Basicu*** nebo ve ***VBScriptu*** vypadat takto:

```
sub OnClick()
    Val = Formular.GetItemValue()
    Alert(Val)
end sub
```

V ostatních jazycích se stejně jako v případě vlastností používá metoda **`Invoke`** z rozhraní **`IDispatch`**. Následující příklad znázorňuje volání **`GetItemValue`** pro načtení hodnoty složky `IDC_JMENO` v záznamu `CurPos` v jazyce C++:

```
VARIANT Arg[2];
VARIANT Res;

DISPPARAMS DispParams =
{
    Arg,                          // Ukazatel na seznam
                                  // parametrů
    NULL,                         // Ukazatel na pole ID
                                  // parametrů
    2,                            // 2 parametry
    2                             // 2 pojmenované parametry
};

VariantInit(Arg);                 // Vynulování Var
VariantInit(Arg + 1);
                                  // Argumenty se ukládají
                                  // v obráceném pořadí
Arg[0].vt   = VT_I4;              // Pozice
Arg[0].lVal = CurPos;
Arg[1].vt   = VT_I4;              // Číslo složky
Arg[1].lVal = IDC_JMENO;

lpDispatch->Invoke
(
    DID_GETITEMVAL,               // ID metody
    IID_NULL,                     // Rezervovano
    LOCALE_SYSTEM_DEFAULT,        // Jazyk implicitní pro daný
                                  // systém
```

```
    DISPATCH_METHOD,              // Volání metody
    &DispParams,                  // Seznam parametrů
    &Res,                         // Ukazatel na výsledek
    NULL,                         // Výjimky a označení chybného
    NULL                          // argumentu nejsou zajímavé
);

BSTR Value = Res.bstrVal;
.
.
.
```

Události komponenty “WinBase602 PohledX“

Prostřednictvím událostí informují ActiveX objekty klientskou aplikaci o svém stavu. Klientská aplikace může, ale nemusí na události reagovat. Komponenta **WinBase602 PohledX** poskytuje tyto události:

OnCreate	Formulář byl otevřen
OnClose	Formulář byl uzavřen
OnNextRecord	Uživatel přešel na nový záznam
OnItemChanged	Uživatel změnil hodnotu složka
OnResetRecord	Záznam se překresluje
OnReset	Celý formulář se překresluje
OnSubcursor	Formulář přechází k subkurzoru QBE
OnSupercursor	Formulář se vrací k superkurzoru
OnOpenQuery	Uživatel přešel do formuláře pro zadání QBE nebo uspořádání
OnCloseQuery	Uživatel uzavřel formulář pro zadání QBE nebo uspořádání
OnOpenEdit	Uživatel otevřel okno pro editaci sloupce typu Text
OnCloseEdit	Uživatel uzavřel okno pro editaci sloupce typu Text
OnOpenPict	Uživatel otevřel okno pro zobrazení obrázku
OnClosePict	Uživatel uzavřel okno pro zobrazení obrázku

Ošetření událostí

Má-li uživatelská aplikace reagovat na některou událost, je třeba vytvořit obslužnou rutinu a napojit ji na příslušnou událost. V ***Microsoft Visual Basicu*** nebo ***Borland Delphi*** se o napojení postará automaticky vývojové prostředí. Pro zvolený objekt nabídne seznam událostí, které jsou k dispozici a po té, co si vývojář nějakou vybere, vygeneruje vývojové prostředí rámec obslužné rutiny, do kterého stačí doplnit pouze vlastní příkazy.

V HTML se obslužná rutina napojí na událost pomocí parametru `FOR` a `EVENT` v prvku `<SCRIPT>`. Například ošetření události **`OnNextRecord`** (výpis čísla záznamu (interního i externího) na status bar browseru) může v JavaScriptu vypadat například takto:

```
<OBJECT
CLASSID="clsid:32F34910-9288-101B-96B8-04021C007002"
ALIGN=CENTER WIDTH=275 HEIGHT=156 BORDER=1 HSPACE=5
ID=Formular>

<SCRIPT FOR=Formular EVENT="OnNextRecord(iRec, eRec, Subf)"
LANGUAGE="JavaScript">
  window.status = iRec.toString(10) + "/" + eRec.toString(10);
</SCRIPT>
```

Ve VBScriptu lze vedle této použít také syntaxi:

```
<SCRIPT LANGUAGE="VBScript">
  Sub Formular_OnNextRecord(iRec, eRec, Subf)
       window.Status  = Cstr(iRec) + "/" + Cstr(eRec)
  end sub
</SCRIPT>
```

V C++ je třeba obslužné rutiny událostí „zabalit“ do rozhraní IDispatch. V inicializační části aplikace se po aktivaci komponenty najde pomocí metody `IconnectionPointContainer::FindConnectionPoint` její styčný bod a tomu se předá ukazatel na rozhraní IDispatch. Události komponenty **WinBase602 PohledX** mají ID = `{32F34913-9288-101B-96B8-04021C007002}`. Inicializace obsluhy událostí může vypadat např. takto:

```
DEFINE_GUID(WBFORM_EVENT_ID, 0x32F34913L, 0x9288, 0x101B, 0x96,
            0xB8, 0x04, 0x02, 0x1C, 0x00, 0x70, 0x02);

void CWBFormItem :: InitEventHandler()
{
    LPCONNECTIONPOINTCONTAINER lpConPtCont = NULL;

    if (m_lpObject->QueryInterface(IID_IConnectionPointContainer,
        (LPVOID *)&lpConPtCont) != NOERROR)
        return;
    if (lpConPtCont->FindConnectionPoint(WBFORM_EVENT_ID,
        &m_EVConPt) == NOERROR)
        m_EVConPt->Advise((LPDISPATCH)this, &m_EVCookie);
    if (lpConPtCont)
        lpConPtCont->Release();
}
```

Komponenta pak vyvolává jednotlivé události prostřednictvím metody `IDispatch::Invoke`.

```
STDMETHODIMP CWBFormItem :: Invoke(DISPID dispid, REFIID riid,
    LCID lcid, WORD Flags, DISPPARAMS *Params, VARIANT *pRes,
    EXCEPINFO *pexcepinfo, UINT *puArgErr)
{
    if (Flags == DISPATCH_METHOD)
    {
        switch (dispid)
        {
        case DID_ONNEXTRECORD:
            .
            .
            .
            break;
       case DID_ONITEMCHANGED:
            .
            .
            .
            break;
       .
       .
       .
    }
    return(NOERROR);
}
```

V rámci uvolňování komponenty je třeba styčný bod uvolnit:

```
void CWBFormItem :: ReleaseEventHandler()
{
    if (m_EVConPt)
    {
        if (m_EVCookie)
            m_EVConPt->Unadvise(m_EVCookie);
        m_EVConPt->Release();
    }
}
```