Official Guide

Add-in Express™ .net

for Microsoft Visual Studio Tools for Microsoft Office

Getting Started

Add-in Express 2007 for VSTO

Add-in Express™ 2007 for VSTO

Document version **3.1**

Revised at **31-Jan-07**

Product version **3.x**

Table of Contents

Introduction

Add-in Express 2007 for VSTO is a tool designed to simplify and speed up the development of Office add-ins as well as Excel and Word customizations in Visual Studio Tools for Office (VSTO 2005 and VSTO 2005 SE). It provides a number of specialized components that allow the developer to jump through the interface-programming phase to the functional programming phase with a minimal loss of time.

System Requirements

Supported IDE Versions

- Visual Studio .NET 2005 Tools for Microsoft Office
- Visual Studio .NET 2005 Tools for Microsoft Office Second Edition

Or Visual Studio .NET 2005 Tools for Microsoft Office installed over:

- Visual Studio.NET 2005 Team System
- Visual Studio.NET 2005 Professional Edition
- Visual Basic .NET 2005 Standard Edition

Host Applications

- Microsoft Outlook 2003 and higher
- Microsoft Excel 2003 and higher
- Microsoft Word 2003 and higher
- Microsoft PowerPoint 2003 and higher
- Microsoft Visio 2003 and higher

Technical Support

Add-in Express is developed and supported by the Add-in Express Team, a branch of Add-in Express Ltd. You can obtain technical support using any of the following methods.

Add-in Express Web Site

The Add-in Express web sire at www.add-in-express.com provides a wealth of information and software downloads for Add-in Express developers, including:

The HOWTOs section, sample projects that answer most common "how to" questions.

ADX Toys, entire and "open sourced" add-ins for popular Office applications.

Forums. We are actively participating in these forums. Really.

Internet E-mail

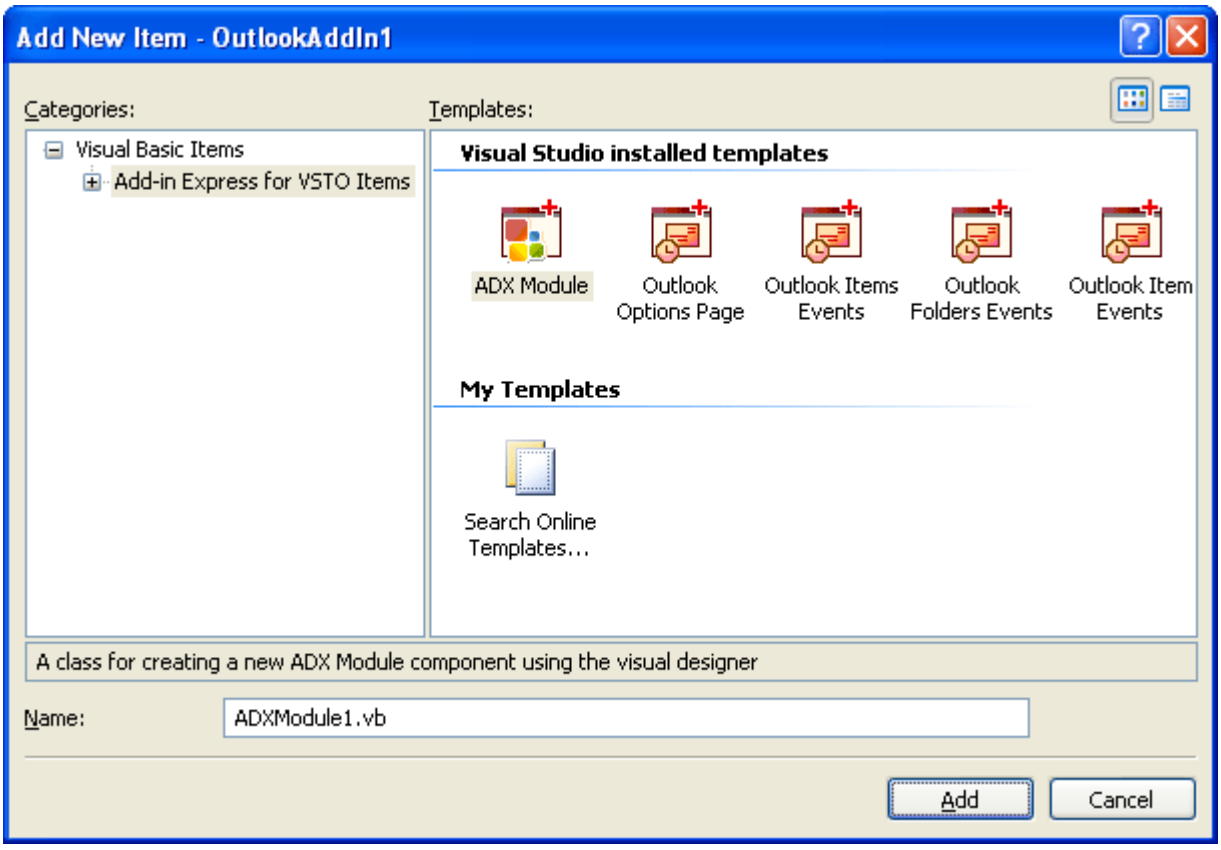
For technical support through the Internet, e-mail us at support@add-in-express.com.

Technical Support via Instant Messengers

If you are a subscriber of our Premium Support Service and need help immediately, you can request technical support via an instant messenger, e.g. Windows/MSN Messenger or ICQ. Please ask us at http://www.add-in-express.com/premium-area/contact-us.php.

Getting Started

After you install Add-in Express 2007 for VSTO, it adds several items to the Add New Item dialog.



When starting any new VSTO project, your first step is to add an Add-in Express Module component to your project.

Add-in Express for VSTO components

Add-in Express Module

The Add-in Express Module is the core component of Add-in Express 2007 for VSTO. It represents an Add-in in an Office application and allows centralizing all programming logics in one place. Its designer allows adding other Add-in Express components and sets their properties at design-time. It also provides all events of the host application. Note, some Outlook-specific events, as well as the Outlook Option page (Folder Propery pages) should be added via the Add New Item dialog.

For Outlook add-ins, you specify the Options page and Folder Property pages (see Outlook Property Page). See the following chapters for the Add-in Express components you add onto the Add-in Module: Office 2007

Ribbon Components, Command Bars, Command Bar Controls, Built-in Control Connector, Keyboard Shortcut, Outlook Bar Shortcut Manager, and Outlook Forms Manager.

Use the OnStartupComplete and OnBeginShutdown events to handle add-in startup and shutdown.

Adding an Add-in Express Component to Add-in Express Designer

To add any Add-in Express for VSTO component onto an Add-in Express Module, activate the Add-in Express Module designer window and use commands available either in the Properties Window or in the context menu. To activate the Add-in Express Module designer window, in the Solution Explorer window, right-click the Add-in Express Module item and choose the View Designer popup menu item.

Office 2007 Ribbon Components

Office 2007 presented a new Ribbon user interface. Microsoft states that the interface makes it easier and quicker for users to get the results they want. The developers extend this interface by using the XML markup that the add-in should return to the host through the appropriate interface.

Add-in Express provides some 20 Ribbon-related components to give you the full power of the Ribbon UI customization features.

You start with ADXRibbonTab, ADXRibbonOfficeMenu, and ADXRibbonQuickAccessToolbar that get the task of creating the markup upon them. You add controls to a tab or menu using the convenient tree-view-like editor that allows you to see all the items of the tab or menu at a glance. Please, note Microsoft require developers to use the StartFromScratch parameter (see the StartFromScratch property of AddinModule) when customizing Quick Access Toolbar.

ADXRibbonTab and ADXRibbonOfficeMenu support all types of Ribbon controls including regular and button groups; regular, edit, combo and check boxes; buttons and split buttons; labels and dropdown lists; galleries and menus; separators and dialog launchers.

Also, note, you can use pre-Office2007 command bars in the Office 2007 add-ins. In this case, your toolbars will be added to the Add-ins ribbon tab.

Office 2007 Custom Task Panes

To allow further customization of its applications, Office 2007 provides custom task panes. Add-in Express supports task panes by equipping the Add-in module with the TaskPanes property. Add a UserControl to your project, add an item to the TaskPanes collection, and set up the item by choosing the control in the

ControlProgId property and filling in the Title property. Add your reaction to the TaskPaneXXX event series of the Add-in module and the DockPositionStateChange and VisibleStateChange events of the task pane.

Command Bars

Microsoft Office 2000-2003 supplied us with a common term for Office toolbars, menus, and context menus. This term is Command Bar. While look-n-feel of all Office command bars is the same, Outlook command bars are different from command bars of other Office applications. They are different for the two main Outlook window types – for Outlook Explorer and Outlook Inspector windows. Additionally, different Outlook Inspector window types show different command bars. Accordingly, Add-in Express provides you with ADXCommandBar, ADXOlExplorerCommandBar, ad ADXOlInspectorCommandBar components.

To add a command bar to your project, use one of the following ADXAddinModule commands:

- Add CommandBar
- Add ExplorerCommandBar - Outlook-specific
- Add InspectorCommandBar - Outlook-specific

The main property of any CommandBar component is CommandBarName. If its value is not equal to the name of any built-in command bar of the host application, then you are creating a new toolbar. If its value is equal to any built-in command bar of the host application, then you are connecting to a built-in command bar. To find out the built-in command bar names, use the free Built-in Controls Scanner utility (http://www.add-in-express.com/downloads/controls-scanner.php).

To position your command bar, use the Position, Left, Top, and RowIndex properties.

To speed up add-in loading when connecting to an existing command bar, set the Temporary property to False. To make the host application remove the command bar when the host application quits, set the Temporary property to True.

Outlook-specific toolbars provide the FolderName, FolderNames, and ItemTypes properties that add context-sensitive features to the toolbar.

Command Bars in Office 2007

You can use pre-Office2007 command bars in the Office 2007 add-ins. In this case, your toolbars will be added to the Add-ins ribbon tab.

Command Bar Controls

The Office Object Model (OOM) includes the following command bar controls CommandBarButton, CommandBarComboBox, and CommandBarPopup. Using the correct property settings of the CommandBarComboBox component, you can extend the list with edits and dropdowns. Nevertheless, this list is extremely short. Add-in Express allows extending this list with any control of your choice using the Add-in Express Extensions for Microsoft Office Toolbars, which is a plug-in for Add-in Express.

What follows below is a list of controls available in the ADXCommandBarControl Collection Editor dialog (it opens when you edit the Controls collection of a command bar):

- ADXCommandBarButton
- ADXCommandBarComboBox
- ADXCommandBarEdit
- ADXCommandBarPopup
- ADXCommandBarDropDownList
- ADXCommandBarControl (you use this item to add built-in controls to your command bars)
- ADXCommandBarAdvancedControl (you use this item with Add-in Express Extensions for Microsoft Office Toolbars, http://www.add-in-express.com/office-toolbar-controls/)

Please, note that due to the nature of command bars (remember a 'command bar' stands for toolbar, menu, and context menu), [context] menu items can be buttons, combo boxes, and pop-ups.

Populate your command bar using the Controls collection both at run-time and at design-time. Every control (built-in and custom) added to this collection will be added to the corresponding toolbar at your project startup.

The main property of any command bar control is the Id property. To add a built-in control to your toolbar, specify its Id in the Id property of the command bar control. To find out the Ids of every built-in control in the host application, use the free Built-in Controls Scanner utility (http://www.add-in-express.com/downloads/controls-scanner.php). To add a custom control to the toolbar, leave the Id property unchanged.

Set up a control's appearance using a great number of its properties, such as Enabled and Visible, Style and State, Caption and ToolTipText, DropDownLines and DropDownWidth, etc. You also control the size (Top, Left, Height, Width) and location (Before, AfterId, and BeforeId) properties. To provide your command bar buttons with a default list of icons, drop an ImageList component to the Add-in Express Module and specify the ImageList in the Images property of the Add-in Express Module. Don't forget to set the button's Style property to either adxMsoButtonIconAndCaption or adxMsoButtonIcon. Use the DisableStandardAction property available for command bar buttons.

Use the OlExplorerItemTypes, OlInspectorItemTypes, and OlItemTypesAction properties to add context-sensitivity to controls on Outlook-specific command bars. The OlItemTypesAction property specifies an action

that Add-in Express will perform on the control when the current item's type coincides with that specified by you. Use the Click event for Button and the Change event for Edit, ComboBox and DropDownList controls.

Built-in Control Connector

Built-in controls of an Office application have predefined IDs. You find the IDs using the free Built-in Controls Scanner utility (http://www.add-in-express.com/downloads/controls-scanner.php).

The Built-in Control Connector component allows overriding the standard action for any built-in control without the necessity to add it onto any command bar.

Add a BuiltinControlConnector onto ADXAddinModule. Set its Id property to the command bar control ID from your host application. To connect the component to the command bar control, leave its CommandBar property empty. To connect the component to the control on a given toolbar, specify the toolbar in the CommandBar property. To override the default action of the control, use the Action event. The component traces the context and when the context changes, it reconnects to the currently active instance of the command bar control with the given Id taking away this task from you.

Keyboard Shortcut

Every Office application provides built-in keyboard combinations that allow shortening the access path for commands, features, and options of the application. Add-in Express allows adding custom keyboard combinations and processing both custom and built-in ones.

Add the component onto ADXAddinModule, choose the keyboard shortcut you need in the ShortcutText property, set the HandleShortCuts property of the Add-in Express Module to true and process the Action event of the KeyboardShortcut component.

Outlook Bar Shortcut Manager

Outlook provides us with the Outlook Bar (Navigation Pane in Outlook 2003). The Outlook Bar displays Shortcut groups consisting of Shortcuts that you can target to a Microsoft Outlook folder, a file-system folder, or a file-system path or URL. You use the Outlook Bar Shortcut Manager to customize the Outlook Bar with your shortcuts and groups.

