Obsah

ÚVOD

Komu je kniha určena

Tato kniha je především o *času*, kterého nikdo z nás nemá nazbyt. Možná jste manažer, který si chce firemní data pravidelně analyzovat sám, aby si byl jistý, že mu údaje předkládané oddělením controllingu nezkreslují skutečnost. Možná jste pracovník oddělení controllingu a každý měsíc do pátého pracovního dne musíte zkonsolidovat tuny dat a zaplnit standardní tabulku definovanou kdesi v mateřské firmě kýmsi, kdo nemá ponětí o databázovém uspořádání dat. Možná jste pracovník IT oddělení, expert na databáze a všechny možné dialekty SQL, ale váš šéf jednoduše vyžaduje reporty v Excelu. Jako databázový expert a zkušený programátor v SQL a ASP.NET považujete něco takového pod vaši úroveň, nicméně šéf je šéf a chce mít svou zprávu co nejdřív.

Pro vás všechny je tato kniha. Naučí vás, jak vám VBA může ušetřit čas, a rovněž vám ukáže, jak ušetřit čas při samotném psaní kódu. Když hovoříme o kódu, rozumíme tím program, v tomto případě ve VBA. V praxi je možno se často setkat s výrazem „makro". Toto slovo v této knize nebudu pro program používat, protože se jedná o slovo matoucí. Slovo „Makro" v Excelu pochází z dob, kdy Excel používal takzvaný XLM jazyk (neplést s XML). Velmi neintuitivní syntaxe byla postupně vytlačena nástupem VBA. Makro příkazy XML mají však své výhody. Ukazuje se, že díky tomu, že jsou přímo součástí jádra Excelu, vykonávají příkazy mnohem rychleji než jejich ekvivalenty v VBA. Samozřejmě, většinou mluvíme v milisekundách, ale co když příkazem zpracováváme jednotlivě 20 000 řádek? A co když chceme dávkově zpracovat 100 takových souborů čekajících na nás ve složce „KeZpracovani"?

Asi vás příliš nepřekvapím, když řeknu, že efektivní kód znamená obvykle kratší kód, čímž ušetříte i čas na samotné programování. Proto, až se naučíte psát efektivní programy, zabijete dvě mouchy jednou ranou. Ušetříte čas jak při vývoji, tak později díky využívání vaší báječné aplikace.

Kniha není primárně určena pro začátečníky v programování, neboť poměrně do hloubky probírá základy VBA a nejpoužívanější části objektového modelu Excelu. V knize je velké množství konkrétních ukázek a řešení běžných problémů, se kterými jsem se setkal ve vlastní praxi a které rovněž vedou k častým dotazům na internetových fórech. Vysvětlení, jak kód pracuje, se spíše týká použitého algoritmu a pastí, než toho, jak ta která klauzule, funkce, metoda nebo vlastnost pracuje. Budu se snažit vysvětlit rovněž netypické a netradiční použití jinak standardních metod, které jsem ve velké míře odkoukal od špičkových mágů ze světa Excelu a v menší míře na ně přišel sám.

Přestože je kniha určena především pro pokročilé uživatele Excelu 2007, většina kódu prošla testem na Excel 2007 CZ, Excel 2007 ENG, Excel 2003 ENG a většina i na Excel 2000 ENG. Operační systémy byly Windows Vista i Windows XP SP2 v anglické verzi s českým místním nastavením. Při extrémnějších pokusech s mezinárodním prostředím jsem používal Windows XP RU a rovněž

Excel v ruské lokalizaci. Pokud bude některý kód nekompatibilní s verzemi nižšími než Excel 2007, výslovně na to upozorním.

