Delphi – podstata, koncepce a metody MDI aplikace

Bc. Tomáš Selucký, Ústav statistiky a operačního výzkumu, Provozně ekonomická fakulta, Mendelova zemědělská a lesnická univerzita v Brně, selucky@selucky.com

Abstrakt

Příspěvek se zabývá MDI aplikacemi, které představují řešení přístupu jedné instance programu k více dokumentům současně. Práce ve své první části vymezuje podstatu a koncepci MDI aplikací, v další je potom kladen velký důraz na obecnou ukázku postupu její tvorby v rámci vývojového prostředí Delphi, a to včetně naznačení algoritmicky řešených problémů, které jsou podrobeny rozboru.

Klíčová slova

Delphi, MDI aplikace, formulář (okno), procedura, komponenta.

Abstract

This contribution deals with MDI applications, which represent a solution of one programme instance access to more documents in the same time. There are a essence and a concept of MDI applications definitions in the first part, the following one is focused on general example of its creating procedure in Delphi develop environment, including algorithmically solved problems which are analysed in detail.

Key Words

Delphi, MDI application, form, procedure, component.

Úvod

Při tvorbě aplikací ve vývojovém prostředí Borland Delphi 2005 Developer Studio for Windows (dále jen Delphi 2005) lze v zásadě využít několika formulářových stylů[1]. Prvním z nich jsou tzv. **SDI aplikace** (single document interface – rozhraní pro práci s jedním dokumentem), jejichž podstatou je, jak překlad napovídá, práce v určitém časovém okamžiku pouze s jedním dokumentem[2]. Druhým z nich jsou pak tzv. **MDI aplikace** (multiple document interface), neboli rozhraní pro práci s dokumenty více[3], které lze s výhodou použít, je-li takových prvků v aplikaci třeba.

MDI aplikace potom zcela logicky musí obsahovat několik důležitých částí, z nichž žádnou při tvorbě nelze opomenout. Jde o jeden MDI container – hlavní formulář, a dále pak o jedno nebo více MDI child – klientských formulářů (oken). Další prvky jsou v aplikaci samozřejmě volitelné. Postup samotné tvorby není nijak složitou procedurou, nicméně je třeba dbát určitých odlišností, které jsou typické pro aplikace jiného typu[4].

Cíl a metodika

Cílem práce je objasnění problematiky MDI aplikací se zaměřením na jejich podstatu, koncepci, nutné součásti a často používané metody.

První část práce se proto zaměřuje na krátké seznámení s MDI aplikacemi a obecné vymezení jejich prvků. Následující část se poté snaží uvést stručný postup tvorby takové aplikace a ošetření vybraných událostí jednotlivých komponent, včetně algoritmických řešení souvisejících problémů.

Při práci byla využita personální edice poslední verze vývojového nástroje Delphi 2005, přičemž veškeré postupy a uvedené metody v něm byly také ověřeny.

Výsledky

Podstata a koncepce MDI aplikace

Jak již vyplývá z předcházejícího textu, MDI aplikaci budeme vytvářet v případě, je-li třeba v jednom časovém okamžiku pracovat v rámci jedné instance aplikace současně s více dokumenty najednou. Nemusí se nutně jednat o práci s dokumenty textovými, na kterých si

---

[1] Článek se omezuje pouze na některé možnosti vlastnosti FormStyle, a to hodnoty fsNormal, fsMDIForm a fsMDIChild.
[2] Příkladem může být třeba Poznámkový blok.
[3] Jako příklad může být uveden MS Word.
[4] Typicky SDI, které je reprezentováno hodnotou fsNormal vlastnosti FormStyle.

danou aplikaci lze nejjednodušeji představit, ale o situaci, kdy je třeba v programu zobrazovat více klientských oken (formulářů).

Koncepce aplikace je potom postavena na třech základních prvcích. Jde o:

- **Hlavní formulář:** nadřazený všem ostatním, hodnota vlastnosti *FormStyle* bude nastavena na *fsMDIForm*. Pro přehlednost bychom jej měli přejmenovat, aby již z názvu byly jasné vztahy nadřízenosti a podřízenosti, například na *frmMain*.
- **Klientská okna:** podřízená hlavnímu formuláři, na ostatních klientech nezávislá, hodnota vlastnosti *FormStyle* bude nastavena na *fsMDIChild*. Daná okna můžeme opět pro přehlednost pojmenovat například *frmChild*.
- Hlavní menu, popřípadě jiné komponenty, které budou sloužit k ovládání aplikace (tlačítka, apod.).

