Official Guide

# Add-in Express™ .net

## Getting Started

# Add-in Express™ 2008 for Microsoft® .NET

Revised at **18-Dec-07**

# Table of Contents

# Introduction

*Add-in Express .NET is a development tool designed to simplify and speed up the development of Office COM Add-ins, Run-Time Data servers (RTD servers), Smart Tags, and Excel Automation Add-ins in Visual Studio 2003, Visual Studio 2005, and Visual Studio 2008 through the consistent use of the RAD paradigm. It provides a number of specialized components that allow the developer to walk through the interface-programming phase to the functional programming phase with a minimal loss of time.*

# Why Add-in Express?

Microsoft introduced the term Office Extensions. This term covers all the customization technologies provided for Office applications. The technologies are:

- COM Add-ins
- Smart Tags
- Excel RTD Servers
- Excel Automation Add-ins
- Excel XLL Add-ins

Add-in Express allows you to overcome the basic problem when customizing Office applications in .NET – building your solutions into the Office application. Based on the True RAD paradigm, Add-in Express saves the time that you would have to spend on research, prototyping, and debugging numerous issues of any of the above-said technologies in all versions and updates of all Office applications. The issues include safe loading / unloading, host application startup / shutdown, as well as user-interaction and deployment issues.

Add-in Express provides you with simple tools for creating version-neutral, secure, insolated, managed, deployable, and updatable Office extensions.

- **Managed Office Extensions**

You develop them in every programming language available for Visual Studio .NET.

- **Isolated Office Extensions**

Add-in Express allows loading Office extensions into separate application domains. So, they do not have a chance to break down the host application.

- **Version-neutral Office Extensions**

The Add-in Express programming model and its core are version-neutral. It gives you a chance to develop one Office extension for all available Office versions, from 2000 to the newest 2007.

- **Deployable Office Extensions**

Add-in Express automatically supplies you with a setup project making your solution ready-to-deploy.

- **Updatable Office Extensions**

Add-in Express uses the start-up and deployment model that allows updating your deployed solutions at run-time.

# Add-in Express Extensions

Depending on a product package, Add-in Express includes plug-ins that allow a deeper customization of Office applications using Add-in Express COM Add-ins.

- **The Add-in Express Extensions .NET for Microsoft Outlook**

The Add-in Express Extensions allows Outlook COM Add-in developers to embed .NET forms into Outlook Explorer and Inspector windows. See the product description at http://www.add-in-express.com/outlook-extension/.

- **The Add-in Express Extensions .NET for Microsoft Excel**

The Add-in Express Extensions allows Excel COM Add-in developers to embed .NET forms into Excel windows.

- **The Add-in Express Extensions for Microsoft Office Toolbars**

Office COM Add-in developers can use the Add-in Express Extensions for Microsoft Office Toolbars to place any .NET controls (such as grids, tree views, lists) on Microsoft Office toolbars. See the product description at http://www.add-in-express.com/office-toolbar-controls/.

- **Outlook Security Manager**

This is a standalone product designed for Outlook solution developers. It allows controlling Outlook E-mail Security Guard by turning it off and on in order to suppress unwanted Outlook Security warnings.

# System Requirements

## Supported IDEs

### Visual Studio 2003 Editions

- Visual Studio .NET 2003 Enterprise Architect Edition
- Visual Studio .NET 2003 Enterprise Developer Edition
- Visual Studio .NET 2003 Professional Edition
- Visual Studio .NET 2003 Standard Edition
- RemObjects Chrome 1.53 and higher

### Visual Studio 2005 Editions

- Visual Studio .NET 2005 Team System
- Visual Studio .NET 2005 Professional Edition
- Visual Studio .NET 2005 Standard Edition
- Visual Basic .NET 2005 Express
- Visual C# 2005 Express
- RemObjects Chrome 1.53 and higher

### Visual Studio 2008 Editions

- Visual Studio .NET 2008 Team System
- Visual Studio .NET 2008 Professional Edition
- Visual Studio .NET 2008 Standard Edition
- Visual Basic .NET 2008 Express
- Visual C# 2008 Express

#### Not supported

*Visual Studio 2003 Trial (Learning) Edition is not supported.*

## Supported Languages

- Visual Basic .NET 2003
- Visual Basic .NET 2005

- Visual Basic .NET 2008
- Visual Basic .NET 2005 Express
- Visual Basic .NET 2008 Express
- Visual C# 2003
- Visual C# 2005
- Visual C# 2008
- Visual C# 2005 Express
- Visual C# 2008 Express
- Visual C++ .NET 2003
- Visual C++ .NET 2005
- Visual C++ .NET 2008

## Host Applications

### COM Add-ins

- Microsoft Excel 2000 and higher
- Microsoft Outlook 2000 and higher
- Microsoft Word 2000 and higher
- Microsoft FrontPage 2000 and higher
- Microsoft PowerPoint 2000 and higher
- Microsoft Access 2000 and higher
- Microsoft Project 2000 and higher
- Microsoft MapPoint 2002 and higher
- Microsoft Visio 2002 and higher
- Microsoft Publisher 2003 and higher
- Microsoft InfoPath 2007

### Real-Time Data Servers

- Microsoft Excel 2002 and higher

### Smart Tags

- Microsoft Excel 2002 and higher
- Microsoft Word 2002 and higher
- Microsoft PowerPoint 2003 and higher

### Excel Automation Add-ins

- Microsoft Excel 2002 and higher

### Excel XLL Add-ins

- Microsoft Excel 2000 and higher

#### Office editions, service packs and updates

*Add-in Express supports all Microsoft Office service packs and updates for all Microsoft Office editions including Student, Basic, Standard, Professional, Small Business, Enterprise, etc.*

# Technical Support

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

## Add-in Express Web Site

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

- The HOWTOs section that contains sample projects answering most common "how to" questions.
- Add-in Express Toys contains entire and "open sourced" add-ins for popular Office applications.
- Built-in Controls Scanner utility. It is free.
-

## Support

For technical support through the Internet, e-mail us at support@add-in-express.com or use our forums. We are actively participating in these forums.

## 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.

# Installing and Activating

There are two main points in the Add-in Express .NET installation. First off, you have to specify the development environments that you need to use Add-in Express .NET in. See Supported IDEs. Second, you need to activate the product. What follows below is a brief guide on activating.

## Activation Basics

During software registration, the registration wizard prompts you to enter your license key. The key is a 30 character alphanumeric code shown in six groups of five characters each (for example, AXN4M-GBFTK-3UN78-MKF8G-T8GTY-NQS8R). Keep the license key in a safe location and do not share it with others. This product key forms the basis for your ability to use the software.

For purposes of product activation only, a non-unique hardware identifier is created from general information that is included in the system components. At no time are files on the hard drive scanned, nor is personally identifiable information of any kind used to create the hardware identifier. Product activation is completely anonymous. To ensure your privacy, the hardware identifier is created by what is known as a "one-way hash". To produce a one-way hash, information is processed through an algorithm to create a new alphanumeric string. It is impossible to calculate the original information from the resulting string.

**Your product key and a hardware identifier are the only pieces of information required to activate the product. No other information is collected on your PC or sent to the activation server.**

If you choose the **Automatic Activation Process** option of the Activation Wizard, the wizard attempts to establish an online connection to the activation server, www.activatenow.com. If the connection is established, the wizard sends both the license key and the hardware identifier over the Internet. The activation service generates an activation key using this information and sends it back to the activation wizard. The wizard saves the activation key to the registry.

