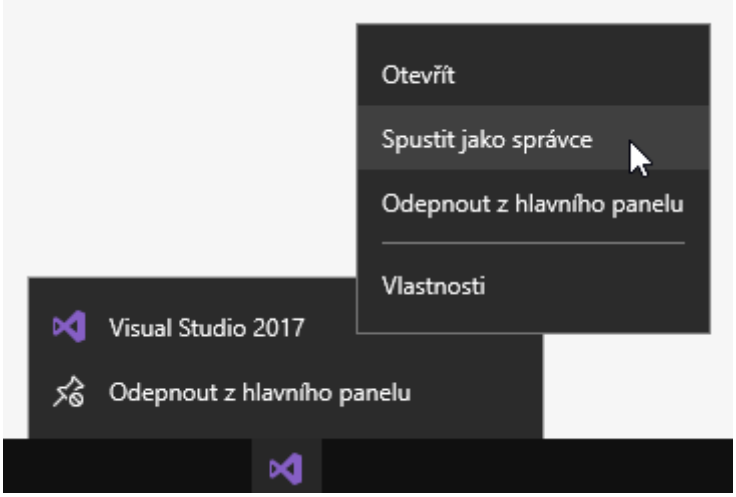
Jak vytvořit .NET komponentu (DLL, COM Class) pro Excel? A proč? A co k tomu budeme potřebovat?

Velký Visual Basic (dnes VB.NET) se rozešel s Visual Basicem pro aplikace (VBA) před cca 16 lety. A i když dnes už VB.NET není zdaleka „in“, přeci jen ušel značný kus cesty a je násobně silnější než stařičké a unavené VBA. Proto není na škodu se na něj obrátit, když už si ve VBA nevíme rady. No jo, ale co nás to bude stát? Dnes už nic. Microsoft již před časem dal k dispozici Visual Studio Community zdarma, které na programování komponent a doplňků pro Microsoft Excel (Microsoft Office) budeme potřebovat (i kdyby ne, pořád je tu projekt SharpDevelop). Takže prostředí máme a pro programátory ve VBA nebude zatěžko porozumět „dialektu“ VB.NET (můžete se rozhodnout i pro jazyk typu C). Přeci jen je ale tento článek určen těm, kteří již nějakou zkušenost s Visual Studiem a VB.NET mají. Jdeme na to.

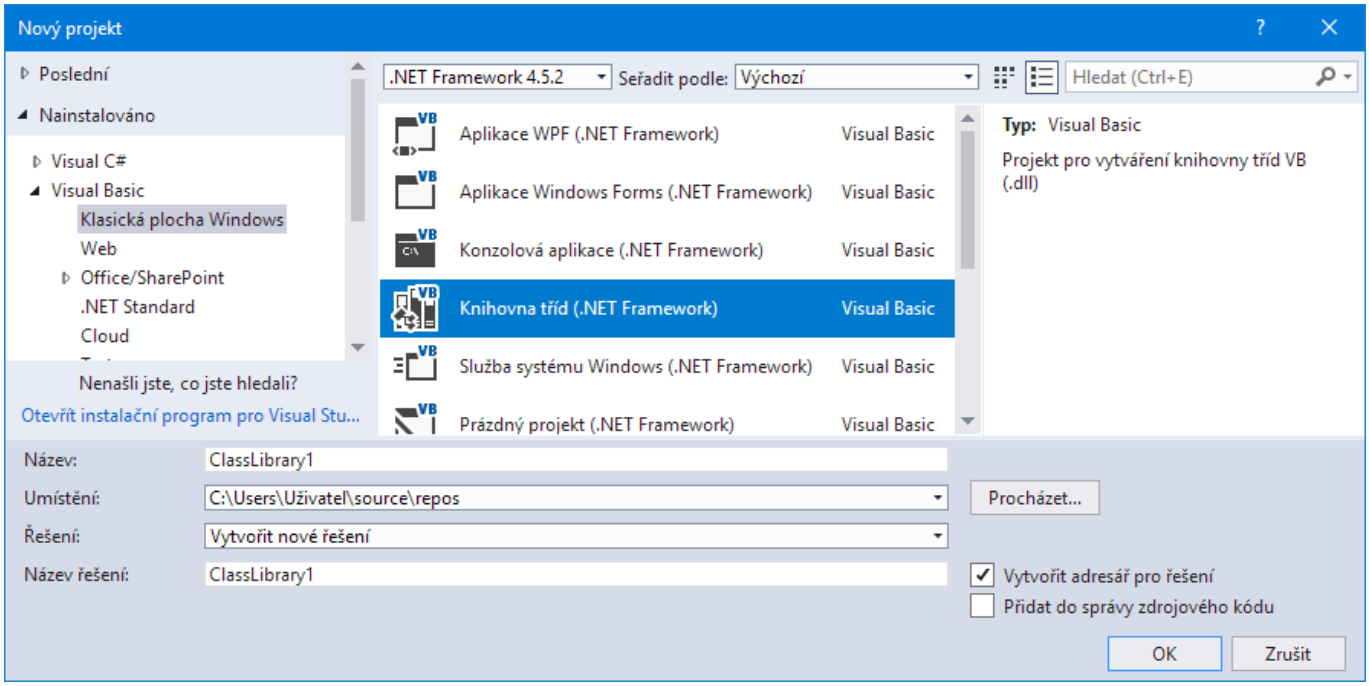
Vytvoření .NET komponenty (DLL) ve Visual Studiu

Spustíme Visual Studio s právy správce (důvod bude zmíněn později).