Některým oblastem jsem úmyslně věnoval menší místo, než by možná bylo v knize o VBA obvyklé. Jedná se především o uživatelské formuláře. Přestože jim v této knize věnuji celou jednu lekci, ve vhodně navržené aplikaci jsou zřídkakdy potřeba, protože list v Excelu je vlastně výkonný a flexibilní formulář, za který můžeme schovat libovolný kód. Naopak, podle mé zkušenosti neustále vyskakující okna a formuláře vyžadující reakci uživatele spíše obtěžují. Objektový model Excelu je velmi rozvětvený a s každou novou verzí se dále rozšiřuje. V současné době se rozšiřování týká zejména propojení na vnější databáze, těsnější integrace XML a podobně. Tradiční funkce Excelu se od verze 2000 příliš nemění a výraznějšími změnami prošel zejména objektový model grafiky a jejího formátování. Podstatně rozšířen ve verzi 2007 byl i podmíněný formát.

Další „opomenutou" oblastí je ovládání vestavěného menu Excelu. Za prvé, Excel 2007 menu zásadně změnil a dosavadní objektový model pro něj neplatí, čímž se vaše aplikace stává nekompatibilní směrem dolů. A za druhé, zvládnutí úplného programování nového menu předpokládá znalost jazyka XML a to už je opravdu mimo rámec této knihy.

Obecně se tato kniha drží spíše hesla „Dej člověku rybu, a nakrmíš ho na jeden den. Nauč jej rybařit, a nakrmíš jej na celý život." Je prakticky nemožné a hlavně zhola zbytečné snažit se popisovat práci se všemi objekty Excelu do detailů. Autor samozřejmě taky nezná nazpaměť všechny objekty a už vůbec ne jejich metody a vlastnosti. Vtip je v tom, umět méně používané prvky nalézt a správně a hlavně efektivně je použít ve svém kódu. A i o tom je tato kniha, zejména její šestá lekce.

Co je to VBA a jak se liší od VB

Visual Basic (VB) a Visual Basic for Applications (VBA) mají společný základ. Lze říci, že jádro jazyka Visual Basic 6.0 je součástí instalace Microsoft Office (od verze 9.0, neboli 2000) a komunikace s ním probíhá přes moduly kódu a uživatelské formuláře, které se od formulářů Visual Basic liší svou určitou omezeností, avšak pro většinu aplikací postačí.

Pokud mluvíme o VBA, nemluvíme o Excel VBA, Word VBA, nebo Access VBA. Syntaxe jazyka je pro všechny aplikace stejná, liší se pouze *objektový model* aplikace, se kterou pracujeme. Pokud se naučíte syntaxi jazyka VBA a programování aplikace Excel, je relativně jednoduché nabyté znalosti použít na jiné aplikace, které VBA rovněž používají. Jediné, co se musíte naučit, je objektový model aplikace, kterou chcete automatizovat.

Podívejte se na následující ukázky jednoduchého kódu, které vykonávají velmi podobnou činnost, totiž zapisují sekvenci čísel 1 až 10 do nové tabulky Excelu, do nového dokumentu Wordu a konečně do existující tabulky v Accessu.

Microsoft Excel:

```
'// Kód zapíše čísla 1 až 10 do oblasti A1:A10
'// listu "Sheet1" nového sešitu v Excelu

Const MAX As Long = 10
Dim i As Long
```

```
With Workbooks.Add
    For i = 1 To MAX
        Sheets(1).Cells(i, "A") = i
    Next i
End With
```

Microsoft Word:

```
'// Kód zapíše čísla 1 až 10, každé do nového odstavce,
'// do nového dokumentu ve Wordu

Const MAX As Long = 10
Dim i As Long

Documents.Add
With Application
    For i = 1 To MAX
        .Selection.TypeText Text:=CStr(i)
        .Selection.TypeParagraph
    Next i
End With
```

Microsoft Access:

```
'// Kód přidá nové záznamy do tabulky "tblMyTable" v databázi Access
'// a zapíše čísla 1 az 10 do pole "MyField"

Const MAX As Long = 10
Dim i As Long
Dim rs As DAO.Recordset
Set rs = CurrentDb.OpenRecordset("tblMyTable")

With rs
    For i = 1 To MAX
        .AddNew
        .Fields("MyField").Value = i
        .Update
    Next i
End With

rs.Close
Set rs = Nothing
```

Objektový model aplikace se skládá z objektů, jejich metod a vlastností. S objekty manipulujeme metodami a jejich charakteristiky stanovujeme a zjišťujeme přes jejich vlastnosti. V běžném jazyce řekneme:

„Natankujeme do automobilu naftu, poté ho nastartujeme, a jestli motor běží, rozjedeme se.“ V pseudokódu by tento proces vypadal takto:

```
Automobil.Natankovat Palivo:=Nafta
Automobil.Nastartovat
If Automobil.Motor.Bezi Then
```

```
....Automobil.RozjetSe
End If
```

Podobně, pokud chcete „nastartovat“ aplikaci Excel například z aplikace Word, spusťte Word, stiskněte Alt+F11, vložte nový modul, napište tento kód, umístěte kurzor kamkoli dovnitř kódu a stiskněte F5.

```
Sub NastartovatExcel()
Dim oAppXL As Object
Set oAppXL = CreateObject("Excel.Application")
If Not oAppXL Is Nothing Then
    oAppXL.Workbooks.Add
    oAppXL.ActiveWorkbook.Sheets(1).Range("A1") = "MS Word"
    oAppXL.Visible = True
End If
End Sub
```

Nevadí, jestli nevíte, jak tento kód pracuje. Vše se dozvíte v dalších částech knihy. Pro tento moment je důležité, abyste si uvědomili, že je jen jeden jazyk VBA, který vychází z jazyka Visual Basic. Co se liší, jsou objekty dané aplikace, jejich metody a vlastnosti.

VBA pomáhá šetřit čas

Toto není vymyšlený příběh.

Představte si, že jste byli jmenováni manažerem v nové firmě, která provozuje desítky obchodů, obsluhuje desetitisíce zákazníků denně a prodává kolem deseti tisíc položek. Ve firmě existuje několik informačních systémů, včetně vlastní databáze nákupů a prodejů na bázi FoxPro a účetního systému na bázi SAP. Protože jsou výstupy z těchto systémů dosti uživatelsky nepřívětivé, každé oddělení si vytvořilo své standardní tabulky v Excelu, které jsou ovšem zčásti zaplňovány a upravovány ručně, a tudíž jsou náchylné k chybovosti. Na existující zprávy se nemůžete spolehnout, protože nevíte, jaká data zahrnují a jestli jsou informace úplné. Souhrnné přehledy integrující výsledky jednotlivých oddělení neexistují. Navíc neexistuje jednotná databáze obchodů, popisující jejich vybavení, příslušnost do regionů a kvalitativní kategorizaci. Databáze prodejů neumožňuje přímo generovat časový vývoj, srovnání s plánem a podobně. Výsledkem je, že nedokážete odpovědět na základní otázky týkající se silných a slabých stránek firmy a jejího možného ohrožení.

Jako profesionál víte přesně, jaké informace potřebujete, abyste dostal firmu rychle pod kontrolu. Vznesete proto dotaz na IT oddělení, jaké budou náklady na vytvoření standardních přehledů a sestav podle vašeho přání. Odpovědí budou pravděpodobně finanční náklady minimálně ve stovkách tisíc a časové náklady v řádu měsíců. Mimoto rozpočet neumožňuje zahrnout takové náklady do letošního roku, a tudíž je možné s pracemi začít nejdříve po schválení rozpočtu na rok příští a i schválení takového projektu je otázkou.

Vy ovšem nemáte čas čekat, protože akcionáři od vás očekávají rychlou analýzu situace a návrhy opatření. Navíc očekávají pravidelnou měsíční zprávu o situaci ohledně prodejů. A to je chvíle, kdy vám znalost programování aplikací Microsoft Office ve VBA může doslova zachránit kůži.

První den si ujasníte, která data budete potřebovat, abyste obdrželi potřebné souhrnné zprávy. Ještě tentýž den navrhnete několik základních tabulek v aplikaci Microsoft Access, které potřebujete pra-

videlně plnit daty, a definujete potřebné relace mezi nimi kvůli integritě dat (toto je velká přednost Accessu oproti Excelu). Uživatelské rozhraní zatím programovat nepotřebujete, protože databáze v Accessu bude sloužit pouze jako sklad dat.

