Delphi – nastavení formulářů

Úvod

Formulář je základní komponentou aplikace a nastavení jeho vlastností má podstatný vliv na její celkový vzhled. Je proto velmi vhodné ihned po jeho vložení a přidání nové jednotky, která je s formulářem svázána, provést v jeho implicitním nastavení dle potřeby několik změn.

Tyto změny lze sice ve většině případů provádět i za běhu samotné aplikace, nicméně v době návrhu je mnohem snazší využít nástroje **Inspektor objektů** (dále *Object Inspector*).

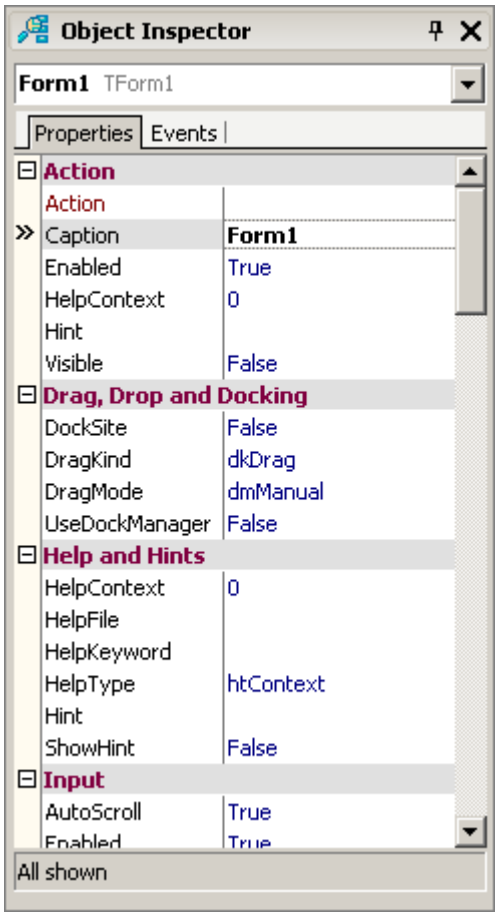
Tvorba i ukázky aplikací jsou prováděny ve vývojovém prostředí **Delphi 2005 Developer Studio for Windows** (dále jen Delphi 2005).

Object Inspector a nastavení vlastností formuláře

V rámci tvorby aplikací je velmi pohodlné nenastavovat veškeré vlastnosti použitých objektů pouze samotnými příkazy, ale lze k tomuto účelu využít v Delphi 2005 implementovaných pomocníků. Takovým nástrojem je i *Object Inspector*.

Ten je ukázán na následujícím obrázku (*Obr. 1*).

Obr. 1: *Object Inspector pro formulář Form1*



Zdroj: *Prostředí Borland Delphi 2005 Developer Studio for Windows (Print Screen).*
Poznámka: *V horní části je vždy jmenován objekt, kterého se vlastnosti týkají.*

Jak je vidět, skládá se ze dvou základních částí: vlastností (*Properties*) a událostí (*Events*), přičemž v obou oblastech jsou vždy v levém sloupci uvedeny vlastnosti nebo události a vpravo jejich hodnoty.

Po tomto krátkém exkurzu se již můžeme podívat na vlastnosti formulářů, které bychom neměli nechávat implicitně nastavené, dále pak na varietu jejich možností. Na úvod je vhodné zapamatovat si číslo **deset**,

neboť je to právě deset vlastností, kterými bychom se měli zabývat. Vlastnosti jsou uvedeny tak, jak je postupně naleznete v nástroji *Object Inspector* (vlastnost *Name* je uvedena jako poslední).

Caption (titulek – v částech Action, Localizable, Visual)

Užívá se pro specifikaci textu titulku formuláře. Nejlépe tuto vlastnost ilustrujme pomocí dvou následujících obrázků (*Obr. 2*, *Obr. 3*). Standardně je tato vlastnost nastavena na jméno formuláře (*Form1*) (*Obr. 2*), my provedeme změnu na *Manažer* (*Obr. 3*).