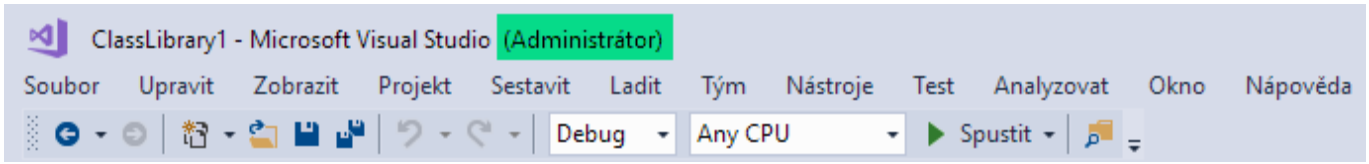
Visual Studio – spuštění s právy správce

Vytvoříme nový projekt postavený na obecné třídě s názvem ClassLibrary1.



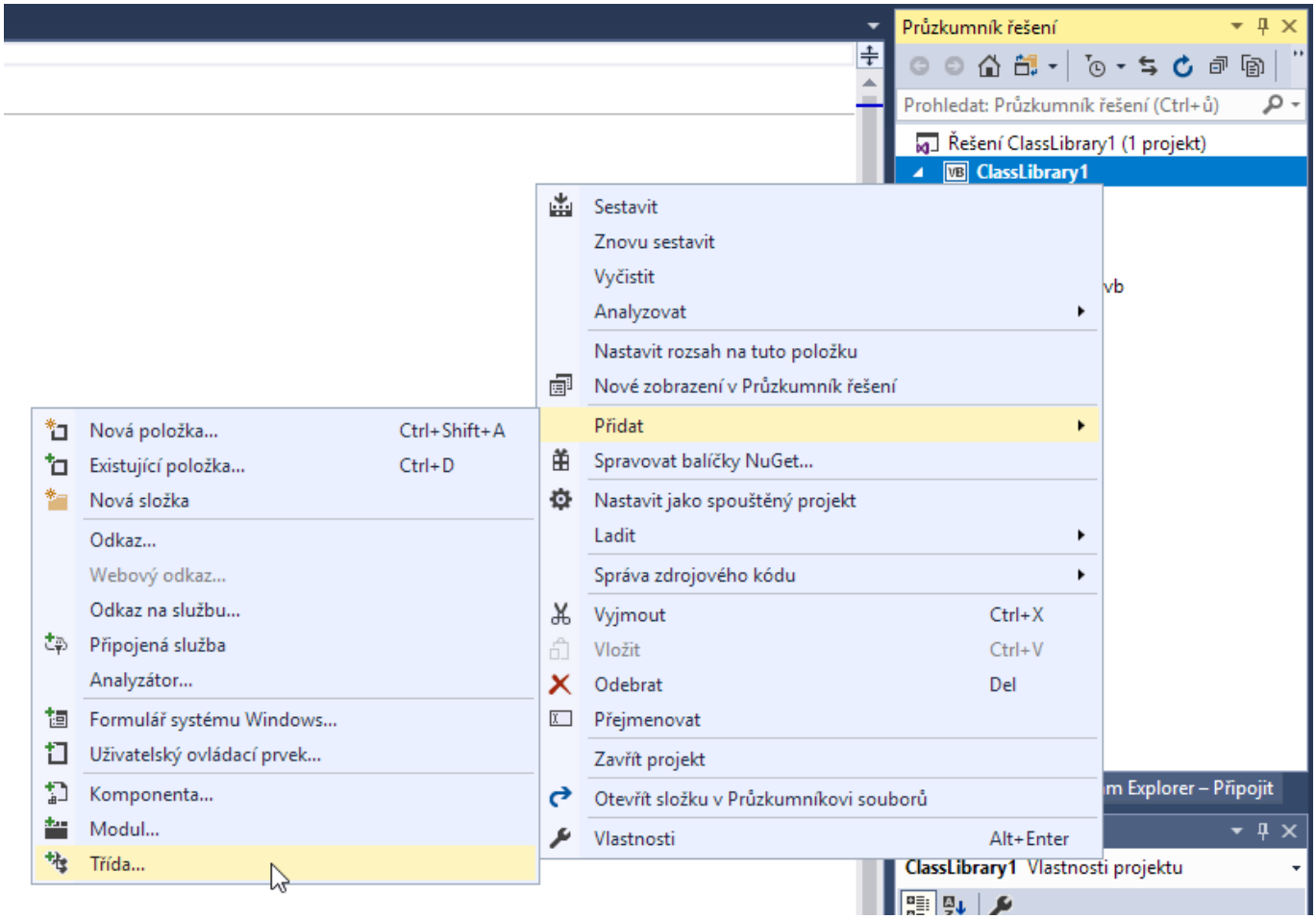
Visual Studio – knihovna tříd (DLL)

Ověříme si, že je aplikace otevřena skutečně s právy správce.

