Programování v objektovém prostředí Cocoa [1]

Což takhle trochu kakaa?

Tímto článkem otevíráme seriál, v němž se postupně naučíme programovat v prostředí Cocoa – v plně objektovém vývojovém systému, který zahrnuje i podporu práce s databázemi nebo tvorbu WWW aplikací. Dnes Cocoa patří firmě Apple, prostředí však vzniklo v rámci firmy NeXT a jejího operačního systému NeXTStep. Právě luxus a flexibilita tohoto vývojového prostředí byly patrně rozhodujícím faktorem, který Apple před třemi roky přiměl za bezmála půl miliardy dolarů firmu NeXT koupit.

Jestliže produkt Cocoa patří Applu, je zřejmé, že jeho přirozeným životním prostředím jsou počítače Macintosh. Ani ostatní platformy však nepřicházejí úplně zkrátka: téměř kompletní Cocoa je součástí hostujícího systému WebObjects for NT, který je na trhu již dávno. Pro uživatele Linuxu a ostatních prostředí je nadějí projekt GNUStep, jehož cílem je uvolnit kompletní prostředí odpovídající “kakau" v rámci GNU licence. Konečně stále existuje naděje, že firma Apple se rozhodne využít bezproblémovou portabilitu prostředí a bude nabízet svůj Mac OS X i pro jiné platformy – kód pro počítače Sun SPARC, Hewlett-Packard PA-RISC a IBM PC kompatibilní má k dispozici, stačilo by jej trochu oprášit...

Copak je to za novinky...

Je vhodné si uvědomit, že ačkoli technologicky je Cocoa patrně nejmodernějším z dnes užívaných vývojových prostředí, novinkou v pravém smyslu slova není: vzniklo totiž jako vývojové prostředí NeXTStepu před patnácti lety. To je jeho nesmírná výhoda, protože díky tomu jsou již jeho dětské chyby dávno vychytány. Navíc existuje po celém světě široká základna programátorů, kteří již s Cocoa – přesně řečeno s něčím velmi, velmi podobným – mají rozsáhlé zkušenosti.

Tehdy ovšem nikdo o kakau ještě nemluvil: vývojové prostředí NeXTStepu se nazývalo OpenStep. Firma Apple jej nejprve z nepříliš jasných důvodů přejmenovala na “Yellow Box", a teprve poměrně nedávno mu přidělila jméno Cocoa – je v tom vidět další hříčka na téma Java, a navíc název Cocoa hezky aliteruje se jmény alternativních prostředí, která Apple nabízí pro zachování kompatibility se starými verzemi Mac OS: Classic a Carbon.

Co nenajdete ve výloze, hledejte uvnitř!

Každé vývojové prostředí se skládá v zásadě z knihoven služeb, jež mohou využívat aplikace, které v něm vytváříme, a ze skupiny prostředků, jež nám v tom pomáhají: překladače programovacích jazyků, editory zdrojových textů, debuggery, generátory obrazovek a podobně. V tomto odstavci si zběžně ukážeme, jak na tom v tomto směru je Cocoa.

Kolik řečí znáš...

Dnes je většina vývojových prostředí postavena kolem některého konkrétního programovacího jazyka. Ne tak ovšem Cocoa – od samého začátku je toto prostředí navrženo pro možnost práce s víceméně libovolným jazykem a v současnosti je jeho standardní součástí jazyků pět:

- *Objective C* je základním jazykem, ve kterém je celý systém vytvořen. Objective C je standardní jazyk C s doplněnou podporou objektů na podobných principech, na jakých je založen SmallTalk. Díky tomu nabízí nesrovnatelně vyšší programátorský luxus pro práci s objekty než C++, nebo dokonce Java. S Objective C se v tomto seriálu seznámíme blíž a budeme jej používat pro většinu příkladů.
- Alternativním jazykem je kromě Objective C také *Java*. Pro práci s objekty nenabízí tak pohodlné služby jako Objective C a vinou částečné interpretace je také pomalejší. Na druhou stranu však má nezastupitelnou roli tam, kde požadujeme plnou přenositelnost: v Javě je možné psát kód distribuované aplikace, který poběží v rámci WWW prohlížeče klienta na libovolné platformě.
- Standardní jazyk *C* je v prostředí Cocoa také plně podporován. Jeho využití je zčásti omezeno tím, že C nepodporuje práci s objekty, takže z něj nejsou přímo přístupné objektové knihovny. Máme-li však hotový kód v C, který chceme do prostředí Cocoa přenést, není v tom žádný problém.
- Velmi podobně jako C podporuje Cocoa také jazyk *C++*. Plné využití objektových knihoven z C++ možné není (v kontrastu k rozšířenému omylu je nutné si uvědomit, že C++ *není* plnohodnotný objektový jazyk). Máme však k dispozici jeho překladač a můžeme jej využít pro přenos libovolného kódu, napsaného již dříve v C++, do prostředí Cocoa.
- Posledním ze standardně podporovaných jazyků je *WebScript*, interpretovaný objektový jazyk se syntaxí odvozenou od Objective C nebo Javy – můžeme si vybrat a podle potřeby a nálady používat tu či onu variantu. Využití WebScriptu je omezeno na internetové aplikace, pro něž je ostatně také určen tím, že je plně interpretovaný.

Za zmínku stojí samozřejmost, která v jiných prostředích zdaleka samozřejmá není: v Cocoa můžeme

bez nejmenších problémů v rámci jediné aplikace jazyky libovolně míchat – jediným omezením je, že více různých jazyků nesmíme používat v jednom zdrojovém souboru...

Knihovny služeb

Mezi rysy, jež dávají prostředí Cocoa jeho výjimečné postavení, bezpochyby patří i nesmírně luxusní knihovny služeb. S těmi se v průběhu našeho seriálu samozřejmě seznámíme podrobněji. Zde si prozatím jen v rychlosti projdeme čtyři základní skupiny knihoven, které Cocoa nabízí – jejich přehled vidíme na obr. 1:

Základem všeho je *Foundation Kit*. V něm jsou soustředěny základní služby potřebné při tvorbě libovolné aplikace, bez ohledu na konkrétní prostředí, ve kterém poběží: kontejnerové objekty, nesmírně luxusní práce s textovými řetězci s plnou podporou UNICODE i řady osmibitových kódování, přístup k souborům, komunikace mezi procesy a mnoho dalších služeb.

Enterprise Objects Framework, zkráceně *EOF*, doplňuje a rozšiřuje služby Foundation Kitu o přístup k databázím. EOF je navržen natolik flexibilně, že databáze mohou být vlastně libovolné, včetně prostých DBF nebo dokonce textových souborů; nejčastěji však spolupracuje s výkonnými SQL servery.

Připravujeme-li aplikaci, jež bude pro interakci s uživatelem využívat standardní služby operačního systému (tj. myš, klávesnici, obrazovková okna a podobně), sáhneme po knihovně *AppKit*. V ní jsou třídy reprezentující všechny standardní prvky uživatelského rozhraní, od oken nebo menu až po poslední tlačítko. Navíc je zde kompletní podpora interakce s uživatelem a plně přenositelné grafické služby, nezávislé na konkrétním Window Serveru.

Knihovna *WebObjects* naopak poslouží v případě, kdy by nám služby AppKitu byly málo platné, protože aplikace má s uživateli komunikovat prostřednictvím internetu. Místo klávesnice a obrazovky se tak vlastně stává uživatelským rozhraním aplikace WWW browser, běžící na jakékoli platformě. Podporuje-li takový prohlížeč Javu, může taková aplikace běžet zčásti i u klienta. Jinak výkonný kód běží jen na serveru a s klientem komunikuje prostřednictvím protokolu HTTP.

