VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

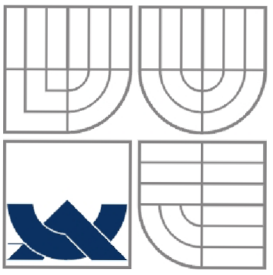
VYTVÁŘENÍ PLUGINŮ PRO MS OUTLOOK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

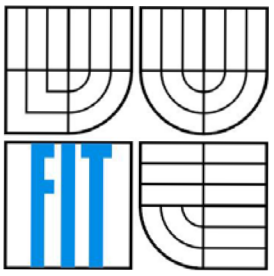
AUTOR PRÁCE
AUTHOR

BOHUMÍR FAJT

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

VYTVÁŘENÍ PLUGINŮ PRO MS OUTLOOK

CREATION OF PLUGINS FOR MS OUTLOOK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

BOHUMÍR FAJT

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ADAM HEROUT, Ph.D.

BRNO 2010

Abstrakt

Tato bakalářská práce se zabývá psaním rozšíření pro MS Outlook. Jejím cílem je projít možné způsoby, jak psát rozšíření pro MS Outlook. Následně vytvořit sadu příkladů využívajících tyto postupy a demonstrovat je, včetně praktických ukázek. Bakalářská práce má sloužit budoucím vývojářům těchto rozšíření, kteří však nemají doposud žádné zkušenosti s tímto tématem. Bude sloužit jako publikace, která je seznámí s daným tématem a ukáže základní možnosti psaní rozšíření.

Abstract

This bachelor's thesis is concerned with the writing of add-ons for MS Outlook. It has an objective of discussing the possible manners of writing add-ons for MS Outlook. Furthermore, it provides a set of examples using these procedures and demonstrates them, including practical exemplification. The bachelor's thesis is intended to serve future developers of the add-ons concerned who have no previous experience in this area. It will serve as a publication which will acquaint them with the subject and show the basic possibilities of writing add-ons.

Klíčová slova

MS Outlook, Visual Basic for Application, VBA, COM Add-in, Visual Studio Tools for Office, VSTO, Ribbon, Form Region

Keywords

MS Outlook, Visual Basic for Application, VBA, COM Add-in, Visual Studio Tools for Office, VSTO, Ribbon, Form Region

Citace

Fajt Bohumír: Vytváření pluginů pro MS Outlook, bakalářská práce, Brno, FIT VUT v Brně, 2010

Vytváření pluginů pro MS Outlook

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením:
Ing. Adama Herouta, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bohumír Fajt
1.5.2010

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Adamu Heroutovi, Ph.D za metodické vedení, pedagogickou a odbornou pomoc při zpracování mojí bakalářské práce.

Obsah

1 Úvod

V dnešní době patří emailová komunikace ke každodenním činnostem, jejíž používání usnadňují emailový klienti. Ti však nemohou svými vestavěnými funkcemi plně pokrýt nejrůznější potřeby všech jejich uživatelů, a proto určitě spoustu z nás napadlo, zda by poštovní klient nemohl umět ještě něco navíc. Z tohoto důvodu jsem se rozhodl, zabývat se tímto tématem v mojí bakalářské práci.

Pracovat budeme s klientem MS Outlook 2007 a postupně si projdeme možnosti psaní pluginů od jednoduchých skriptů, až po složitější pluginy v podobě sdílených DLL knihoven. Seznámíme se také s distribucí napsaných kódů a jejich instalací. Na jednoduchých příkladech si ukážeme možnosti pluginů, jako např.: přizpůsobení Ribbon funkcí, úprava aplikačního menu, úprava panelu nástrojů, či spolupráce s dalšími aplikacemi ze sady MS Office 2007.

2 Možnosti psaní pluginů

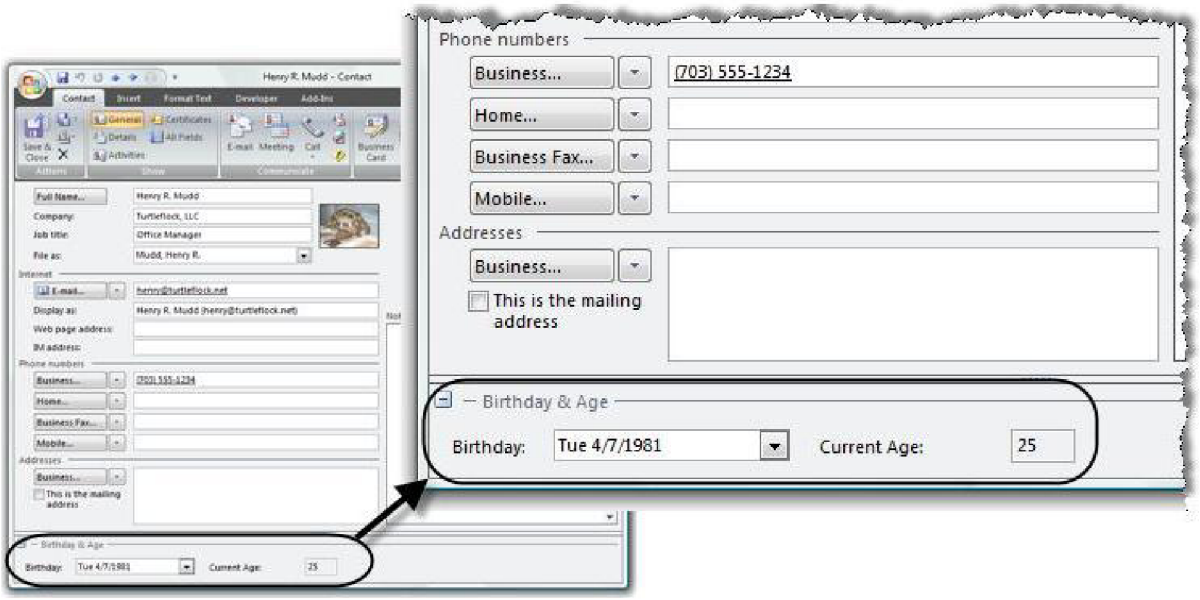
Plugin nebo též add-in je zásuvný modul, který rozšiřuje standardních funkce. Vytváření pluginů pro MS Outlook se dělí na dva hlavní směry, a to buď na pluginy tvořené pomocí VBA (Visual Basic for Applications) a úprav formulářů, napsané přímo ve vývojovém prostředí integrovaném v Outlooku, nebo napsané ve formě COM (Component Object Model) Add-in, jedná se o DLL knihovnu nebo spustitelný soubor (EXE). Každá z těchto možností má své výhody i nevýhody, které si postupně představíme, popíšeme a rozebereme.

Před samotným začátkem psaní si musíme rozmyslet rozsah, účel i požadavky na bezpečnost kódu atd. Na základě těchto požadavků vybereme, jednu z předchozích možností. Pro psaní kratších a jednodušších kódů, kde nepotřebujeme chránit napsaný kód, je vhodnější využít maker. Naopak COM Add-in využijeme všude tam, kde je potřeba chránit obsah vytvářeného pluginu. Mezi typická užití COM Add-in patří:

- přidávání tlačítek a nabídek na Panel nástrojů
- reakce na události vyvolané v Outlook Application object
- reakce na další události z Outlook Object Model
- rozšíření funkcí pomocí formulářů
- práce s rozhraním ribbon

2.1 Úprava Outlook formulářů

První možnost jak napsat plugin pro Outlook je, upravit si jeho formuláře s využitím integrovaných nástrojů. Cokoli co otevřeme v Outlooku, jako např.: emailová zpráva, kontakty, kalendář atd., je možné upravit. Můžeme přidávat či odebírat data, která tyto formuláře zobrazují, reagovat na uživatelský vstup a spouštět další činnosti Outlooku. Podrobněji se úpravě formulářů budeme věnovat v kapitole 3. pomocí COM Add-in.



Obr. 2.1 Upravený formulář [1]

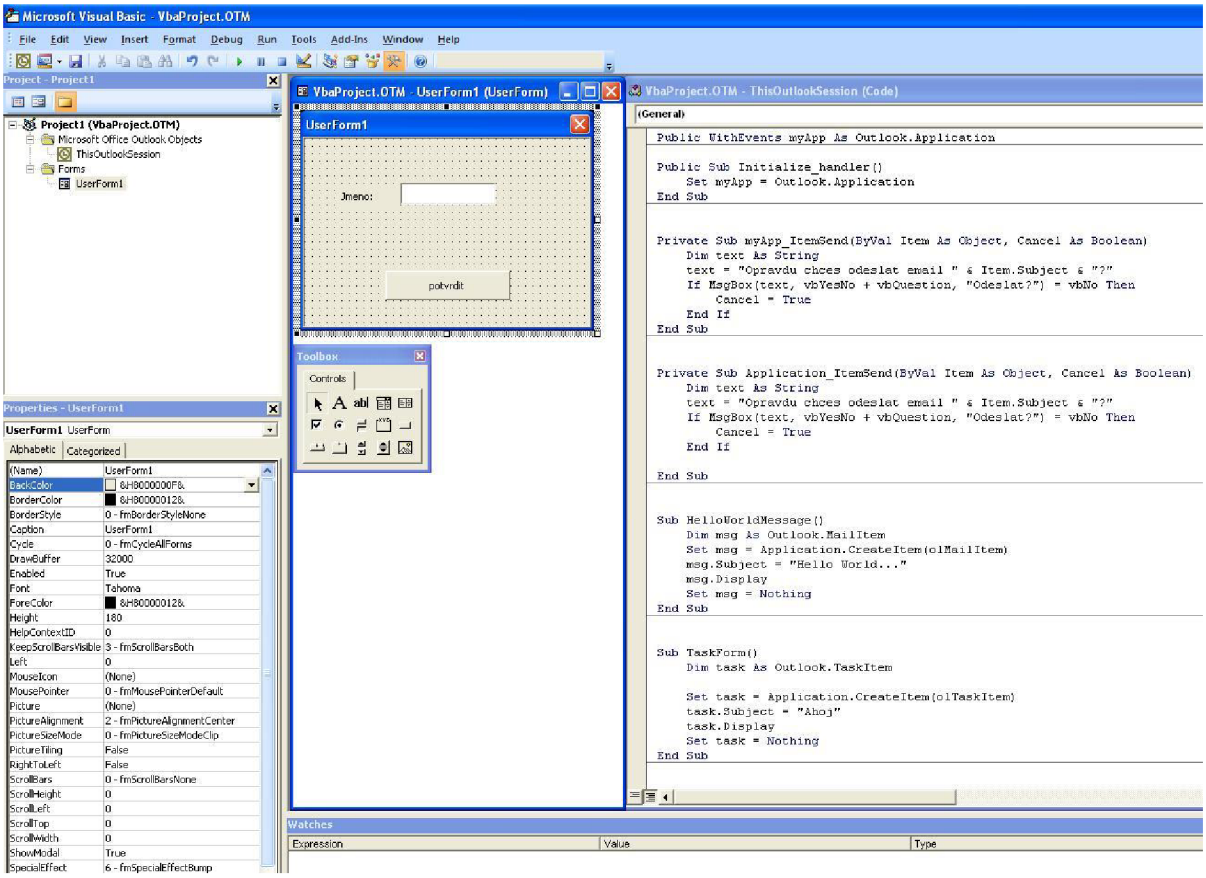
2.2 Visual Basic for Applications

Typické použití VBA:

- správa a údržba emailů (hromadné mazání příloh či celých emailů, přesouvání emailů do složek)
- zpracování a filtrování emailů
- vyhledávání v kalendáři a kontaktech
- spolupráce s dalšími aplikacemi ze sady Office (obsah emailu překopírovat do Excelu pro další zpracování)

Velkou výhodou tohoto způsobu vytvářeni pluginů je, že veškeré potřebné nástroje pro psaní kódu jsou integrovány přímo v Outlooku. Je zde vývojové prostření pro tvorbu maker, odchytávání událostí a tvorbu procedur. Integrované nástroje pro psaní pomocí VBA:

- Visual forms designer (pro vytváření dialogových oken ve VBA, nedají se zde upravovat formuláře v Outlooku)
- editor kódu se zvýrazňováním syntaxe
- a další



Obr 2.2 Editor VBA

Naopak nevýhodou tohoto způsobu psaní pluginů je, že zdrojový kód je možné prohlížet, upravovat a kopírovat.