Druhý den si necháte od IT oddělení předvést, jak a v jaké formě můžete data z jednotlivých systémů dostat. Je důležité, abyste dostali data na co nejnižší úrovni, protože nemůžete vědět, jestli agregační zprávy zahrnují všechny potřebné informace. Pokud bude mít předpokládaný objem dat do blízké budoucnosti několik set tisíc až několik málo milionů řádků, nevadí, protože SQL dotazy si s tímto objemem dat relativně hravě poradí. Omezení databáze Access (2GB) by se vás zdaleka týkat nemělo, navíc je možno data rozdělit a databáze propojit.

Když budete mít štěstí (autor ho většinou neměl), IT oddělení vám nainstaluje potřebné databázové ovladače a dá práva čtení k tabulkám ve firemní databázi. Pokud budete mít o něco méně štěstí, obdržíte ze systémů data v tabulární formě s jasně oddělenými sloupci ve formátu Microsoft Excel. Často ale nebudete mít štěstí vůbec, protože se bude jednat o textový soubor obsahující nepotřebné informace v záhlaví a v těžko čitelném formátu. Navíc není možno dostat všechna data v jednom souboru, protože systém umožňuje buď export prodejů po jednotlivých dnech pro jeden obchod, nebo pro všechny obchody pro zadané období. Pokud chcete sledovat denní vývoj, potřebujete obojí, a tudíž každý den exportujete data pro všechny obchody.

Nyní již víte, jaká data jsou k dispozici, a máte v Accessu připravené tabulky, do kterých budete data importovat. Je čas napsat první program ve VBA, jenž data upraví do podoby, kterou lze do tabulek správně naimportovat. Samozřejmě že elegantní je data upravovat přímo interakcí s textovým souborem, ale na psaní takového kódu nemáte čas a ostatně, je to zbytečné, neboť aplikaci Excel máte stejně otevřenou.

Založíte nový sešit, spustíte záznamník maker, otevřete soubor se syrovými daty a data upravíte do přijatelné podoby. Poté záznamník vypnete a napíšete efektivní kód, přičemž jako základ pro psaní složitějších metod a vlastností (`TextToColumns`, `FormulaR1C1`, apod.) použijete řádky nahrané záznamníkem. Během několika minut jste hotovi a kód sám otevře sešit, upraví data a opět jej uloží.

Dalším krokem je kód upravit tak, aby prohledal adresář na všechny podobné soubory, všechny upravil podobným způsobem a data zkopíroval pod sebe do hlavního sešitu. Odladění takového kódu chvíli trvá, ale odměnou vám bude konsolidace dat z několika desítek sešitů během několika sekund.

Dalším krokem bude napsání jednoduchého kódu, který data z konsolidovaného sešitu neimportuje do dočasné tabulky připravené v Accessu a spustí přidávací dotazy, které odpovídající data nahrají do příslušných tabulek. Během řádově několika desítek sekund máte ve vaší databázi nová aktuální data. Navíc jste si jisti, že pokud import proběhl zdárně, není narušena integrita dat (např. neexistují prodeje nepřiřazené prodejně).

Nyní je čas daty nakrmit vaše standardní sestavy v Excelu. Na kartě „Data“, skupina „Načíst externí data“ klepnete na „Z aplikace Access“. Postupujete podle instrukcí průvodce a případně nezapomenete přidat do buněk parametry, pokud nechcete do listu načíst všechny data, která dotaz připravený v Accessu poskytuje.

Jako konečný krok propojíte buňky ve vašich sestavách na listu Excelu s importovanými daty a jste hotovi. Pokud změníte parametr v buňce, během řádově několika sekund (podle objemu dat) získáte přehled pro nové období.

Poté se můžete zabývat doplňkovými tabulkami a údaji (adresy, vybavení, kategorie), abyste mohli udělat přesnou analýzu situace, ale s hlavní částí jste hotovi. Každodenní aktualizace dat zabere méně času, než pití ranní kávy a vy máte každý den k dispozici aktuální přehled.

Dále můžete automatizovat automatické rozesílání standardních zpráv na definované adresy přes aplikaci Microsoft Outlook a rovněž automaticky vygenerovat měsíční prezentaci pro akcionáře v aplikaci Microsoft Powerpoint.