If an online connection cannot be established (or you choose the **Manual Activation Process** option), you can activate the software using your web-browser. In this case, you will be prompted to enter the product key and a hardware identifier on a Web page, and will get an activation key in return. This process finishes with saving the activation key to the registry.

Activation is completely anonymous; no personally identifiable information is required. The activation key can be used to activate the product on that computer an unlimited number of times. However, if you need to install the product on several computers, you will need to perform the activation process again on every PC. Please refer to your end-user license agreement for information about the number of computers you can install the software on.

## Setup Package Contents

The Add-in Express 2008 for .NET setup program installs the following folders on your PC:

- Bin – Add-in Express .NET binary files
- Demo Projects – demo projects for Visual Studio 2003 and Visual Studio 2005
- Docs - Add-in Express .NET documentation
- Docs \ Samples – sample documentation projects
- Images – Add-in Express .NET icons
- Redistributables – Add-in Express .NET redistributable files
- Sources - Add-in Express .NET source code (see the note below).

### Where is the Add-in Express .NET source code?

*Please note that we include the source code of Add-in Express .NET according to the product package you purchased. See the Packages & Pricing page on our web site for details.*

Add-in Express .NET setup program installs the following text files on your PC:

- licence.txt – EULA
- readme.txt – short description of the product, support addresses and such
- whatsnew.txt – this file describes the latest information on the product features added and bugs fixed.

# Getting Started

In this chapter, we guide you through the following steps of developing Add-in Express Projects:

- Create an Add-in Express project
- Add an Add-in Express Designer to the project
- Add Add-in Express components to the Add-in Express Designer
- Add some business logics
- Build, register, and debug the Add-in Express project
- Tune up the Add-in Express Loader based setup project
- Deploy your project to a target PC

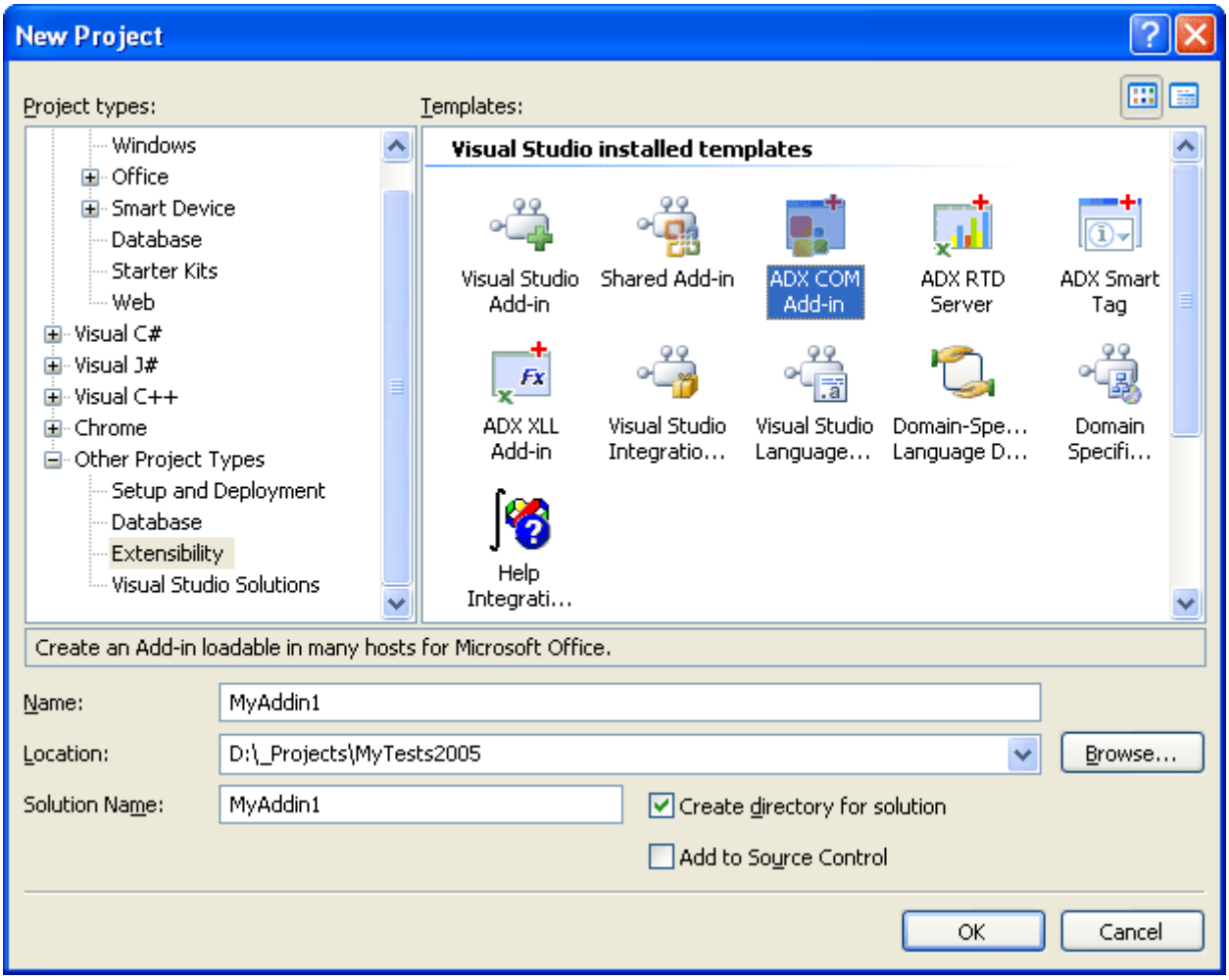
**Note.**

*You find all the below described samples in the following folder on your PC:*

*C:\Program Files\Add-in Express\Add-in Express 2008 for .NET <ProductPackage>\Docs\Samples*

# Creating Add-in Express Projects

Add-in Express .NET adds several project templates to the Extensibility folder of the New Project dialog. To see the dialog, choose the “File | New | Project…” menu.




Whichever Add-in Express project template you use, it starts the Add-in Express project wizard that allows selecting a programming language for your Add-in Express project as well as other options. The project wizard creates the solution and adds two projects: an Add-in Express project and setup project. It also adds an appropriate Add-in Express designer to the Add-in Express project (see Add-in Express Designers). The designers are the core components of Add-in Express .NET. They allow adding other Add-in Express components and setting their properties at design-time as well as at run-time (see Add-in Express Components).

# Add-in Express Designers

There are several Add-in Express designer types responsible for common tasks in customizing Office in .NET. Below we list the Office automation tasks. Each task includes an introductory description, the Add-in Express designer type available for the task, and Add-in Express components available.

- COM Add-ins - ADXAddinModule
- RTD Servers - ADXRtdServerModule
- Smart Tags - ADXSmartTagModule
- Excel Automation Add-ins - ADXExcelAddinModule
- Excel Workbooks - ADXExcelSheetModule
- Word Documents – ADXWordDocumentModule
- Excel XLL Add-ins – ADXXLLModule.