2.2.1 Distribuce kódu

Kód VBA byl navržen pro osobní použití, proto se nepočítalo s jeho distribucí mezi velké množství uživatelů. Kód VBA tak můžeme distribuovat pouze dvěma možnostmi: distribucí pomocí exportu a importu modulu nebo zkopírováním souboru VbaProject.OTM. Veškerý napsaný kód se ukládá do jediného souboru VbaProject.Otm, umístěného ve složce: `\Documents and Settings\%uživatel%\Data aplikací\Microsoft\Outlook\VbaProject.OTM`

Informace z podkapitoly 2.2 jsem čerpal z [2].

2.2.1.1 Distribuce pomocí exportu a importu modulu

Nejjednodušší možností, jak distribuovat napsaný kód, je využít exportu v integrovaném editoru. Z důvodu, že stačí vybrat pouze z nabídky *File->Export File...* nebo využít klávesovou zkratku Ctrl+E. Jestliže hodláme distribuovat také dialogová okna, je potřeba poslat jak soubor *.frm, tak také soubor *.frx.

Nevýhody tohoto způsobu distribuce jsou:

- nepříjemný a zdlouhavý proces v případě většího množství modulů
- jestliže exportujeme ThisOutlookSession modul, uživateli se při importu změní název modulu na ThisOutlookSession1, takže kód nebude spouštěn automaticky jako v případě ThisOutlookSession
- uživatel musí vyřešit shodné názvy procedur mezi importovaným kódem a kódem již existujícím ručně

2.2.1.2 Zkopírování souboru VbaProject.OTM

Jelikož je veškerý kód uložen v jediném souboru, nabízí se vcelku jednoduchá možnost a tou je zkopírování celého souboru s kódem.

Ovšem i tato metoda má své nevýhody:

- veškerá dosud napsaná makra budou přepsána
- v Outlooku 2000 a 2002 musí uživatel kód spustit ručně pomocí Alt + F8 [2]

2.2.2 Příklady

V této kapitole si ukážeme několik příkladů s kódem a také, jak tento kód spouštět. Makra můžeme v podstatě rozdělit na dvě kategorie.

První z nich jsou makra pracující automaticky po startu Outlooku. Tyto makra zpravidla reagují na události generované Outlookem, které následně zpracovávají, jako např.: příchod emailu nebo odeslání emailu.

Druhou skupinou jsou makra, která jsou spouštěna uživatelem. Ty mohou také reagovat na události, ale až po jejich ruční aktivaci.

2.2.2.1 Hello World

První marko, které si ukážeme je "Hello World!". Nejdříve spustíme editor VBA *Tools ->Macro->Visual Basic Editor* nebo pomocí klávesové zkratky Alt+F11. Po spuštění tohoto makra se zobrazí dialog s textem „Hello World!".

```
Sub HelloWorld()
    MsgBox ("Hello World!")
End Sub
```

Obr. 2.3 Hello World!

2.2.2.2 Vytvoření nové zprávy

Začneme makrem, které vytvoří novou zprávu a vepíše do ní příjemce, předmět a obsah. Tímto postupem můžeme zefektivnit naši práci, zejména, pokud často komunikujeme se stejnou skupinou lidí či odesíláme v příloze stále stejné soubory.

```
Sub NewMessage()
    Dim msg As Outlook.MailItem
    Set msg = Application.CreateItem(olMailItem)
    msg.Subject = "Pozdrav"
    msg.To = "xfajt00@stud.fit.vutbr.cz"
    msg.Body = "Dobry den."
    msg.Display
    Set msg = Nothing
End Sub
```

Obr. 2.4 Makro – Nová zpráva

2.2.2.3 Zpracování událostí

Události generované Outlookem můžeme zpracovávat automaticky po startu nebo až po aktivaci uživatelem. Jestliže chceme odchytávat událost automaticky po startu Outlooku, složíme název obslužné procedury následovně `Application_JménoObsluhovanéUdálosti`.

V této ukázce budeme zpracovávat událost `ItemSend`. Událost je generována ještě před odesláním emailu. Celé makro pak bude před odesláním každého emailu chtít potvrzení od uživatele, že email chce skutečně odeslat.

```
Private Sub Application_ItemSend(ByVal Item As Object, Cancel As Boolean)
    Dim text As String
    text = "Opravdu chces odeslat email " & Item.Subject & "?"
    If MsgBox(text, vbYesNo + vbQuestion, "Odeslat?") = vbNo Then
        Cancel = True
    End If

End Sub
```

Obr. 2.5 Makro – Automatické zpracování události

Jestliže však potřebujeme zpracovávat události až po aktivaci uživatelem, musíme nejdříve vytvořit proměnou pro práci s událostmi, inicializovat pro odchytávání událostí (provede uživatel) a napsat obslužnou proceduru, jejíž jméno bude složeno následujícím způsobem: `JménoProměnné_JménoObsluhovanéUdálosti`.

```
Public WithEvents myApp As Outlook.Application

Public Sub Initialize_handler()
    Set myApp = Outlook.Application
End Sub


Private Sub myApp_ItemSend(ByVal Item As Object, Cancel As Boolean)
    Dim text As String
    text = "Opravdu chces odeslat email " & Item.Subject & "?"
    If MsgBox(text, vbYesNo + vbQuestion, "Odeslat?") = vbNo Then
        Cancel = True
    End If
End Sub
```

Obr. 2.6 Makro – Uživatelem spuštěná obsluha události

2.2.2.4 Spouštění dalších aplikací ze sady MS Office

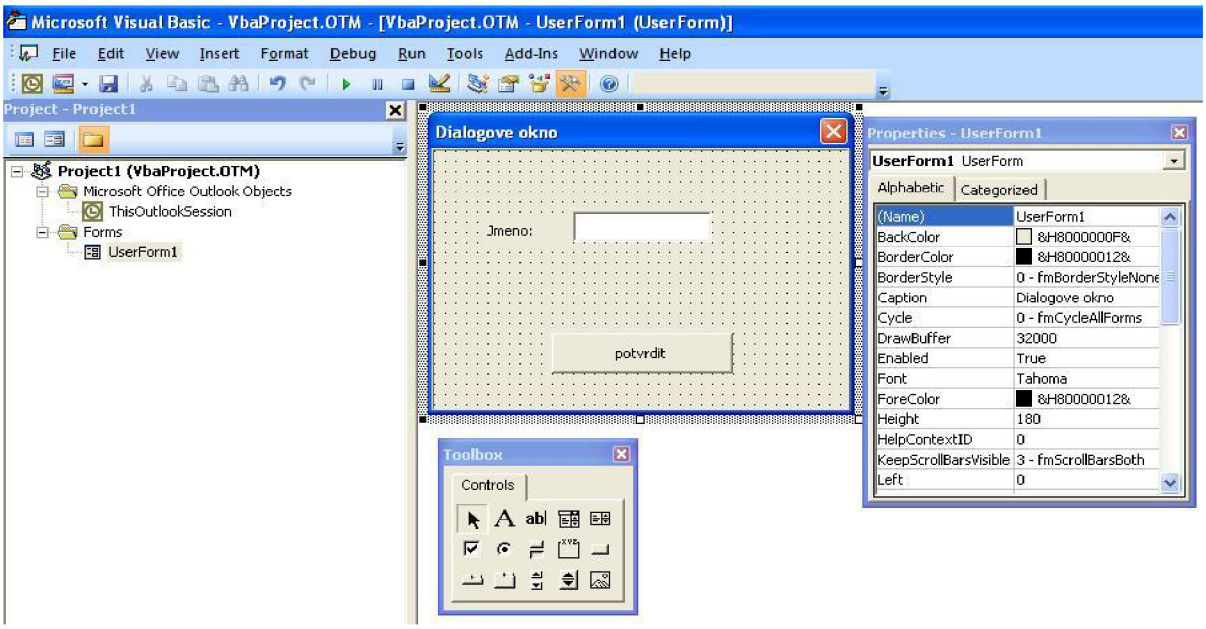
Další skupinou maker, kterou můžeme vytvořit, jsou makra spolupracující s dalšími aplikacemi ze sady MS Office. Můžeme jednoduše spustit Word či Excel a předvyplnit je údaji z příchozí zprávy. V našem příkladu si ukážeme spuštění MS Word. Vytvoříme dva objekty, jeden bude reprezentovat aplikaci Word, druhý bude dokument aplikace Word.

```
Sub WordDoc()
    Dim wApp As Object ' Aplikace Word
    Dim wDoc As Object ' Dokument
    Set wApp = CreateObject("Word.Application") ' Spusteni wordu
    Set wDoc = wApp.Documents.Add   'vytvoreni noveho dokumentu
    wApp.Visible = True
    Set wApp = Nothing
    Set wDoc = Nothing
End Sub
```

Obr. 2.7 Makro – Spuštění Wordu

2.2.2.5 Vytvoření dialogového okna

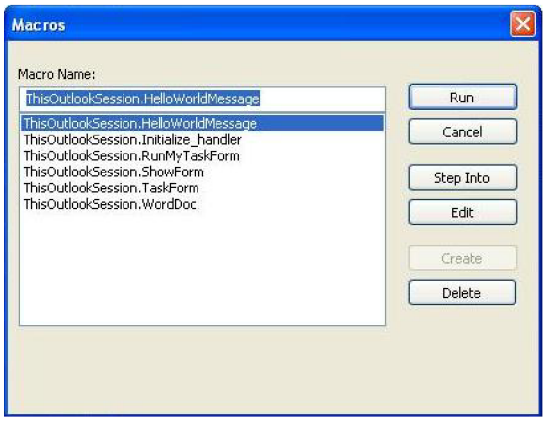
Doposud napsaná makra sloužila k automatizaci a zefektivnění naší práce s Outlookem, někdy ale potřebujeme interakci s uživatelem např.: zadat určitá data. A právě k tomuto účelu vytvoříme dialogové okno. Nejprve spustíme editor VBA a v okně Project Explorer vyvoláme kontextové menu. Z nabídky vybereme *Insert -> UserForm*. Zobrazí se prázdné dialogové okno, do kterého můžeme vkládat ovládací prvky (ListBox, CheckBox, OptionButton atd.) a měnit vlastnosti (viditelnost, název,barvu atd.).