Domníváme se, že při investici řádově desítek hodin to není špatný výsledek, a určitě je nyní jasné, proč má smysl se učit VBA. Tato kniha se věnuje téměř výhradně jazyku VBA a objektovému modelu Excel, a proto v ní nenaleznete podrobný návod, jak řešit všechny kroky z našeho příběhu. Přesto ale po nastudování všech příkladů byste měli být schopni podobné programy psát.

Jak používat tuto knihu a vůbec, jak se učit VBA

Tato kniha si neklade za cíl stát se podrobným přehledem jazyka VBA a úplným popisem objektového modelu aplikace Microsoft Excel. O obou tématech byly napsány celé knihy a pouhý popis všech objektů aplikace Excel včetně jejich vlastností a metod by přesáhl obsah této knihy.

Jak jsme již řekli v úvodu, tato kniha je zaměřena na praktické využití jazyka VBA při programování aplikací v Excelu a řešení nejčastějších úloh. Autor je původcem více než 10000 příspěvků na mezinárodních fórech a zodpověděl tisíce dotazů. Samozřejmě že se dotazy často týkají podobných úloh, a přestože není možné kód napsat všeobecně, tato kniha se bude snažit představit techniky programování tak, aby je při rozumné míře přizpůsobení bylo možné použít pro vaše konkrétní potřeby.

I když předpokládáme, že čtenář zná základy programování ve VB nebo VBA, v prvních částech se zaměříme na popis nejpoužívanějších konstrukcí jazyka VBA a rovněž popis základních objektů aplikace Microsoft Excel. Zatímco jazyk VBA je obecný a program může být zapsán do modulu kterékoli aplikace, která hostování jazyka podporuje, objektový model Excelu je specifický pro tuto aplikaci a jazyk VBA ho přes standardní objekty, metody a vlastnosti využívá. Všechny důležité funkce jazyka VBA i základních objektů aplikace Excel se budeme vždy okamžitě snažit demonstrovat na příkladech konkrétního použitelného kódu. Někdy uvedeme vedle sebe kód běžný, který je snadno pochopitelný, a kód optimalizovaný pro výkon, který často obsahuje nezvyklé konstrukce. Takové konstrukce vždy hned objasníme.

V dalších částech knihy se zaměříme na praktické použití VBA pro řešení nejčastějších úloh, se kterými se pokročilý uživatel Excelu setkává, například:

1. Hromadná změna dat ve sloupci
2. Transformace dat do databázové podoby
3. Konsolidace dat z více sešitů

Všechny tyto techniky umožňují enormní ušetření času a nákladů na pracovní sílu. Vzhledem k tomu, s jakým množstvím různých sešitů Excel z různých zdrojů se uživatel setkává, je časová úspora prakticky neomezená.

Mocným nástrojem při konsolidaci dat může být jazyk SQL (Structured Query Language), což je obecný jazyk určený pro tvorbu dotazů nad databázovými tabulkami. Přestože jeho použití v Excelu má určitá omezení, ukážeme jeho použití na konkrétních příkladech. Samozřejmě že podrobné

vysvětlení jeho syntaxe není vzhledem k rozsahu této knihy možné. O tomto jazyku a jeho použití v aplikacích Microsoft Office přes technologii ADO (ActiveX Data Objects) byly rovněž napsány celé knihy.

Co se týče použití uživatelských formulářů, tato kniha vás podrobně provede tvorbou uživatelského formuláře pro editaci dat v databázové formě. Formulář bude záměrně navržen tak, aby představil použití všech praktických ovládacích prvků, které budete při návrhu vašeho konkrétního formuláře potřebovat. Pokud budete navrhovat váš vlastní formulář, měli byste v této lekci najít vždy odpověď na otázku, jak konkrétní ovládací prvek použít ve vašem kódu.

Samostatným tématem Excelu jsou kontingenční tabulky (Pivot Tables). Tento nástroj je velmi mocný při analýze databázových dat a jejich agregaci do přehledů. V praxi jsme se příliš často nesetkali s nutností kontingenční tabulku vytvořit od začátku v kódu, ale přesto si tento postup ukážeme, neboť se při něm seznámíte s jejím objektovým modelem.