## COM Add-ins

COM Add-ins have been around since Office 2000 when Microsoft allowed Office applications to extend their features with COM DLLs supporting the IDTExtensibility2 interface (it is a COM interface, of course). Since then thousands of developers have racked their brains over this interface and the Office Object Model that provided COM objects representing command bars, command bar controls, etc. These were the sources of Add-in Express.

ADXAddinModule represents a COM Add-in in any Office application. To add another add-in to your assembly, add another ADXAddinModule to your project. For the add-in, you specify its name, host application(s) and load behavior. The typical value for the LoadBehavior property is Connected & LoadAtStartup. 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 ADXAddinModule: Office 2007 Ribbon Components, Command Bars, Command Bar Controls, Built-in Control Connector, Keyboard Shortcut, Outlook Bar Shortcut Manager, Advanced Form Regions in Outlook 2000-200, and Application-level Events.

Use the AddinStartupComplete and AddinBeginShutdown events to handle add-in startup and shutdown.

## COM Add-ins and ClickOnce

ClickOnce is a manifest-driven deployment technology introduced in .NET Framework 2.0. In ClickOnce Overview, you can find that the technology doesn't allow installing a COM add-in. However, with the Add-in Express ClickOnce Solution you are able to achieve this. In the base of the ClickOnce Solution, you find a special application (a ClickOnce application) tailored for COM add-in requirements. Being supplied with add-in files and properly configured, the application executes COM Add-in specific actions when installed and uninstalled. You can handle the corresponding events via Add-in Express ClickOnce Module. When added to your Add-in Express Loader based add-in project, the ClickOnce module allows handling the following

add-in-related actions: install, uninstall, register, unregister, update to a newer version, and revert to the previous version. In addition, the ClickOnce module allows you to show custom GUI when the user invokes the ClickOnce application from the Start menu: if you don't process the corresponding event, the standard GUI of the Add-in Express ClickOnce application will be shown.

There are no Add-in Express components to use with the ClickOnce module.

## RTD Servers

RTD Server is a technology introduced in Excel XP. It is a great way to display constantly changing data such as stock quotes, currency exchange rates, inventory levels, price quotes, weather information, sports scores, and so on.

A short terminology list follows below:

- An RTD server is a Component Object Model (COM) Automation server that implements the IRtdServer interface. Excel uses the RTD server to communicate with a real-time data source on one or more topics.
- A real-time data source is any source of data that you can access programmatically.
- A topic is a string (or a set of strings) that uniquely identifies a piece of data that resides in a real-time data source. The RTD server passes the topic to the real-time data source and receives the value of the topic from the real-time data source; the RTD server then passes the value of the topic to Excel for display. For example, the RTD server passes the topic "New Topic" to the real-time data source, and the RTD server receives the topic's value of "72.12" from the real-time data source. The RTD server then passes the topic's value to Excel for display.

ADXRtdServerModule represents an RTD Server. The only Add-in Express component allowed for this designer is RTD Topic. The module provides the Interval property that indicates the time interval between updates (in milliseconds).

You refer to an existing RTD Server using the RTD worksheet function in Excel:

```
=RTD(ProgID, Server, String1, String2, ... String28)
```

The ProgID parameter is a required string value representing the programmatic ID (ProgID) of the RTD server. See attributes of the RTDServerModule class for the ProgId of your RTD Server. The current version of Add-in Express requires the Server parameter to be an empty string. Use two quotation marks (""). The String1 through String28 parameters represent topics of the RTD server. Only the String1 parameter is required; the String2 through String28 parameters are optional. In most cases, the String1 parameter will be enough for you. The actual values for the String1 through String28 parameters depend on the requirements of the real-time data server.

## Smart Tags

Office XP bestowed Smart Tags upon us in Word and Excel. Office 2003 added PowerPoint to the list of smart tag host applications. This technology provides Office users with more interactivity for the content of their Office documents. A smart tag is an element of text in an Office document having custom actions associated with it. Smart tags allow recognizing such text using either a dictionary-based or a custom-processing approach. An example of such text might be an e-mail address you type into a Word document or an Excel workbook. When smart tag recognizes the e-mail address, it allows the user to choose one of the actions associated with the text. For e-mail addresses, possible actions are to look up additional contact information or send a new e-mail message to that contact.

ADXSmartTagModule lies at the base of the Add-in Express Smart Tags. It represents a set or a library of smart tag recognizers in Excel, Word, and PowerPoint. The only Add-in Express component you add to the designer is Smart Tag.

## Excel Automation Add-ins

Excel 2002 brought in Automation Add-ins – a technology that allows writing user-defined functions for use in Excel formulas. Add-in Express .NET provides you with a specialized module, COM Excel Add-in Module, that reduces this task to just writing one or more user-defined functions. A typical function accepts one or more Excel ranges and/or other parameters. Excel shows the resulting value of the function in the cell where the user calls the function.

Add-in Express .NET provides developing Excel Automation Add-ins using the COM Excel Add-in Module in COM Add-in projects. You add this module to your COM Add-in project by choosing the appropriate item in the Add New Item Dialog. The module represents an Excel Automation add-in in Add-in Express. It does not provide any properties. Also, there are no Add-in Express components to use with the module.

## Excel XLL Add-ins

An XLL is a DLL written in such a way that Excel can open it directly. Like Excel Automation add-ins, XLL add-ins are mostly used to create user-defined functions, however much faster. This technology was introduced in Excel 4.0 (Wikipedia states that this Excel version was released in 1992!). Since then, XLL interfaces have been available for C and C++ developers only. For .NET developers, Add-in Express includes the XLL Add-in project template (in the Extensibility folder of the New Project dialog) and the XLL Add-in Module that hides XLL complexities.

The XLL Add-in module contains a special class, XLLContainer, where you add your **public static** (in VB, **Public Shared**) functions. Just adding a function is enough for a quick start. Using the module's designer, you are able to specify all other function-related stuff: description, help reference, category, descriptions of the function's parameters, etc. In addition, you can instruct Excel to call your function whenever recalculation is required (IsVolatile property). Another option is specifying a parameter of the Object type to accept Excel ranges as a reference to an object of the ADXExcelRef type or as a 2D array of values.

**Note**

*The current version of Add-in Express supports XLL except for multi-threaded calculations introduced in Excel 2007.*

### Excel Workbooks

Sometimes you need to automate a given Excel workbook (template). You can do it with ADXExcelSheetModule that represents one worksheet of the workbook. The Document property allows creating and browsing for the workbook. If you choose creating a new workbook, the dialog appears where you specify the name and location of the workbook as well as the Property Name and Property Value textboxes. Add-in Express adds this property to the list of custom properties of the workbook and uses the name and value of the property in order to recognize the workbook. Accordingly, you specify the PropertyId and PropertyValue properties of the module. The module provides a full set of events available for Excel workbook.

For the Add-in Express components available for the module see the following chapters: Command Bars, Command Bar Controls, Built-in Control Connector, and Application-level Events.

### Word Documents

To automate a given Word document, you use the ADXWordDocumentModule. The module allows creating and browsing for the document. If you choose creating a new document, the dialog appears where you specify the name and location of the document as well as the Property Name and Property Value textboxes. Add-in Express adds this property to the list of custom properties of the document and uses the name and value of the property in order to recognize the document. Accordingly, you specify the PropertyId and PropertyValue properties of the module. The module provides a full set of events available for Word document.