Obr. 2.8 Dialogové okno

2.2.3 Spouštění maker

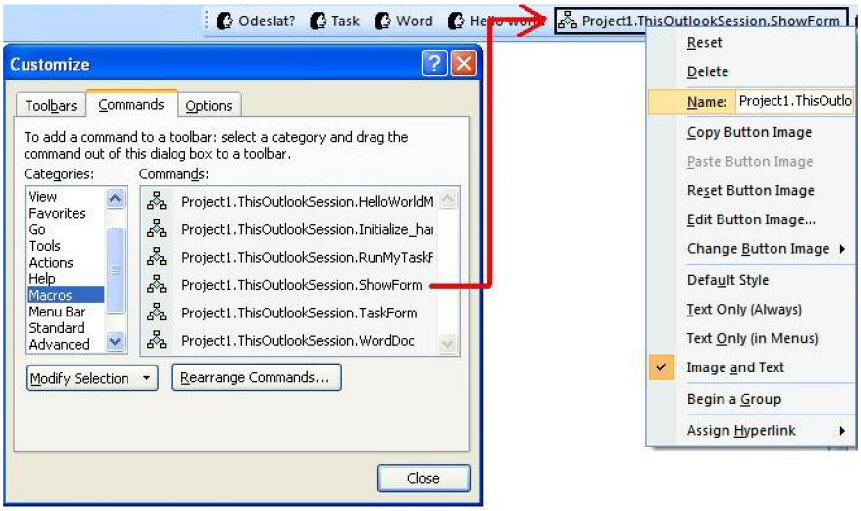
Nyní máme makra napsaná a zbývá nám je jen spustit. I zde existuje více možností, jak toho dosáhnout. První z nich je, zobrazit si veškerá makra a z nich si vybrat to, které potřebujeme. V případě, že máme napsáno velké množství maker, je tato metoda neefektivní. Zobrazí se nám totiž všechny napsané procedury, jejichž název je pevně daný, komplikovaný a nepřehledný. Pro zobrazení tohoto seznamu vybereme *Tools->Macro->Macros...* nebo pomocí Alt+F8.



Obr. 2.9 Seznam maker

Jak je vidět na obrázku 2.8 je tato metoda velice nepřehledná, je proto vhodnější vytvořit si na Panelu nástrojů tlačítka a těmi následně spouštět makra. Vybereme *Tools->Customize...* v záložce *Toolbars* vytvoříme nový toolbar. Přepneme na záložku *Commands,* z nabídky *Categories* vybereme *Macros.*

Z nabídky *Commands* vybereme požadované makro a přetáhneme jej do vytvořeného toolbaru. Poté můžeme vyvolat kontextové menu, ve kterém můžeme změnit název nebo ikonu tlačítka.



Obr. 2.10 Vytvoření toolbaru pro spouštění maker

2.2.4 Bezpečnost

Makra mohou využít útočníci k získání citlivých informací, jako jsou emaily, rozpisy schůzek, seznam kontaktů a osobních informací. Je proto nutné, sledovat jaká makra máme nainstalována a vždy si nové makra instalovat z důvěryhodných zdrojů. Máme také možnost nastavit úroveň zabezpečení přímo v Outlooku. Chceme-li změnit úroveň zabezpečení, zvolíme *Tools->Macro->Security...* Zde máme mimo jiné možnost vybrat z nabídky, zda makra úplně povolíme či úplně zakážeme.

3 COM add-in

Hlavní výhodou COM add-in je jejich snadná distribuce a ochrana napsaného kódu, což vyplývá z jejich podstaty DLL či EXE souboru. Tato vlastnost umožňuje psaní komerčních pluginů. Většinou se setkáme s DLL souborem, který je oproti EXE rychlejší.

Existuje velké množství nástrojů pro tvorbu pluginů např.: Visual C++, Visual Basic,Visual Studio .NET, Visual Studio Tools for Office (VSTO). Z toho vyplývá i to, že vývoj aplikací se může provádět v různých programovacích jazycích, z nichž nejpoužívanější jsou C# a Visual Basic .NET.

Pro vytváření COM pluginů existují dvě odlišné architektury. První z nich se nazývá sdílené add-in (shared add-in), která je postavena na IDTExtensibility2 interface. Druhou možností jsou pluginy sestavené pomocí Visual Studio Tools for Office (VSTO). Tato architektura je dostupná pro verzi Outlook 2003 a novější.Visual Studio Tools for Office vytváří podporu pro psaní kódu v jazyce Visual Basic .NET nebo C#. Jestliže porovnáme architekturu IDTExtensibility2 interface s VSTO, zjistíme, že VSTO nabízí několik výhod[3]:

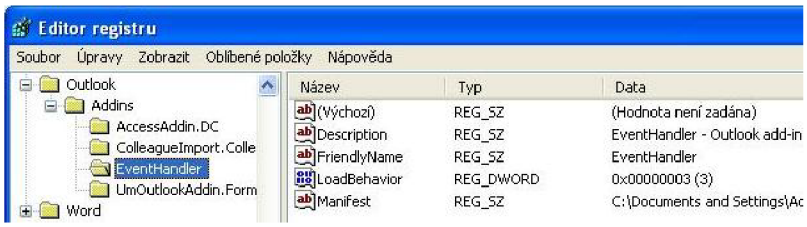
- lepší ladění již při návrhu
- lepší odchytávání chyb za běhu programu
- elegantní ukončení pluginu bez nutnosti spravovat všechny odkazy na COM objekty
- jednoduché události při spuštění a vypnutí programu
- integrovaný designér pro formuláře a ribbon panel

3.1 Spuštění DLL pluginu

Nejdříve se podíváme na způsob, jak nainstalovat plugin. Jeho kód je uložen v DLL souboru a my teď musíme dát Outlooku vědět o jeho existenci. Všechny pluginy jsou registrovány v registrech Windows v následujícím umístění:

`HKEY_CURRENT_USER\Software\Microsoft\Office\Outlook\Addins\název`

V registru jsou také uloženy informace o názvu, popisu a umístění souboru s pluginem. Dále zde můžeme najít položku s názvem LoadBehavior, která určuje způsob načtení makra. V tabulce 3.1 naleznete hodnoty s vysvětlením jejich významu.



Obr. 3.1 Záznam pluginu v registru

Hodnota LoadBehavior	Stav pluginu	Chování pluginu při spuštění	Popis
0	nenačten	nenačítat automaticky	Aplikace nenačítá plugin automaticky. Plugin může být načten uživatelem nebo programově. Pokud je plugin načten LoadBehavior zůstává na hodnotě 0, ale status pluginu v COM Add-in je aktualizován a ukazuje načtení.
1	načten	nenačítat automaticky	Aplikace nenačítá plugin automaticky. Plugin může být načten uživatelem nebo programově. Přestože COM Add-in ukazuje, že plugin je načten po startu aplikace, ve skutečnosti načten není, až do okamžiku manuálního nebo programového načtení. Pokud je plugin načten, je LoadBehavior změněn na hodnotu 0 a zůstává tak i po ukončení aplikace.
2	nenačten	načíst při spuštění	Aplikace se nepokouší načíst plugin automaticky. Plugin může být načten uživatelem nebo programově. Pokud je plugin načten LoadBehavior je změněn na hodnotu 3 a zůstává tak i po skončení aplikace.
3	načten	načíst při spuštění	Aplikace se pokusí načíst plugin automaticky. Toto je základní hodnota po vytvoření pluginu ve Visual Studiu. Pokud je plugin načten LoadBehavior zůstává hodnota 3. Pokud dojde při načítání k chybě LoadBehavior, je změněn na hodnotu 2 a zůstává tak i po skončení aplikace.
8	nenačten	načíst na vyžádání	Aplikace se nepokouší načíst plugin automaticky. Plugin může být načten uživatelem nebo programově. Pokud je plugin načten, LoadBehavior je změněn na hodnotu 9.
9	načten	načíst na vyžádání	Plugin je načten, pouze pokud je aplikací vyžadován. Pokud je plugin načten, LoadBehavior zůstává 9, ale status pluginu v COM Add-in je aktualizován a ukazuje načtení. Pokud dojde při načítání k chybě, LoadBehavior je změněn na hodnotu 8.
16	načten	načíst při prvním spuštění, poté načíst na vyžádání	Při této hodnotě je plugin načten na vyžádání. Aplikace načte plugin při prvním spuštění. Při příštím spuštění aplikace načte veškeré uživatelské rozhraní pluginu, ale plugin není načten, dokud uživatel neaktivuje element z uživatelského rozhraní pluginu. Po úspěšném načtení zůstává LoadBehavior na hodnotě 16, dokud je plugin načten. Po skončení aplikace je LoadBehavior změněn na hodnotu 9.

Tab. 3.1 Hodnoty LoadBehavior v registru a jejich význam [4]

3.2 Application-Specific Add-in

Při psaní pluginu můžeme narazit na Application –Specific Add-in. Jedná se o pluginy určené přímo pro konkrétní program ze sady MS Office. Jestliže tedy napíšeme Application –Specific Add-in, uložíme jej zvlášť pro Word a zvlášť pro Excel. I přestože mají oba pluginy velmi podobný kód, budou to dva samostatné pluginy. Při použití COM Add-in však můžeme napsat jen jediný plugin, který bude obsahovat kód pro Word, kód pro Excel a kód sdílený pro oba programy. Pokud takový plugin registrujeme v registrech, bude použitelný pro oba programy [5].

Add-in	Přípona souboru	Použitelné pro
Word add-in	.dot, .wll, .wiz	Pouze pro Word
Excel add-in	.xla, .xll	Pouze pro Excel
PowerPoint add-in	.ppa, .pwz	Pouze pro PowerPoint
Access add-in	.mda, .mde	Pouze pro Access
Exchange klient	.dll	Pouze pro Outlook a Microsoft Exchange klient
COM add-in	.dll, .exe	Word, Excel, PowerPoint, Outlook, Frontpage a další aplikace ze sady MS Office

Tab. 3.2 Přípony pluginů a jejich použitelnost v aplikacích [5]

3.3 Přehled Outlook object modelu

Microsoft Office Outlook object model je založen na COM (Component Object Model). Velké množství objektů z COM je použitelných pro vývoj a úpravu Outlooku. V Outlook object modelu třída reprezentuje každý objekt z uživatelského rozhraní např.: `Microsoft.Office.Interop.Outlook.Application` reprezentuje celou aplikaci a třída `Microsoft.Office.Interop.Outlook.MailItem` reprezentuje emailovou zprávu.

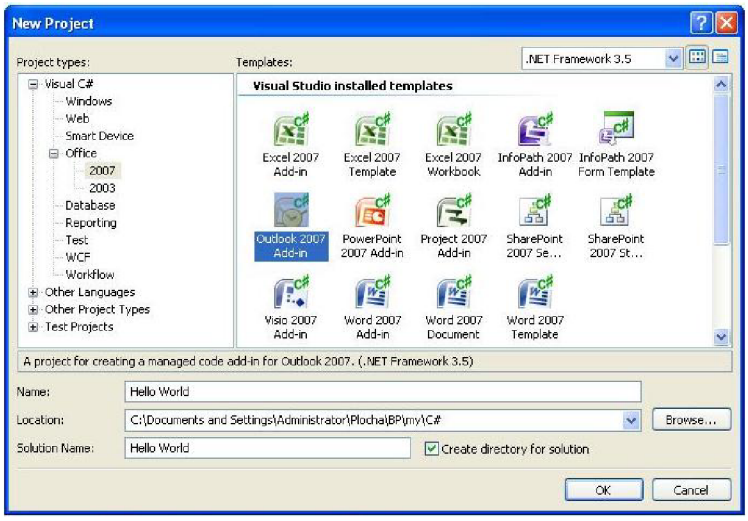
Před samotným začátkem psaní pluginů je dobré seznámit se s některými základními objekty. `Application` objekt reprezentuje aplikaci Outlook, jedná se o nejvyšší třídu v Outlook object modelu. `Explorer` koresponduje s oknem zobrazujícím složky a datové záznamy Outlooku jako např.: emaily, úkoly, schůzky atd. `MAPIFolder` reprezentuje složku obsahující emaily, kontakty, úkoly a další datové záznamy Outlooku. Je zde šestnáct základních `MAPIFolder` záznamů. `Inspector` koresponduje s oknem zobrazujícím jeden záznam např.: jeden email nebo jeden kontakt[6].