Obr. 2: *Nastavení **Object Inspector → Caption: Form1***

***Zdroj:** Prostředí Borland Delphi 2005 Developer Studio for Windows (Print Screen).*
***Poznámka:** Ikona není nastavena v menu Project → Options, ani v Object Inspector → Icon.*

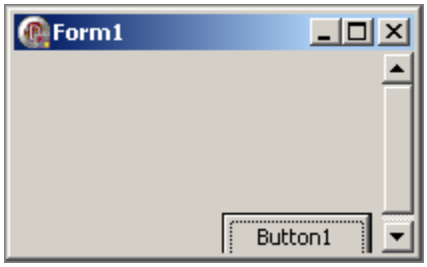
Obr. 3: *Nastavení **Object Inspector → Caption: Manažer***

***Zdroj:** Prostředí Borland Delphi 2005 Developer Studio for Windows (Print Screen).*
***Poznámka:** Nastaví-li se ikona v menu Project → Options, netřeba ji nastavovat v Object Inspector → Icon.*

AutoScroll (automatické zobrazení rolovacích lišt – v částech Input, Layout)

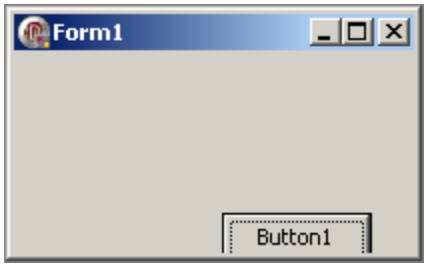
Pokud je tato vlastnost nastavena na *true*, pak, je-li to třeba, objevují se na formuláři rolovací lišty (*Scroll Bars*), tzn. sahá-li některá z ostatních komponent mimo klientskou část formuláře, pak je možné rolovat až do jejího úplného zobrazení. Více nám ukazují následující obrázky (*Obr. 4*, *Obr. 5*).

Obr. 4: *Nastavení **Object Inspector → AutoScroll: true***



***Zdroj:** Prostředí Borland Delphi 2005 Developer Studio for Windows (Print Screen).*
***Poznámka:** O zobrazení všech komponent se stará aplikace sama rolovacími lištami.*

Obr. 5: *Nastavení **Object Inspector → AutoScroll: false***



***Zdroj:** Prostředí Borland Delphi 2005 Developer Studio for Windows (Print Screen).*
***Poznámka:** Automatické rolování je vypnuto; nevíme, zda jsou všechny komponenty zobrazeny.*

Polemika nad tím, které nastavení je výhodnější, se přesouvá do jiné roviny. V určitých situacích je velmi výhodné nechat aplikaci rolovat samotnou, nicméně náhodné zobrazení rolovacích lišt, které naprosto poruší kompaktní design celé aplikace, také není nejlepším řešením.

Jako ideální se potom zdá automatické zobrazení rolovacích lišt vypnout a jinými metodami zajistit zobrazení komponent, které zobrazeny být mají. Pokud je však zcela zřejmé, že vizuální prvky aplikace se na formulář současně umístit nedají, je nutné tuto vlastnost ponechat v implicitním nastavení, tedy na hodnotě *true*.

ClientHeight, ClientWidth (výška a šířka klientské části formuláře – v částech Localizable, Visual)

Jde o velmi důležitou vlastnost, která ovlivňuje velikost samotné aplikace. Není však možné tuto dvojici vlastností zaměňovat s vlastnostmi *Height* a *Width*. V této situaci jde o **velikost klientské části formuláře**, nikoliv o velikost aplikace celé, i když jde o jeden z rozhodujících prvků pro její velikost. Jelikož nám však většinou jde o zobrazení všech vizuálních komponent na formuláři, je proto třeba nastavovat vlastnosti *ClientHeight* a *ClientWidth*.

Je důležité také dodat, že velikost celé aplikace záleží buď na vlastnostech *Height* a *Width*, nebo na kombinaci vlastností *ClientHeight*, *ClientWidth* a *BorderStyle*. Veškeré rozměry jsou zadávány v pixelech.

Font (nastavení vlastností písma – v částech Localizable, Visual)

Jedná se o soubor vlastností, kterými lze nastavit znakovou sadu písma, jeho barvu, velikost, styl, apod. Vzhledem k tomu, že veškeré komponenty, které jsou na formuláři umístěny, dědí implicitně tento soubor vlastností, je vhodné ho volit co nejuniverzálněji (tzn. pro co největší počet komponent beze změny).

Kromě barvy, velikosti a stylu však není třeba parametry měnit. Velikost je uvedena v pixelech se záporným znaménkem.

Icon (specifikace ikony formuláře – v částech Localizable, Visual)

Ikonu, která bude zobrazena přímo na určitém formuláři, je možné měnit pomocí vlastnosti *Icon*. Zde stačí pouze zapsat cestu k ikoně, kterou chcete použít, nebo pomocí průvodce vyhledat ikonu ve složkách. Pokud ikonu nenastavíte, bude použita ikona implementovaná v samotných Delphi 2005.

Pokud bude nastavena alespoň ikona v menu **Project → Options...**, pak se tato stává ikonou implicitní, tzn. bude použita ve formulářích automaticky.