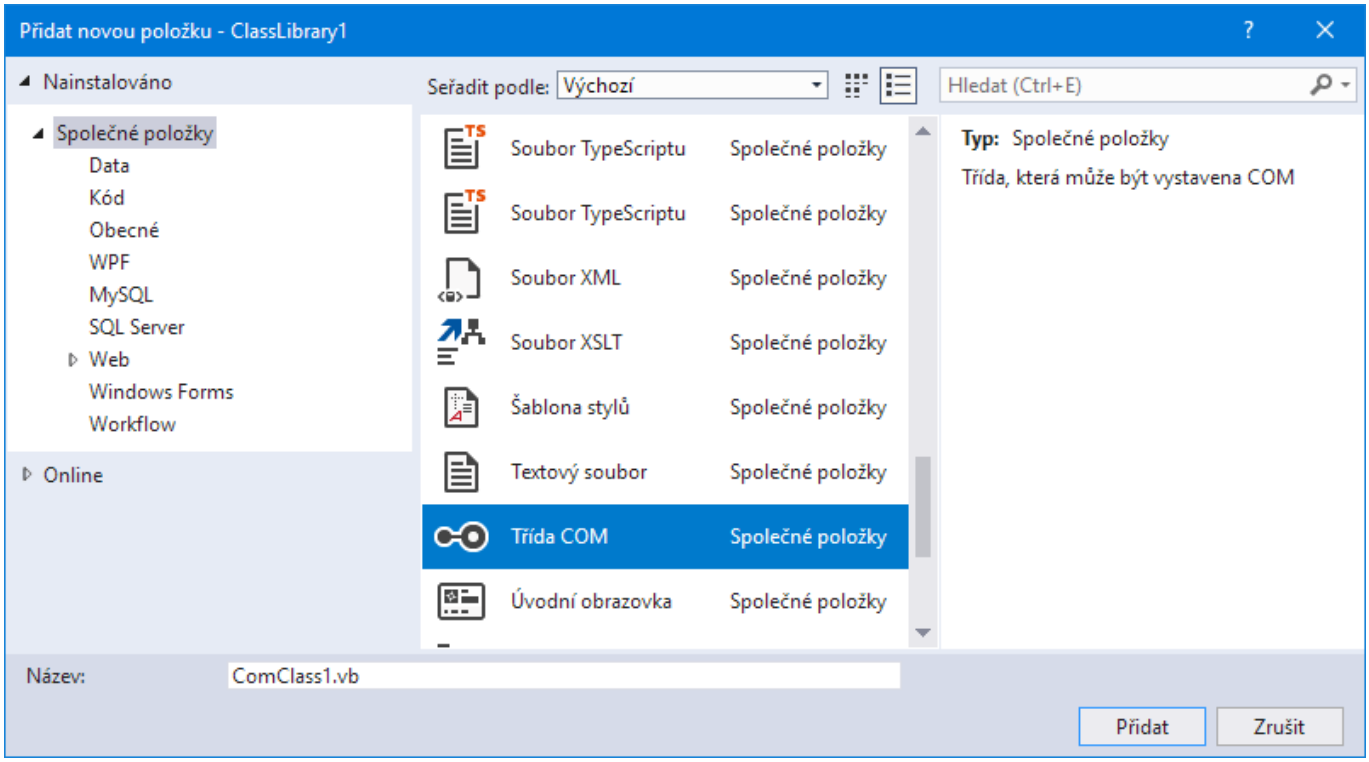


Visual Studio – práva správce

Hned vzápětí založíme i třídu typu COM Class s názvem ComClass1 (jakýsi interface, poslouží pro vlastní komunikaci s aplikací Excel).



Visual Basic – třída COM



Visual Studio – třída COM

Její obsah bude vypadat nějak takto:

```vb
<ComClass(ComClass1.ClassId, ComClass1.InterfaceId, ComClass1.EventsId)> _
Public Class ComClass1

#Region "identifikátory GUID modelu COM"
    ' Tyto GUID identifikátory poskytují modelu COM identitu této třídy
    ' a jejích COM rozhraní. Pokud je změníte, existující
    ' klienti nebudou moci ke třídě přistupovat.
    Public Const ClassId As String = "de836f9b-1b1c-4195-ab08-096493fd0de2"
    Public Const InterfaceId As String = "ac93b09e-4381-44fe-958c-2ba805812e2f"
    Public Const EventsId As String = "de5919c2-0c63-400d-8550-4b1a0d7bb463"
#End Region

    ' Vytvořitelná COM třída musí obsahovat proceduru Public Sub New()
    ' bez vstupních parametrů, jinak třída nebude
    ' registrována v COM registru a nebude ji možné vytvořit
    ' pomocí metody CreateObject.
    Public Sub New()
        MyBase.New()
    End Sub

End Class
```