3.4 Příklady

Zabývat se budeme psaním pluginů architektury Visual Studio Tools for Office. Díky tomu můžeme psát pluginy pro šest aplikací z balíčku Microsoft Office. My však budeme psát pluginy pouze pro Outlook. Pro psaní kódu použijeme vývojové prostředí Microsoft Visual Studio 2008 Professional Edition, jehož součástí je Microsoft Visual Studio 2008 Tools for Office, programovacím jezykem je C#. Pro spouštění a testování napsaných pluginů použijeme MS Outlook 2007. Informace pro podkapitolu 3.4 jsem čerpal hlavně z [6] a msdn.microsoft.com.

3.4.1 První plugin „Hello World“

Nyní můžeme začít s prvním pluginem. Spustíme Visual Studio a zvolíme *File->New->Project...* Vpravo z nabídky vybereme *Visual C# -> Office -> 2007* a z nabídnutých šablon *Outlook 2007Add-in*. Vymyslíme jméno projektu a určíme jeho umístění.



Obr. 3.2 Nový projekt

Vytvoří se projekt s třídou `ThisAddIn` a dvě procedury `ThisAddIn_Startup` a `ThisAddIn_Shutdown`. Procedura `ThisAddIn_Startup` je volána po startu aplikace, takže do ní vepíšeme náš pozdrav.

```
System.Windows.Forms.MessageBox.Show("Hello World!");
```

První plugin je hotový, jeho funkce je jednoduchá, po spuštění Outlooku otevře dialogové okno s nápisem „Hello World!“.

3.4.2 Přidání menu

Nyní se podíváme na daleko zajímavější téma. Při psaní pluginů, se určitě nevyhneme potřebě interakce s uživatelem. To se samozřejmě netýká jen rozsáhlých pluginů, ale i přidání třeba jen jedné funkce vyžaduje, aby měl uživatel možnost tuto funkci spustit. A v mnoha případech je právě přidání menu vhodným řešením. V Microsoft Office jsou všechny menu a toolbary CommandBars. CommandBar je kolekce obsahující standardní toolbary, menu, kontextová menu atd. Budeme pracovat s objektem `Explorer`.

Při přidávání menu budeme provádět tyto kroky:

- určíme, kam nové menu přidáme (v našem případě do hlavního menu)
- vložíme nové menu do starého

- pojmenujeme menu
- přidáme ovládací tlačítko a nastavíme obsluhu při kliknutí na něj.

Při vytváření menu můžeme určit, zda chceme menu vpřidat trvale, či pouze dočasně. Dočasným přidáním se rozumí do konce běhu aplikace. V našem případě přidáváme menu trvale, poslední paramert ve funkci `Add` je nastaven na `false`.

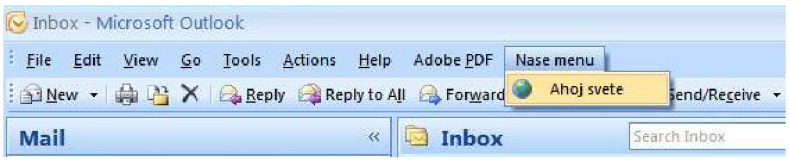
```
//definujeme kam budeme pridavat menu
menuBar = this.Application.ActiveExplorer().CommandBars.ActiveMenuBar;
//pridame menu
newMenuBar = (Office.CommandBarPopup)menuBar.Controls.Add(
    Office.MsoControlType.msoControlPopup, missing,
    missing, missing, false);
if (newMenuBar != null)
{
    //nazev menu
    newMenuBar.Caption = "Nase menu";
    newMenuBar.Tag = menuTag;
    //pridani tlacitka do menu
    buttonOne = (Office.CommandBarButton)newMenuBar.Controls.
    Add(Office.MsoControlType.msoControlButton, missing,
        missing, 1, true);
    //nastaveni tlacitka
    buttonOne.Style = Office.MsoButtonStyle.
        msoButtonIconAndCaption;
    buttonOne.Caption = "Ahoj svete";
    //nastavime ikonu tlacitka
    buttonOne.FaceId = 610;
    buttonOne.Tag = "c123";
    //obsluha pri kliknuti na tlacitko
    buttonOne.Click += new Office._CommandBarButtonEvents_ClickEventHandler(buttonOne_ClickM);
    newMenuBar.Visible = true;
}
```

Obr. 3.3 Kód přidání menu

Když máme menu vytvořené jako trvalé, může nastat situace, kdy jej budeme chtít odstranit. K odstranění menu jej musíme nejdříve najít. Hledání provádíme pomocí popisu menu.

```
Office.CommandBarPopup foundMenu = (Office.CommandBarPopup)
    this.Application.ActiveExplorer().CommandBars.ActiveMenuBar.
    FindControl(Office.MsoControlType.msoControlPopup,
    missing, menuTag, true, true);
if (foundMenu != null)
{
    foundMenu.Delete(true);
}
```

Obr.3.4 Kód odebrání menu



Obr.3.5 Přidané menu

3.4.3 Úprava toolbaru

Obdobně jako jsme přidali menu, můžeme přidat i toolbar. I v tomto případě budeme pracovat s `Explorer` objektem. Při vytváření budeme provádět tyto kroky:

- nejdříve určíme, kam toolbar přidáme
- vložíme nový toolbar mezi ostatní toolbary
- pojmenujeme toolbar
- přidáme ovládací tlačítko a nastavíme obsluhu při kliknutí na něj.

```
toolBar = this.Application.ActiveExplorer().CommandBars;
//vytvoreni toolbaru
newToolBar = toolBar.Add(tollBarTag, Office.MsoBarPosition.msoBarTop, false, true);

//pridani tlacitka do toolbaru
Office.CommandBarButton button1 = (Office.CommandBarButton)
    newToolBar.Controls.Add(1,missing,missing,missing,missing);
//styl tlacitka
button1.Style = Office.MsoButtonStyle.msoButtonIconAndCaption;
//popisky
button1.Caption = "Ahoj svete";
button1.Tag = buttonTag;
//kontrola jestli se povedlo tlacitko vytvorit
if(this.buttonOne == null)
{
    //obsluha kliknuti na tlacitko
    this.buttonOne = button1;
    buttonOne.Click += new Office._CommandBarButtonEvents_ClickEventHandler(buttonOne_ClickM);
}
```

Obr. 3.6 Kód přidání toolbaru

Obr. 3.7 Přidaný toolbar

Na rozdíl od nového menu nemusíme řešit odstranění toolbaru, protože při jeho vytváření jsme jej definovali jako dočasný. Poslední parametr příkazu je nastaven na true.

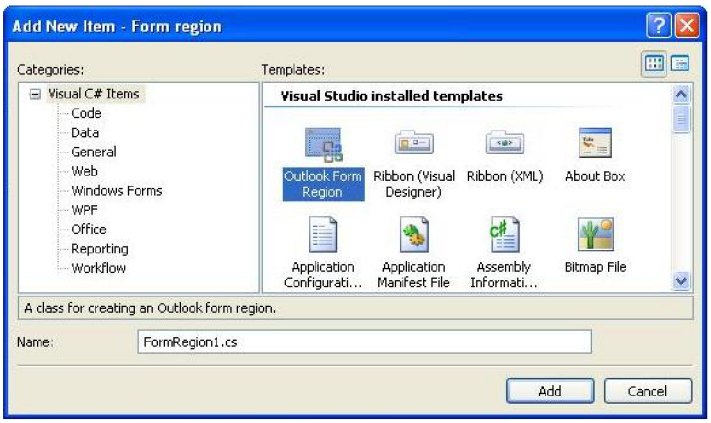
```
newToolBar=toolBar.Add(toolBarTag,Office.MsoBarPosition.msoBarTop,
    false,true)
```

To znamená, že při každém vypnutí Outlooku se toolbar sám smaže.

3.4.4 Form region

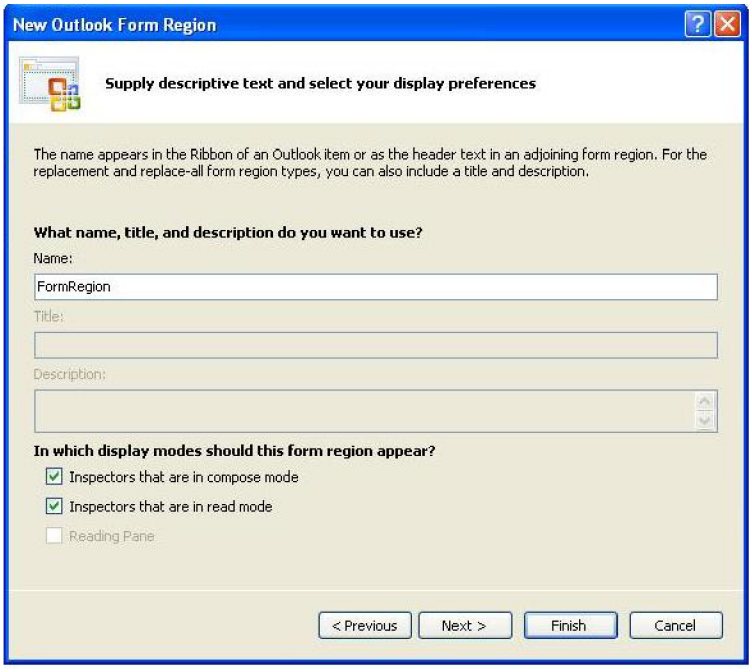
Form region přichází s Outlookem 2007 a představují další možnost, jak vytvořit vlastní uživatelské rozhraní. Vytvoříme nový projekt a v Solution Exploreru vyvoláme kontextové menu.

Vybereme *Add->New Item* a z nabízených šablon zvolíme Outlook Form region. Tímto způsobem spustíme průvodce, který nám pomůže s tvorbou Form region.



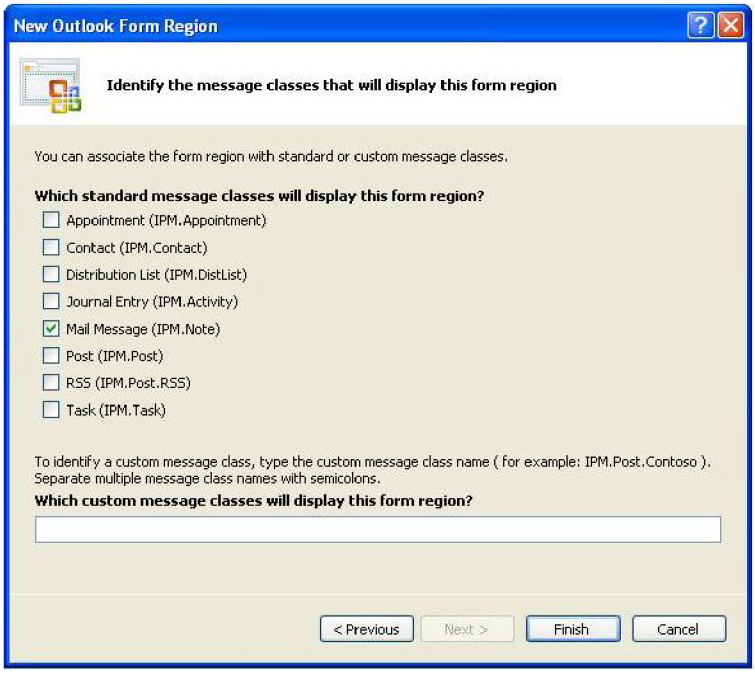
Obr. 3.8 Vytvoření Form region

V prvním kroku vybereme možnost Design a new form region, dále vybíráme typ Form region, zvolíme Separete, tím přidáme novou stránku. Další možnosti jsou Adjoining (přidá naše úpravy na konec standardního formuláře), Replacement (nahradí základní stránku formuláře), Replace-all (nahradí všechny formuláře). Dalším krokem zvolíme název a také, kdy se námi vytvořený Form region bude zobrazovat. Můžeme jej zobrazit při vytváření položky nebo při čtení položky. Adjoining Form region, můžeme zobrazit v náhledovém podokně Outlooku. Musíme vybrat alespoň jednu z uvedených možností.