Postup tvorby

Při tvorbě aplikace typu MDI se můžeme opřít o nápovědu příslušného vývojového prostředí [1], případně o dokumenty ve webovém archivu o programování, jehož autorem je Zarko Gajic [2], [3], [4]. Určitou modifikací zmíněných je následující:

1. vytvoření hlavního formuláře aplikace (lze jej nazývat „parent – rodič“),
2. vytvoření klientského formuláře (nazýván jako „child – potomek, klient“),
3. přiřazení klienta do programových jednotek hlavního formuláře,
4. použití správných příkazů k ošetření události k uzavírání formulářů,
5. vytvoření ovládacích prvků v aplikaci (hlavní menu, tlačítka, apod.),
6. ošetření událostí příslušných ovládacích prvků.

Posledním bodem je po provedení předcházejících pochopitelně odladění případných nesrovnalostí, kompilace a následné spuštění vytvořené aplikace.

Rozbor postupu tvorby

První tři body tvorby MDI aplikace lze shrnout následovně. Po spuštění Delphi 2005 postupujeme v hlavním menu například tímto způsobem: **File → New → VCL Form Applications – Delphi for Win32**[5]. Tím vytvoříme základ celé aplikace. Automaticky bude totiž vytvořen první formulář, který budeme považovat za hlavní. V nástroji *Object Inspector* nastavíme vlastnost *FormStyle* na hodnotu *fsMDIForm*. Tamtéž provedeme změnu hodnot

[5] *Tento postup samozřejmě záleží na konkrétním prostředí, ve kterém hodláme aplikaci tvořit. Delphi 2005, které lze nazvat komplexním řešením programování pro Windows, nám nabízí více možností.*

u vlastnosti *Name* a celý projekt uložíme[6] (**File → Save Project As…**). Po uložení aplikace postupujeme v hlavním menu takto: **File → New → Form – Delphi for Win32**. Tím vytvoříme další formulář, který musíme nastavit jako „potomka" – klienta. To provedeme opět změnou hodnoty u vlastnosti *FormStyle*, a to na *fsMDIChild*, přejmenujeme jej ve vlastnosti *Name* a příslušnou jednotku uložíme (**File → Save As…**). V souvislosti s klientským formulářem musí docházet k jeho „ruční" tvorbě, neboli v hlavním menu **Project → Options…** je nutné jej přesunout z *Auto-create Forms* do *Available Forms*. Pro zajištění vazby mezi klientským a hlavním formulářem je též nutné v seznamu programových jednotek – začíná klíčovým slovem **uses** – přidat název souboru klienta.

Nyní je dále třeba deklarovat proceduru, kterou se bude nové okno klientského formuláře otevírat. Ještě než se však dostaneme k samotnému obsahu procedury (Příklad 2), je třeba ji uvést v seznamu procedur v části **private** (Příklad 1).

Jednoduchou procedurou (Příklad 2) poté zajistíme, že bude konstruktorem „ručně" vytvořen formulář a bude nastaven jeho požadovaný titulek (nápis se zobrazí v titulkovém pruhu okna – vlastnost *Caption*).

Příklad 1

```
private
  procedure CreateChildForm(const child_name: string);
```

Příklad 2

```
procedure TfrmMain.CreateChildForm(const child_name: string);
  var
    child_form: TfrmChild;
  begin
    child_form := TfrmChild.Create(Application);
    child_form.Caption := child_name;
  end;
```

Po provedení těchto operací je nutné dále ošetřit události při zavření formuláře. V nástroji *Object Inspector* se přepneme do části *Events* (události) a zde zvolíme položku *OnClose*.

Celou proceduru včetně textu, který je nutné dopsat, nám ukazuje následující příklad (Příklad 3).

---

[6] *Nejprve ukládáme hlavní soubory aplikace (formáty BDSPROJ a bez dotazu DPR), poté jednotky (PAS). U jednotek se snažíme volit takové jméno, z něhož bude jasné, o kterou jednotku (a související formulář) se jedná. Například následujícím způsobem* `[nazev_aplikace]_[nazev_formulare].pas`.

Příklad 3

```
procedure TfrmMain.FormClose(Sender: TObject; var Action: TCloseAction);
  begin
    Action := caFree;
  end;

procedure TfrmChild.FormClose(Sender: TObject; var Action: TCloseAction);
  begin
    Action := caFree;
  end;
```

Příslušné akci (*Action*) při události zavírání hlavního i klientského formuláře zde přiřazujeme hodnotu *caFree*, a to proto, aby při uzavření formuláře došlo k uvolnění veškeré paměti, která je alokována pro příslušný formulář.