For the Add-in Express components available for the module see the following chapters: Command Bars, Command Bar Controls, Built-in Control Connector, and Application-level Events.

## Add-in Express Components

### How to Add an Add-in Express Component to Add-in Express Designer

To add any Add-in Express .NET 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 Solution Explorer, right-click the Add-in Express module 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 COM 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 that 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.

Note also that to use pre-Office2007 command bars in the Office 2007 add-ins, you must explicitly set to True the UseForRibbon property of the appropriate command bar components. In this case, your toolbars are 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 COM Add-in module with the TaskPanes property. Add an instance of 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 COM Add-in module and the DockPositionStateChange and VisibleStateChange events of the task pane item. Use the OfficeColorSchemeChanged event and the OfficeColorScheme property for your task pane colors to conform to the current Office 2007 color scheme.

## Advanced Form Regions in Outlook 2000-2007

Customizing Outlook has a long-dated history. To customize an Outlook form you can use the built-in tools for form designing and publishing (see the Tools | Forms | Design a Form menu in Outlook). 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, which is a core component of the form-embedding technology, is available for ADXAddinModule only. You can find the detailed information on this technology at http://www.add-in-express.com/outlook-extension/.

## Custom Task Panes in Excel 2000-2007

The only Excel version that allows adding Custom Task Panes is Excel 2007. To be more precise, VSTO 2003 as well as VSTO 2008 allows customizing Excel 2003 workbooks and templates with Action Panes, not Task Panes. Now, with Add-in Express, you can create custom task panes in Excel 2000-2007. To achieve this, Add-in Express implements a special .NET form class (a descendant of Windows.Forms.Form) that can be embedded into Excel windows. The class is called ADXExcelTaskPane. In order to be embedded into Excel windows, all your Excel task panes forms must be descendants of ADXExcelTaskPane. To add such a task pane to your add-in project, you choose the corresponding item in the Add New Item dialog (see the Add New Item Dialog). ADXExcelTaskPane provides several Excel-specific properties and events that can be used to access Excel objects from the task pane form.

The Excel Forms Manager component, which orchestrates custom task panes, is available for ADXAddinModule only.

## 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. 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 our 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.



*To see a pre-Office2007 command bar in the Ribbon UI, set the UseForRibbon property of the appropriate Add-in Express command bar component to true.*

## Command Bar Controls

The Office Object Model 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 .NET 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)

Please note that due to the nature of command bars, [context] menu items can be buttons, combo boxes, and pop-ups. See also Terminology.

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 corresponding property of the command bar control component. To find out the Ids of every built-in control in the host application, use our 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 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.

## Smart Tag

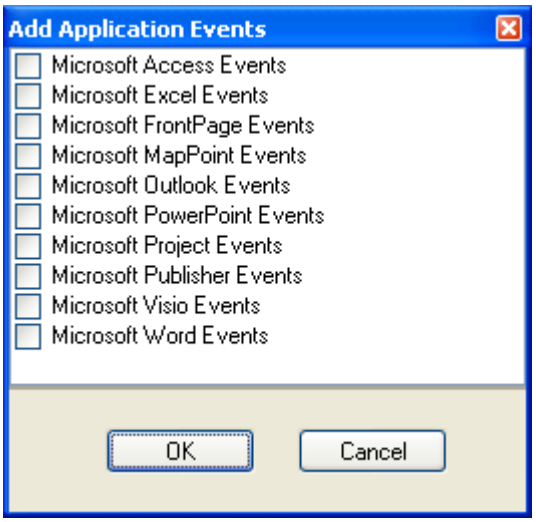
The Kind property of the ADXSmartTag component allows you to choose one of two text recognition strategies: either using a list of words in the RecognizedWords string collection or implementing a custom recognition process based on the Recognize event of the component. Use the ActionNeeded event to change the Actions collection according to the current context. The component raises the PropertyPage event when the user clicks the Property button in the Smart Tags tab (Tools / AutoCorrect Options menu) for your smart tag.

## RTD Topic

Use the String## properties to identify the topic of your RTD server. To handle RTD Server startup situations nicely, specify the default value for the topic and, using the UseStoredValue property, specify, if the RTD function in Excel returns the default value (UseStoredValue = false) or doesn't change the displayed value (UseStoredValue = true). The RTD Topic component provides you with the Connect, Disconnect, and RefreshData events. The last one occurs (for enabled topics only) whenever Excel calls the RTD function.

## Application-level Events

ADXAddinModule provides events for all Office applications through the Add Events command that shows the following dialog:




Select the application you need and click OK. This adds and/or removes appropriate Add-in Express event components to the module. Use the event handlers of an Add-in Express event component to respond to the host application's events. You may need to process other events provided by Outlook and Excel. If this is the case, see Add-in Express Event Classes.

## 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 version independent components that ease the pain of handling such events. 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.

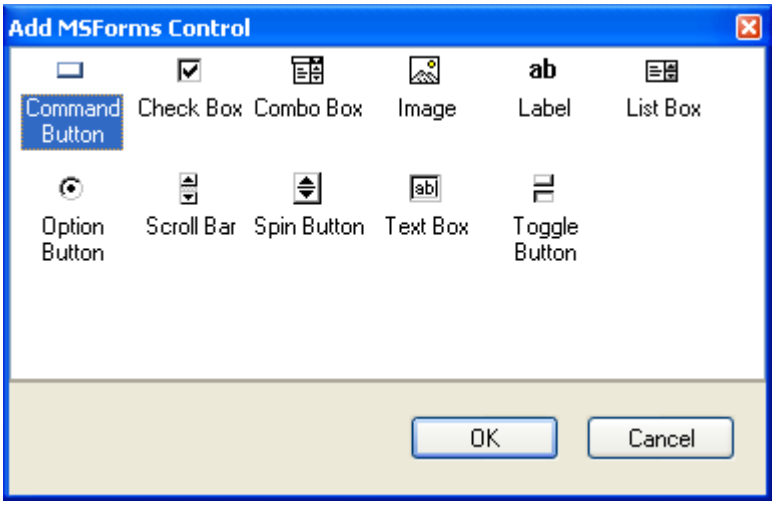
At run-time, you connect an Add-in Express event class instance to an event source using its ConnectTo method. To disconnect the Add-in Express event class from the event source you use the RemoveConnection method. To apply the same business rules to another event source of the same type (say, to items 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 (in Outlook 2007, you can also refer the Folder class).

```
Imports System

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

   Public Sub New(ByVal ADXModule As AddinExpress.MSO.ADXAddinModule)
      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
```

### MSForms Control

This command is available for ADXExcelSheetModule and ADXWordDocumentModule. When run, it displays the following dialog:




Select the control you need to connect to and click OK. Add-in Express adds an appropriate MS Forms Control Connector to the module. Use the Control Name property of the connector to specify the underlying control on the Excel worksheet or Word document. Respond to the events provided by the control connector.