Obr. 3.9 Průvodce vytvoření Form region 1

Posledním krokem vybereme, kde se bude náš Form region zobrazovat viz obrázek 3.10.

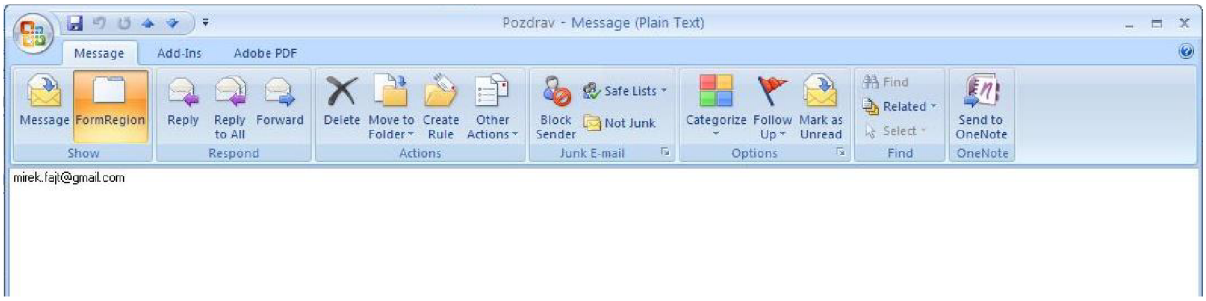


Obr. 3.10 Průvodce vytvoření Form region 2

Abychom nenapsali jen prázdný Form region, vložíme do něj kód, který zobrazí odesílatele zprávy. Musíme se však ujistit, že skutečně pracujeme se zprávou, jestliže jsme při tvorbě Form region umožnili, aby se zobrazoval i v jiných položkách, než jen při tvorbě/čtení emailů. Také musíme ověřit, zda jsme v psacím nebo čtecím režimu. V psacím režimu zpráva z pohledu Outlooku odesílatele ještě nemá.

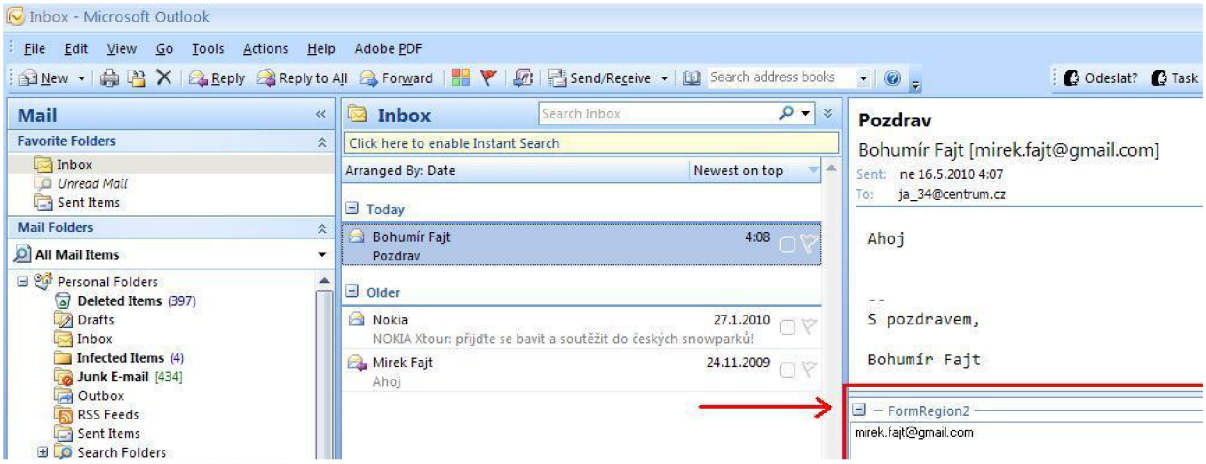
```
private void FormRegion1_FormRegionShowing(object sender, System.EventArgs e)
{
    //overime jestli pracujeme se zpravou
    if (this.OutlookItem is Outlook.MailItem)
    {
        Outlook.MailItem mail = (Outlook.MailItem)this.OutlookItem;
        //overime jestli existuje odesilatel zpravy
        if (mail.SenderEmailAddress != null)
            listBox1.Items.Add(mail.SenderEmailAddress);
    }
}
```

Obr. 3.11 Kód Form region



Obr. 3.12 Výsledný Form region

Obrázek 3.13 ukazuje, jak by vypadalo hlavní okno Outlooku, pokud bychom jako typ Form region zvolili Adjoining a zaškrtli zobrazování “Reading pane“ v průvodci.



Obr. 3.13Adjoining Form region v hlavním okně Outlooku

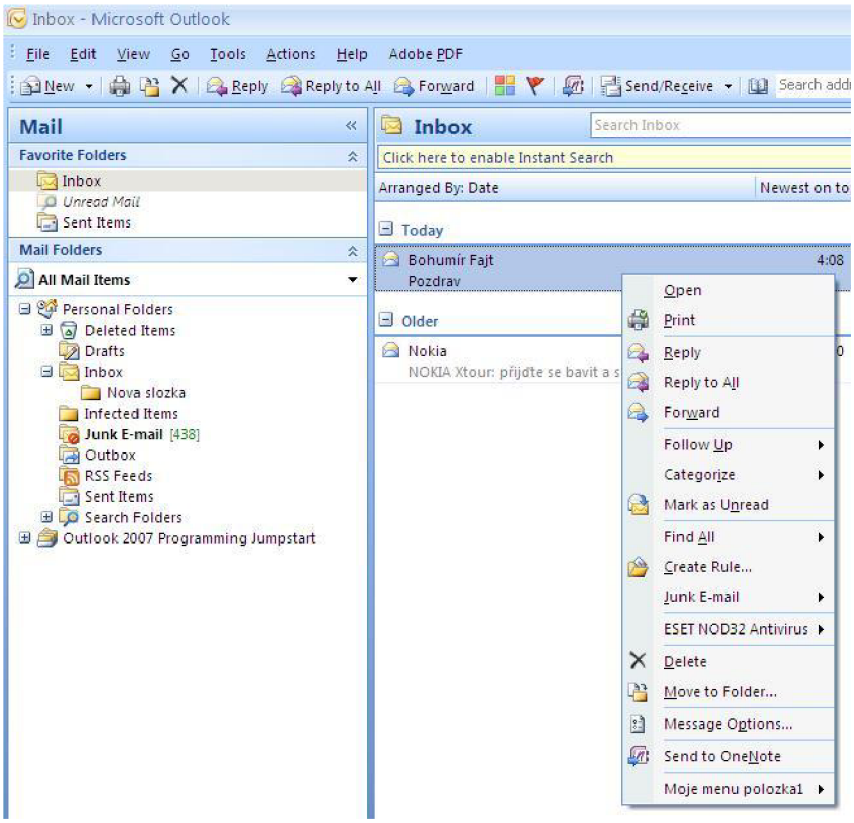
3.4.5 Kontextové menu

Pomocí pluginu můžeme upravovat i kontextová menu. Následující ukázka přidá do kontextového menu jednu novou položku. Naše upravené kontextové menu se zobrazí, pokud jej vyvoláme nad nějakou záznamovou položkou Outlooku (email, kontakt atd.). Při přidávání položky do kontextového menu budeme postupně provádět tyto kroky:

- inicializujeme obslužnou proceduru, ve které přidáme položku do kontextového menu, obslužná procedura bude reagovat na událost vzniklou vyvoláním kontextového menu nad položkou v Outlooku
- v obslužné proceduře přidáme položku do kontextového menu
- pojmenujeme novou položku
- nastavíme její popisky

```
private void ThisAddIn_Startup(object sender, System.EventArgs e)
{
    //pridani obsluhy udalosti vyvolani menu
    this.Application.ItemContextMenuDisplay += new Outlook.
        ApplicationEvents_11_ItemContextMenuDisplayEventHandler(ObsKontextMenu);
}

void ObsKontextMenu(Office.CommandBar commBar, Outlook.Selection Selection)
{
    Office.CommandBarPopup myItem = (Office.CommandBarPopup)
        commBar.Controls.Add(Office.MsoControlType.msoControlPopup, missing, "Moje menu",
        commBar.Controls.Count + 1, missing);

    //vytvorime novou skupinu v menu
    myItem.BeginGroup = true;
    //nazev menu
    myItem.Tag = "myMenu";
    //popis menu
    myItem.Caption = "Moje menu polozka1";
    //zobrazime vytvorene menu
    myItem.Visible = true;
}
```

Obr. 3.14 Kód vytvoření kontextového menu



Obr. 3.15 Kontextové menu

3.4.6 Složky

Pro lepší přehlednost příchozích emailů máme možnost, organizovat emaily do složek a podsložek. Jejich vytváření patří k základním dovednostem při psaní pluginů. Velmi často jsou pluginy psány právě proto, aby zpracovávaly příchozí poštu a třídily ji do složek.

3.4.6.1 Vytvoření složky

Podíváme se jak takovou složku vytvořit. Při vytváření složky určíme, kde ji budeme vytvářet, její nadřazenou složku.

```
Outlook.NameSpace nameSpace = this.Application.GetNamespace("MAPI");
//urcime kde budeme novou slozku vytvaret
//v tomto pripade vytvorime podslozky do slozky Inbox
Outlook.MAPIFolder folder = nameSpace.GetDefaultFolder(
    Microsoft.Office.Interop.Outlook.OlDefaultFolders.olFolderInbox);
//vytvorime novou slozku
folder.Folders.Add("Nova slozka", missing);
```

Obr. 3.16 Kód vytvoření nové složky

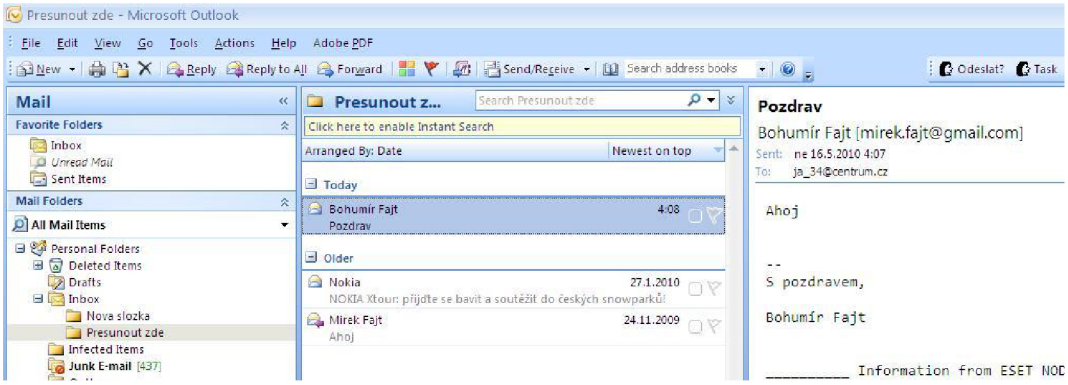
3.4.6.2 Přesun emailů

V předchozím příkladu jsme si ukázali, jak vytvořit novou složku. Nyní se podíváme, jak manipulovat s emaily. V následujícím příkladu přesuneme všechny emaily ze složky Inbox do námi vytvořené složky.