Pro přehlednost lze také doporučit přesunout ikonu do složky, kde se nacházejí ostatní soubory použité při tvorbě aplikace, a pojmenovat ji jako jednotku, v jejímž formuláři je použita, nebo jako hlavní soubor aplikace.

Position (pozice formuláře na obrazovce – v části Miscellaneous)

Pozice samotné aplikace je taktéž důležitou vizuální vlastností. V tomto případě můžeme volit z několika možností:

- **poDesigned** – v tomto případě budou pozice (zleva a shora) a rozměry (výška a šířka) formuláře takové, jaké je měl v době návrhu aplikace.
- **poDefault** – je opakem předcházející možnosti, neboť jak pozice, tak rozměry budou určeny operačním systémem. Při každém spuštění aplikace se formulář posunuje dolů a doprava (bez ohledu na rozlišení obrazovky).
- **poDefaultPosOnly** – formulář se zobrazí s rozměry, které byly určeny v době návrhu, avšak operační systém určí pozici na obrazovce. Při každém spuštění aplikace se formulář posunuje dolů a doprava (rozlišení obrazovky je zohledněno – formulář se v případě, že by nebyl zcela zobrazen, objeví v levém horním rohu).
- **poDefaultSizeOnly** – pozice formuláře je odvozena z doby návrhu a operační systém ovlivňuje jeho rozměry. Opět dochází k posunování formuláře směrem doprava a dolů (bez ohledu na rozlišení obrazovky).
- **poScreenCenter** – formulář si zachovává rozměry z doby návrhu a je umístěn ve středu obrazovky.
- **poDesktopCenter** – pozice formuláře je stejná jako v předchozím případě. Rozdíl spočívá v užití ve vícemonitorových aplikacích; v předchozím případě může být totiž formulář přesunut.
- **poMainFormCenter** – formulář si zachovává rozměry z doby návrhu a je umístěn na střed hlavního formuláře aplikace; je tedy logické, že vždy půjde o tzv. *sekundární* formulář.

- **poOwnerFormCenter** – jedná se o úpravu předchozí možnosti s tím rozdílem, že formulář, na jehož střed je formulář umísťován, nemusí být hlavním – záleží na nastavení vlastnosti *Owner*. Pokud tato vlastnost není specifikována, půjde o situaci z předchozí hodnoty.

Pokud tvoříte formulář menších rozměrů, který nebude roztažen přes celou obrazovku, je velmi vhodné jako hodnotu této vlastnosti nastavit **poScreenCenter**, popřípadě jej umístit na střed formuláře nadřazeného; při každém zobrazení ho totiž pak naleznete na středu obrazovky/formuláře.

Avšak i v opačném případě je této možnosti také možné využít, ale pouze při prvním zobrazení formuláře, při němž je vhodné ho přizpůsobit obrazovce a zarovnat na střed. Další zobrazení je však již výhodné ponechat v režii systémových položek.

BorderIcons (místní menu a tlačítka v titulkovém pruhu – v části Visual)

Co jsou to *BorderIcons*, nám nejlépe ukazuje následující obrázek (*Obr. 6*). Jde o soubor hodnot, která nám říkají, která tlačítka v titulkovém pruhu budou zobrazena a která nikoliv. Standardně jde o tři (celkem čtyři) ikony:

- **biSystemMenu** – je-li systémové menu nastaveno na *true*, pak je možné vyvolat místní nabídku užitím pravého tlačítka na titulkovém pruhu aplikace, též bude zobrazen *křížek* (tlačítko zcela vpravo) pro ukončení aplikace. Pokud by však hodnota byla nastavena na *false*, na hlavním panelu nebude možné vyvolat místní nabídku a **žádná** z ikon nebude zobrazena.
- **biMinimize** – po kliknutí na tlačítko dochází k minimalizování aplikace.
- **biMaximize** – po kliknutí na tlačítko dochází k roztažení aplikace na celou obrazovku.
- **biHelp** – toto tlačítko souvisí mimo jiné také s užitím vlastnosti *BorderStyle* (je rozebrána jako následující vlastnost). Pokud je např. vlastnost *BorderStyle* nastavena jako *bsDialog*, pak nedojde k zobrazení maximalizačního ani minimalizačního tlačítka, a to i přesto, že budou nastavena na hodnotu *true*. Zobrazeno bude systémové menu a tlačítko nápovědy (*Obr. 7*). Po kliknutí na něj dojde ke změně kurzoru na *šipku s otazníkem* a po kliknutí na určitou komponentu bude moci být vyvolána místní nápověda (pokud pochopitelně bude připojen soubor nápovědy a definována vlastnost *HelpContext* v komponentě). Další možností, jak zobrazit tlačítko nápovědy, je vypustit minimalizační nebo maximalizační tlačítko bez ohledu na vlastnost *BorderStyle*.