This component is available for ADXAddinModule. Use the Groups collection of the component to create a new shortcut group. Use the Shortcuts collection of a short group to create a new shortcut. To connect to an existing shortcut or shortcut group, set the Caption properties of the corresponding ADXOlBarShortcut and/or ADXOlBarGroup components equal to the caption of the existing shortcut or shortcut group. Please note, there are no other ways to identify the group or shortcut.

That is why your shortcuts and shortcut groups must be named uniquely for Add-in Express to remove them (and not those with the same names) when the add-in is uninstalled. That is why you have to do this yourself.

Depending on the type of its value, the Target property of the ADXOlBarShortcut component allows you to specify different shortcut types. If the type is MAPIFolder, the shortcut represents a Microsoft Outlook folder. If the type is a String, the shortcut represents a file-system path or a URL. No events, thanks to MS.

Outlook Forms Manager

Customizing Outlook has a long-dated history. To customize Outlook forms you can use the built-in tools providing for designing and publishing the form. You can use HTML and VBScript to customize folder views, and Outlook Today is an example of this approach. Now you can embed custom .NET forms into the Outlook Explorer and Inspector windows and replace folder views with custom .NET forms.

The Outlook Forms Manager component is available for ADXAddinModule only. It is the core component of the Add-in Express 2007 Extensions for Microsoft Outlook that is a plug-in for Add-in Express. You find the detailed information on this product at http://www.add-in-express.com/outlook-extension/.

Outlook Property Page

Outlook allows extending its Options dialog with custom pages. You see this dialog when you choose Tools | Options menu. In addition, Outlook allows adding such a page to the Folder Properties dialog. You see this dialog when you choose the Properties item in the folder context menu. You create such pages using the Outlook Property Page component.

In the Add New Item Dialog, choose the Outlook Options Page item to add a class to your project. This class is a descendant of the System.Windows.Forms.UserControl class. It allows creating Outlook property pages using its visual designer. Just set up the property page properties, place your controls onto the page, and add your code. To add this page to the Outlook Options dialog, select the name of your control class in the PageType combo of ADXAddinModule and enter some characters into the PageTitle property.

To add a page to the Folder Properties dialog for a given folder(s), you use the FolderPages collection of the Add-in Express Module. Run its property editor and add an item (of the ADXOlFolderPage type). You connect the item to a given property page through the PageType property. Note, the FolderName, FolderNames, and ItemTypes properties of the ADXOlFolderPage component work in the same way as those of Outlook-specific command-bars.

Specify reactions required by your business logics in the Apply and Dirty event handlers. Use the OnStatusChange method to raise the Dirty event, the parameters of which allow marking the page as Dirty.

Add-in Express Event Classes

Outlook and Excel differ from other Office applications because they have event-raising objects not only at the topmost level of their object models. These exceptions are Worksheet in Excel, and Folders, Items and all Item sorts in Outlook. Naturally, you need to handle events from these sources. Add-in Express Event

Classes provide you with components that ease the pain of handling such events. Add-in Express Event classes are host-version independent. The Add-in Express event classes also handle releasing of COM objects required for their functioning.

At design-time, you add an Add-in Express event class to the project (see Add New Item Dialog) and use its event procedures to write the code for just one set of event handling rules for a given event source type (say, Items collection of an Outlook folder). To implement another set of event handling rules for the same event source type, you add another Add-in Express event class to your project.

At run-time, you connect an Add-in Express event class instance to an event source using its ConnectTo method. To disconnect the Add-in Express event class from the event source you use the RemoveConnection method. To apply the same business rules to another event source of the same type (say, to items of another folder), you create a new instance of the same event class.

What follow below is the source of a newly added Event Class that processes the events of the Items collection of the MAPIFolder class in Outlook.

```
Imports System

'Add-in Express for VSTO Outlook Items Events Class
Public Class OutlookItemsEventsClass1
   Inherits AddinExpress.VSTO.ADXOutlookItemsEvents

   Public Sub New(ByVal ADXModule As AddinExpress.VSTO.ADXOutlookModule)
      MyBase.New(ADXModule)
   End Sub

   Public Overrides Sub ProcessItemAdd(ByVal Item As Object)
      'TODO: Add some code
   End Sub

   Public Overrides Sub ProcessItemChange(ByVal Item As Object)
      'TODO: Add some code
   End Sub

   Public Overrides Sub ProcessItemRemove()
      'TODO: Add some code
   End Sub

End Class
```

Your First Microsoft Office Add-in

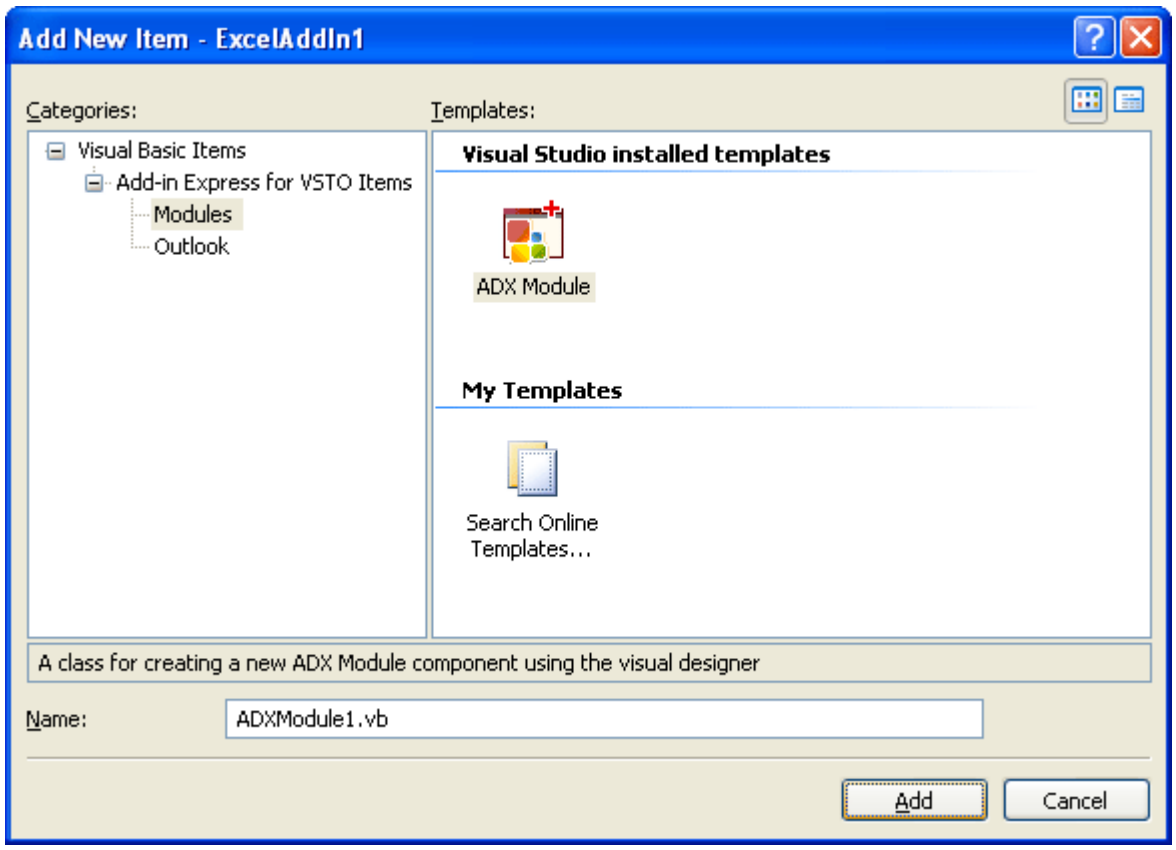
This chapter highlights almost every aspect of creating Add-ins for Microsoft Office applications in VSTO 2005 SE (Second Edition). You find the sample project developed in this chapter in the Demo Projects folder of the Add-in Express install folder.

Outlook and Add-in Express

Please note, Add-in Express provides additional components for Add-ins in Outlook. See Your First Microsoft Outlook Add-in.

Step #1 – Creating the Excel Add-in Project

To create a new Excel (Word, PowerPoint, Visio, or InfoPath) add-in project, close all opened solutions, choose “File | New | Project…”, then select the “Excel Add-in" item in either Visual Basic / Office / 2003 Add-ins (2007 Add-ins) or Visual C# / Office / 2003 Add-ins (2007 Add-ins) folder in the New Project dialog window, and click the OK button. This creates a solution with two projects: the add-in project (ExcelAddin1 in this sample) and the setup project (ExcelAddin1Setup). Now you open the Add New Item dialog for the add-in project, choose the Add-in Module item, and click OK.



This modifies the code of ThisAddin.vb (or ThisAddin.cs) as follows:

```
Public Class ThisAddIn

   Private Sub ThisAddIn_Startup(ByVal sender As Object, _
      ByVal e As System.EventArgs) Handles Me.Startup
      'Add-in Express for VSTO generated code
      ADXModule1.Initialize(Me, _
         System.Type.GetType("ExcelAddIn1.ADXModule1"))

      ' Start of VSTO generated code
      Me.Application = _
         CType(Microsoft.Office.Tools.Excel.ExcelLocale1033Proxy.Wrap _
         (GetType(Excel.Application), Me.Application), Excel.Application)
      ' End of VSTO generated code
   End Sub

   Private Sub ThisAddIn_Shutdown(ByVal sender As Object, _
      ByVal e As System.EventArgs) Handles Me.Shutdown
      'Add-in Express for VSTO generated code
      ADXModule1.Finalize(Me)
   End Sub
End Class
```

Also, this adds the ADXModule1.vb (or ADXModule1.cs) file to the add-in project. We discuss this file in the next step.

Step #2 – Add-in Express Module

The ADXModule1.vb (or ADXModule1.cs) is an Add-in Express Module that is the core part of the add-in project (see Add-in Express Module). It is the placeholder of the Add-in Express components, which allow you to concentrate on the functionality of your add-in. You specify the add-in properties in the module's properties, add the Add-in Express components to the module's designer, and write the functional code of your add-in in this module. To review its source code, in the Solution Explorer window, right-click the AddinModule1.vb (or AddinModule1.cs) file and choose the View Code popup menu item.

The code for ADXModule1.vb is as follows:

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel
Imports System.Windows.Forms
Imports Excel = Microsoft.Office.Interop.Excel
Imports Office = Microsoft.Office.Core

'Add-in Express for VSTO Module
```

```
<ComVisible(True)> _
Public Class ADXModule1
   Inherits AddinExpress.VSTO.ADXExcelAddin

#Region " Component Designer generated code. "
   'Required by designer
   Private components As System.ComponentModel.IContainer

   'Required by designer - do not modify
   'the following method
   Private Sub InitializeComponent()

   End Sub
#End Region

#Region " ADX automatic code "
   'Required by Add-in Express - do not modify
   'the methods within this region

   Public Overrides Function GetContainer() _
      As System.ComponentModel.IContainer
      If components Is Nothing Then
         components = New System.ComponentModel.Container
      End If
      GetContainer = components
   End Function
#End Region

   Public Sub New(ByVal Application As Object)
      MyBase.New(Application)
      'This call is required by the Component Designer
      InitializeComponent()
      'Add any initialization after the InitializeComponent() call
   End Sub

   Public Sub New()
      MyBase.New()

      'This call is required by the Component Designer
      InitializeComponent()

      'Add any initialization after the InitializeComponent() call
   End Sub

   Public ReadOnly Property ExcelApp() As Excel._Application
```

```
        Get
            Return HostApplication
        End Get
    End Property
End Class

Partial Public Class ThisAddIn

    Protected Overrides Function RequestService(ByVal serviceGuid As Guid) _
        As Object
        If serviceGuid = GetType(Office.IRibbonExtensibility).GUID Then
            ADXModule1.Initialize(Me, _
                System.Type.GetType("ExcelAddIn1.ADXModule1"))
            Return ADXModule1.CurrentInstance
        End If

