Borland Delphi 5.0

Když jsem před rokem dostal do rukou verzi 4, první nadšení z řady nových možností vyprchávalo stejně rychle, jako prýštil od firmy Inprise pramen oprav této verze. Pomalu se dostává do života pravidlo sudých a lichých verzí Delphi – zatímco se sudými verzemi jsou problémy anebo nebývají zcela dotažené, liché se jeví výborné. Nechci předbíhat dobu, třeba se za pár dnů ukáže předpoklad jako neoprávněný. Zatím to vypadá, že se vývojáři firmy Inprise tentokrát velmi snažili vystříhat problémů s předchozí verzí a "číslo 5" bude tezi podporovat.

Číslo 5 žije, a má se k světu!

Krátce před příchodem verze 5 se objevil v Chipu (viz Poodhrňme roušku, Chip 8/99, str. 98.) poměrně podrobný popis jejích novinek a změn, vycházející z "beta" verze. Od verze beta k verzi oficiální se většinou už nic zásadního nezmění a stejně tak je tomu i tentokrát. A tak pro mne vznikla otázka, čemu vlastně tuto recenzi věnovat.

Lze konstatovat, že tato verze neobsahuje tolik výrazně nových věcí jako verze předchozí. Akcent byl v této verzi kladen na skutečně značné vylepšení komfortu všech programátorských prostředků s důrazem na vysokou produktivitu vývoje programu. Co považuji za nejdůležitější, je fakt, že se podstatně zlepšila stabilita produktu.

V závěru loňské recenze verze 4 jsem kritizoval to, že její autoři nevyužívají ohromný potenciál inspirace, který hlavně na internetu nabízí třetí strana – jednotliví vývojáři a firmy používající Delphi a vyvíjející vlastní komponenty, třídy, aplikace a experty. Tuto výtku mohu dnes s klidným svědomím škrtnout a konstatovat, že řada námětů byla skutečně akceptována a vtělena do VCL i IDE. Navíc je produkt distribuován s dalším CD, obsahujícím množství produktů třetích stran, od jednotlivých komponent až po celé (velmi bohaté) kolekce ve formě freewaru, sharewaru i demoproduktů.

Namísto opakování toho, co je vše nového, podívejme se na některé z novinek trochu podrobněji a pokusme se zamyslet, kam Delphi spěje.

Body přerušení

V ladicím systému je několik významných vylepšení. Pokud klepnete pravým tlačítkem na označení bodu přerušení (puntík na levém okraji editačního okna) a v kontextovém menu zvolíte "breakpoint properties", nabídne se vám okno s obdobným nastavením jako v předchozí verzi, ale tlačítkem "advandced" můžete okno (viz obr. 1) rozšířit o nastavení akcí, které chcete při přerušení provést. Můžete nastavit či potlačit přerušení při výpočtu, můžete zadat zprávu, kterou chcete vypsat do logu při průchodu bodem přerušení, a nakonec lze vypsat i hodnotu zadaného výrazu. V logu (menu View/Debug windows/Event log) lze pak vyhledat (viz obr. 2) příslušné zprávy a hodnoty.

V okně na obr. 1 lze dále seskupit body přerušení do skupin tak, že do položky “Group” zapíšeme nebo z nabídnutého seznamu vyberete název skupiny, do níž má být bod přerušení zařazen. V průběhu ladění pak mohou být skupiny zapínány a vypínány podle konkrétní situace (položky “Enable group” a “Disable group”).

Smysl je zřejmý – pokud ladíte několik oblastí problémů, můžete definovat všechny potřebné kroky a pak je podle potřeby přepínat.

Komplexní přehled o úplném nastavení všech bodů přerušení pak získáte z menu View/Debug windows/Breakpoints – viz obr. 3.

Informaci o nastavení jednotlivých bodů přerušení můžete získat nově i v editačním okně. Podobně jako se objeví hodnota proměnné během ladění při umístění kurzoru na tuto proměnnou, při umístění kurzoru na puntík označující bod přerušení se objeví podmínky platné pro tento bod přerušení (viz obr. 4).

ADO

Firma Inprise, ještě jako firma Borland, formulovala jako jeden ze svých hlavních cílů dosažení datového propojení všech typů datových bází. Vyjádřením byla koncepce BDE. Firma Microsoft se nenechala zahanbit a zformulovala – jak již to bývá jejím zvykem – svou vlastní strategii pro dosažení stejného cíle. Touto strategií je UDA (Universal Data Access), jehož cílem je – jak je formulováno na serveru firmy Microsoft – rychlý, jazykově nezávislý přístup k relačním i nerelačním datovým zdrojům. Tento teoretický základ je dnes realizován ve formě MDAC, čili UDA a MDAC jsou v obdobném vztahu jako OLE a COM. MDAC zahrnuje tři hlavní oblasti, a to ODBC (Open Database Connectivity), OLE DB a ADO (Active Data Objects). Zatímco ODBC byla zahrnuta především kvůli dodržení zpětné kompatibility, OLE DB definuje hlavně všechna potřebná COM propojení na datové zdroje a ADO, propojení směrem k uživatelské aplikaci.