Vzhledem k tomu, že tato kniha není referencí jazyka, ale jeho praktickou učebnicí, doporučujeme knihu alespoň jednou byť jen zběžně přečíst. Až se v budoucnu setkáte s konkrétním problémem, budete vědět, kde v knize najít konkrétní řešení, nebo alespoň návod k němu.

Jazyk VBA (a vlastně žádný programovací jazyk) se nenaučíte čtením popisu jednotlivých funkcí, objektů a možností jejich manipulace. Nejspolehlivější metodou je praxe. Programování je možno přirovnat k umění, jako je hudba. Můžete znát všechny noty a pravidla harmonie, ale to ještě neznamená, že jste schopni napsat skladbu, která se bude líbit. Podobně můžete do podrobností znát příkazy jazyka VBA a celý objektový model Excelu, ale to ještě neznamená, že napíšete program, který náročnou úlohu vyřeší za několik sekund. Přestože programování podobně jako hudba vyžaduje určitou dávku talentu, při určité dávce praxe a studia technik zkušených programátorů je možno za několik měsíců psát efektivní kód prakticky pro jakoukoli úlohu. A právě ke zkrácení této doby vám chce pomoci tato kniha. Najdete v ní mnoho ověřených řešení, která byste jinak dlouho hledali na internetu nebo v literatuře (podobně jako autor).

Otázky k částem lekcí

Za většinou částí lekcí najdete kontrolní otázky, které prověří, jestli jste porozuměli vykládanému textu. I když na konci knihy naleznete řešení, snažte se otázky nejprve zodpovědět samostatně, protože se většinou jedná o základy dané lekce.

Pokud je v řešení uveden příkladový kód, nejedná se samozřejmě většinou o jedinou možnost. Snažili jsme se však napsat vždy kód co nejefektivnější, který splňuje nároky na rychlost a současně jednoduchou upravitelnost.

V řešení jsme vždy danou otázku zopakovali, a tak vlastně příloha slouží i jako rychlá příručka řešení mnoha malých úloh, se kterými se budete v praxi setkávat.

Kód není vždy nejefektivnější řešení

Přestože tato kniha je téměř výhradně o programování aplikace Excel v jazyce VBA, musíme dát důrazné upozornění. Aplikace Microsoft Excel je složitý program a veškeré jeho části jsou zkompilovány (přeloženy) do jazyka, kterému procesor ve vašem počítači přímo rozumí a rychle reagu-

je. Oproti tomu váš kód VBA, zapsaný do normálních modulů nebo do modulů tříd, zůstává i po kompilaci (Debug => Compile Project) jazykem interpretovaným. To znamená, že příkazy nejsou předány procesoru přímo, nýbrž musí být nejprve za běhu programu přeloženy a vykonány. To znamená, že i ta nejefektivněji zapsaná uživatelská funkce ve VBA bude v naprosté většině pomalejší, než vestavěný vzorec Excelu.

Pokud programujete aplikaci pro Excel, snažte se vždy o spojení obou světů. Vytvořte list nebo sadu listů, která využívá metody dostupné přes uživatelské rozhraní (vzorce, ověření dat, podmíněné formátování, apod.), a programem psaným ve VBA tuto aplikaci pouze podpořte. V praxi je mnohem jednodušší a rychlejší vytvořit dokonalou šablonu, kterou přes VBA naplníte libovolnými daty, než vytvářet celý sešit přes VBA! A to nemluvíme o údržbě. Pokud si váš šéf vymyslí nový formát zprávy, musíte kód měnit na mnoha místech. Pokud budete mít vytvořenou šablonu, změníte formát jednoduše přes uživatelské rozhraní, případně v kódu upravíte oblasti, kam budete zapisovat hodnoty, a jste hotovi.