- určíme si složku, ze které budeme emaily přesouvat
- vytvoříme složku, do které chceme emaily přesunout
- přesuneme všechny emaily

```
Outlook.NameSpace nameSpace = this.Application.GetNamespace("MAPI");
//slozka ze ktere budeme presouvat
Outlook.MAPIFolder srcFolder = nameSpace.GetDefaultFolder(
    Microsoft.Office.Interop.Outlook.OlDefaultFolders.olFolderInbox);
//polozky ve zdrojove slozce
Outlook.Items items = (Outlook.Items)srcFolder.Items;
Outlook.MailItem mail = null;
//vytvoreni cilove slozky
srcFolder.Folders.Add("Presunout zde", missing);
//cilova slozka
Outlook.MAPIFolder dstFoder = srcFolder.Folders["Presunout zde"];

foreach (object eMail in items)
{
    try
    {
        mail = eMail as Outlook.MailItem;
        if (mail != null)
        {
            mail.Move(dstFoder);
        }
    }
    catch (Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}
```

Obr. 3.17 Kód přesunu emailů



Obr. 3.18 Přesunuté emaily

3.4.6.3 Smazání složky

Zbývá už je vytvořené složky smazat. Při odstraňování bychom měli zkontrolovat, zda ve složce nezůstaly zprávy, které smazat nechceme. Odstraníme totiž nejen složku, ale i celý její obsah. Odstraňovanou složku můžeme určit indexem (indexování složek začíná od 1) jako v ukázce nebo použít název složky: `folder.Folders["nazevMazaneSlozky"].Delete()`

V následujícím příkladu budeme mazat druhou podsložku složky Inbox.

- najdeme nadřazenou složku složky, kterou chceme smazat
- odstraňovanou složku určíme jejím indexem

```
Outlook.NameSpace nameSpace = this.Application.GetNamespace("MAPI");
//urcime kde budeme mazat
Outlook.MAPIFolder folder = nameSpace.GetDefaultFolder(
    Microsoft.Office.Interop.Outlook.OlDefaultFolders.olFolderInbox);
//vytvorime smazem druhou podslozku ve slozce Inbox
folder.Folders[2].Delete();
```

Obr. 3.19 Kód odstranění složky

3.4.7 Kontakty

Kontakty obsahují velké množství nejrůznějších informací (jméno, příjmení, email, telefonní spojení, adresu atd.) o osobách, se kterými komunikujeme. Není proto divu, že velmi často potřebujeme měnit tyto informace dynamicky, a právě s tímto problémem nám pomůže VSTO plugin.

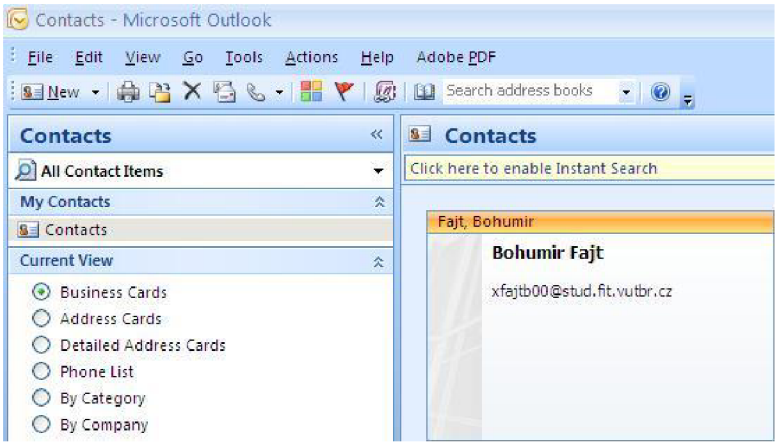
3.4.7.1 Vytvoření kontaktu

Předvedeme si, jak jednoduše vytvořit kontakt:

- vytvoříme nový kontakt
- vložíme údaje, které chceme uložit
- uložíme kontakt

```
//Vytvorime novy kontakt
Outlook.ContactItem newContact = (Outlook.ContactItem)
    this.Application.CreateItem(Outlook.OlItemType.olContactItem);
//zadame jmeno
newContact.FirstName = "Bohumir";
//zadame prijmeni
newContact.LastName = "Fajt";
//zadame email
newContact.MailingAddress = "xfajtb00@stud.fit.vutbr.cz";
//zadame skupinu
newContact.Categories = "VUT";
//ulozime kontakt
newContact.Save();
```

Obr. 3.20 Kód vytvoření nového kontaktu



Obr. 3.21 Vytvořený kontakt

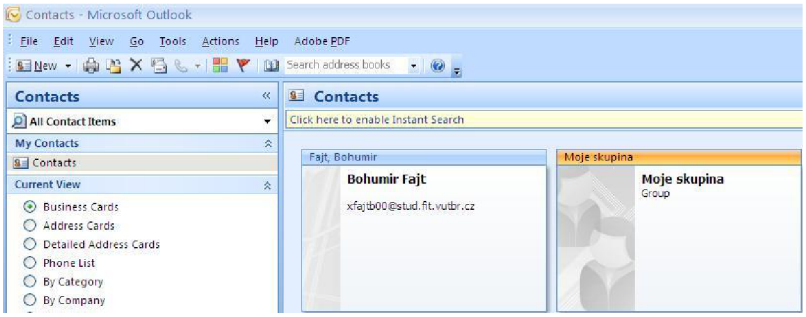
3.4.7.2 Skupina kontaktů

Jednou z nejdůležitějších funkcí Outlooku, co se týče správy kontaktů, je vytváření skupin kontaktů. Jedná se o rychlou a efektivní cestu, jak rozeslat hromadný email určité skupině lidí. Skupina obsahuje seznam kontaktů a jejich emailových adres, kterým můžeme posílat emaily.

- vytvoříme skupinu
- pojmenujeme skupinu
- vytvoříme nový email, ten však nebudeme ukládat ani odesílat
- do emailu vložíme příjemce (jméno, emailová adresa)
- příjemce z emailu vložíme do skupiny
- uložíme skupinu

```
//vytvoreni skupiny prijemcu
Outlook.DistListItem group = (Outlook.DistListItem)
    this.Application.CreateItem(Outlook.OlItemType.olDistributionListItem);
//zadame nazev skupiny
group.DLName = "Moje skupina";
Outlook.MailItem mail = (Outlook.MailItem)
    this.Application.CreateItem(Outlook.OlItemType.olMailItem);
//seznam prijemcu
mail.Recipients.Add("Bohumir Fajt <xfajtb00@stud.fit.vutbr.cz>");
mail.Recipients.Add("Jan Novak <novak@seznam.cz>");
//pridani seznamu prijemcu do skupiny
group.AddMembers(mail.Recipients);
//ulozeni skupiny
group.Save();
```

Obr. 3.22 Kód vytvoření skupiny kontaktů



Obr. 3.23 Vytvořená skupina

3.4.8 Emailové zprávy

3.4.8.1 Vytvoření nové zprávy

MS Outlook je v první řadě emailový klient a jeho hlavním úkolem je přijímání, odesílání zpráv a jejich správa. Nyní si tedy ukážeme, jak takovou zprávu vytvořit. Emailovou zprávu reprezentuje třída `Microsoft.Office.Interop.Outlook.MailItem`

- vytvoříme nový email
- vložíme potřebné údaje (příjemce, předmět, tělo, důležitost emailu atd.)
- email buď rovnou odešleme nebo jej můžeme zobrazit uživateli pro editaci

```
//vytvorime novou zpravu
Outlook.MailItem newMail = (Outlook.MailItem)
    this.Application.CreateItem(Outlook.OlItemType.olMailItem);
//vlozime prijemcovu emailovou adresu
newMail.To = "xfajtb00@stud.fit.vutbr.cz";
//vlozime predmet
newMail.Subject = "Pozdrav";
//vlozime obsah zpr8vy
newMail.Body = "Dobry den.";
//dale muzeme nastavit treba dulezitost zpravy
newMail.Importance = Outlook.OlImportance.olImportanceNormal;
//jestlize chceme napsanou zpravu dale upravovat rucne zobrazime ji
newMail.Display(true);
//jestlize chceme zpravu rovnou odeslet pouzijeme nasledujici prikaz
//newMail.Send();
```

Obr. 3.24 Vytvoření nové zprávy

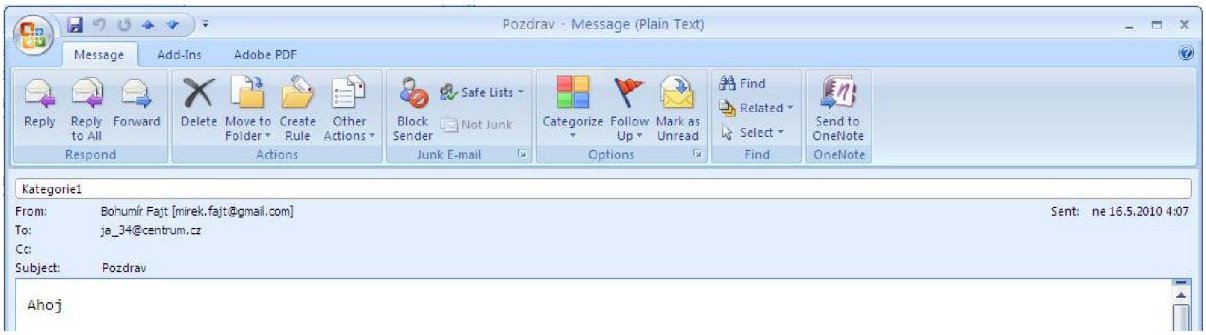
3.4.8.2 Kategorizace zpráv

Při větším množství zpráv se může stát orientace v nich velmi složitou, zprávy tak můžeme organizovat nejen do složek, ale můžeme je i barevně označit a přiřadit jim kategorii. Označit zprávu můžeme ručně nebo také programově. Následující příklad vytvoří kategorii a přiřadí ji první zprávě ze složky Inbox.

- určíme zprávu, kterou chceme označit. V našem případě první zprávu ze složky Inbox
- přiřadíme zprávě kategorii
- uložíme změny

```
Outlook.NameSpace nameSpace = this.Application.GetNamespace("MAPI");
Outlook.MAPIFolder srcFolder = nameSpace.GetDefaultFolder(
    Microsoft.Office.Interop.Outlook.OlDefaultFolders.olFolderInbox);
Outlook.Items items = (Outlook.Items)srcFolder.Items;

Outlook.MailItem mail = (Outlook.MailItem) items.GetLast();
mail.Categories = "Kategorie1";
mail.Save();
```

Obr. 3.25 Kód vytvoření kategorie



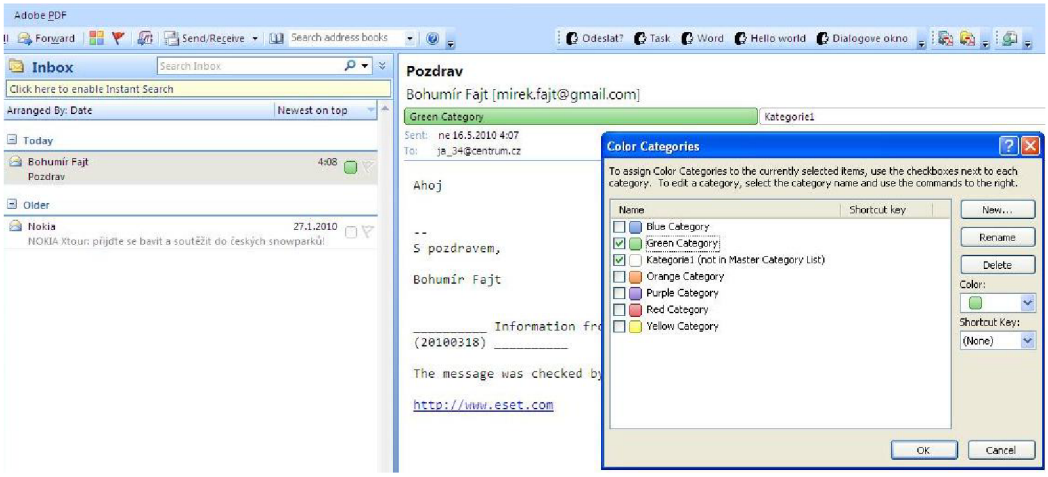
Obr. 3.26 Označení zprávy vytvořenou kategorií

Nebo můžeme nechat uživatele, ať kategorii zvolí ručně.