Základními komponentami při realizaci aplikace s podporou ADO jsou ADOConnection a ADODataSet. Pokud umístíte první z nich na formulář, je třeba nastavit stěžejní vlastnost – ConnectionString. Ta určuje to, co jinak zajišťuje BDE – to je určení charakteristik datového zdroje a napojení na tento zdroj. Vlastnost ConnectionString je skutečně řetězec, který obsahuje řadu parametrů k tomuto napojení ve tvaru jméno=hodnota a může být opravdu dlouhý. Proto je k dispozici průvodce (viz obr. 6) , který celou situaci zjednoduší. Poklepáním na ikonu komponenty (resp. volbou z kontextového menu nebo stisknutím tlačítka – mimochodem mi to připomíná školení o operačním systému sálového počítače IBM před mnoha roky, kdy nám školitel kladl v testu otázky typu “Kolika způsoby lze provést…”) se vyvolá editor vlastnosti ConnectionString, který má na sobě tlačítko “Build” pro vyvolání zmíněného průvodce. Nejprve si zvolíte typ OLE DB providera, který určí okruh dat, s nimiž se bude pracovat. Na záložce “Connection” (viz první část obr. 7) se definují další detaily závislé na prvním kroku. Pro definici řetězce s parametry můžete opět využít tlačítko “Build” a vybrat si (v našem případě jako důsledek předchozí volby) ODBC driver (viz druhá část obr. 7). Konfigurace a nabídka obsahu dalších záložek vždy závisí na předchozích volbách. Pokud zvolíte v prvním kroku jiný OLE DB provider, budou obsah a nabídky dalších záložek jiné. Na třetí záložce lze doplnit detailní požadavky na přístup k souboru a činnost v síti. Na poslední záložce jsou pak všechny požadavky přehledně vypsány a je ještě možnost je ručně doladit. Součástí řetězce parametrů může být i implicitní adresář, kde jsou tabulky či soubory umístěny.

Tím je podstatná část práce vykonána a nyní je třeba nastavit vlastnosti komponenty ADODataSet. Jednak musíte propojit vlastnost Connection s již definovanou komponentou ADOConnection (měla by se objevit v nabídce vlastnosti), jednak je nutno nastavit dvě vlastnosti – CommandType a v závislosti na ní CommandText. Tyto dvě vlastnosti jsou velmi zajímavé, protože pomocí první z nich můžete zadat řadu možností toho, jaký bude mít vlastnost CommandText význam. Tak můžete určit, zda je jejím obsahem například jméno uložené procedury, zda se jedná o jméno tabulky databáze či o SQL příkaz. Pokud například zvolíte hodnotu vlastnosti cmdText, pak při poklepání na vlastnost CommandText se vám nabídne jednoduchý (ale praktický) průvodce pro vytvoření SQL příkazu (viz obr. 8). Nyní již stačí umístit na formulář klasické komponenty DataSource, třeba DBGrid s DBNavigator, a obvyklým způsobem je propojit (DataSource se ve vlastnosti DataSet - propojí s ADODataSet). Po nastavení vlastnosti Active komponenty ADODataSet na True tak již v době návrhu můžete vidět výsledek podobný jako na obrázku 9. Podstatné na tom je to, že výsledný program nepotřebuje mechanismus BDE.

Obě ADO komponenty mají samozřejmě ještě celou řadu dalších vlastností, a tak škála možností nastavení vytváří hodně rozsáhlou množinu. K dispozici jsou další komponenty, odpovídající komponentám Table, Query a StoredProc, což usnadňuje přechod z BDE na technologii ADO. Vzhledem k tomu, že jde o dítě z dílny Microsoftu, a navíc o potomka, který se má opravdu k světu, lze předpokládat, že technologie se bude dále rozšiřovat a využívat v budoucích operačních systémech firmy Microsoft. Výhodou bude pravděpodobně i okamžitá podpora případných novinek v dalších verzích produktů Microsoftu.