Třída COM – kód

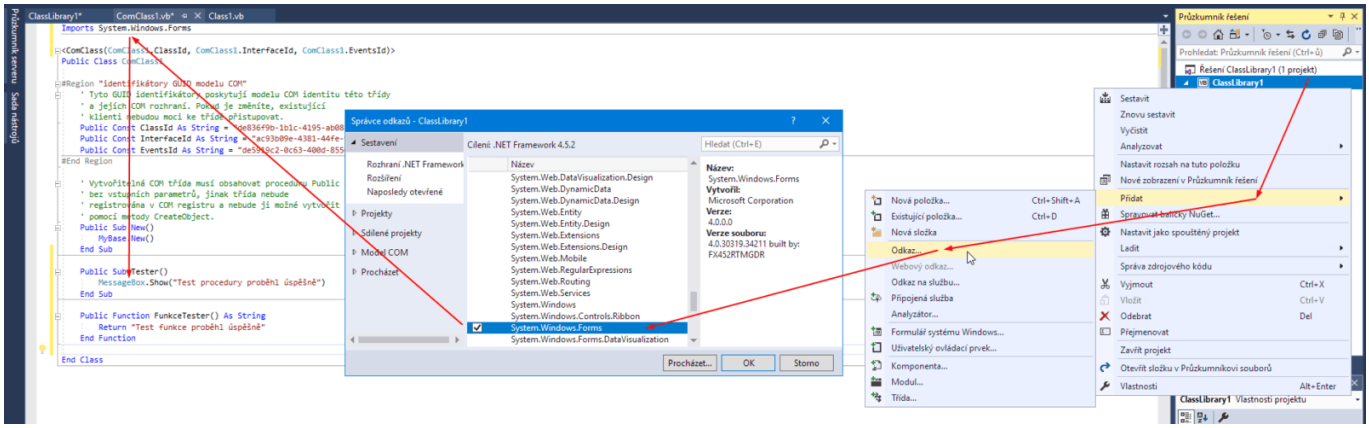
Teoreticky by bylo možné tuto třídu vytvořit i z obecné třídy – identifikátory GUID lze vygenerovat s pomocí menu Nástroje, s atributem ComClass už jsem ale narazil na problém v rámci InteropServices, zkrátka, nestojí to za to. Doplníme ještě testovací proceduru a funkci.

ComClass1.vb | Class1.vb

```
Imports System.Windows.Forms

<ComClass(ComClass1.ClassId, ComClass1.InterfaceId, ComClass1.EventsId)>
Public Class ComClass1

#Region "identifikátory GUID modelu COM"
    ' Tyto GUID identifikátory poskytují modelu COM identitu této třídy
    ' a jejích COM rozhraní. Pokud je změníte, existující
    ' klienti nebudou moci ke třídě přistupovat.
    Public Const ClassId As String = "de836f9b-1b1c-4195-ab08-096493fd0de2"
    Public Const InterfaceId As String = "ac93b09e-4381-44fe-958c-2ba805812e2f"
    Public Const EventsId As String = "de5919c2-0c63-400d-8550-4b1a0d7bb463"
#End Region

    ' Vytvořitelná COM třída musí obsahovat proceduru Public Sub New()
    ' bez vstupních parametrů, jinak třída nebude
    ' registrována v COM registru a nebude ji možné vytvořit
    ' pomocí metody CreateObject.
    Public Sub New()
        MyBase.New()
    End Sub

    Public Sub Tester()
        MessageBox.Show("Test procedury proběhl úspěšně.")
    End Sub

    Public Function FunkceTester() As String
        Return "Test funkce proběhl úspěšně."
    End Function

End Class
```