        Return MyBase.RequestService(serviceGuid)
    End Function
End Class
```

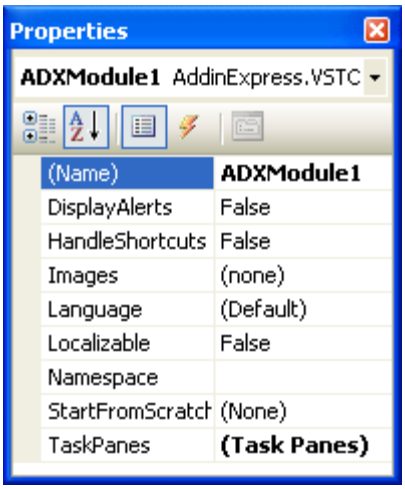
Please, pay attention to the ExcelApp property of the module. You can use it in your code to get access to Excel objects. You can use it in your code to get access to the host application object.

Step #3 - Add-in Express Designer

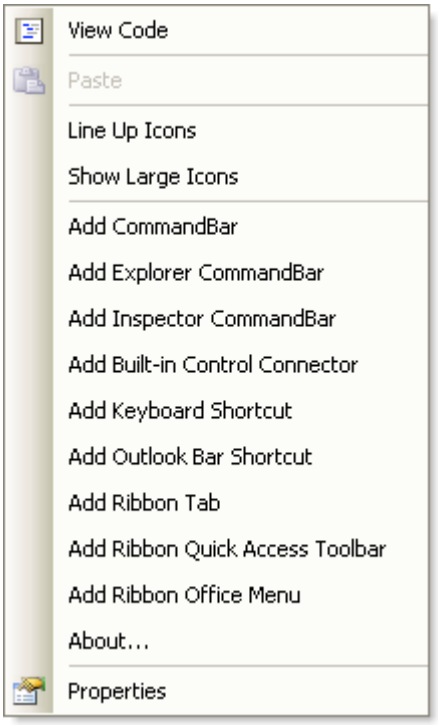
The Add-in Express Designer allows setting add-in properties and adding components to the module.

In the Solution Explorer window, right-click the AddinModule.vb (or AddinModule.cs) file and choose the View Designer popup menu item.

In the Properties window, you set the name and description of your add-in module (see Add-in Express Module).



To add an Add-in Express Component to the module, you use an appropriate command in the Properties window, or you can right-click the designer surface and choose the same command in the context menu.



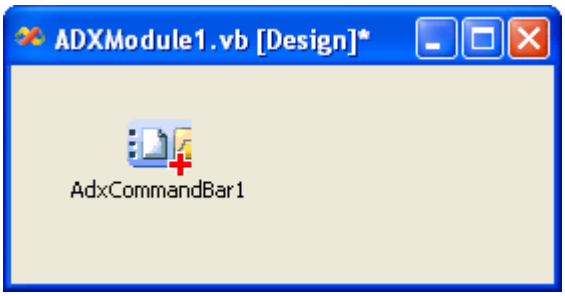
The following commands add the following components to the module:

- Add CommandBar – adds a command bar to your add-in (see Command Bars)
- Add Explorer CommandBar – adds an Outlook Explorer command bar to your add-in (see Command Bars)
- Add Inspector CommandBar – adds an Outlook Inspector command bar to your add-in (see Command Bars)

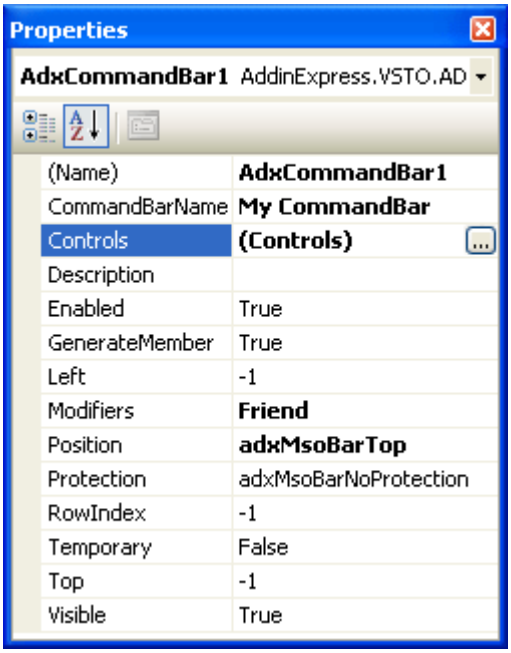
- Add Built-in Control Connector – adds a component that allows intercepting the action of a built-in control of the host application(s) (see Built-in Control Connector)
- Add Keyboard Shortcut– adds a component that allows intercepting application-level keyboard shortcuts (see Keyboard Shortcut)
- Add Outlook Bar Shortcut Manager – adds a component that allows adding Outlook Bar shortcuts and shortcut groups (see Outlook Bar Shortcut Manager)
- Add Outlook Forms Manager – adds a component that allows embedding custom .NET forms into Outlook windows (see Outlook Forms Manager)
- Add Ribbon Tab – adds a Ribbon tab to your add-in (see Office 2007 Ribbon Components)
- Add Ribbon Quick Access Toolbar – adds a component that allows customizing the Ribbon Quick Access Toolbar in your add-in (see Office 2007 Ribbon Components)
- Add Ribbon Office Menu – adds a component that allows customizing the Ribbon Office Menu in your add-in (see Office 2007 Ribbon Components)

Step #4 - Adding a New Command Bar

To add a command bar to your add-in, use the Add CommandBar command that adds an ADXCommandBar component to the Add-in Module (see Command Bars).



Select the command bar component and, in the Properties window, specify the command bar name using the CommandBarName property. In addition, you select its position in the Position property.



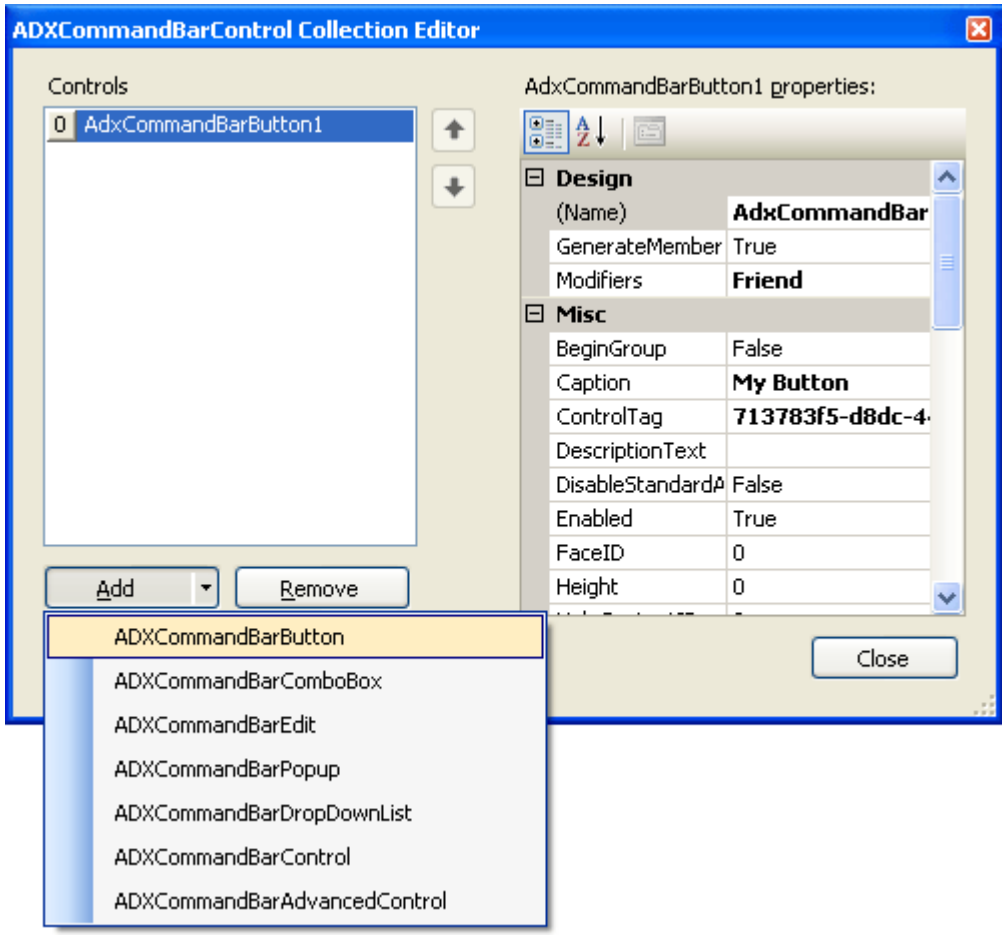
Excel 2003 and Excel 2007

In Excel 2003, this command bar will be positioned at the top of the Excel workbook window. In Excel 2007, this command bar will be shown in the Add-ins tab if the Visible property of the command bar is set to True and if the controls of the command bar are visible.The same rules applies to all Office applications.

Step #5 – Adding a New Command Bar Button

To add a new button to the command bar, in the Properties window, you select the Controls property of an appropriate command bar component and click the property editor button (the button in the property value field).

In the CommandBar Control Collection Editor, you select the command bar control type in the Add button (see also Command Bar Controls).



Specify the button's Caption property, set the Style property (default value = adxMsoButtonCaption), and close the collection editor. To handle the Click event of the button, select the added button in the topmost combo of the Properties window and add the Click event handler: The code of the event handler follows below (it's empty as you can see)

```
    Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
        Handles AdxCommandBarButton1.Click

    End Sub
```

Step #6 - Accessing Host Application Objects

The Add-in Module provides the HostApplication property that returns the Application object (of the Object type) of the host application the add-in is currently running in. For your convenience, the Add-in Express Project Wizard adds host-related properties to the Add-in module. You use these properties to access host application objects. For instance, this sample add-in includes the following properties in the Add-in Module:

```
    Public ReadOnly Property ExcelApp() As Excel._Application
        Get
            Return HostApplication
        End Get
    End Property
```

The _Application object provides the same properties and methods as the Application object but it doesn't provide events.

This allows us to write the following code to the Click event of the button just added.

```
Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
   Handles AdxCommandBarButton1.Click
   MsgBox("The current cell is " + _
      Me.ExcelApp.ActiveCell.AddressLocal(False, False)) 'relative address
End Sub
```

Step #7 - Handling Host Application Events

You might see that the Click event handler in the previous step will fire an exception when there are no workbooks open. To prevent this, you may disable the button when a window deactivates and enable it when a window activates. This will cover the situations mentioned above.

To process host application events (it's Excel in this case), you use Add-in Module events that provide all the events of the host application. The code of the module is as follows:

```
Private Sub ADXModule1_WindowActivate(ByVal sender As Object, _
   ByVal hostObj As Object, ByVal window As Object) _
   Handles Me.WindowActivate
   Me.AdxCommandBarButton1.Enabled = True
End Sub