- najdeme první zprávu ze složky Inbox
- vyvoláme nabídku, ve které uživatel zvolí kategorii

```
Outlook.NameSpace nameSpace = this.Application.GetNamespace("MAPI");
Outlook.MAPIFolder srcFolder = nameSpace.GetDefaultFolder(
    Microsoft.Office.Interop.Outlook.OlDefaultFolders.olFolderInbox);
Outlook.Items items = (Outlook.Items)srcFolder.Items;

Outlook.MailItem mail = (Outlook.MailItem)items.GetLast();
mail.ShowCategoriesDialog();
```

Obr. 3.27 Kód vyvolání menu s nabídkou kategorií



Obr. 3.28 Vyvolání nabídky pro zvolení kategorie

3.4.9 Kalendář

Komponenta kalendář v MS Outlooku nám pomáhá organizovat náš čas. Tato komponenta je plně integrovaná, takže nám umožňuje spolupracovat s dalšími komponentami, jako jsou emaily či kontakty.

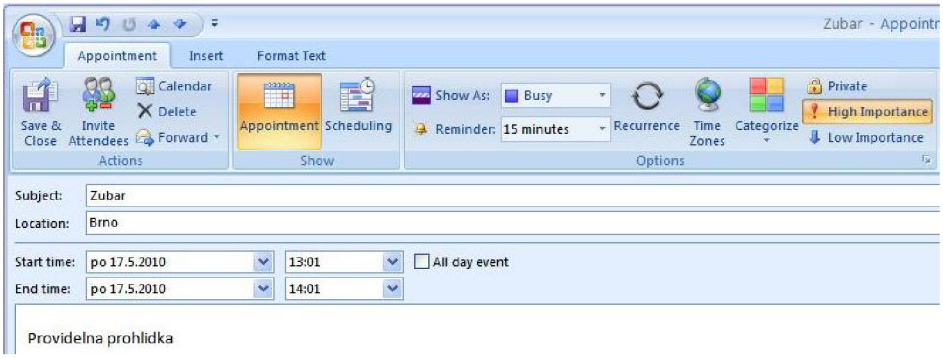
3.4.9.1 Nový záznam v kalendáři

Při vytváření záznamu do kalendáře máme možnost nastavit velké množství doplňkových informací, které nám pomohou ve snadnější orientaci a organizaci záznamů např.: důvod schůzky, podrobnější popis, důležitost, opakování, upozornění před schůzkou atd. Záznamu můžeme také přiřadit kategorii, jako v případě emailu. Jak vytvořit záznam do kalendáře:

- vytvoříme záznam
- vložíme předmět, místo
- vepíšeme podrobnější popis
- určíme začátek a konec
- nastavíme důležitost
- uložíme do kalendáře

```
Outlook.AppointmentItem appoint = (Outlook.AppointmentItem)
    this.Application.CreateItem(Outlook.OlItemType.olAppointmentItem);
appoint.Subject = "Zubar";
appoint.Location = "Brno";
appoint.Body = "Providelna prohlidka";
appoint.Start = new DateTime(2010, 5, 17, 13, 1, 2);
appoint.End = new DateTime(2010,5,17,14,1,2);
appoint.Importance = Outlook.OlImportance.olImportanceHigh;
appoint.Save();
```

Obr. 3.29 Kód vytvoření nové schůzky



Obr. 3.30 Vytvořená nová schůzka

3.4.9.2 Odstranění záznamu z kalendáře

Při odstraňování záznamu z kalendáře je postup podobný jako při mazání složky:

- vstoupíme do složky kalendáře
- vybereme datové záznamy
- najdeme odstraňovanou složku pomocí předmětu
- smažeme položku

```
//nacteme slozku se zaznami z kalendare
Outlook.MAPIFolder calFolder = Application.Session.
    GetDefaultFolder(Outlook.OlDefaultFolders.olFolderCalendar);
//ziskame datove zaznamy ze slozky
Outlook.Items item = calFolder.Items;
//vyhledame zaznam podle predmetu

Outlook.AppointmentItem calItem = item["Zubar"] as
    Outlook.AppointmentItem;

calItem.Delete();
```

Obr. 3.31 Kód smazání záznamu v kalendáři

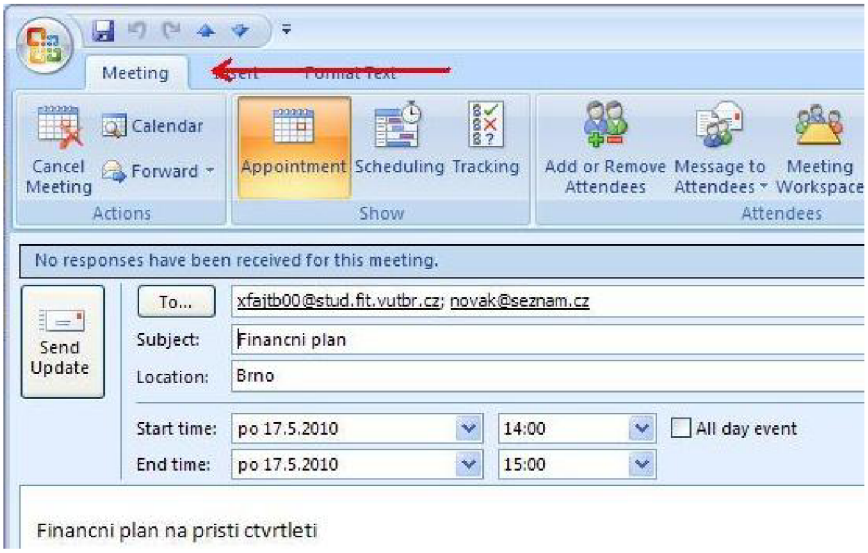
3.4.9.3 Práce s meetingy

Další užitečnou funkci, kterou kalendář Outlooku nabízí, je plánování meetingů. Jedná se v podstatě o stejný druh záznamu jako v předchozím případě, pouze s tím rozdílem, že do záznamu můžeme vložit účastníky a odeslat jim emailem oznámení.

- vytvoříme záznam kalendáře
- označíme jej jako meeting
- vyplníme údaje (předmět, tělo, místo, začátek, konec atd.)
- přidáme účastníky
- určíme, zda je účastník na meetingu vyžadován nebo je jeho účast volitelná
- uložíme záznam

```
Outlook.AppointmentItem meeting = (Outlook.AppointmentItem)
    this.Application.CreateItem(Outlook.OlItemType.olAppointmentItem);
//oznaceni zaznamu jako meeting
meeting.MeetingStatus = Outlook.OlMeetingStatus.olMeeting;
//vyplneni udaju
meeting.Subject = "Financni plan";
meeting.Body = "Financni plan na pristi ctvrtleti";
meeting.Location = "Brno";
meeting.Start = new DateTime(2010, 5, 17, 14, 0, 0);
meeting.End = new DateTime(2010, 5, 17, 15, 0, 0);
//pridani ucastniku
Outlook.Recipient ucastnik = meeting.Recipients.Add("xfajtb00@stud.fit.vutbr.cz");
Outlook.Recipient ucastnik2 = meeting.Recipients.Add("novak@seznam.cz");
ucastnik.Type = (int)Outlook.OlMeetingRecipientType.olRequired;
ucastnik2.Type = (int)Outlook.OlMeetingRecipientType.olOptional;
//muzeme zobrazit meeting pro dalisi editaci uzivatelem
//meeting.Display(true);
//ulozeni meetingu
meeting.Save();
```

Obr. 3.32 Kód vytvoření meetingu

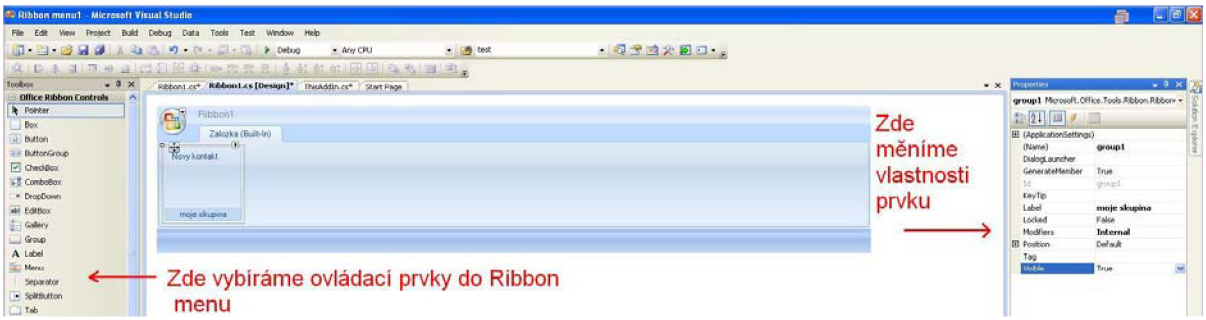


Obr. 3.33 Vytvořený meeting

3.4.10 Ribbon menu

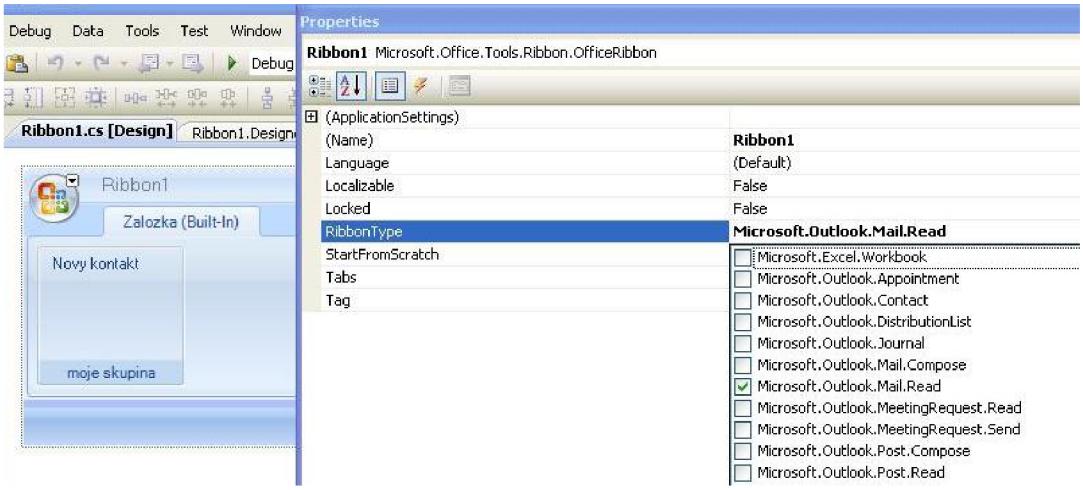
Se sadou Office 2007 přichází nový způsob organizování nástrojů pro práci s aplikací. Ribbon menu se snaží nabídnout uživateli přehlednější způsob organizování nástrojů jako změna velikosti písma, změna fontu, vkládání obrázků, vytváření grafů, vkládání vzorců a spoustu dalšího. Ribbon menu je dostupné ve většině aplikací ze sady Office 2007 (Word, Excel, Outlook...). InfoPath a Visio jej ale nepodporují.

Vytvoříme nový projekt a v Solution Exploreru vyvoláme kontextové menu. Vybereme *Add->New Item* a z nabízených možností vybereme Ribbon (visual designer). Otevře se nám návrhové okno s podobou budoucího Ribbon menu. Zde můžeme měnit podobu budoucího menu a jeho vlastnosti, popisy a podobně.