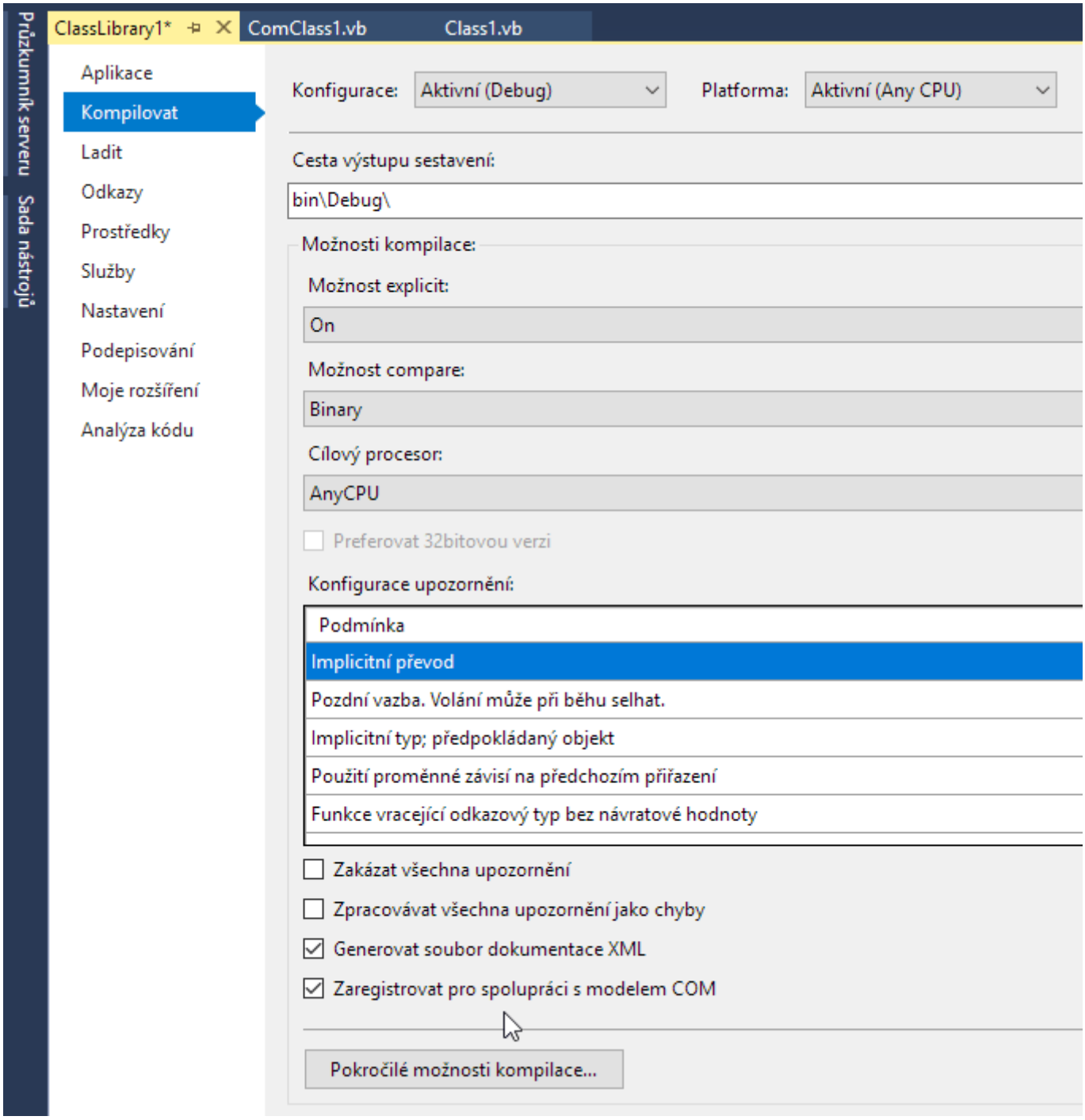
Třída COM – vlastní procedura a funkce

V proceduře Tester je užit MessageBox, univerzální bratříček našeho známého MsgBoxu z VBA. I zde je sice stále dostupný, nicméně se odkážeme na modernější formu dialogového okna. Na to potřebujeme do projektu zahrnout odkaz na knihovnu a jmenný prostor (namespace), v němž se MessageBox nachází. Viz také Projekt / Přidat odkaz...



Visual Studio – reference

Pojďme se nyní podívat na vlastnosti projektu (Projekt / Vlastnosti ...).

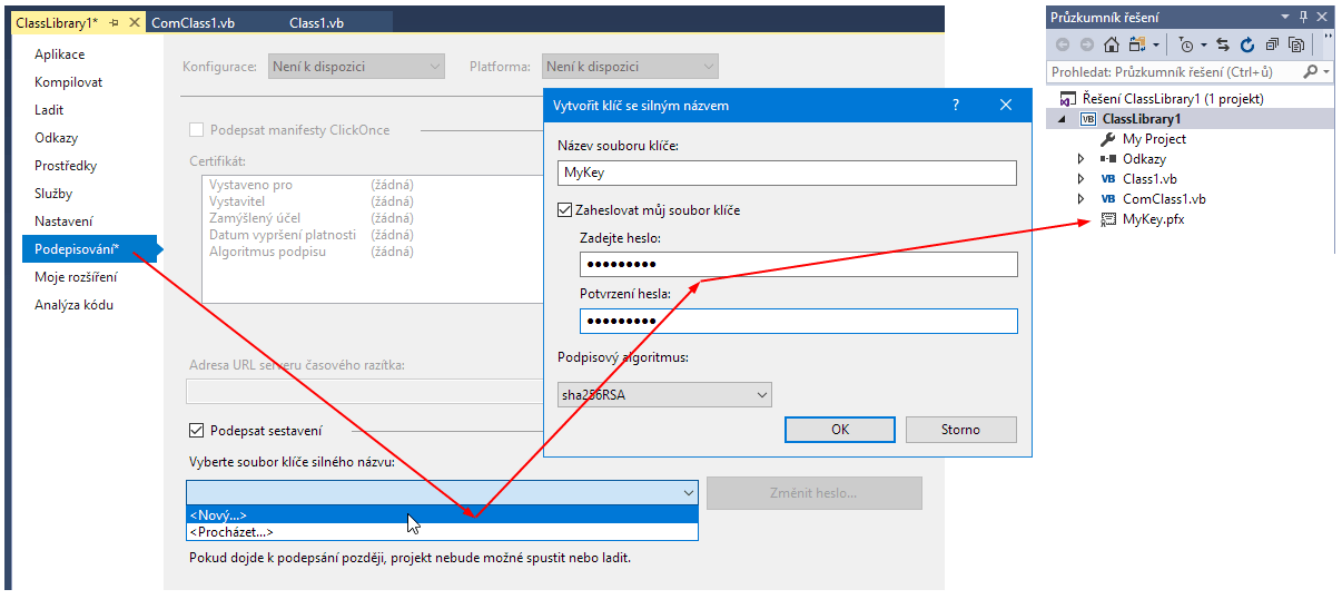


Visual Studio – registrace COM

Pod položkou Kompilace najdeme volbu Zaregistrovat pro spolupráci s modelem COM. Jejím cílem je vytvoření tzv. TLB souboru (Type Library File) a následné registrace informací v něm uložených (operace vyžaduje práva správce). Jedná se o jakýsi definiční soubor obsahující údaje o vlastnostech, metodách a funkcích samotné knihovny DLL. Odkazuje se na něj VBA prostřednictvím svých referencí