Private Sub ADXModule1_WindowDeactivate(ByVal sender As Object, _
   ByVal hostObj As Object, ByVal window As Object) _
   Handles Me.WindowDeactivate
   Me.AdxCommandBarButton1.Enabled = False
End Sub
```

Step #8 - Handling Excel Worksheet Events

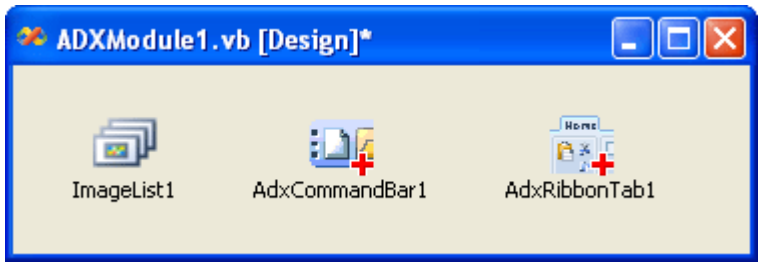
In the same way, you process worksheet-level events.

In the code of the event class, you add the following code to the procedure that handles the BeforeRightClick event of the Worksheet class

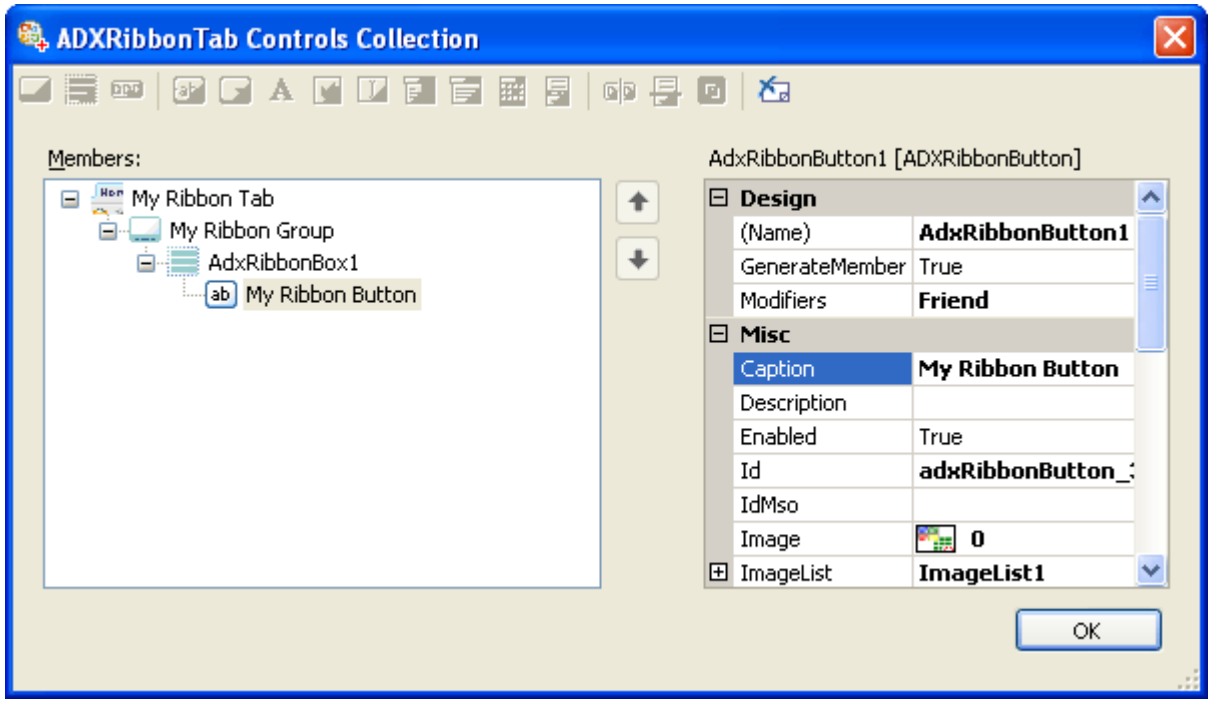
```
Private Sub ADXModule1_SheetBeforeRightClick(ByVal sender As Object, _
   ByVal e As AddinExpress.VSTO.ADXExcelSheetBeforeEventArgs) _
   Handles Me.SheetBeforeRightClick
   Dim R As Excel.Range = CType(e.Range, Excel.Range)
   'Cancel right-clicks for the first column only
   If R.Address(False, False).IndexOf("A") = 0 Then
      MsgBox("Context menu will not be shown!")
      e.Cancel = True
   Else
      e.Cancel = False
   End If
End Sub
```

Step #9 – Customizing the Office 2007 Ribbon User Interface

To add a new tab to the Ribbon, you use the Add Ribbon Tab command that adds an ADXRibbonTab component to the module.



In the Properties window, run the editor for the Controls collection of the tab. In the editor, use the toolbar buttons or context menu to add or delete Add-in Express components that form the Ribbon interface of your add-in. First, you add a Ribbon tab and change its caption to My Ribbon Tab. Then, you select the tab component, add a Ribbon group, and change its caption to My Ribbon Group. Next, you select the group, and add a button group. Finally, you select the button group and add a button. Set the button caption to My Ribbon Button. Use the ImageList and Image properties to set the icon for the button.



Click OK, and, in the Properties window, find the newly added Ribbon button. Now add the event handler to the Click event of the button. Write the following code:

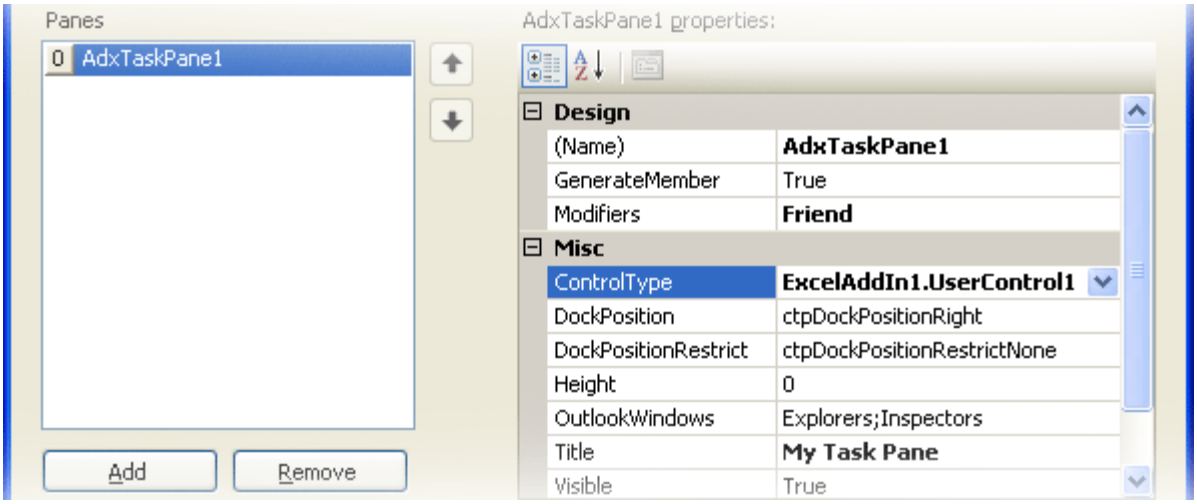
```
Private Sub AdxRibbonButton1_OnClick(ByVal sender As System.Object, _
    ByVal control As AddinExpress.VSTO.IRibbonControl, _
    ByVal pressed As System.Boolean) Handles AdxRibbonButton1.OnClick
    AdxCommandBarButton1_Click(Nothing)
End Sub
```

Remember, the ADXRibbonTab Controls editor performs the XML-schema validation automatically, so from time to time you will run into the situation when you cannot add a control to some Ribbon level. It is a restriction of the Ribbon XML-schema.

See also Office 2007 Ribbon Components.

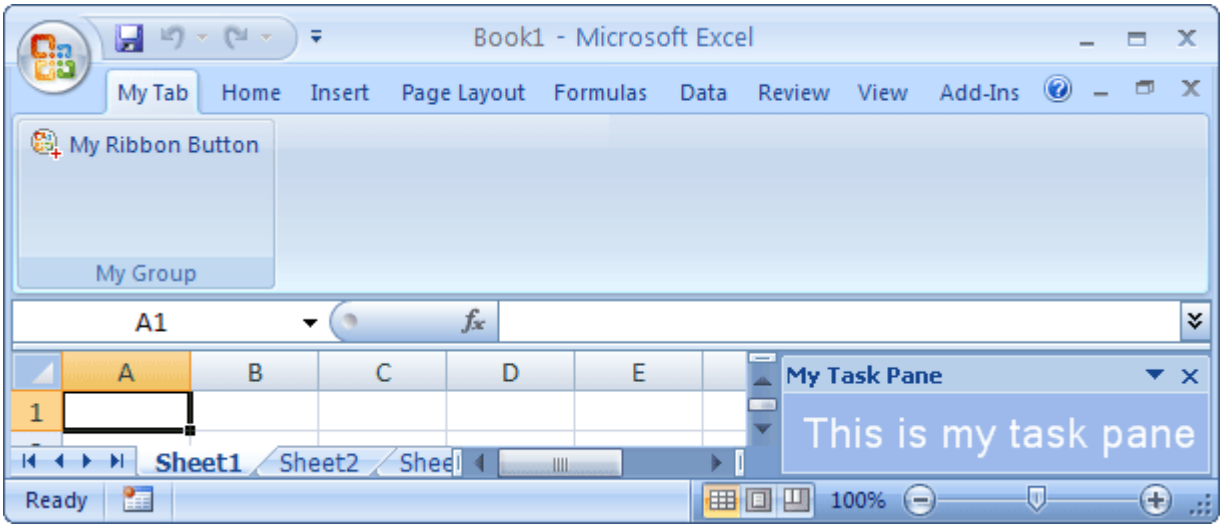
Step #10 – Adding Custom Task Panes in Office 2007

Add a UserControl to the add-in project and add some controls to the UserControl. For the UserControl to become a custom task pane, you add an item to the TaskPanes collection of the Add-in Module and set the properties of the item: Title, ControlType, and OutlookWindows (optional).

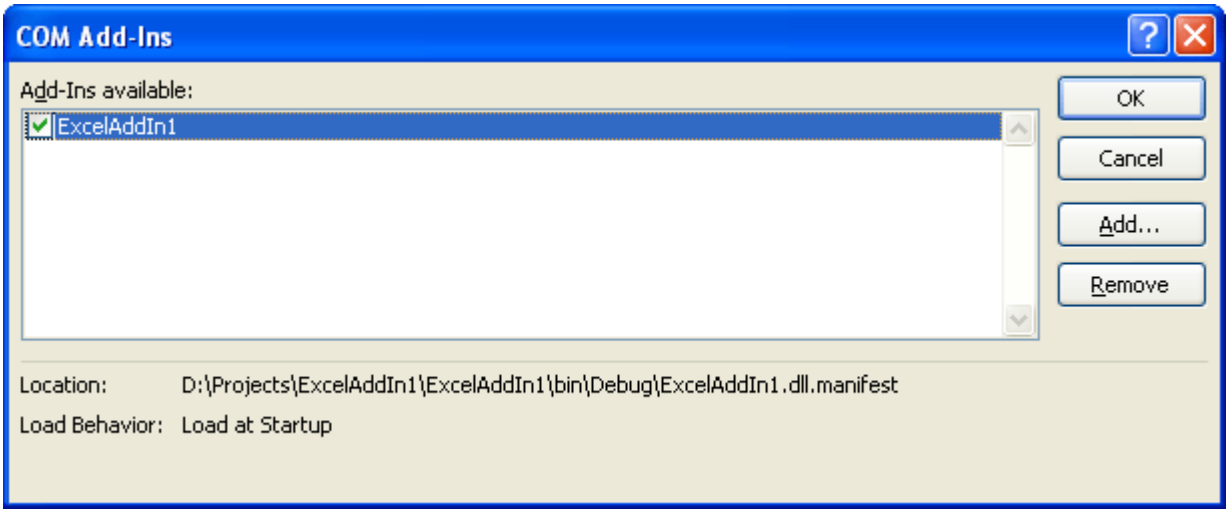


Step #11 - Running the Add-in

Choose the Build <Add-in Project Name> item in the Build menu, then restart the host application (Excel), and find your command bars, ribbon tabs, and custom task panes.



You find your add-in in the COM Add-ins dialog:



See also Add the COM Add-ins Command to a Toolbar or Menu.

Step #12 – Debugging the Add-in

To debug your add-in, just choose the Start Debugging item in the Debug menu of Visual Studio.

Step #13 – Deploying the Add-in

Please use Microsoft recommendations on deployment of VSTO solutions. You can find some useful information in VSTO solution deployment below.

Your First Microsoft Outlook Add-in

For Outlook customizatrions, Add-in Express provides two Outlook-specific command bar components: Explorer Command Bar and Inspector Command Bar. The first adds a command bar to the Outlook Explorer window and solves many problems with custom Outlook command bars. The latter adds a command bar to the Outlook Inspector window. Both components have the FolderName(s) and ItemTypes properties that add context-sensitivity to Outlook command bars. The OlExplorerItemTypes, OlInspectorItemTypes, and OlItemTypeAction properties add context-sensitivity to Outlook command bar controls.

Additionally, Add-in Express supplies the Outlook Property Page component that helps you to add your pages to the Options (Tools | Options menu) and folder Properties dialogs.

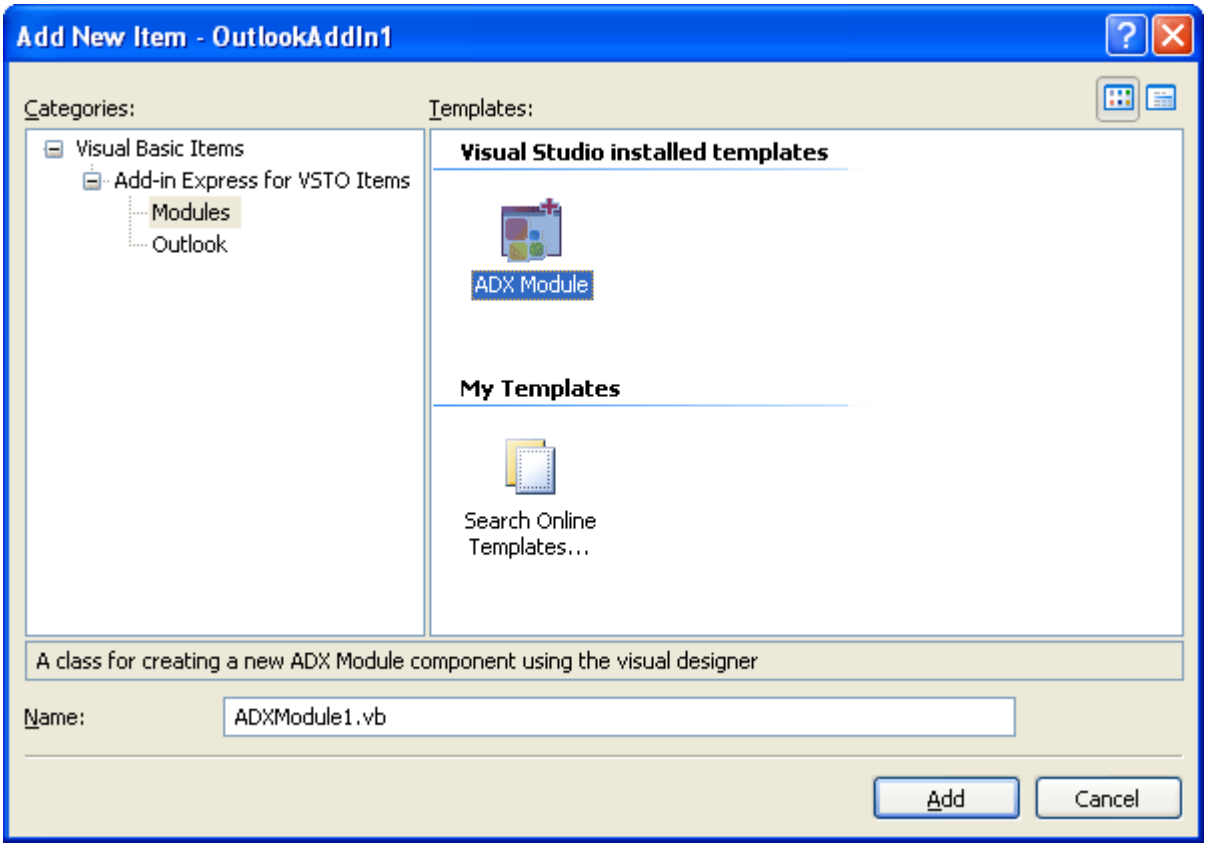
Also, Add-in Express includes Outlook Event classes that can simplify event handling for Outlook objects: Item Event Class, Items Event Class, and Folder Event class.

As to application-level Outlook events, they are provided by the Add-in Module, the core component of Add-in Express.

Step #1 – Creating an Outlook Add-in Project

To create a new Outlook add-in project, close all opened solutions, choose “File | New | Project…”, select the “Outlook Add-in" item in either Visual Basic / Office / 2003 Add-ins (2007 Add-ins) or Visual C# / Office / 2003 Add-ins (2007 Add-ins) folder in the New Project dialog window, and click the OK button. This creates a solution with two projects: the add-in project (OutlookAddin1 in this sample) and the setup project (OutlookAddin1Setup).

Open the Add New Item dialog for the add-in project, choose the Add-in Module item, and click OK.



This modifies the code of ThisAddin.vb (or ThisAddin.cs) as follows:

```
Public Class ThisAddIn
   Private Sub ThisAddIn_Startup(ByVal sender As Object, _
      ByVal e As System.EventArgs) Handles Me.Startup
      'Add-in Express for VSTO generated code
      ADXModule1.Initialize(Me, _
         System.Type.GetType("OutlookAddIn1.ADXModule1"))
   End Sub

   Private Sub ThisAddIn_Shutdown(ByVal sender As Object, _
      ByVal e As System.EventArgs) Handles Me.Shutdown
```