Nyní již máme vytvořen precizní základ MDI aplikace a můžeme se vrhnout na ovládací prvky. Je pochopitelně na vývojáři, kterých komponent k tomu využije. Obvykle půjde o hlavní menu (komponenta *TMainMenu*) nebo o příslušnou formu tlačítek (komponenty *TButton*, *TBitBtn*). Příklady v této souvislosti jsou uvedeny na variantě použití hlavního menu. Vložíme tedy komponentu *TMainMenu* na hlavní formulář a v jeho návrháři přidáváme požadované komponenty *TMenuItem*. V rámci událostí *OnClick* (při kliknutí na prvek menu) ošetříme tyto záležitosti:

- vytvoření nového klientského okna,
- zavření klientského okna,
- zavření všech spuštěních klientských oken.

První z nich provedeme způsobem, který udává následující příklad (Příklad 4). Jde pouze o použití procedury, kterou jsme již dříve vypisovali v sekci *private*. Vlastnost (funkce) *MDIChildCount* vrací hodnotu, kterou je počet klientských oken pod správou hlavního formuláře.

Příklad 4

```
procedure TfrmMain.menu_new_childClick(Sender: TObject);
  begin
    CreateChildForm('Child' + IntToStr(frmMain.MDIChildCount + 1));
  end;
```

Druhá z nich je poměrně složitější, neboť je nejprve nutné zjistit, zda nějaká klientská okna existují, a pokud ano, musíme nechat proběhnout počítaný cyklus, při němž se ověří, které z klientských oken je aktivní – právě to totiž chceme zavřít. Vše nám podrobně ukazuje další příklad (Příklad 5).

Příklad 5

```
procedure TfrmMain.menu_close_childClick(Sender: TObject);
  var
    i: byte;
  begin
    if (frmMain.MDIChildCount > 0)
      then
        for i := 0 to (frmMain.MDIChildCount - 1) do
          if (frmMain.MDIChildren[i].Active = true)
            then
              begin
                frmMain.MDIChildren[i].Close;
                Exit;
              end;
  end;
```

Poslední uvedenou záležitostí je ošetření uzavření všech klientských oken, které se v aplikaci vyskytují. Opět nejprve musíme ověřit, zda taková okna existují, a poté příkazem počítaného cyklu zajistit, aby došlo k jejich postupnému uzavírání. Celá procedura je uvedena v následujícím příkladu (Příklad 6).

Příklad 6

```
procedure TfrmMain.menu_closeall_childClick(Sender: TObject);
  var
    i: byte;
  begin
    if (frmMain.MDIChildCount > 0)
      then
        for i := 0 to (frmMain.MDIChildCount - 1) do
          frmMain.MDIChildren[i].Close;
  end;
```

Tímto posledním krokem jsme v podstatě vyčerpali postup uvedený v předchozí kapitole a můžeme začít naši aplikaci odlaďovat a kompilovat. Pokud vše proběhne v pořádku, tzn. všechny procedury a události fungují, začínáme s touto kostrou pracovat dále a můžeme se konečně věnovat obsahu samotné aplikace.

Závěr

Užití MDI aplikace, jejíž podstatou je možnost práce s více dokumenty v určitém časovém okamžiku, je při vývoji softwaru velmi zajímavou možností.

Při řešení určitých algoritmických problémů se totiž bezesporu dá tvrdit, že jiné řešení by bylo příliš složité, a nebo nemožné. V těchto případech lze použít koncepci MDI aplikace, která se skládá z hlavního formuláře, klientských oken (formulářů) a ovládacích prvků.

Postup tvorby probíhá v několik krocích, a to od návrhové části při vytváření samotných formulářů, až po vkládání událostí jednotlivým položkám hlavního menu.

Samotné metody, které je nutné ošetřovat, nejsou příliš algoritmicky složité, nicméně je třeba si uvědomit posloupnost a vztah jednotlivých částí a komponent.

Literatura

[1] Borland, Delphi 2005 Developer Studio for Windows: *Building a VCL Forms MDI Application Without Using a Wizard* [HDD]. c2004, [citováno 2005-09-21]. Dostupné z: ⟨ms-help://borland.bds3/bds3win32tasks/html/Win32_BuildMDIWithoutWizard.htm⟩.

[2] Gajic, Z..: *Your First MDI Delphi Project* [online]. c2005, [citováno 2005-09-21]. Dostupné z: ⟨http://delphi.about.com/od/beginners/1/aa031103a.htm⟩.

[3] Gajic, Z.: *MDI Development in Delphi I.* [online]. c2005, [citováno 2005-09-21]. Dostupné z: ⟨http://delphi.about.com/od/objectpascalide/1/aa042500a.htm⟩.

[4] Gajic, Z.: *MDI Development in Delphi II.* [online]. c2005, [citováno 2005-09-21]. Dostupné z: ⟨http://delphi.about.com/od/objectpascalide/1/aa050200a.htm⟩.