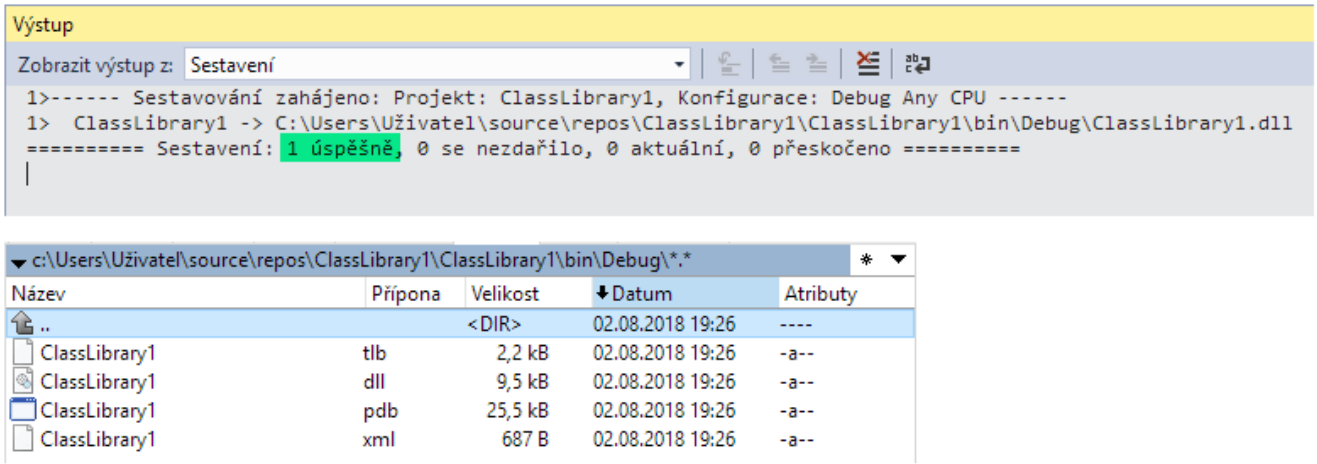
a poslouží i pro Object Browser a IntelliSense. Pro první testování nechte danou volbu zaškrtnutou, nicméně ve finále bude TLB soubor vygenerován až po distribuci DLL knihovny na konkrétní počítač.

Dále si projekt podepíšeme. Důvody si zde rozebírat nijak nebudeme, k tématu se váží pojmy jako „strong assembly name“, „GAC“ a další.



Visual Studio – podepsání projektu

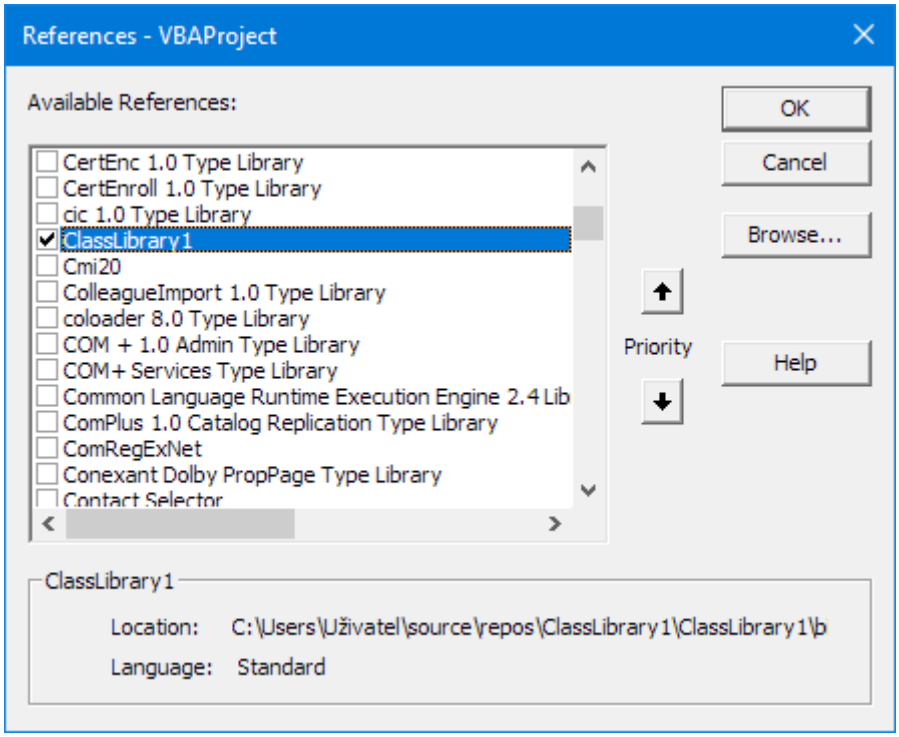
Jsme ve finále a zbývá už jen vygenerovat potřebné soubory (Sestavit /Sestavit řešení)



Visual Studio – sestavení řešení (DLL pro Excel)

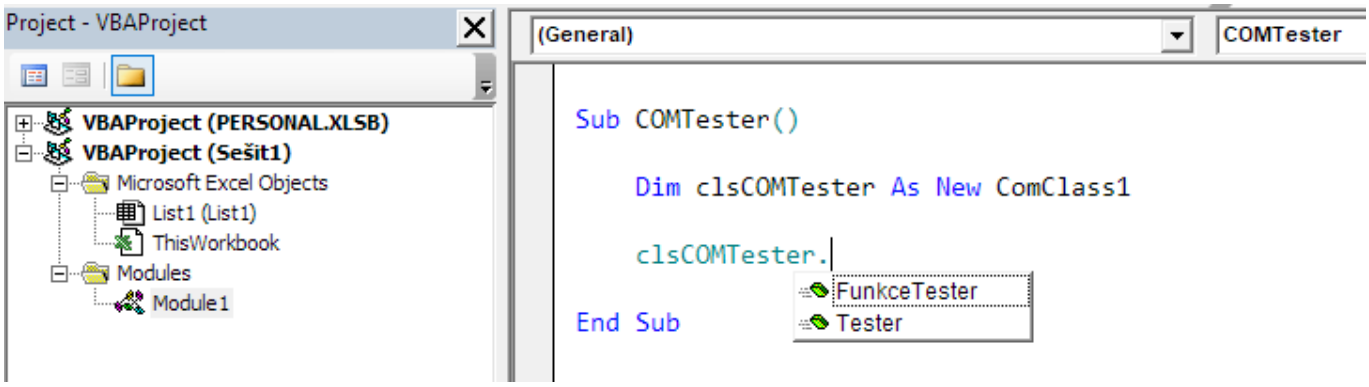
A hurá do Excelu a VBA vyzkoušet náš výtvor...