```
        'Add-in Express for VSTO generated code
        ADXModule1.Finalize(Me)
    End Sub
End Class
```

Also, this adds the ADXModule1.vb (or ADXModule1.cs) file to the add-in project. We discuss this file in the next step.

Step #2 - Add-in Express Module

The ADXModule1.vb (or ADXModule1.cs) is an Add-in Module that is the core part of Add-in Express based add-in projects. It is the placeholder of the Add-in Express components, which allow you to concentrate on the functionality of your add-in. You specify add-in properties in the module's properties, add Add-in Express components to the module's designer, and write the functional code of your add-in in this module. To review its source code, in the Solution Explorer window, right-click the AddinModule1.vb (or AddinModule1.cs) file and choose the View Code popup menu item.

The code for ADXModule1.vb is as follows:

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel
Imports System.Windows.Forms
Imports Outlook = Microsoft.Office.Interop.Outlook
Imports Office = Microsoft.Office.Core

'Add-in Express for VSTO Module
<ComVisible(True)> _
Public Class ADXModule1
    Inherits AddinExpress.VSTO.ADXOutlookAddin

#Region " Component Designer generated code. "
    'Required by designer
    Private components As System.ComponentModel.IContainer

    'Required by designer - do not modify
    'the following method
    Private Sub InitializeComponent()

    End Sub

#End Region

#Region " ADX automatic code "

    'Required by Add-in Express - do not modify
```

```
    'the methods within this region

    Public Overrides Function GetContainer() As _
       System.ComponentModel.IContainer
       If components Is Nothing Then
          components = New System.ComponentModel.Container
       End If
       GetContainer = components
    End Function

#End Region

    Public Sub New(ByVal Application As Object)
       MyBase.New(Application)

       'This call is required by the Component Designer
       InitializeComponent()

       'Add any initialization after the InitializeComponent() call

    End Sub

    Public Sub New()
       MyBase.New()

       'This call is required by the Component Designer
       InitializeComponent()

       'Add any initialization after the InitializeComponent() call

    End Sub

    Public ReadOnly Property OutlookApp() As Outlook._Application
       Get
          Return HostApplication
       End Get
    End Property

End Class

Partial Public Class ThisAddIn

    Protected Overrides Function RequestService(ByVal serviceGuid As Guid) _
       As Object
       If serviceGuid = GetType(Office.IRibbonExtensibility).GUID Then
```

```
            ADXModule1.Initialize(Me, _
               System.Type.GetType("OutlookAddIn1.ADXModule1"))
            Return ADXModule1.CurrentInstance
         End If

         Return MyBase.RequestService(serviceGuid)
      End Function

   End Class
```

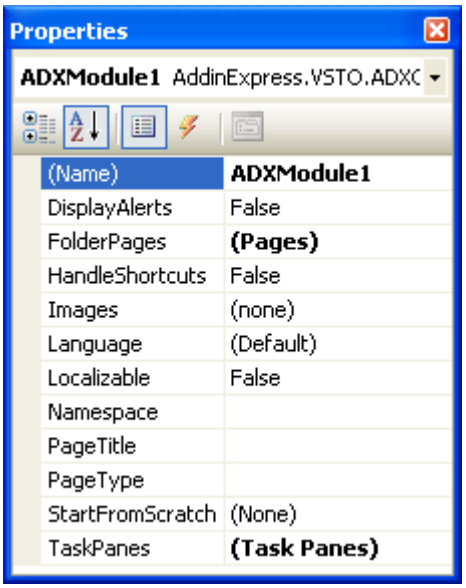
Please, pay attention to the OutlookApp property of the module. You can use it in your code to get access to Outlook objects.

Step #3 – Add-in Express Designer

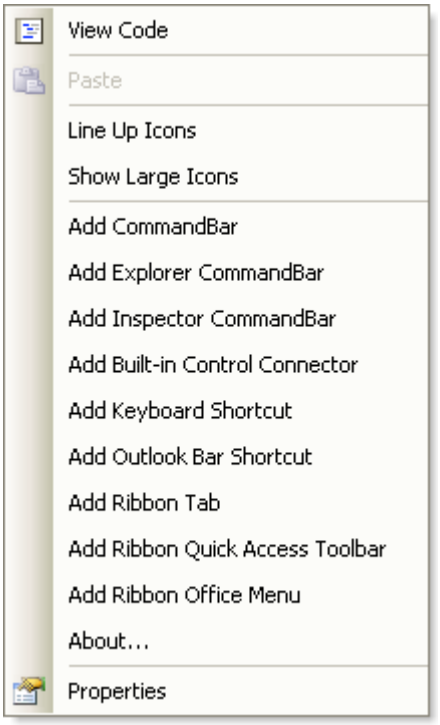
The Add-in Express Designer allows setting add-in properties and adding components to the module.

In the Solution Explorer window, right-click the AddinModule.vb (or AddinModule.cs) file and choose the View Designer popup menu item.

In the Properties window for the Add-in Module designer, you set the name and description of your add-in module (see Add-in Express Module).



To add an Add-in Express Component to the module, you use an appropriate command in the Properties window, or you can right-click the designer surface and choose the same command in the context menu.

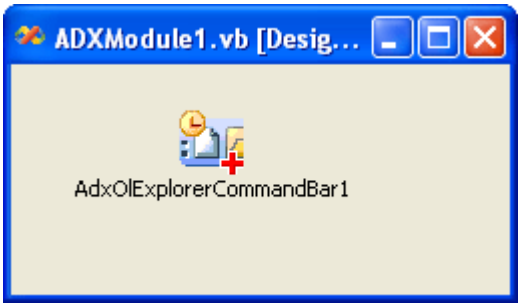


The following commands add the following components to the module:

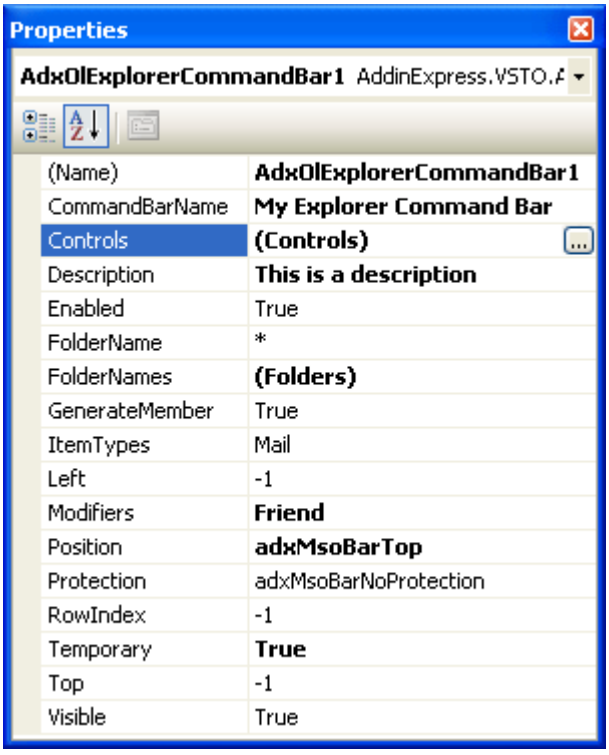
- Add CommandBar – adds an Office command bar to your add-in (see Command Bars)
- Add Explorer CommandBar – adds an Outlook Explorer command bar to your add-in (see Command Bars)
- Add Inspector CommandBar – adds an Outlook Inspector command bar to your add-in (see Command Bars)
- Add Built-in Control Connector – adds a component that allows intercepting the action of a built-in control of the host application(s) (see Built-in Control Connector)
- Add Keyboard Shortcut– adds a component that allows intercepting application-level keyboard shortcuts (see Keyboard Shortcut)
- Add Outlook Bar Shortcut Manager – adds a component that allows adding Outlook Bar shortcuts and shortcut groups (see Outlook Bar Shortcut Manager)
- Add Outlook Forms Manager – adds a component that allows embedding custom .NET forms into Outlook windows (see Outlook Forms Manager)
- Add Ribbon Tab – adds a Ribbon tab to your add-in (see Office 2007 Ribbon Components)
- Add Ribbon Quick Access Toolbar – adds a component that allows customizing the Ribbon Quick Access Toolbar in your add-in (see Office 2007 Ribbon Components)
- Add Ribbon Office Menu – adds a component that allows customizing the Ribbon Office Menu in your add-in (see Office 2007 Ribbon Components)

Step #4 - Adding a New Explorer Command Bar

To add a command bar to the Outlook Explorer window, use the Add Explorer CommandBar command that adds an ADXOlExplorerCommandBar component to the Add-in Module.



Select the Add-in Express Explorer Command Bar component, and, in the Properties window, specify the command bar name using the CommandBarName property, and choose its position (see the Position property). Outlook-specific versions of Add-in Express Command Bar component provide context-sensitive properties. They are FolderName, FolderNames, and ItemTypes (see Outlook CommandBar Visibility Rules and Add-ins for Outlook – Template Characters in FolderName).



In the screenshot, you see the Outlook Explorer command bar that will be shown for every Outlook folder (FolderName = "*") the default item type of which is Mail.

Outlook 2003 and Outlook 2007

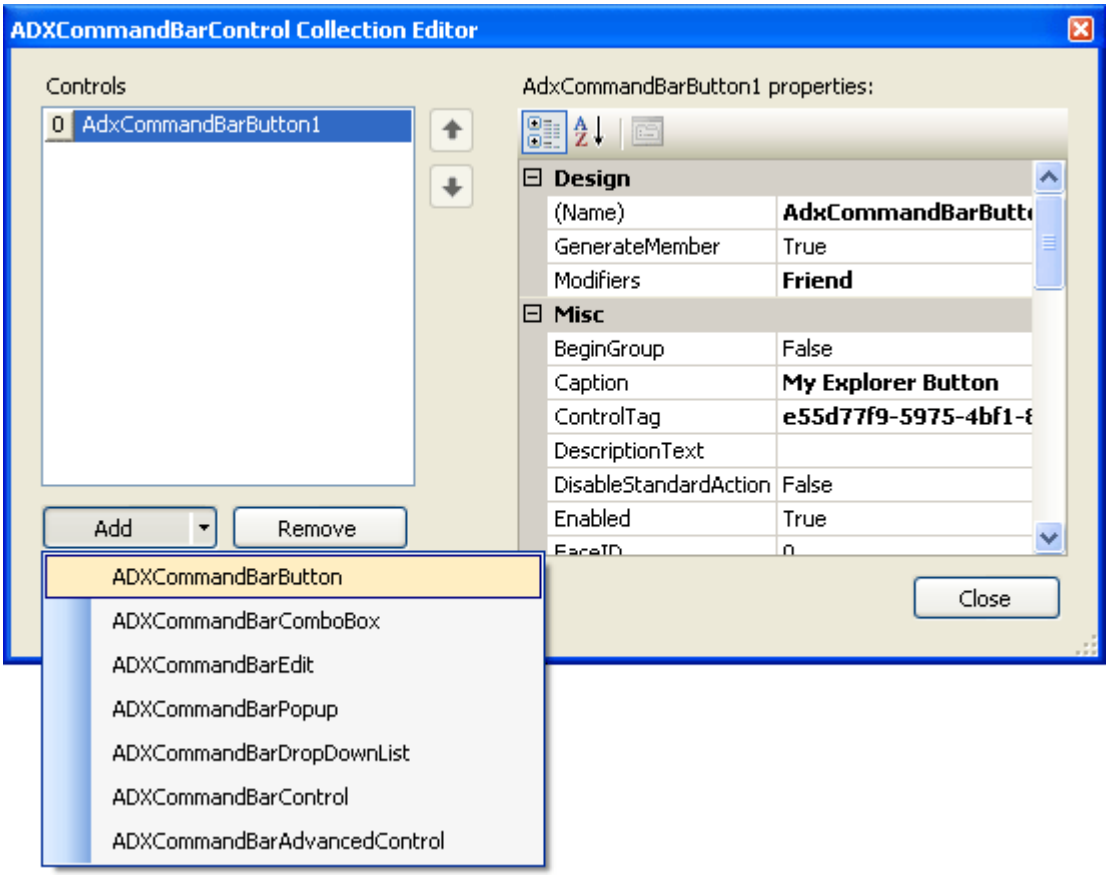
In Outlook 2003, this command bar will be positioned at the top of the Outlook Explorer window. In Outlook 2007, this command bar will be shown in the Add-ins tab if the Visible property of the command bar is set to True and if the controls of the command bar are visible.

See also Command Bars.

Step #5 – Adding a New Command Bar Button

To add a new button to the Explorer command bar, in the Properties window, you select the Controls property and click the property editor button (the button in the property value field).

In the editor window, you select the command bar control type in the Add button (see also Command Bar Controls).



Specify the button's Caption property, set the Style property (default value = adxMsoButtonCaption), and close the collection editor. To handle the Click event of the button, select the added button in the topmost

combo of the Properties window and add the Click event handler. The code of the event handler follows below (it is empty as you can see)

```
Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
    Handles AdxCommandBarButton1.Click

End Sub
```

Step #6 - Accessing Outlook Objects

The Add-in Module provides the HostApplication property that returns the Application object (of the Object type) of the host application the add-in is currently running in. For your convenience, Add-in Express also adds the OutlookApp property in the Add-in Module:

```
Public ReadOnly Property OutlookApp() As Outlook._Application
   Get
      Return HostApplication
   End Get
End Property
```

The _Application object provides the same properties and methods as the Application object but it doesn't provide events. This allows you to write the following code to the Click event of the button just added:

```
Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
   Handles AdxCommandBarButton1.Click
   MsgBox("The subject is:" _
      + vbCrLf _
      + Me.OutlookApp.ActiveExplorer.Selection.Item(1).Subject)
End Sub
```

ThisApplication property

This property of the Add-in Module returns the value of the ThisApplication VSTO property.