Obr. 3.34 Návrh Ribbon menu

Dalším krokem je určit, kde se námi vytvářené menu bude zobrazovat. V návrhu vybereme celé menu a v jeho vlastnostech rozvineme položku RibbonType a zaškrtneme položky, kde chceme zobrazit naše menu.

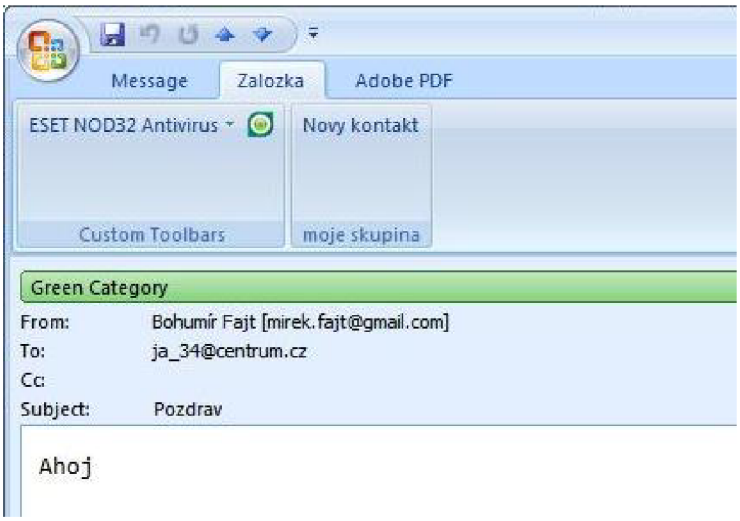


Obr. 3.35 Zobrazení Ribbon Menu

Na závěr vytvoříme činnost našemu menu. Při psaní emailu kliknutím na tlačítko otevřeme dialog pro vytvoření nového kontaktu. Do skupiny vložíme tlačítko a provedeme na něm dvojklik. Vytvoří se nám obslužná procedura pro kliknutí na dané tlačítko.

```
Outlook.ApplicationClass app = new Outlook.ApplicationClass();
Outlook.NameSpace nameSpace = app.GetNamespace("MAPI");
Outlook.MAPIFolder mapiFolder = nameSpace.GetDefaultFolder(
    Outlook.OlDefaultFolders.olFolderContacts);
Outlook.ContactItem newContact = (Outlook.ContactItem)
    mapiFolder.Items.Add(Outlook.OlItemType.olContactItem);
newContact.Display(true);
```

Obr. 3.36 Kód obslužné procedury tlačítka



Obr. 3.37 Vytvořené Ribbon menu

3.4.11 Obsluha událostí

Outlooku za svého běhu generuje velkou spoustu událostí, tyto události můžeme odchytávat a napsáním vlastní obslužné procedury na ně reagovat.

Ukážeme si, jak to vypadá v praxi. Odchytávanou událostí je `ItemSend`. Tato událost je vyvolána poté co uživatel klikne na tlačítko Odeslat, ale ještš před odesláním emailu. V následujícím příklad odchytí událost `ItemSend` a pokud uživatel odesílá email bez vyplněného předmětu, zeptá se jej, jestli chce takový email odeslat.

Jako první nastavíme obslužnou proceduru pro událost `ItemSend`.

```
//Nastaveni obsluzne procedury pro udalost ItemSend
this.Application.ItemSend += new
  Outlook.ApplicationEvents_11_ItemSendEventHandler(Obsluha_ItemSend);
```

Obr. 3.38 Kód nastavení obslužné procedury

V obslužné proceduře zjistíme, jestli má odesílaný email vyplněný předmět. Jestliže ne, zeptáme se uživatele, zda chce i přesto email odeslat.

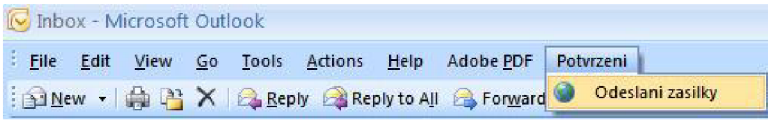
```
void Obsluha_ItemSend(object item, ref bool cancel)
{
    //zprava, ktera vyvolala udalost
    Outlook.MailItem mail = item as Outlook.MailItem;
    string otazka = "Opravdu chces odeslat email bez predmetu?";
    string popis = "Odeslat?";
    DialogResult vysl;
    MessageBoxButtons tlac = MessageBoxButtons.YesNo;

    if (mail.Subject == null)
    {
        //dotaz jestli chce uzivatel odeslat zpravu i bez predmetu
        vysl = MessageBox.Show(otazka, popis, tlac);
        //vyhodnoceni vysledku dotazu
        if (vysl == DialogResult.No)
            cancel = true;

    }
}
```

Obr. 3.39 Kód obslužné události ItemSend

3.4.12 Komplexní ukázka

Nyní si ukážeme plugin spojující několik předchozích podkapitol. Plugin vytvoří nové menu s jedním tlačítkem. Po jeho aktivaci se zobrazí dialogové okno, ve kterém doplníme jméno, příjmení, číslo objednávky a email. V dialogovém okně bude také tlačítko Send, na které kliknutím vytvoříme emailovou zprávu, obsahující údaje z dialogového okna. Dále bude plugin odchytávat událost `ItemSend`. Po jejím příchodu zavře dialogové okno.

- nastavíme obslužnou proceduru pro událost `ItemSend` viz kapitola 3.4.11
- vytvoříme menu viz kapitola 3.4.2
- vytvoříme dialogové okno (v Project Exploreru vyvoláme kontextové menu a zvolíme *Add->New Item...* ze šablon zvolíme *Windows Form)*
- navrhneme grafickou podobu dialogového okna
- napíšeme obslužný kód, vyvolaný kliknutím na tlačítko v dialogovém okně

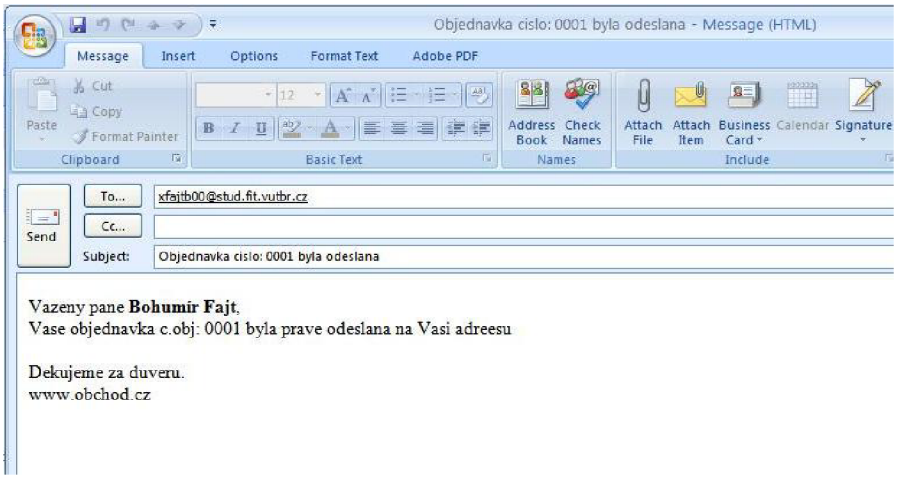


Obr. 3.40 Vytvořené menu

Obr. 3.41 Navržené dialogové okno

```
#region Button1 Event
private void button1_Click(object sender, EventArgs e)
{

    SendEmail(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text);

}
#endregion


#region odeslani emailu
private void SendEmail(string name, string sourname, string cObj, string email)
{
    string body;
    //slozeni tela zpravy
    body = "Vazeny pane <STRONG>%name%</STRONG> <STRONG>%sourname%</STRONG>, <BR /> Vase ob
    StringBuilder sb = new StringBuilder(body);
    sb.Replace("%name%", name.ToString());
    sb.Replace("%sourname%", sourname.ToString());
    sb.Replace("%cObj%", cObj.ToString());

    Outlook.Application app1 = new Microsoft.Office.Interop.Outlook.Application();
    Outlook.MailItem mail1 = (Outlook.MailItem)
                app1.CreateItem(Microsoft.Office.Interop.Outlook.OlItemType.olMailItem);
    mail1.To = email;
    mail1.Subject = ("Objednavka cislo: " + cObj+" byla odeslana");
    mail1.BodyFormat = Microsoft.Office.Interop.Outlook.OlBodyFormat.olFormatHTML;
    mail1.HTMLBody = sb.ToString();
    mail1.Display(true);
}
```

Obr. 3.42 Kód pro tlačítko v dialogovém okně



Obr. 3.43 Výsledný email

4 Závěr

Cílem mojí bakalářské práce bylo, projít si možnosti psaní rozšíření pro MS Outlook a demonstrovat je na příkladech. Ukázal jsem dva možné způsoby psaní těchto rozšíření, a to jak psaní maker přímo ve vývojovém prostředí integrovaném do Outlooku a programovacím jazyce Visual Basic for Application (VBA), tak také psaní COM Add-in ve vývojovém prostředí Visual Studio 2008 na architektuře Visual Studio Tools for Offce (VSTO). Programovací jazyk pro COM Add-in jsem zvolil C#.

Na názorných příkladech, včetně ukázek zdrojových kódů a výsledků činnosti napsaného pluginu, jsem ukázal všechny základní možnosti přizpůsobení si MS Outlook, zejména pak vytváření menu a toolbaru, úpravy kontextového menu a Ribbon menu. Předvedl jsem práci se základními komponentami (složky, email, kalendář), jak je vytvářet, mazat a upravovat. Prošel jsem krok za krokem postup při vytváření pluginu od založení nového projektu ve Visual Studiu, až po spuštění napsaného kódu.

Formu textu práce jsem zvolil tak, aby byl srozumitelný pro ty, kteří chtějí s psaním pluginů teprve začít a doposud nemají žádné zkušenosti s tímto tématem. Spousta příkladů z této práce jsou využitelné nejen pro Outlook, ale dají se využít i v dalších aplikacích ze sady MS Office. Umožnit budoucím vývojářům, rychle se zorientovat v daném tématu a ukázat základní prvky při tvorbě pluginu tak, aby mohli rychleji přejít k vývoji svého pluginu.

Zdroje informací

[1] Sue Mosher, Microsoft Outlook programing: Jumpstartfor Power Users and Administrators, Elsevier Science Ltd, 2002, ISBN: 1555582869

[2] To distribute Microsoft Outlook VBA code to other users, [online], http://www.outlookcode.com/article.aspx?id=28

[3] Outlook Add-ins with Visual Studio Tools for Office, [online], http://www.outlookcode.com/article.aspx?ID=42

[4] Registry Entries for Application-Level Add-Ins, [online], http://msdn.microsoft.com/en-us/library/bb386106.aspx

[5] COM Add-ins vs. Application-Specific Add-ins, [online], http://msdn.microsoft.com/en-us/library/aa165197(office.10).aspx

[6] VSTO 3.0 for Office 2007 Programming, Packt Publishing, 2009, ISBN: 1847197523

Seznam příloh

Příloha A: Struktura dat na přiloženém CD

Tato příloha popisuje obsah CD nosiče, na kterém jsou umístěny zdrojové kódy a technická zpráva k této práci.

Složka nazvaná: “Ukázky COM Add-in“ obsahuje Visual Studio 2008 projekty včetně zdrojových kódů příkladů. Podsložky jsou nazvány čísly, odpovídající číslům kapitol, ve kterých byla daná ukázka použita.

Složka nazvaná:“Ukázky VBA“ obsahuje soubor “VbaProject.otm“, ve kterém je kód příkladů použitý v kapitole 2.

Složka “Dokumentace“ obsahuje technickou zprávu pod názvem BP.pdf