Rámy

Další z novinek v této verzi jsou rámy. Jde o jakýsi hybrid mezi komponentou a formulářem a podle toho vypadají také jeho vlastnosti. Ukažme si opět, jak lze s tímto prvkem pracovat, a z toho nám pak průběžně vyplynou případné výhody. Rám je “container class”, který může být umístěn na formuláři nebo na jiném rámu. Pro jeho vytvoření je – stejně jako pro formulář – určena speciální položka menu: File / New Frame. Dostaneme plochu podobnou formuláři, na kterou můžeme naskládat potřebné komponenty – v našem dále uvedeném příkladě třeba RadioGroup pro výběr polévky, tlačítko pro potvrzení výběru a editační pole (viz obr. 10), kam vypíšeme po stisknutí tlačítka svoji volbu.
K tlačítku proto připojíme následující proceduru k události OnClick:

```
procedure Tnabidka.Button1Click(Sender: TObject);

begin

if with RadioGroup1 do ItemIndex<>-1 then

Edit1.Text:=Items[ItemIndex];

end;
```

Rám je definován stejně jako formulář (čili vizuální složka) a je popsán v souboru s příponou DFM, v unitě jsou ostatní definice a výkonná část. Rám je možno uložit, a to třemi způsoby – pokud jej využijeme jen “v malém”, stačí jej uložit jako běžný formulář. Dalšími možnostmi je uložit jej do

repository nebo jako komponentu na paletu. Obě tyto volby jsou v kontextovém menu po poklepání na rám pravým tlačítkem. Zatímco ukládání do repository je známé, na obrázku 11 je dialog k uložení rámu na paletu. Způsob uložení má samozřejmě vliv na rozdíly při použití. Zatímco použití rámu jako komponenty vede k vytvoření nové instance třídy, umístění z repository vede k definici nové třídy (kopie nebo potomka) a závisí na formě, jakou při výběru z repository zvolíte.

Nyní můžete vytvořit formulář, na který umístíte dvě kopie nadefinovaného rámu. K tomu se využije komponenta Frames, která je na paletě na začátku záložky Standard. Jejím umístěním na formulář se vyvolá nabídka pro výběr rámu (viz obr. 12).

To platí jen pro případ, kdy rám není uložen na paletě nebo v repository – odtud můžete vybírat přímo požadovaný rám.

Na druhém rámu můžete změnit nabídku polévek třeba na nabídku hlavních jídel přímo tak, že si vyberete komponentu RadioGroup z tohoto rámu a předefinujete potřebné položky v inspektoru - objektů. Takto umístěné rámy jsou připojeny (inherited), čili pokud se vrátíte k definici rámu a uděláte v něm nějakou změnu (umístíme třeba další komponentu Label), pak po přepnutí zpět na formulář bude na obou rámech tato změna již promítnuta.

Výše popsané přepsání komponenty v rámci instance rámu se projeví v definici rámu (přepnete-li do zobrazení formuláře jako textu) například takto:

```
inline nabidka2: Tnabidka

Left = 8

Top = 176

TabOrder = 1

inherited RadioGroup1: TRadioGroup

Caption = ’hlavní jídlo:‘

Items.Strings = (

’vepřo-knedlo‘

’rizoto‘

’pizza‘

’guláš‘)

end

end
```

Původní vlastnosti jsou v této instanci rámu předefinovány (overriden). Na instanci rámu ale nelze umístit žádnou novou komponentu a pokus o odstranění komponenty z takovéto instance vede ke

zprávě na obrázku 13. Pokud jde jen o estetickou záležitost, pro komponentu v takovém případě můžete namísto toho použít nastavení vlastnosti Visible=False.

Vlastníkem komponenty na rámu není formulář, ale rám, takže pokud chcete například tlačítkem umístěným na rámu vyvolat akci s objektem na formuláři, musíte tomu příslušně přizpůsobit i příkazy.

Při doplňování akcí ke komponentám na rámu se automaticky generuje i připojené volání akcí zadaných v definici rámu. Pokud bychom nyní například chtěli v uváděném příkladě zobrazovat volbu po stisknutí tlačítka i na komponentě Label umístěné na formuláři, pak po poklepání na tlačítko na rámu se vygeneruje procedura, která již obsahuje připojené (inherited) volání deklarované výše v definici rámu:

```
procedure TForm1.nabidka1Button1Click (Sender: TObject);
begin
nabidka1.Button1Click(Sender);
Label1.Caption:=Edit1.Text;
end;
```

Z několika výše uvedených postupů je vidět rozdíl mezi použitím rámu a ostatními již známými prvky Delphi. Na konec ještě dodejme, že díky specifickým vlastnostem rámu lze šetřit zdroje a zkrátit tak délku programu.