Step #7 - Handling Outlook Events

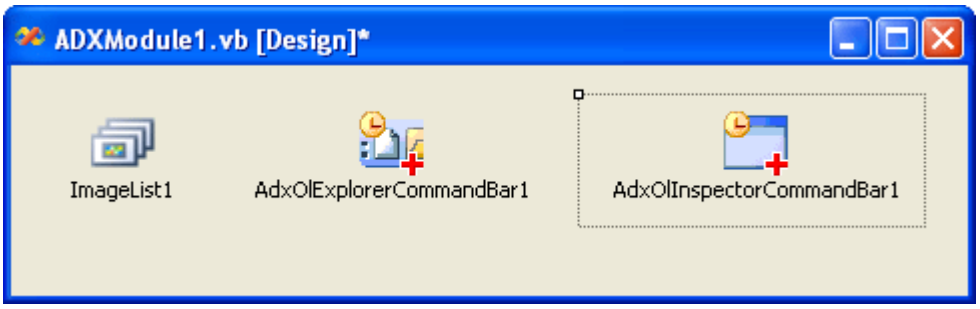
The Add-in Module provides all application-level Outlook events. For instance, the following code handles the BeforeFolderSwitch event of the Outlook Explorer class:

```
Private Sub ADXModule1_ExplorerBeforeFolderSwitch _
   (ByVal sender As Object, _
```

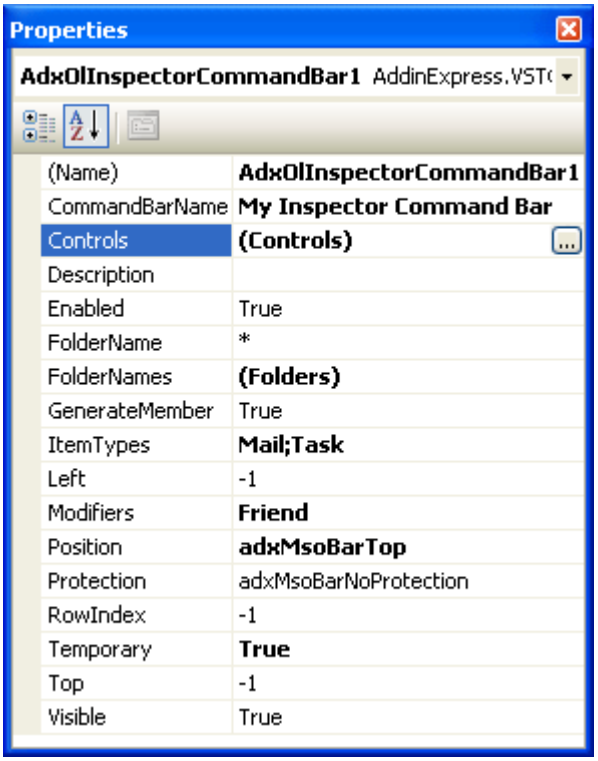
```
        ByVal e As _
            AddinExpress.VSTO.ADXOlExplorerBeforeFolderSwitchEventArgs) _
        Handles Me.ExplorerBeforeFolderSwitch
        MsgBox("You are switching to the " + e.NewFolder.Name + " folder")
    End Sub
```

Step #8 – Adding a New Inspector Command Bar

To add a command bar to the Outlook Inspector window, use the Add Inspector CommandBar command that adds an ADXOlInspectorCommandBar component to the Add-in Module.



The Inspector command bar component provides the same properties as the Explorer command bar component. In the screenshot below you see the properties of the Inspector command bar that will be shown for Mail and Task Items.



Outlook add-ins, 2003 and 2007

> *In Outlook 2003, this command bar will be positioned at the top of the Outlook Inspector window. In Outlook 2007, this command bar will be shown in the Add-ins tab if the Visible property of the command bar is set to True and if the controls of the command bar are visible.*

Add a new command bar button to the command bar (see Step #5 – Adding a New Command Bar Button for details).

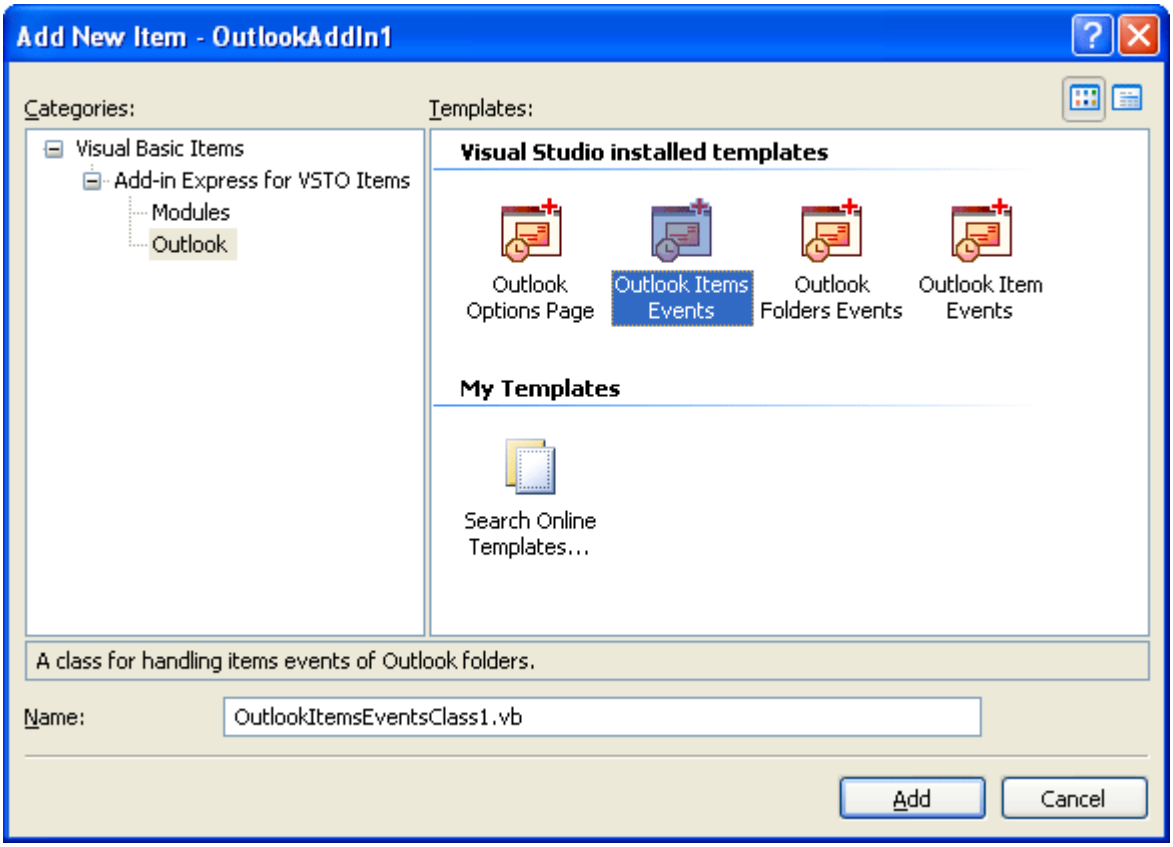
Now you can show the subject of the currently open mail or task item using the following code that handles the Click event of the button:

```
Private Sub AdxCommandBarButton2_Click(ByVal sender As System.Object) _
        Handles AdxCommandBarButton2.Click
   MsgBox("The subject is:" _
      + vbCrLf _
      + Me.OutlookApp.ActiveInspector.CurrentItem.Subject)
End Sub
```

See also Command Bars, Outlook CommandBar Visibility Rules and Add-ins for Outlook – Template Characters in FolderName.

Step #9 – Handling Events of Outlook Items Object

The Outlook MAPIFolder class provides the Items collection. This collection provides the following events: ItemAdd, ItemChange, and ItemRemove. To process these events, you use the Outlook Items Events Class located in the Add-in Express Items folder in the Add New Item dialog:



This will add the OutlookItemsEventsClass1.vb class to the add-in project. You handle the ItemAdd event by entering some code into the ProcessItemAdd procedure of the class:

```
Imports System

'Add-in Express for VSTO Outlook Items Events Class
Public Class OutlookItemsEventsClass1
   Inherits AddinExpress.VSTO.ADXOutlookItemsEvents

   Public Sub New(ByVal ADXModule As AddinExpress.VSTO.ADXAddinModule)
      MyBase.New(ADXModule)
   End Sub

   Public Overrides Sub ProcessItemAdd(ByVal Item As Object)
      MsgBox("The item with subject '" + Item.Subject + _
         "' has been added to the Inbox folder")
   End Sub

   Public Overrides Sub ProcessItemChange(ByVal Item As Object)
      'TODO: Add some code
   End Sub
```

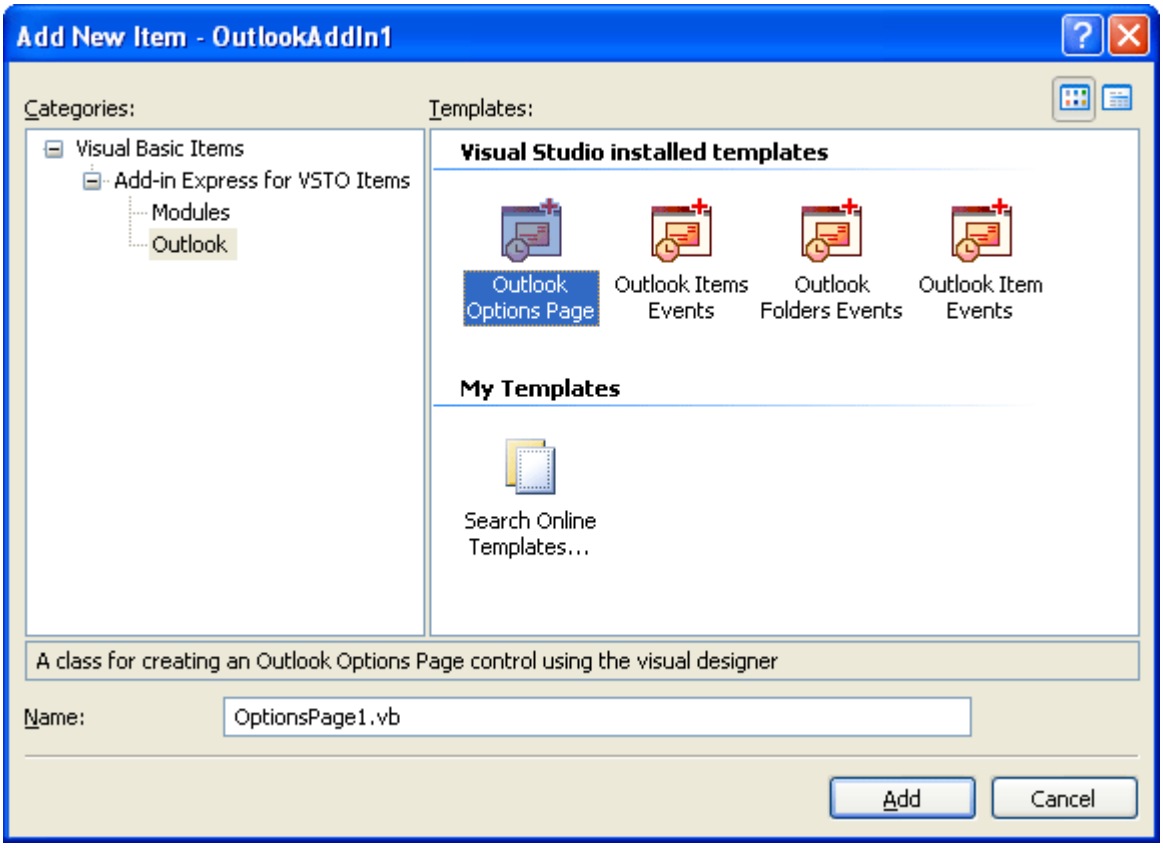
```
    Public Overrides Sub ProcessItemRemove()
        'TODO: Add some code
    End Sub
End Class
```

This requires you to add the following declarations and code to the Add-in Module:

```
    Dim ItemsEvents As OutlookItemsEventsClass1 = _
        New OutlookItemsEventsClass1(Me)
...
    Private Sub ADXModule1_OnBeginShutdown(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.OnBeginShutdown
        If ItemsEvents IsNot Nothing Then
            ItemsEvents.RemoveConnection()
            ItemsEvents = Nothing
        End If
    End Sub

    Private Sub ADXModule1_OnStartupComplete(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.OnStartupComplete
        ItemsEvents.ConnectTo( _
            AddinExpress.VSTO.ADXOlDefaultFolders.olFolderInbox, True)
    End Sub
```

Step #10 – Adding Folder Property Pages

Unlike othe Office applications, Outlook allows you to add custom option pages to the Options dialog box (the Tools | Options menu) and / or to the Properties dialog box of any folder. To automate this task, Add-in Express provides the Outlook Property Page component. You find it in the Add New Item dialog box.



Click the Add button to add a new property page instance, a descendant of the ADXOlPropertyPage class that implements the IPropertyPage interface:

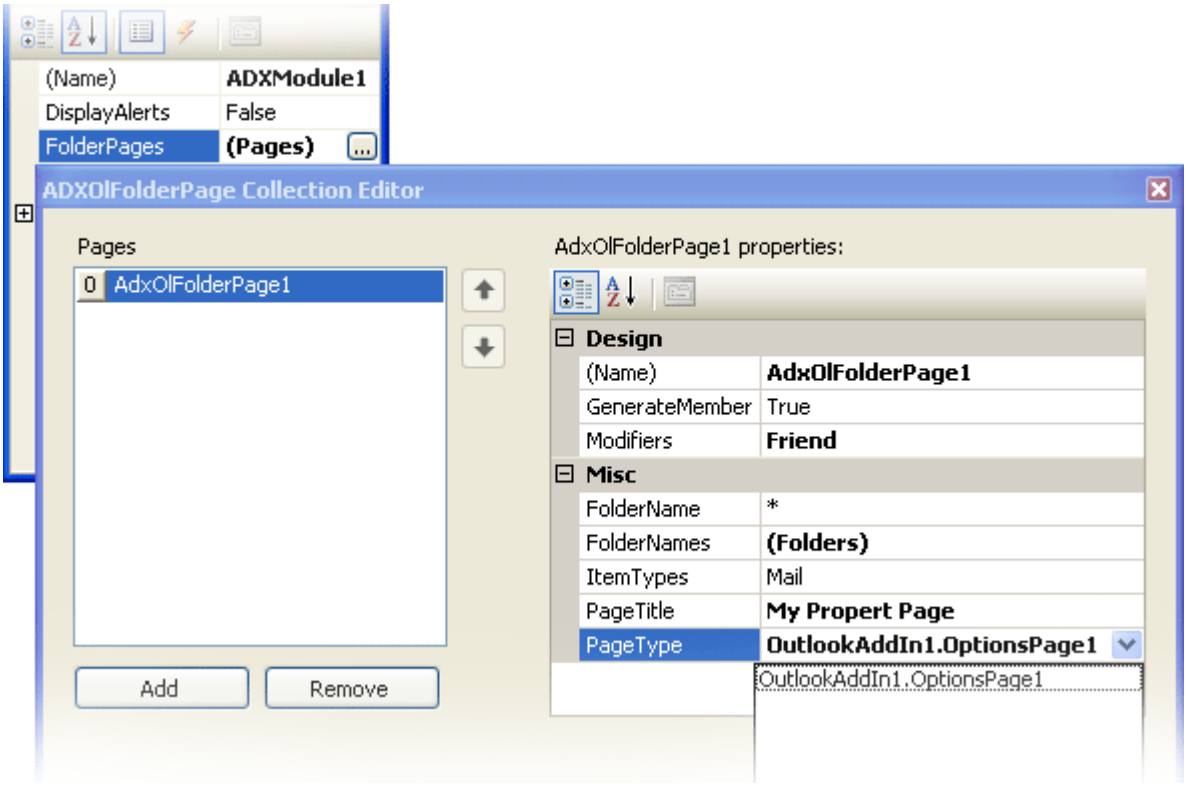
```
Imports System.Runtime.InteropServices

'Add-in Express for VSTO Outlook Options Page
Public Class OptionsPage1
   Inherits AddinExpress.VSTO.ADXOlPropertyPage

#Region " Component Designer generated code "

   Public Sub New()
      MyBase.New()
```

```
        'This call is required by the Component Designer
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Clean up any resources being used
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by designer
    Private components As System.ComponentModel.IContainer

    'Required by designer - do not modify
    'the following method
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        components = New System.ComponentModel.Container()
        '
        'OptionsPage1
        '
        Me.Name = "OptionsPage1"
        Me.Size = New System.Drawing.Size(413, 358)
    End Sub

#End Region
End Class
```

You can customize the page as an ordinary form: add the controls and handle their events.