# Your First Microsoft Office COM Add-in

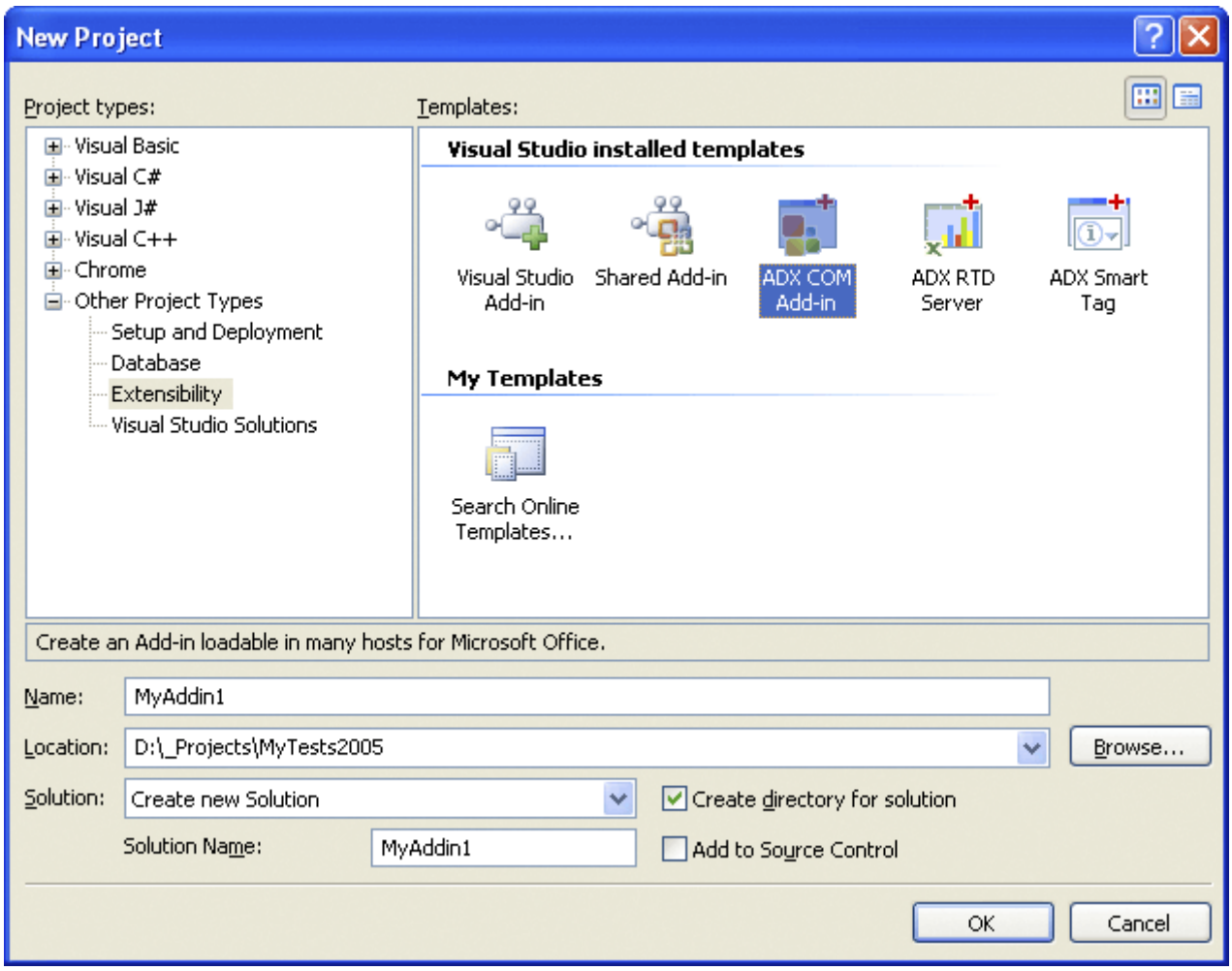
This chapter highlights almost every aspect of creating COM Add-ins for Microsoft Office applications.

### Outlook and Add-in Express

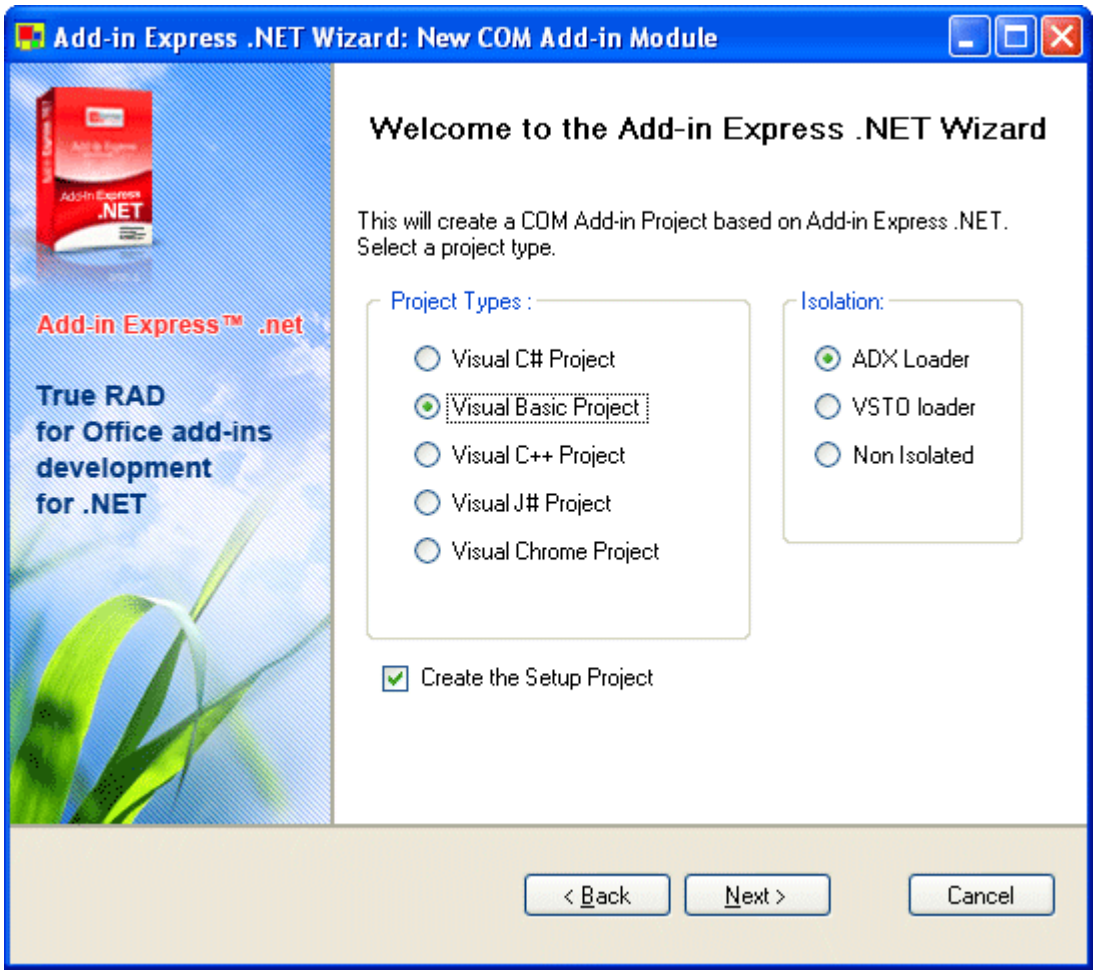
*Please note, Add-in Express provides additional components for COM Add-ins in Outlook. See Your First Microsoft Outlook COM Add-in.*