Další začátečnickou chybou je duplikace vestavěných dialogů a funkcí. Často se setkáváme s dotazem typu:

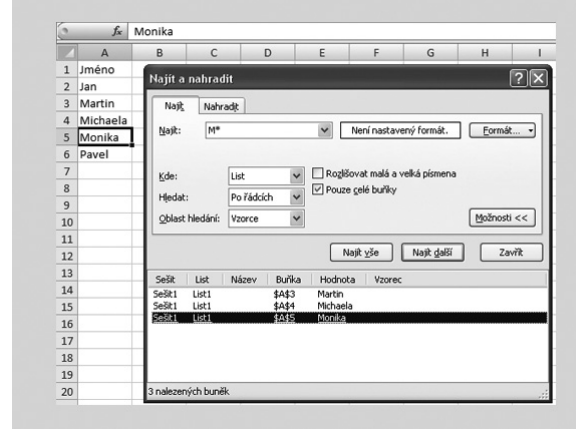
„Chci vytvořit formulář, do něhož uživatel zadá písmeno a program vybere první buňku, která začíná tímto písmenem."

Samozřejmě že Excel má pro tuto úlohu vestavěný poměrně mocný dialog „Najít a vybrat" ve skupině Úpravy. Tento dialog navíc umožňuje klepnout na tlačítko „Najít vše" a klikáním na jednotlivé odkazy se pohybovat po zvolených buňkách!

Je určitě jednodušší uživatelům vysvětlit použití tohoto dialogu (případně pokročilé uživatele odkázat na nápovědu), než vytvářet vlastní formulář, který bude funkčně omezený, a navíc budete muset kód měnit při každé změně zadání. Kromě toho lze například původní dialog „Najít" zobrazit přímo z VBA včetně parametrů:

```
Application.Dialogs(64).Show "M*", , , , , False, True
```

Jedinou výjimkou jsou takzvané diktátorské aplikace (*Dictatorship Applications*). V těchto vysoce profesionálních aplikacích, ušitých klientu na míru, se uživatelské rozhraní nahrazuje rozhraním vlastním a běžné funkce Excelu jsou pro uživatele nedostupné. Důvodem bývá nutnost omezit lidský faktor při zadávání dat, kdy například vložení sloupce nebo změna hlavičky může zcela zmást proceduru importu do jiné aplikace. Důvodem bývá i bezpečnost dat, ale jak se zmíníme v závěru knihy, tato bezpečnost je v Excelu dosti rozporuplná.



Obrázek Ú.1: Hromadné hledání buněk s obsahem odpovídajícím kritériu

Vyvarujte se i přílišného obtěžování uživatele vyskakujícími okny se zprávami (funkce `MsgBox`). Nejenže takové chování je nestandardní a běžného

uživatele může bez důkladné nápovědy zcela zmást, ale nevhodné naprogramování může celou aplikaci prakticky paralyzovat!

Příkladem může být tento dotaz:

„Ve sloupci A mám čísla faktur a ve sloupci B jejich lhůtu splatnosti. Chci napsat program ve VBA, který zobrazí upozornění pro každou fakturu po splatnosti."

Standardní odpověď může obsahovat tento kód, který úlohu vyřeší:

Soubor 0 Uvod.xlsm
modul Module1

```
Sub AlertOverdue()
Const MSG_TITLE As String = "Faktura po splatnosti"
Dim cl As Range, rg As Range
Dim msg As String
With ActiveSheet
    Set rg = .Range("B2", Cells(.Rows.Count, "B").End(3))
End With
For Each cl In rg
    If cl < Date Then
        MsgBox cl(, 0) & ": " & cl, vbOKOnly, MSG_TITLE
    End If
Next
End Sub
```

Jeho použití ovšem nedoporučujeme, pokud se – což je pravděpodobné – na seznamu vyskytuje po splatnosti faktur více. Jestliže má seznam několik desítek nebo dokonce stovek takových položek, uživatel by se z kódu nedostal, dokud by neodklepal všechna vyskakující okna se zprávou! Pokud chcete zobrazit všechny problematické položky v jedné zprávě, použijte tento kód:

Soubor 0 Uvod.xlsm
modul Module1

```
Sub AlertOverdue2()
Const MSG_TITLE As String = "Faktury po splatnosti"
Dim cl As Range, rg As Range
Dim msg As String
With ActiveSheet
    Set rg = .Range("B2", Cells(.Rows.Count, "B").End(3))
End With
For Each cl In rg
    If cl < Date Then
        msg = msg & cl(, 0) & ": " & cl & vbLf
    End If
Next
MsgBox msg, vbOKOnly, MSG_TITLE
End Sub
```

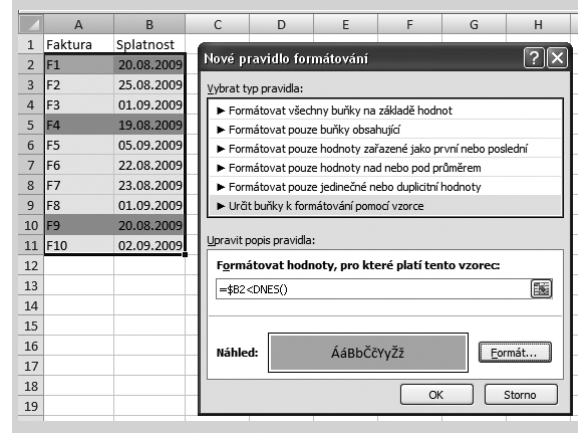
V praxi však zobrazení okna se zprávou mnoho nepomůže. Co s ním uživatel udělá? Opíše si všechna čísla, než klepne na OK? Práce s takovou aplikací by určitě nebylo nic příjemného.

Proto doporučujeme použít podmíněný formát:

1. Vyberte souvislou oblast s daty ve sloupcích A a B tak, aby buňka A2 byla aktivní
2. Na kartě „Domů“ klepněte na „Podmíněné formátování“ ve skupině „Styly“ a vyberte položku „Nové pravidlo“.
3. V následujícím dialogu klepněte na položku „Určit hodnoty k formátování pomocí vzorce“ a do textového pole zapište vzorec: =$B2<DNES(). Dialogové okno ještě nezavírejte.
4. Klepněte na tlačítko „Formát“, upravte formát podle potřeby (např. výplň buňky) a potvrďte klepnutím na „OK“.
5. Dialogové okno podmíněného formátu uzavřete rovněž klepnutím na tlačítko „OK“.

V naprosté většině případů je tazatel udiven robustností podmíněného formátu a volí toto řešení. Následujícím dotazem může být, jak implementovat podmíněný formát automaticky, pokud se seznam faktur dynamicky mění. Jeho hlavní problém je ale vyřešen rychle a elegantně, a to i ke spokojenosti ostatních uživatelů.

Chtěli jsme uvést tyto příklady, abyste si uvědomili, že než začnete psát kód, je vždy vhodné si ujasnit, jaké je vlastně zadání. V posledním příkladu bylo přáním upozornit na faktury po splatnosti. Ačkoli je úlohu možné řešit přes VBA, podmíněný formát je o mnoho vhodnější a nabízí větší flexibilitu, například zobrazení různých dob prodlení různými barvami. Mimochodem, podmíněné formátování bylo v Excelu 2007 podstatně rozšířeno a podle nás je jedním z důvodů, proč přejít na vyšší verzi.



Obrázek Ú.2: Použití podmíněného formátu k upozornění na kritické řádky

Shrnutí

Nesnažte se napodobovat standardní funkce Excelu. Ve většině případů je to zbytečné a uživatele to jedině zmate. Navíc se vaše aplikace zpomaluje. Raději vysvětlete, jak se ta která funkce používá, a pokročilé uživatele odkažte na nápovědu.

Pamatujte si, že v případě vyskakovacích oken a zpráv je méně více. Neustálé zobrazování opakujících se zpráv a upozornění nesvědčí o vaší profesionalitě, ale o tom, že neumíte navrhnout list tak, aby uživatele logicky vedl ke správnému zadávání dat.

Standardní vestavěné dialogy můžete zobrazit přímo z VBA. V nápovědě hledejte výraz „XlBuiltInDialog Enumeration“ pro podrobný seznam dialogů. Pro seznam možných parametrů hledejte výraz „Built–In Dialog Box Argument Lists“.