To add a property page to the <FolderName> Properties dialog box of an Outlook folder(s), you do the following:

- In the AddinModule properties, run the editor of the FolderPages property,
- Click the Add button,
- Specify the folder you need in the FolderName property,
- Set the PageType property to the PropertyPage component you've added
- Specify the Title property and close the dialog box.



The screenshot above shows the settings you need to display your page in the Folder Properties dialog for all Mail folders (FolderName = '*' and ItemTypes = Mail).

In order to limit the property page to the Inbox folder only, change the code of the OnStartupComplete event in the Add-in Module as follows:

```
Private Sub ADXModule1_OnStartupComplete(ByVal sender As Object, _
   ByVal e As System.EventArgs) Handles Me.OnStartupComplete
   ItemsEvents.ConnectTo( _
      AddinExpress.VSTO.ADXOlDefaultFolders.olFolderInbox, True)
   Dim FullPath As String = _
      Me.OutlookApp.GetNamespace("Mapi"). _
         GetDefaultFolder(Outlook.OlDefaultFolders.olFolderInbox).FolderPath
      ' remove leading backslashes
      Me.FolderPages.Item(0).FolderName = _
         FullPath.Substring(2, FullPath.Length - 2)
    End Sub
```

In order to control the events for the folder, add a checkbox to the page and handle its CheckedChanged event as well as the Dirty, Apply, and Load events of the page as follows:

```
    ...
    Friend WithEvents CheckBox1 As System.Windows.Forms.CheckBox
    Private TrackStatusChanges As Boolean
```

```
    ...

Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged
   If Not TrackStatusChanges Then Me.OnStatusChange()
End Sub

Private Sub OptionsPage1_Apply(ByVal sender As Object, _
   ByVal e As System.EventArgs) Handles Me.Apply
   CType(AddinExpress.VSTO.ADXOutlookModule.Instance, _
      OutlookAddIn1.ADXModule1).IsFolderTracked = _
         Me.CheckBox1.Checked
End Sub

Private Sub OptionsPage1_Dirty(ByVal sender As Object, _
   ByVal e As AddinExpress.VSTO.ADXDirtyEventArgs) Handles Me.Dirty
   e.Dirty = True
End Sub

Private Sub OptionsPage1_Load(ByVal sender As Object, _
   ByVal e As System.EventArgs) Handles Me.Load
   TrackStatusChanges = True
   Me.CheckBox1.Checked = _
         CType(AddinExpress.VSTO.ADXOutlookModule.Instance, _
      OutlookAddIn1.ADXModule1).IsFolderTracked
   TrackStatusChanges = False
End Sub
```

Finally, you add the following property to the AddinModule code:

```
    ...
    Friend Property IsFolderTracked() As Boolean
       Get
          Return ItemsEvents.IsConnected
       End Get
       Set(ByVal value As Boolean)
          If value Then
             ItemsEvents.ConnectTo _
                (AddinExpress.VSTO.ADXOlDefaultFolders.olFolderInbox, True)
          Else
             ItemsEvents.RemoveConnection()
          End If
       End Set
    End Property
```

Don't forget

You must check the Make Assembly COM-Visible checkbox to make your property pages work. Find it in Project Properties / Application / Assenbly Information.

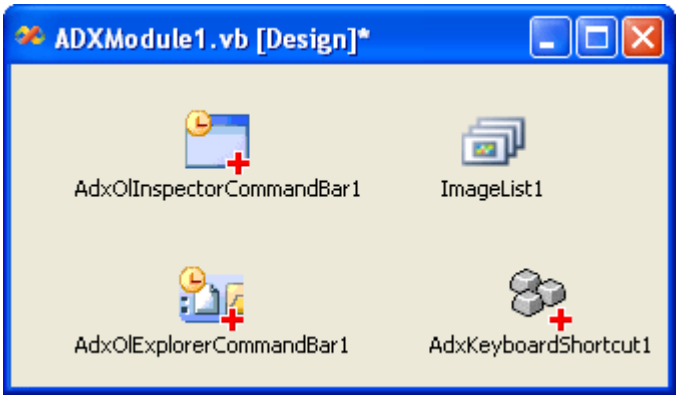
Adding a page to the Outlook Options dialog

To add this or other property page to the main Options dialog box, you use the PageType and PageTitle properties of the add-in module.

See also Outlook Property Page.

Step #11 - Intercepting Keyboard Shortcut

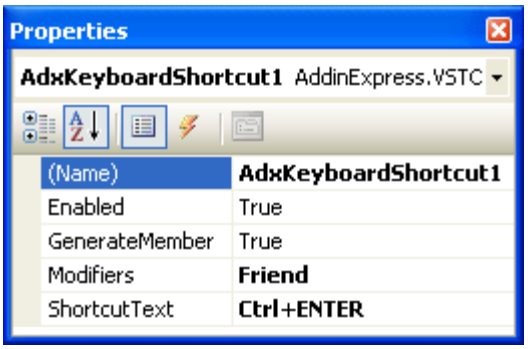
To intercept a keyboard shortcut, you add an ADXKeyboardShortcut component to the Add-in Express Module using the Add Keyboard Shortcut command of the module.



In the Properties window you select (or enter) the desired shortcut in the ShortcutText property. We chose the shortcut for the Send button in the mail Inspector's Standard command bar. It is Ctrl+Enter.

HandleShortcuts property

To use keyboard shortcuts, you need to set the HandleShortcuts property of AddinModule to true.

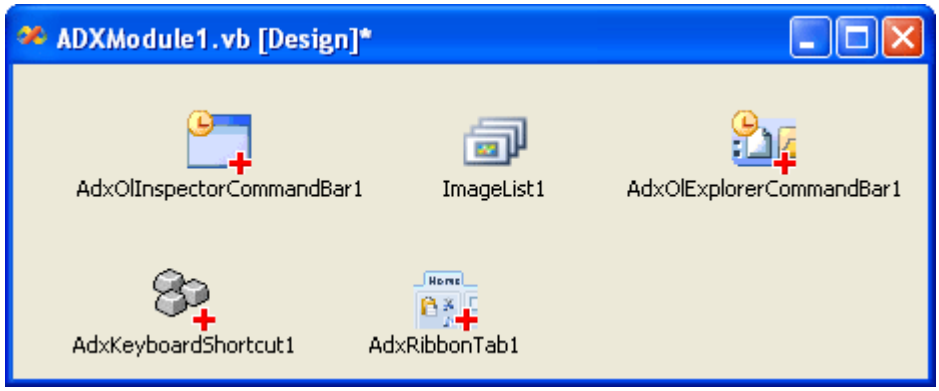


Now you handle the Action event of the component:

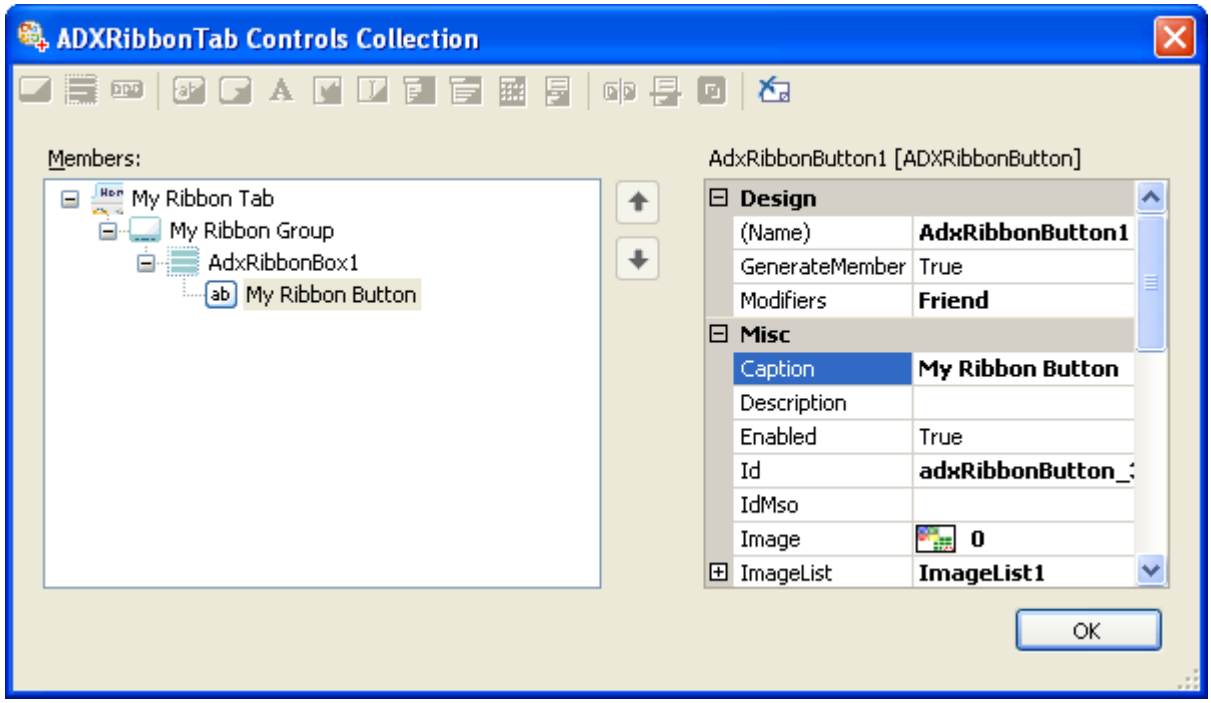
```
Private Sub AdxKeyboardShortcut1_Action(ByVal sender As System.Object) _
   Handles AdxKeyboardShortcut1.Action
   MsgBox("You've pressed " + _
      CType(sender, AddinExpress.VSTO.ADXKeyboardShortcut).ShortcutText)
End Sub
```

Step #12 – Customizing the Outlook 2007 Ribbon User Interface

To add a new tab to the Ribbon, you use the Add Ribbon Tab command that adds an ADXRibbonTab component to the module.



In the Properties window, run the editor for the Controls collection of the tab. In the editor, use the toolbar buttons or context menu to add or delete Add-in Express components that form the Ribbon interface of your add-in. First, you add a Ribbon tab and change its caption to My Ribbon Tab. Then, you select the tab component, add a Ribbon group, and change its caption to My Ribbon Group. Next, you select the group, and add a button group. Finally, you select the button group and add a button. Set the button caption to My Ribbon Button. Use the ImageList and Image properties to set the icon for the button.



Click OK, and, in the Properties window, find the newly added Ribbon button. Now add the event handler to the Click event of the button. Write the following code:

```
Private Sub AdxRibbonButton1_OnClick(ByVal sender As System.Object, _
    ByVal control As AddinExpress.VSTO.IRibbonControl, _
    ByVal pressed As System.Boolean) Handles AdxRibbonButton1.OnClick
    AdxCommandBarButton2_Click(Nothing)
End Sub
```

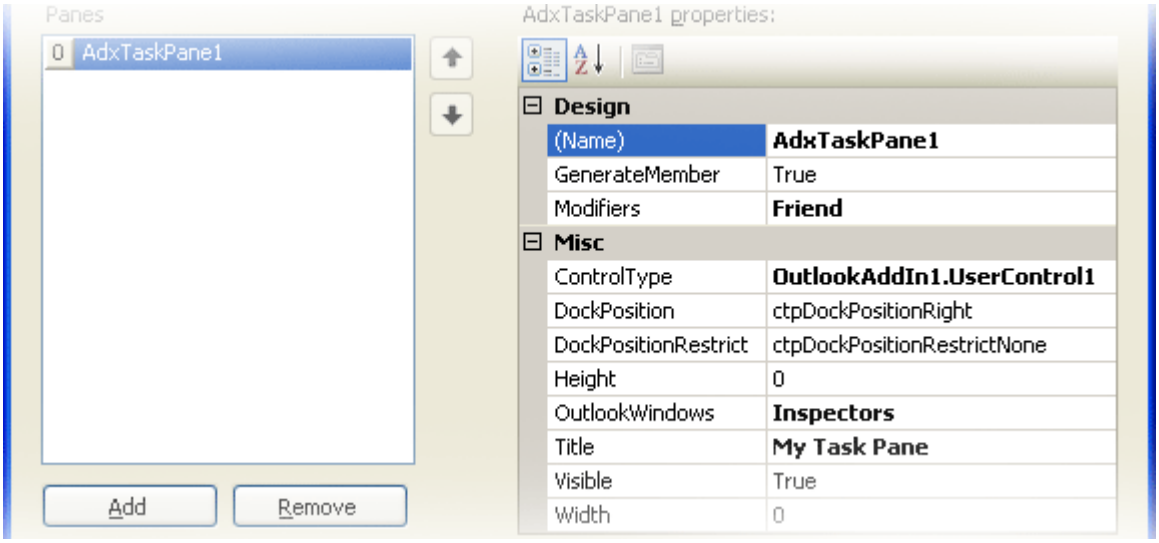
Remember, the ADXRibbonTab Controls editor performs the XML-schema validation automatically, so from time to time you will run into the situation when you cannot add a control to some Ribbon level. It is a restriction of the Ribbon XML-schema.