## Step #1 – Creating an Add-in Express COM Add-in Project

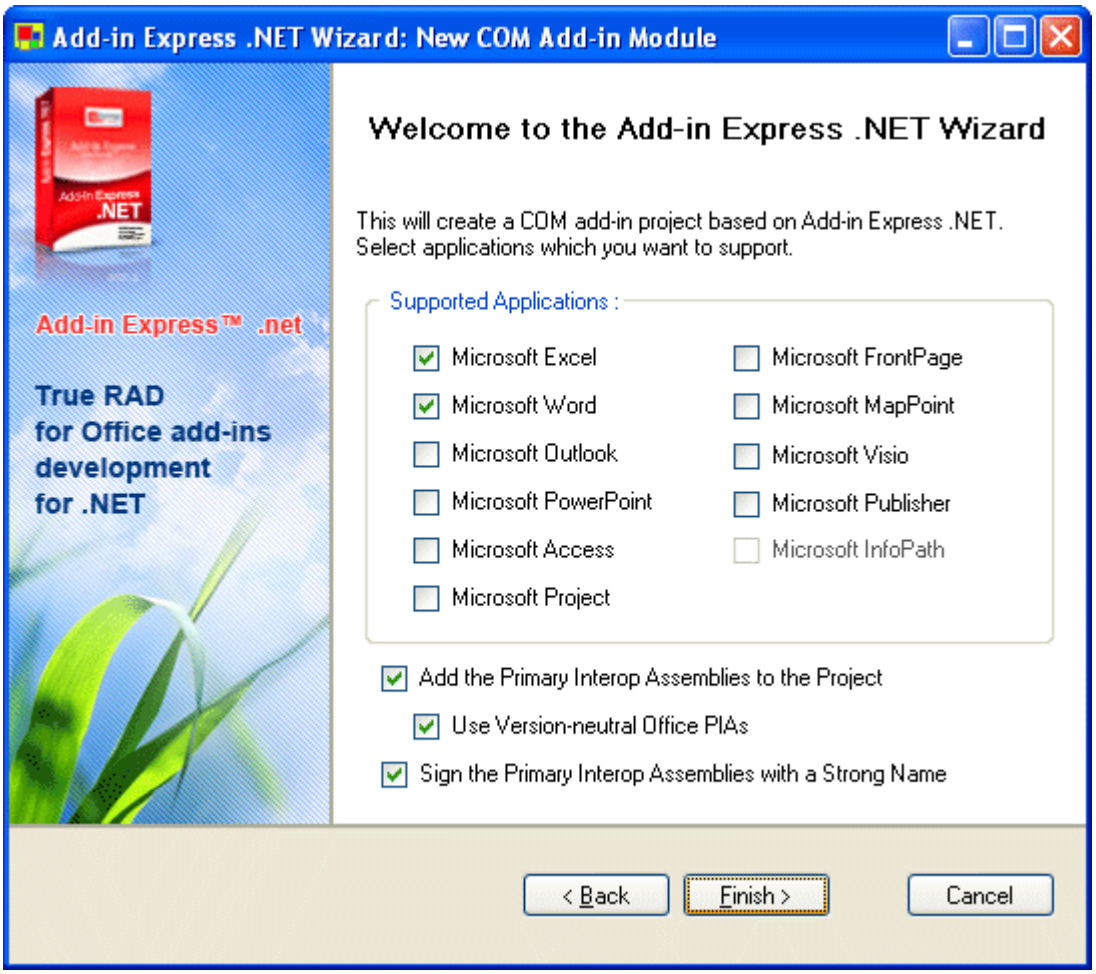
Add-in Express adds the Add-in Express COM Add-in project template to the Visual Studio IDE.



When you select the template and click OK, the Add-in Express COM Add-in project wizard starts. In the wizard windows, you choose the programming language, setup project options, applications supported by your add-in, and PIAs options. See Project Wizard Options for more details.



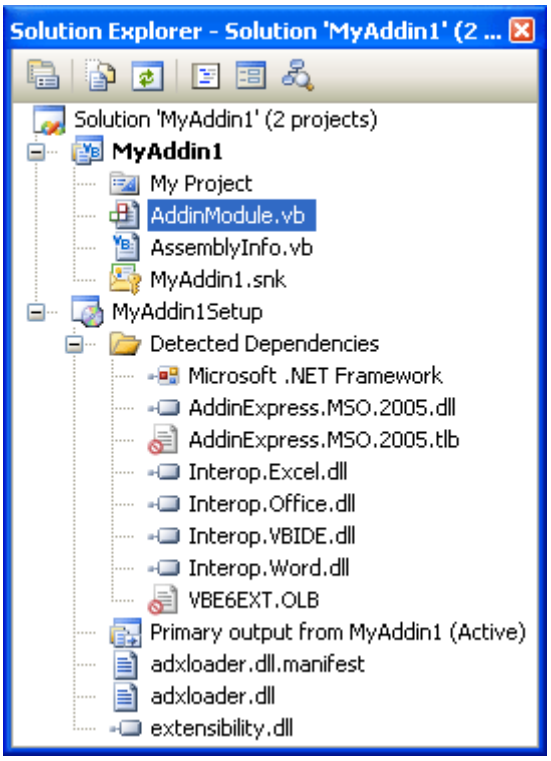

This VB.NET sample shows an Add-in Express COM Add-in project implementing a COM add-in for Excel and Word with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.




The Add-in Express Project Wizard creates and opens the COM Add-in solution in the IDE. The solution includes the COM Add-in project and the setup project.

### Your add-ins are version-neutral

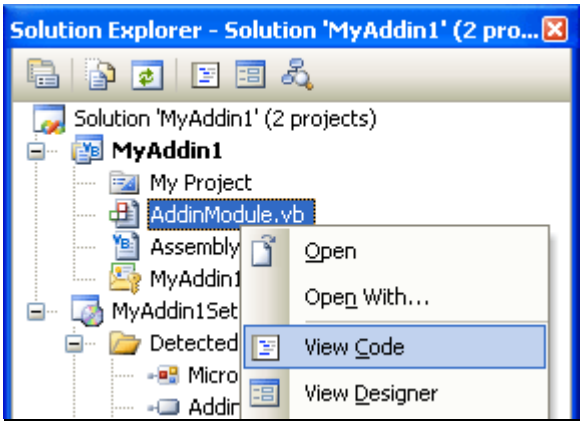
*Add-in Express delivers version-neutral Office interop assemblies that allow develop extensions for Office 2000+. To use them, simply check the Use Version-neutral Office PIAs check box when creating your projects. See also Why Version-Neutral PIAs?*




The COM Add-in project contains the AddinModule.vb (or AddinModule1.cs) file discussed in the next step.

## Step #2 – Add-in Express COM Add-in Module

The AddinModule.vb (or AddinModule1.cs) is a COM Add-in Module that is the core part of the COM add-in project (see COM Add-ins). 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 Solution Explorer, right-click the AddinModule1.vb (or AddinModule1.cs) file and choose the View Code popup menu item.