Návrhář datových modulů

Datový modul se proměnil z běžné “plochy pro umístění DB komponent” v mini-CASE nástroj. To je samozřejmě přehnané, ale předpokládám, že se najdou tací, kteří na tuto proměnu navážou, a vbrzku se nějaký ten ad-in-průvodce do IDE s více možnostmi na toto téma objeví. Jak tedy nyní vypadá práce při návrhu datového modulu? Vyberete-li z repository datový modul, zjistíte, že plocha je rozdělena na dvě části, přičemž v levé části je zobrazena stromová struktura se strukturou komponent vložených na datový modul. Na pravé části jsou dvě záložky, přičemž první zobrazuje komponenty tak, jak tomu bylo v předchozích verzích, a druhá zobrazuje diagram struktury použitých dat. Na obrázku 14 je datový modul v základní podobě po přidání jedné tabulky, zatím bez jakékoliv další specifikace. V levé části jsou ve stromu červeně označena místa, která je třeba ještě určit, a ikonka u položek, které jsou definovány implicitně, je v matných barvách. Komponentu můžete umisťovat buď na plochu, nebo přímo formou drag & drop na strukturu stromu. Pokud ji přidáte na úroveň session, předpokládá se, že se jedná o nový alias nebo adresář, a přidá se nový uzel na úroveň aliasu. Pokud ji přidáte na úroveň aliasu, předpokládá se, že se jedná o stejný alias, a tabulka jej má již doplněný v příslušné vlastnosti – prostě návrhář se snaží být maximálně vstřícný. Na obrázku 15 je ukázáno, jak může vypadat konkrétní návrh datového modulu. Je zde zobrazena pouze záložka Data Diagram, protože záložka Components vypadá stejně jako v minulých verzích.

Vlastní vytvoření diagramu není automatické. Tabulky, s nimiž se má pracovat, se umístí na záložku přenesením drag & drop. V rámečku tabulky se objeví pouze položky, které jsou definovány field editorem. K tabulkám je možno dodefinovat a připojit komentáře a různá propojení. K tomu slouží pět tlačítek na levém okraji záložky Data Diagram (viz obr. 15). První z nich definuje vztah vlastnosti, který vzniká v případě, že vlastnost jedné komponenty odkazuje na jinou komponentu. Druhé definuje vztah master-detail. Příklad takového vztahu je na obrázku 15 a při jeho definici je vyvolán "návrhář propojení polí" (viz obr. 16), kde se specifikují detaily tohoto propojení. Třetím tlačítkem se aktivuje vztah definovaný v tabulkách jako "lookup" a poslední dvě slouží k vytvoření bloku komentáře a k propojení komentáře s tabulkou. Kromě těchto propojení se provede ještě jedno další automaticky, a to tehdy, když na záložku umístíte některý nadřízený uzel ze stromu v levé části návrháře. V tom případě se veškeré odkazy automaticky zobrazí tak, jak je vidět na obrázku 15 v odkazu na Default {Session}.

Návrhář umožňuje i tisk datového modelu, ale není to nějaká dokumentace, nýbrž pouze kopie výše zobrazené záložky Data Diagram pro tiskárnu nebo obrázek do souboru.

Jazyková podpora

Pokud budete chtít svůj program v Delphi přeložit do jiného jazyka, budete mít od této verze úlohu velmi usnadněnu. Součástí prostředí je několik nástrojů, které vám umožní vytvořit tabulku se všemi řetězci, které se v programu vyskytují jako součást vlastností typu "localizable" nebo jako součást "resources". Pokud chcete, aby byly nabídnuty k překladu všechny texty, musíte i v kódu programu striktně využít mechanismu "resourcestring". Pokud tyto konvence dodržíte, je vlastní postup jednoduchý. (Princip je obdobný tomu, který je použit v programu ing. Z. Hlinky pro lokalizaci programů Delphi – viz Chip 7/98, Chip CD 9/98.) V menu Project/Languages zvolíte "Add…". Touto volbou se vyvolá průvodce pro přidání jazyka (viz obr. 17). Za jeho pomoci si postupně zvolíte projekt, jazyk a jeho subvariantu a nakonec adresář, v němž se bude překlad realizovat. Poté průvodce provede analýzu formulářů a zdrojů ve zvoleném projektu a v zadané podknihovně založí nový projekt pro vytvoření DLL pro zvolený jazyk. Hned poté vám průvodce předloží výsledek své analýzy v tabulce, zobrazené na obrázku 18.