Note, unlike other Ribbon-based applications Outlook, has numerous ribbons. Please use the Ribbons property of your ADXRibbonTab components to specify the ribbons you customize with your tabs.

See also Office 2007 Ribbon Components.

Step #13 – Adding Custom Task Panes in Outlook 2007

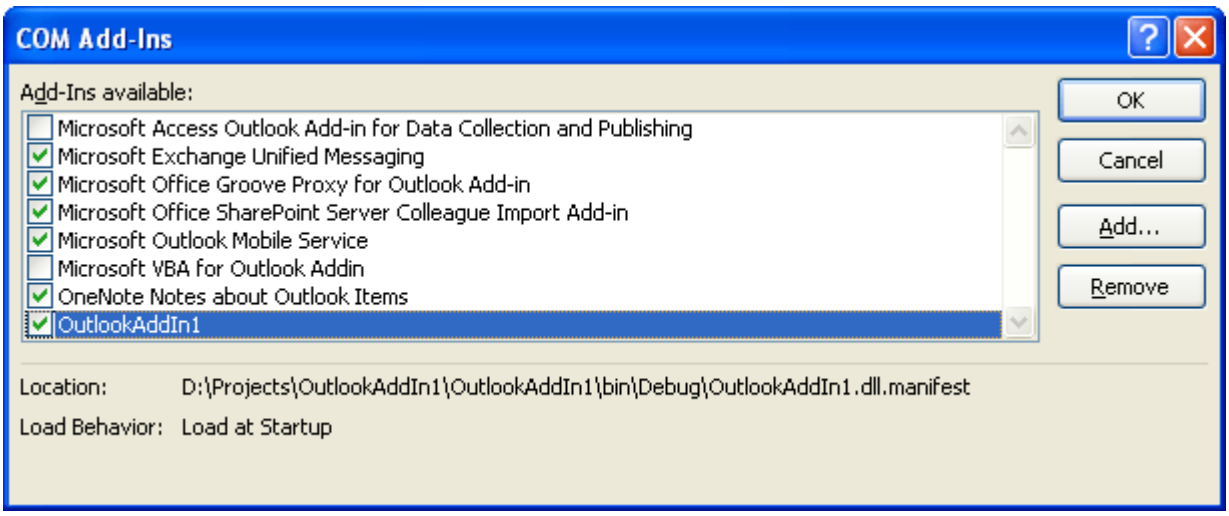
Add a UserControl to the add-in project and add some controls to the UserControl. For the UserControl to become a custom task pane, you add an item to the TaskPanes collection of the Add-in Module and set the properties of the item: Title, ControlType, and OutlookWindows (optional).



Step #14 - Running the Outlook Add-in

Choose the Build <Add-in Project Name> item in the Build menu, then restart Outlook, and find your option page(s), command bars, ribbon tabs, and custom task panes.

You find your add-in in the COM Add-ins dialog:



Step #15 – Debugging the Outlook Add-in

To debug your add-in, just choose the Start Debugging item in the Debug menu of Visual Studio.

Step #16 – Deploying the Outlook Add-in

Please use Microsoft recommendations on deployment of VSTO solutions. You can find some useful information in VSTO solution deployment below.

Several notes

Don't you have an impression that creating add-ins is a very simple task? Sure, Add-in Express makes embedding your code into Office applications very simple, but you should write the applied code yourself, and we guess it would be something more complex than a single call of MessageBox.

Here we describe some tips and important issues you will need when developing your add-ins.

Terminology

In this document, on our site, and in all our texts we use the terminology suggested by Microsoft for all toolbars, their controls, and for all interfaces of the Office Type Library. For example:

- Command bar is a toolbar, a menu bar, or a context menu.
- Command bar control is one of the following: a button, an edit box, a combo box, or a pop-up.
- Pop-up can stand for a pop-up menu, a pop-up button on a command bar or a submenu on a menu bar.

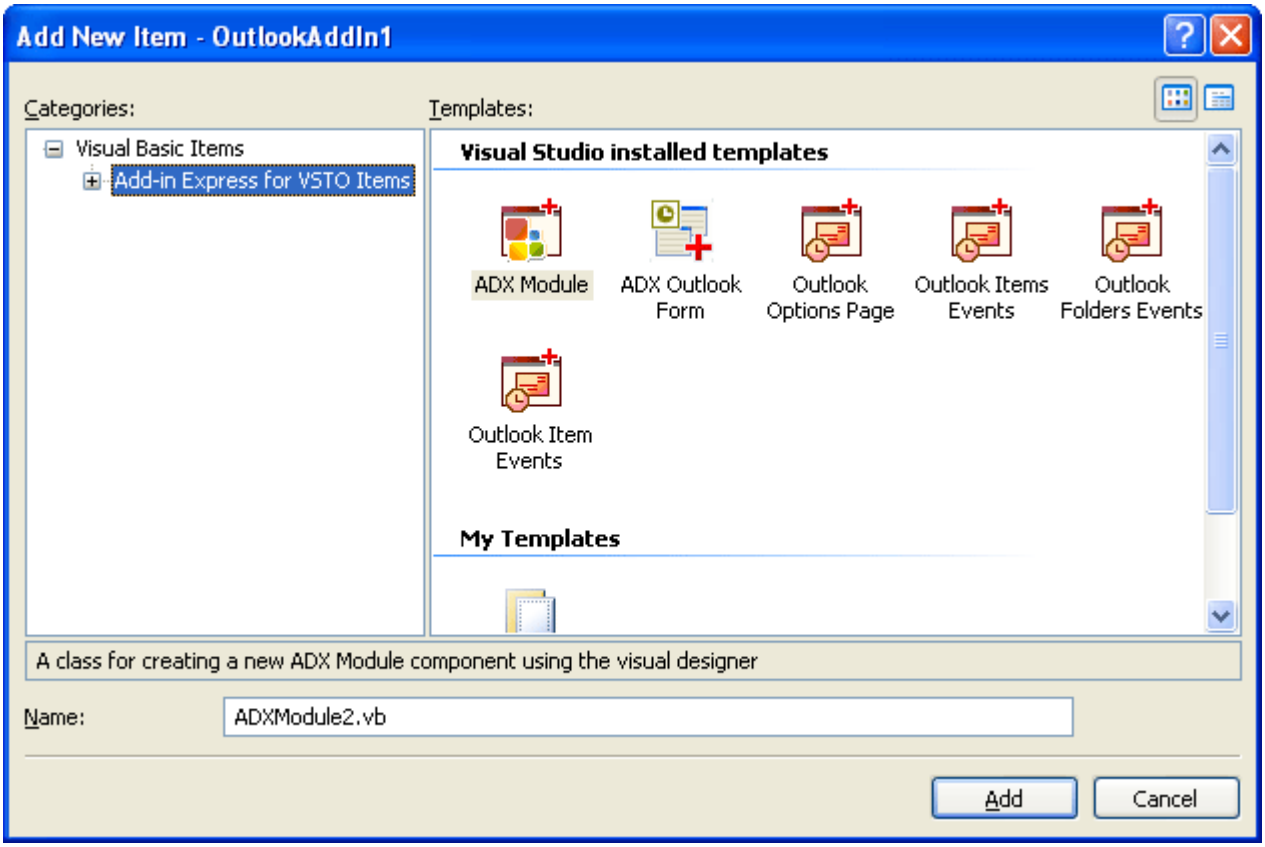
Add-in Express uses interfaces from the Office Type Library. We do not describe them here. Please refer to the VBA help and to application type libraries.

Getting Help on COM Objects, Properties and Methods

To get assistance with host applications' objects, their properties and methods as well as help info, use the Object Browser. Go to the VBA environment (in the host application, choose menu Tools / Macro / Visual Basic Editor or just press Alt+F11), press F2, select the host application (also Office and MSForms) in the topmost combo and/or specify a search string in the search combo.

Add New Item Dialog

Add-in Express 2007 for VSTO installs the following items to the Add New Item dialog (right-click you project item in the Solution Explorer and choose the Add New Item menu).



- Add-in Express Module – the core of any Add-in Express based add-in. See Add-in Express Module.
- Add-in Express Outlook Form – a form designed for embedding into the Outlook Explorer and Inspector windows. To use this functionality, you will have to install appropriate Add-in Express 2007 for VSTO package. See the Pricing page for details.
- Outlook Options Page – the form designed for extending Outlook Options and Folder Properties dialogs with custom pages. See Outlook Property Page and Your First Microsoft Outlook Add-in.
- Outlook Items Event Class – provides easy access to the events of the Items class of Outlook.
- Outlook Folders Event Class – provides easy access to the events of the Folders class of Outlook.
- Outlook Item Event Class – provides easy access to the events of the MailItem, TaskItem, ContactItem, etc classes of Outlook.

Add the COM Add-ins Command to a Toolbar or Menu

To add the COM Add-ins command to a toolbar or menu in Office 2000-2003 as well as to Office 2007 applications that do not have Ribbon user interface you do the following:

- Open the host application (Outlook, Excel, Word, etc)
- On the Tools menu, click Customize.

- Click the Commands tab.
- In the Categories list, click the Tools category.
- In the Commands list, click COM Add-Ins and drag it to a toolbar or menu of your choice.

How to Get Access to the Add-in Host Applications

For your convenience, the Add-in Module provides host-related properties to the Add-in module, such as OutlookApp and ExcelApp. Also, the ThisApplication property returns the ThisApplication property of the VSTO add-in.

Registry Entries

COM Add-ins registry entries are located in the following registry branches:

```
   HKEY_CURRENT_USER\Software\Microsoft\Office\<OfficeApplication>\AddIns\<Add-
in ProgID>
   HKEY_CLASSES_ROOT\CLSID\<Add-in Express Project GUID>
```

Outlook CommandBar Visibility Rules

Add-in Express displays the Explorer command bar for every folder, which name **AND** type correspond to the values of FolderName, FolderNames, and ItemTypes properties. For the Inspector toolbar, the same rule applies to the folder in which an Outlook Item is opened or created.

Event classes

An Event class cannot be connected to several event sources (say, Items collections of several folders). Instead, you create several instances of the Event class.

ControlTag vs Tag

Add-in Express identifies all its controls (command bar controls) using the ControlTag property (the Tag property of the CommandBarControl interface). The value of this property is generated automatically and you don't need to change it. For your own needs, use the Tag property instead.

Pop-ups

According to the Microsoft's terminology, the term "pop-up" can be used for several controls: pop-up menu, pop-up button, and submenu. With Add-in Express you can create your own pop-up as a command bar control and populate it using the Controls property.

But pop-ups have a very annoying feature: if an edit box or a combo box is added to a pop-up, their events are fired very oddly. Don't regard this bug as that of ADX. Seems, it was introduced by MS intentionally.

CommandBar.Position = adxMsoBarPopup

This option allows displaying the commandbar as a popup (context) menu. In the appropriate event handler, you write the following code:

```
AdxOlExplorerCommandBar1.CommandBarObj.GetType.InvokeMember("ShowPopup", _
 Reflection.BindingFlags.InvokeMethod, Nothing, _
 AdxOlExplorerCommandBar1.CommandBarObj, Nothing)
```

The same applies to other commandbar types.

CommandBar.Position = adxMsoBarMenuBar

This option can be used in some scenarios with Excel solutions only.

Edits, Combos, and the Change Event

The Change event appears only after the control's value is changed AND the focus is shifted. This is not our bug either, but MS people's "trick".

Removing Custom Command Bars and Controls

Add-in Express removes custom command bars and controls while add-in is uninstalled. However, this doesn't apply to Outlook and Access add-ins. You should set the Temporary property of custom command bars (and controls) to true to notify the host application that it can remove them itself. If you need to remove a toolbar or button yourself, use the Tools | Customize dialog.

Built-in Controls and Command Bars

You can connect an ADXCommandBar instance to any built-in command bar. For example, you can add your own controls to the "Standard" command bar or remove some controls from it. To do this, just add to the add-in module a new ADXCommandBar instance and specify the name of the built-in command bar you need via the CommandBarName property.

Also, you can add any built-in controls to your own command bars. Just add an ADXCommandBarControl instance to the ADXCommandBar.Controls collection and specify the Id of the built-in control you need via the Id property. Pay attention please that in this case you just add the built-in control but do not handle its event excluding buttons with witch you also can disable its standard action. To handle events of other built-in controls use the ADXBuiltInControl object.

Add-ins for Outlook - Template Characters in FolderName

Regardless of the fact that the default value of the FolderName property is '*' (asterisk), which means "every folder", the current version doesn't support template characters in the FolderName(s) property value. Moreover, this is the only use of the asterisk recognizable in the current version.

VSTO solution deployment

To understand the deployment of VSTO projects, use the MSDN site. Below there are some of the links that could be of interest for you:

1. Deploying Office Solutions: http://msdn2.microsoft.com/en-us/library/hesc2788.aspx

2. Deploying Visual Studio 2005 Tools for Office Solutions Using Windows Installer (Part 1 of 2): http://msdn.microsoft.com/office/default.aspx?pull=/library/en-us/odc_vsto2005_ta/html/OfficeVSTOWindowsInstallerOverview.asp

Notes

1. Add-in Express 2007 for VSTO adds a single assembly to your setup project. The assembly is AddinExpress.VSTO.dll. In your setup project, you should add it to the Application Folder.

2. The links were correct at the moment of writing.

3. God bless you, VSTO solution deployers!

Final Note

If your questions are not answered here, please see the HOWTOs section on www.add-in-express.com. From time to time, we update these pages ;-).