The code for AddinModule1.vb is as follows:

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express Add-in Module
<GuidAttribute("AB07BADE-56F7-414B-ACA0-D3E2DDAABCCB"), _
    ProgIdAttribute("MyAddin1.AddinModule")> _
Public Class AddinModule
    Inherits AddinExpress.MSO.ADXAddinModule

#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()
      Me.components = New System.ComponentModel.Container
      '
      'AddinModule
      '
      Me.AddinName = "MyAddin1"
      Me.SupportedApps = CType((AddinExpress.MSO.ADXOfficeHostApp.ohaExcel _
          Or AddinExpress.MSO.ADXOfficeHostApp.ohaWord), _
          AddinExpress.MSO.ADXOfficeHostApp)
   End Sub
#End Region

#Region " Add-in Express 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

    <ComRegisterFunctionAttribute()> _
    Public Shared Sub AddinRegister(ByVal t As Type)
        AddinExpress.MSO.ADXAddinModule.ADXRegister(t)
    End Sub

    <ComUnregisterFunctionAttribute()> _
```

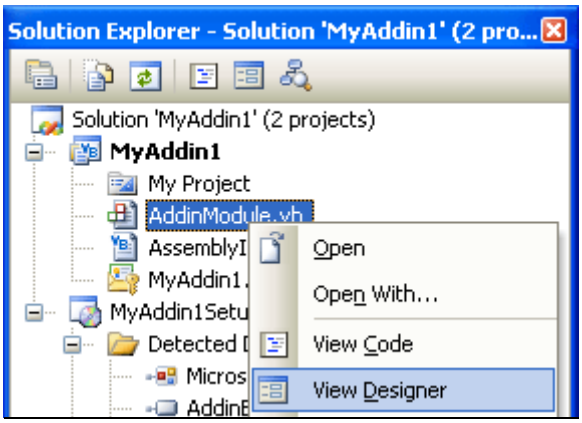
```
        Public Shared Sub AddinUnregister(ByVal t As Type)
            AddinExpress.MSO.ADXAddinModule.ADXUnregister(t)
        End Sub

        Public Overrides Sub UninstallControls()
            MyBase.UninstallControls()
        End Sub

    #End Region

        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 CType(HostApplication, Excel._Application)
            End Get
        End Property