Obr. 6: *BorderIcons*

Zdroj: *Prostředí Borland Delphi 2005 Developer Studio for Windows (Print Screen).*
Poznámka: *BorderIcons = [biSystemMenu, biMinimize, biMaximize].*

Obr. 7: *BorderIcons vs. BorderStyle*

Zdroj: *Prostředí Borland Delphi 2005 Developer Studio for Windows (Print Screen).*
Poznámka: *BorderIcons = [biSystemMenu, biMinimize, biMaximize, biHelp],* ***ale*** *BorderStyle = bsDialog.*

Při volbě konkrétního systému zobrazení těchto tlačítek (ikon, menu) je třeba zvážit především samotnou povahu formuláře. V dialogu by některá tlačítka (alespoň vizuálně) překážela, v hlavním formuláři aplikace by zase snad kromě tlačítka nápovědy nemělo chybět žádné.

BorderStyle (styl okraje formuláře – v části Visual)

Jak již název napovídá, bude se jednat o styl okraje formuláře. Tato vlastnost může nabývat šesti hodnot:

- **bsDialog** – tento styl okraje formuláře se vyznačuje skutečností, že jej nelze uchopit, tudíž velikost formuláře není měnitelná tažením myši, z *BorderIcons* je možné zobrazit pouze *biSystemMenu* a *biHelp*. Tento typ se používá především u dialogových formulářů.
- **bsNone** – okraj formuláře nebude zobrazen, titulkový pruh bude taktéž chybět (nemusíme si alespoň lámat hlavu s použitím *BorderIcons* :-)). Okraj formuláře nebude uchopitelný. Používá se především pro úvodní formuláře aplikací, na nichž obvykle sledujeme proces spouštění aplikace.
- **bsSingle** – jde o standardní nicméně neuchopitelný okraj. Titulkový pruh se systémovou nabídkou a ikonami bude zcela funkční. Používáme u aplikací, u nichž existuje předpoklad, že nedosáhnou velikosti celé obrazovky.
- **bsSizeable** – implicitně nastavený, s možností změny velikosti (uchopitelný myší). Lze použít univerzálně.

- **bsToolWindow** – obdoba hodnoty *bsSingle*, ale s menší hlavičkou (titulkem).
- **bsSizeToolWin** – obdoba hodnoty *bsSizeable*, ale s menší hlavičkou (titulkem).

Name (jméno komponenty – v části Miscellaneous)

Jde o jméno komponenty (v našem případě formuláře); to nemůže být zadáno s diakritikou. V rámci aplikace též musí být dané jméno tzv. *unikátní* – neboli žádná jiná komponenta nesmí nést stejné jméno.

Obecně lze doporučit pojmenování stylem `frm[nazev_jednotky]`. Jde-li tedy o formulář hlavní, pak bude jeho jméno *frmMain*, pro formulář nápovědy použijeme *frmHelp*, atp.

Ostatní vlastnosti formuláře v nástroji Object Inspector

AutoSize

Chcete-li, aby se velikost formuláře měnila automaticky dle použitého obsahu, stačí nastavit tuto vlastnost na hodnotu *true*.

Menu

Tato vlastnost slouží k nastavení hlavního menu formuláře. Vlastnost však nemusíme zapisovat, neboť sem bude přiřazena hodnota názvu první komponenty *TMainMenu*, kterou na formulář umístíme.

HelpFile, HelpContext, Hint, ShowHint

Vlastnost *HelpFile* slouží k nastavení konkrétního souboru, který má být použit pro vyvolání nápovědy. Na něj navazuje vlastnost *HelpContext*, která udává unikátní číslo (ID) určité **stránky** souboru nápovědy, která se váže k dané komponentě (v tomto případě formuláři). Chcete-li při najetí myší nad formulář navíc zobrazit plovoucí nápovědu, pak je třeba ji nastavit ve vlastnosti *Hint* (zde se bude nalézat text, který má být zobrazen). Aby však byla plovoucí nápověda zobrazena, musí být vlastnost *ShowHint* nastavena na volbu *true*.