Test DLL knihovny ve VBA

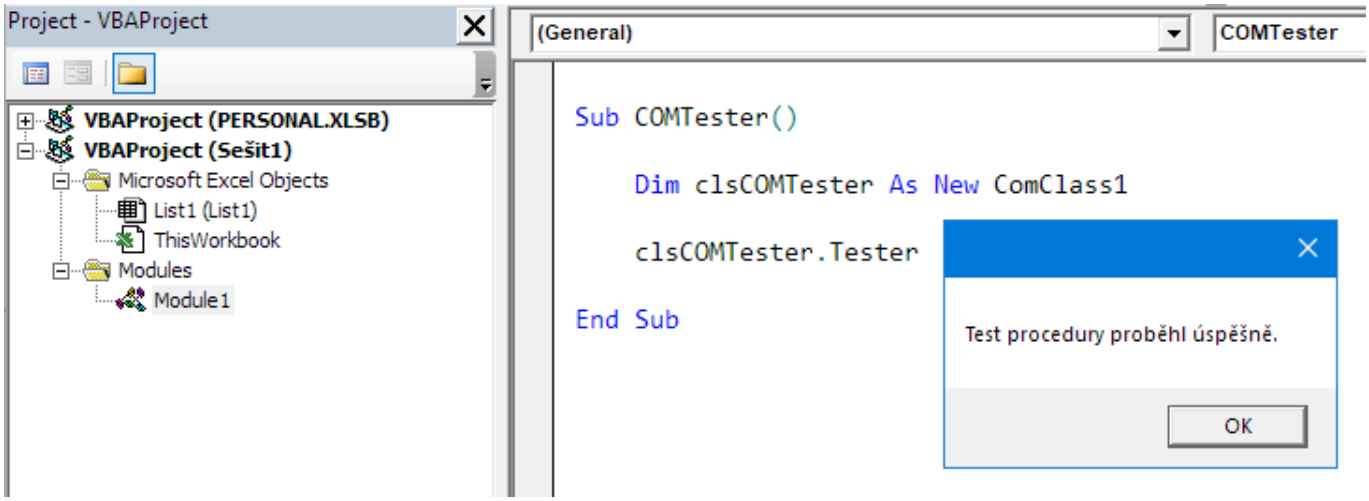


Microsoft Excel – test knihovny pod VBA

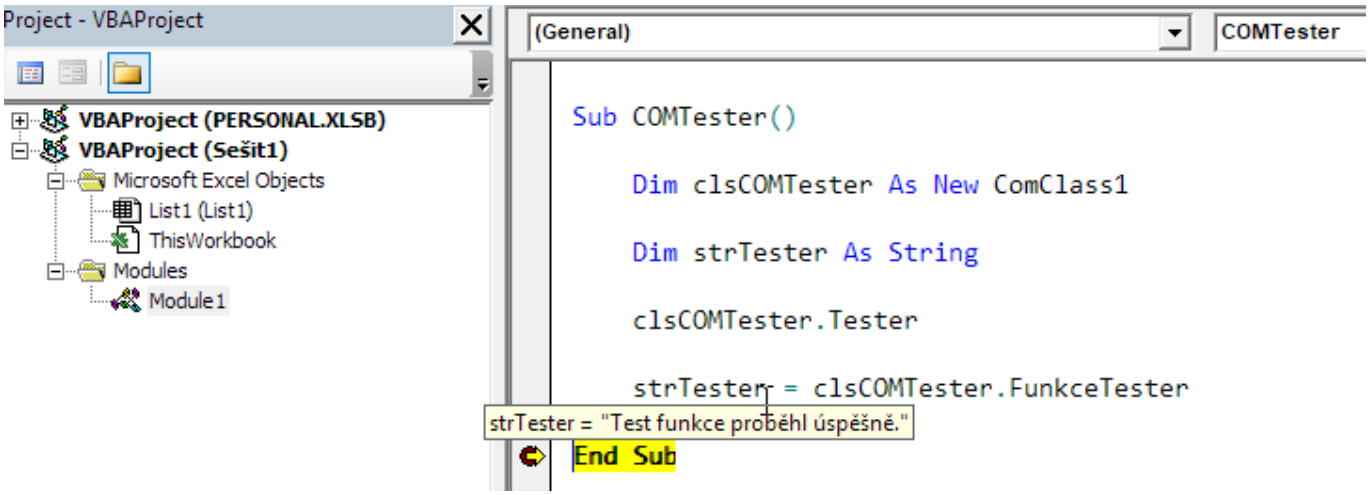
V deklaraci třídy nezapomeňte na klíčové slovo New. Jak je vidět níže, IntelliSense se chytne.