Samozřejmě že Cocoa obsahuje řadu dalších knihoven. Součástí Mac OS X Serveru je například knihovna AIAT obsahující služby pro fulltextový přístup k datům, kit Interceptor pro přímý přístup na obrazovku nebo knihovnu Zip se službami pro komprimaci dat. Ty jsou však relativně podružné – my se v tomto seriálu soustředíme na základní služby čtyř výše popsaných knihoven.

Za zmínku stojí ještě to, jak jsou knihovny v prostředí Cocoa reprezentovány: na rozdíl od většiny ostatních systémů zde nejde o “nějakou DLLku", ale o propracovaný mechanismus nazývaný *framework*. Součástí frameworku jsou kromě vlastních knihoven také podpůrné soubory, hlavičkové soubory pro překladač, dokumentace a podobně. Tak je všechno na jednom místě a nemůže dojít ke zmatkům. Frameworky navíc využívají propracovaný systém verzí a v případě zásadních změn je možné v jediném frameworku uložit i starší verze knihoven, aby starší aplikace nadále bez problémů pracovaly.

Aplikace a pomocné prostředky

Samotné skvělé knihovny a překladače nestačí – je třeba mít k dispozici přinejmenším dobrý editor zdrojového kódu a debugger. Nadto právě Cocoa ukazuje, jak obrovským usnadněním práce může být vizuální programování, když se udělá pořádně – v dlouhodobém průměru trvá tvorba stejné aplikace v prostředí Cocoa zhruba desetinu času oproti jiným prostředím!

O integraci celého vývojového řetězu se stará aplikace *ProjectBuilder* – sama obsahuje editor zdrojového kódu a umí spolupracovat s debuggerem, a podle potřeby sama spouští ostatní aplikace. Okno ProjectBuilderu se dvěma zdrojovými soubory v Javě a v Objective C vidíte na obr.2.

Pracujeme-li s databázemi, potřebuje EOF znát podrobnou strukturu dat a jejich vzájemných relací. K tomu slouží *EOModeler*, který je vidět na obr. 3. Strukturu dat v něm specifikujeme pomocí E-R modelů. EOF pak již sám na základě modelu vytváří a udržuje objekty reprezentující data v databázi a jejich vzájemné vazby.

Skutečnou magii, umožněnou plnohodnotným objektovým systémem, nabízí *InterfaceBuilder* – aplikace pro interaktivní tvorbu a údržbu objektových sítí. Jak jméno naznačuje, nejčastěji se takové sítě skládají z objektů uživatelského rozhraní. To ilustruje také příklad na obrázku 4a – natažením černé čáry mezi posuvníkem a textovým polem jsme právě “naprogramovali", že hodnota zobrazená v poli bude vždy odpovídat pozici posuvníku. InterfaceBuilder však není v žádném případě omezen jen na práci s grafickými objekty: na obr. 4b vidíme jiné propojení, jímž jsme určili, že databázový objekt *Studio* bude reprezentovat položku studia odpovídající zvolenému filmu v databázovém objektu *Movie*.

Podobná kouzla pro internetové aplikace dokáže *WebObjectsBuilder*. Obr. 6 ukazuje, jak jsme navázali tlačítko ve vytvářené WWW stránce na metodu *performTest*. Vedle je navíc vidět již dříve vytvořená vazba mezi proměnnou *userName* a textovým polem. Jakmile pak někdo otevře URL takovéto aplikace ve svém WWW prohlížeči, zobrazí se stránka vypadající nějak jako obr. 6 – na místě “OCSoftware" samozřejmě bude libovolný text, který byl zrovna uložen v proměnné *userName*. Jestliže pak uživatel prohlížeče klepne na tlačítko, spustí se v aplikaci metoda *performTest*...

Ovšemže i podpůrných prostředků je v systému Cocoa mnohem víc. Máme zde k dispozici například luxusní profiler pro vyhledávání “bottlenecků" v pomalých aplikacích nebo třeba aplikace pro vyhledávání a odstraňování alokované nevyužité paměti. Podobně jako u knihoven, v našem seriálu se soustředíme jen na ty základní.

Ondřej Čada