Zde si můžete postupně pro jednotlivé formuláře a zdroje procházet nalezené texty a provádět jejich překlad do příslušného jazyka. Po ukončení je možno provést překlad nově definované DLL knihovny (s příponou odpovídající zvolenému jazyku namísto DLL) – projekt je uložen v podknihovně, kterou jste si zvolili. Poté se můžete vrátit k původnímu projektu a z menu Project/Languages/Set active si vyberete příslušný jazyk. Po překladu program akceptuje DLL, která je přítomna ve stejném adresáři jako spouštěný program. Pokud zde tato DLL není, program běží v originálním jazyku. Proces je velmi transparentní a jednoduchý.

CD-ROM

V této verzi je poprvé distribuován s Delphi i CD, který obsahuje poměrně rozsáhlou škálu produktů třetích stran. Na internetovém serveru Inprise je již dlouho stránka s rozsáhlým, dobře utříděným seznamem firem, které nabízejí produkty související nějakým způsobem s Delphi. Několik z nich se objevilo i na zmíněném CD a je to pro tyto firmy i dobrá vizitka a dobrá reklama. Zatímco některé firmy zde nabízejí třeba jen jedinou komponentu, jsou zde i takové, které nabízejí jako freeware, lite nebo opensource i kolekce stovek komponent (LMD, ABC for Delphi) nebo balík expertů

(GExpert). Většinu těchto produktů můžete nalézt a získat i na internetu (některé i případně každý měsíc na Chip CD), ale jsou i takové, které bylo možno (alespoň v době začátku prodeje) získat pouze na tomto CD. Fakt, že se staly součástí tohoto CD, na druhé straně jistě znamená i určitou visačku kvality produktu.

Kromě CD s produktem Delphi a zmíněného CD třetích stran je ve verzi Enterprise distribuována i verze 3 produktu C++Builderu a verze 2 JBuilderu.

Závěr

Jak jsem uvedl na začátku, je v aktuální verzi Delphi řada dalších významných novinek. Uveďme jen namátkou komponenty pro přístup k InterBase bez prostřednictví BDE, celkem 32 komponent ActiveX na paletě Servers pro přístup k produktům firmy Microsoft, mnoho nových technologií v přístupu k internetu atd. Hlavní dojem pak vytváří "vyladěné" vývojové prostředí, řada nových průvodců a stabilita produktu.

Signifikantní je u této verze značný pokrok ve směru podpory pro technologie firmy Microsoft. Je to zcela pochopitelné. I přes řadu odpůrců Microsoftu, od zavilých až po občasné, nelze přehlédnout jeho trvalou a spíše sílící pozici. Microsoft dnes pokrývá bezpečně prakticky bez větších mezer celou škálu oblastí softwaru a brousí si zuby i na další obory. Totéž platí zcela beze zbytku i pro technologie na poli softwaru.

V tomto světle se jeví smlouva uzavřená nedávno mezi Inprise a Microsoftem jako jednoznačný přínos pro uživatele. Pokud pomineme ekonomické aspekty, které určitě byly i při realizaci této smlouvy dominantní, pak možná již v této verzi Delphi je náznak úrody, kterou by smlouva mohla (a měla) přinést.

Kolekci produktů firmy Inprise (Delphi společně s CBuilderem a JBuilderem) dnes tvoří vývojové nástroje, které lze zcela určitě označit jako jedny z nejlepších, nejcelistvějších a nejúplnějších na dané, tedy windowsové platformě. Snadnost tvorby programů pracujících současně s technologiemi COM, DCOM, CORBA a UDA, rychlost vytvoření spolehlivých víceúrovňových aplikací, pracujících na jednotlivém počítači nebo na jakékoliv síti, šíře palety možností použitelnosti kombinovaná se snadností tvorby, toho dnes nedosahuje žádný jiný vývojový prostředek. Rostoucí popularitu je zřetelně vidět právě například na internetu. Tolik stránek a tolik autorů souvisejících prostředků a komponent, kolik jich má Delphi, nemá jednoznačně žádný jiný vývojový prostředek. O rostoucí popularitě svědčí i skutečnost, že firma přikročila ke klíčování instalace, která navíc probíhá ve zcela novém hávu (viz obr. 19). Popularita zcela určitě ještě vzroste, pokud začne Delphi koketovat s platformou Linuxu, jak nedáno naznačily zprávičky "od Inprajsů" (nemohu si pomoci, ale stále si ještě myslím, že by "od Borlandů" znělo lépe). A tak o budoucnost snad nemusíme mít obavy.

Tím nechci tvrdit, že by se nenašla ještě spousta věcí, které bychom v Delphi rádi viděli. O to napjatější bude čekání na příští léto a na další verze (ale pozor, bude to zase verze sudá!), jaká překvapení nám přinesou…

Jiří Ventluka