Microsoft Excel – test knihovny pod VBA



Microsoft Excel – test knihovny pod VBA

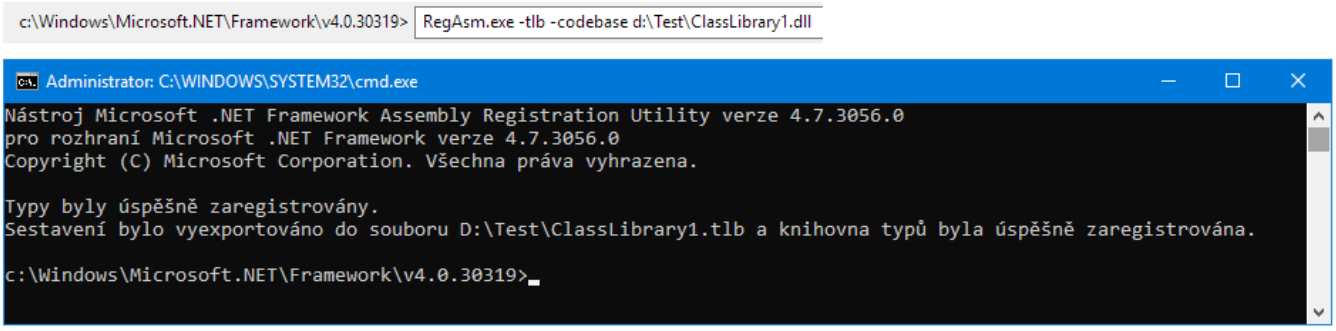


Microsoft Excel – test knihovny pod VBA

Nutno poznamenat, že o registraci knihovny se na počítači postaralo samotné Visual Studio (reference se z VBA odkazuje do složky projektu). To samozřejmě nic neříká o tom, jak danou knihovnu distribuovat uživatelům. Žádný instalátor ovšem nebude potřeba. Postačí si zkopírovat obsah složky Debug (bez TLB souboru), přenést ho na počítač klienta (v příkladu dále složka D:\Test) a provést registraci DLL knihovny. Jak?

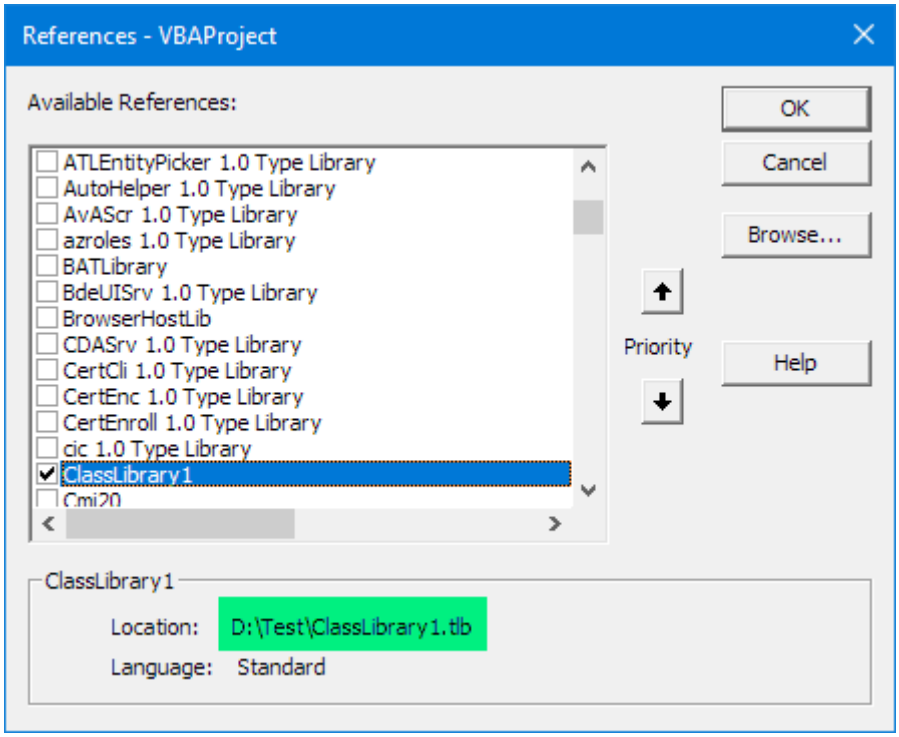
Registrace DLL

Možná si vybavujete příkaz Regsvr32 ve Windows. Ten pro dané účely ovšem není vhodný. My potřebujeme utilitku s názvem Regasm. Najdeme ji v instalačních složkách Frameworku. Příklad níže uvádí spuštění v Total Commanderu (opět s právy správce nutnými v rámci příkazového řádku). Po spuštění (Shift + Enter) dojde k vytvoření nového definičního souboru TLB a registraci samotné knihovny. Pozn. Právě parametr codebase potřebuje „strong assembly name“, tj. podepsané řešení.



Registrace knihovny

Reference pod VBA se poté bude vázat na reálné umístění TLB souboru u klienta. Další postup užití se neliší od již uvedeného.



Vlastní funkce (UDF)

Jednu věc jsem úmyslně vynechal – návrh vlastních funkcí pod VB.NET, které mají být dostupné na listu Excelu (tzv. User Defined Functions – UDF). Ke stávajícímu bychom museli přidat problémy s knihovnou mscoree.dll, hrát si s registry a nakonec stejně nejspíš ručně vybírat funkci v okně Doplňky Excelu pod tlačítkem Automatizace. Jako jednodušší mi přijde jen o kus dál posunuté předložené řešení – jádro naprogramované ve Visual Studiu pouze obalíme deklarací procedury funkce pod VBA (vytvoříme tzv. wrapper).

Soubory ke stažení:
excel_com_class_dll.zip