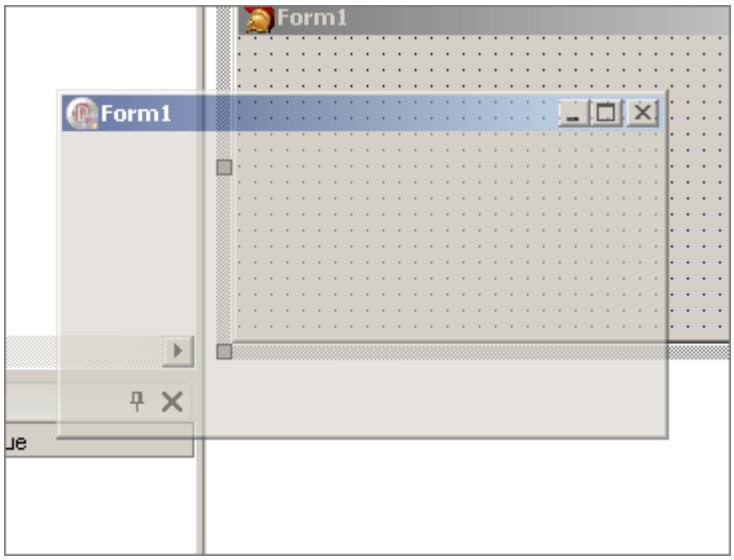
Left, Top

Chcete-li přesně nastavit vzdálenost formuláře od levého a horního okraje obrazovky, vepište hodnoty vlastností do připravených polí Object Inspectoru (zadává se v pixelech).

AlphaBlend, AlphaBlendValue

Velmi efektní možností, jak zpestřit vaši aplikaci, je použití vlastností *AlphaBlend* a *AlphaBlendValue*. První případ pouze rozhoduje o použití efektu volbou *true* a *false*, druhý je pak intenzitou průhlednosti celé aplikace.

Obr. 8: *Nastavení **Object Inspector → AlphaBlend: true, AlphaBlendValue : 100***



***Zdroj:** Prostředí Borland Delphi 2005 Developer Studio for Windows (Print Screen).*
***Poznámka:** V pozadí formuláře Delphi 2005 :-).*

Hodnota *0* znamená zcela transparentní formulář, hodnota *255* pak opak. Více na předcházejícím obrázku (*Obr. 8*).

Ukázky zdrojového kódu

Vlastnosti formuláře je také možné (kromě nastavení v nástroji *Object Inspector*) měnit přímo příkazem ve zdrojovém kódu některé z procedur, tedy za běhu aplikace. V příkladu je použita procedura, která se vykoná po kliknutí na příslušné tlačítko (událost *OnClick)*.

Aby však k takovéto změně mohlo dojít, je nutné, aby vlastnost měla atribut tzv. *za běhu změnitelné* (typicky například hodnotu vlastnosti *Name* za běhu změnit nelze; není možné měnit název něčeho, na co je někde jinde odkazováno, muselo by totiž dojít k rekompilaci).

```
procedure TForm1.Button1Click(Sender: Tobject);
        //demonstrace příkazů při kliknutí na tlačítko (událost OnClick)
  begin
    Form1.Caption := 'Změněný titulek';
        //nastavení nápisu v titulkovém pruhu
    Form1.AutoScroll := false;
        //nastavení zamezení automatického rolování
    Form1.ClientHeight := 300;
        //nastavení výšky klientské části formuláře
    Form1.ClientWidth := 500;
        //nastavení šířky klientské části formuláře
    with Form1.Font do
        //nastavení vlastností písma, příkaz with (typ record) do
      begin
        Name := 'Tahoma';
        Size := -11;
        Color := clRed;
        Style := [fsBold, fsUnderline];
      end;
    Form1.Icon := Application.Icon;
        //nastavení souboru ikony (třída TIcon)
    Form1.Position := poScreenCenter;
        //nastavení nápisu v titulkovém pruhu
    Form1.BorderIcons := [biSystemMenu];
        //nastavení lokálního menu a tlačítek v titulkovém pruhu
    Form1.BorderStyle := bsSingle;
        //nastavení okraje formuláře
  end;
```

Jak je na našem příkladu vidět, tak většinu výše zmíněných vlastností formuláře lze měnit za běhu aplikace; pokud však zapisujete zdrojový kód, je třeba vždy také zohlednit datový typ a samotnou reálnost hodnoty.

Závěr

Formulář aplikace je její výchozí a zcela zásadní komponentou. Proto je také nutné věnovat nastavení hodnot jeho vlastností zvýšenou pozornost.

Některé vlastnosti je možné pominout, jiné však nikoliv, vždy záleží na účelu a charakteru vytvářené aplikace. Pokud však chceme vytvořit graficky i funkčně kvalitní program, je nutné se touto problematikou podrobně zabývat.