        Public ReadOnly Property WordApp() As Word._Application
            Get
                Return CType(HostApplication, Word._Application)
            End Get
        End Property
```

Please, pay attention to the ExcelApp and WordApp properties of the module generated by the Add-in Express Project Wizard. You can use them in your code to get access to the host application objects.

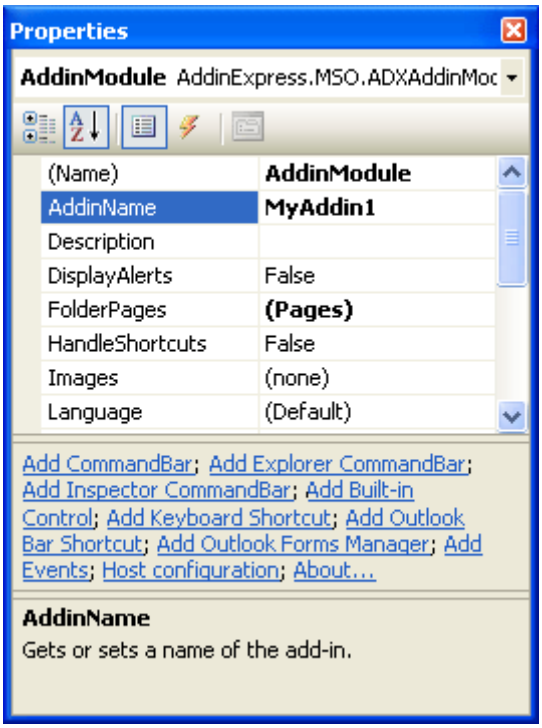
### Step #3 – Add-in Express COM Add-in Designer

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

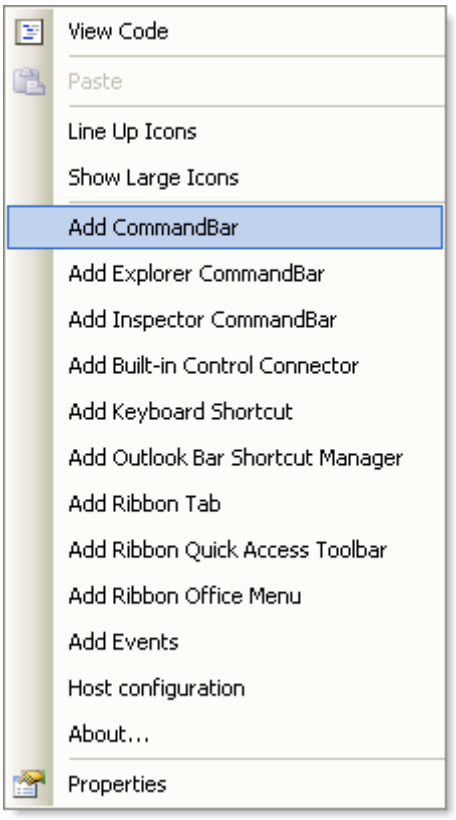
In Solution Explorer, right-click the AddinModule.vb (or AddinModule.cs) file and choose the View Designer popup menu item.




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




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.



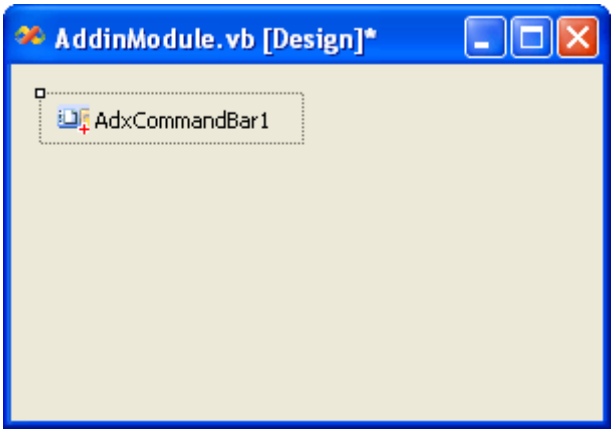

The following commands add the following components to the module:

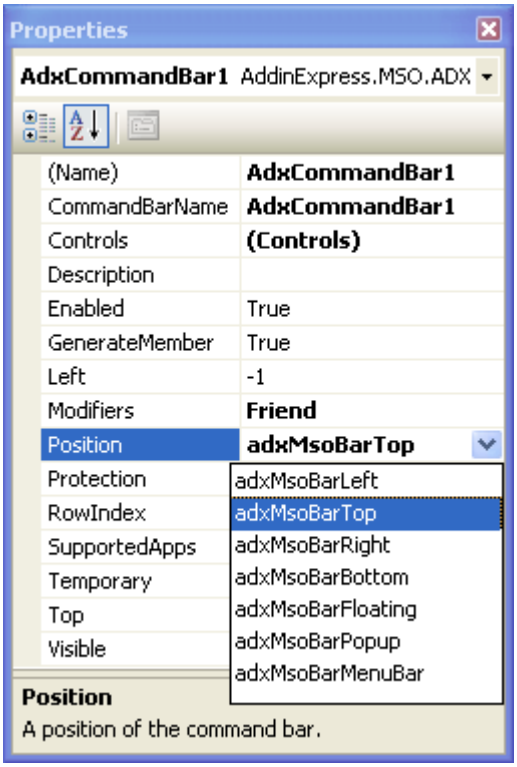
- Add CommandBar – adds a command bar to your add-in (see Command Bars)
- Add Explorer CommandBar – adds an Outlook Explorer command bar to your add-in (see Command Bars)
- Add Inspector CommandBar – adds an Outlook Inspector command bar to your add-in (see Command Bars)
- Add Built-in Control Connector – 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 Advanced Form Regions in Outlook 2000-200)
- 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)
- Add Events – adds or deletes components that provide access to application-level events of the add-in host applications (see Application-level Events)

## 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 COM 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.

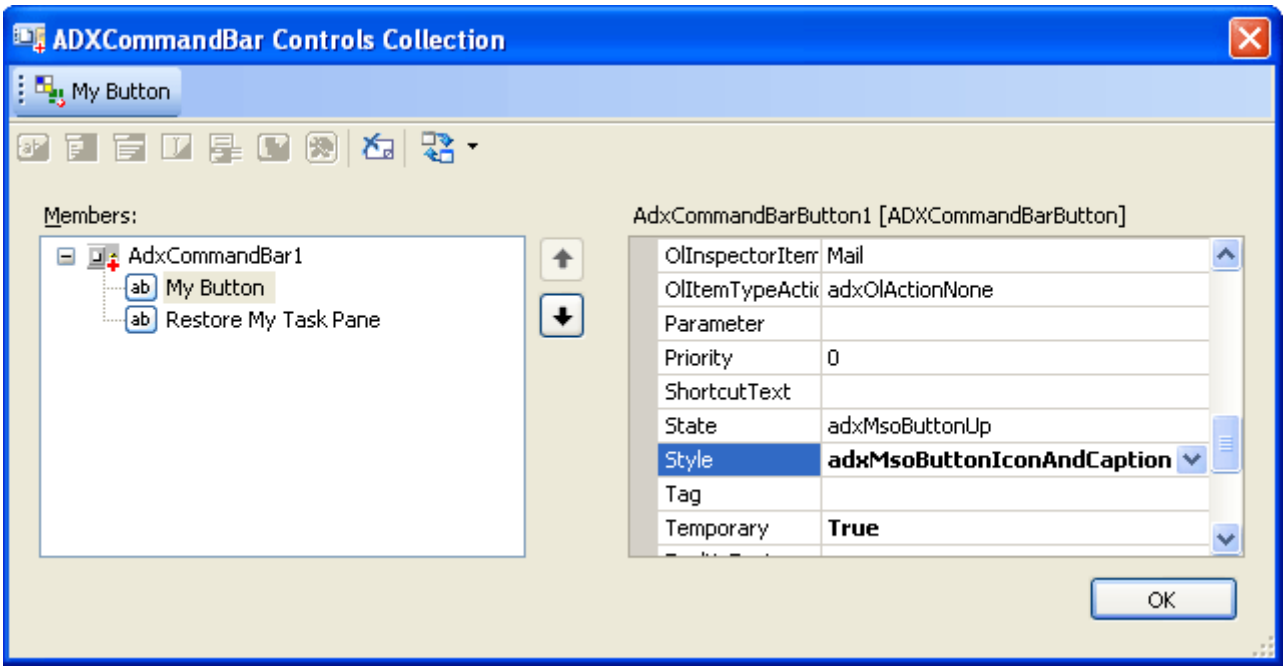
### Command Bars in Office 2007

*To display the command bar in Office 2007 you must explicitly set the UseForRibbon property of the command bar component to True.*

## 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).

This runs the visual designer. Use its toolbar to add or remove Command Bar Controls. Just click the appropriate button and see the result.




In this sample, we add a button. When it is added, select it in the tree. Then specify the button's Caption property, change the Style property if you need to show an icon in the button (default value = adxMsoButtonCaption), and close the collection editor. To handle the Click event of the button, close the editor, select the newly 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, we write the following code to the Click event of the button just added.

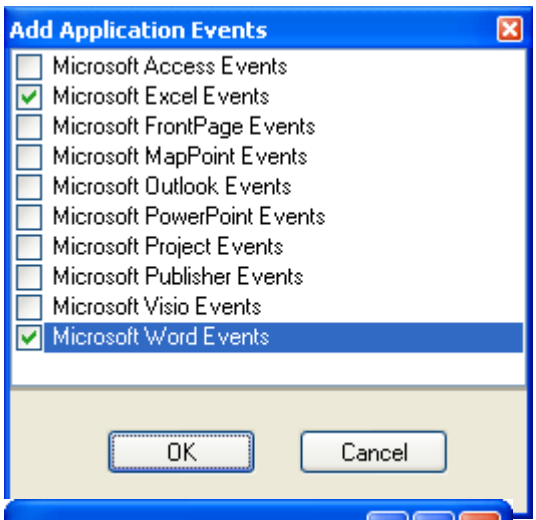
```
Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
                                    Handles AdxCommandBarButton1.Click
   Dim ActiveWindow As Object = Me.HostApplication.ActiveWindow
   MsgBox(GetInfoString(ActiveWindow))
   If Not ActiveWindow Is Nothing Then
      Marshal.ReleaseComObject(ActiveWindow)
```

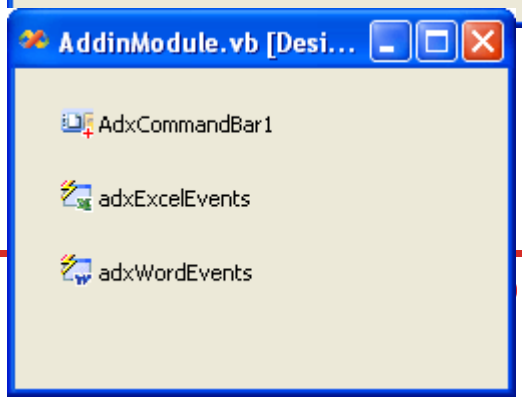
```
        End If
    End Sub

    Private Function GetInfoString(ByVal Window As System.Object) As String
        If Window IsNot Nothing Then
            Select Case Me.HostName
                Case "Excel"
                    Dim ActiveCell As Excel.Range = CType(Window, _
                        Excel.Window).ActiveCell
                    'relative address
                    Dim Address As String = ActiveCell.AddressLocal(False, False)
                    Marshal.ReleaseComObject(ActiveCell)
                    Return "The current cell is " + Address
                Case "Word"
                    Dim Selection As Word.Selection = CType(Window, _
                        Word.Window).Selection
                    Dim Range As Word.Range = Selection.Range
                    Dim Words As Word.Words = Range.Words
                    Dim WordCountString = Words.Count.ToString()
                    Marshal.ReleaseComObject(Selection)
                    Marshal.ReleaseComObject(Range)
                    Marshal.ReleaseComObject(Words)
                    Return "There are " + WordCountString _
                        + " words currently selected"
            End Select
        End If
        Return AddinName + " doesn't support " + HostName
    End Function
```

Note the use of Marshal.ReleaseComObject in the code above. See also Releasing COM objects and How Do I Find the Source Code of This Sample?

## Step #7 – Handling Host Application Events

The COM Add-in Designer provides the Add Events command that adds (and remove) event components that allow handling application-level events. When you choose this command, the Add Application Events dialog box opens allowing you to select the application Events components you need (see Application-level Events).




With the Events components, you handle any application-level events of the host application. You might see that the Click event handler in the previous step will fire an exception

when there are no workbooks or documents open. To prevent this, you disable the button when a window deactivates and enable it when a window activates. This covers the situations mentioned above.

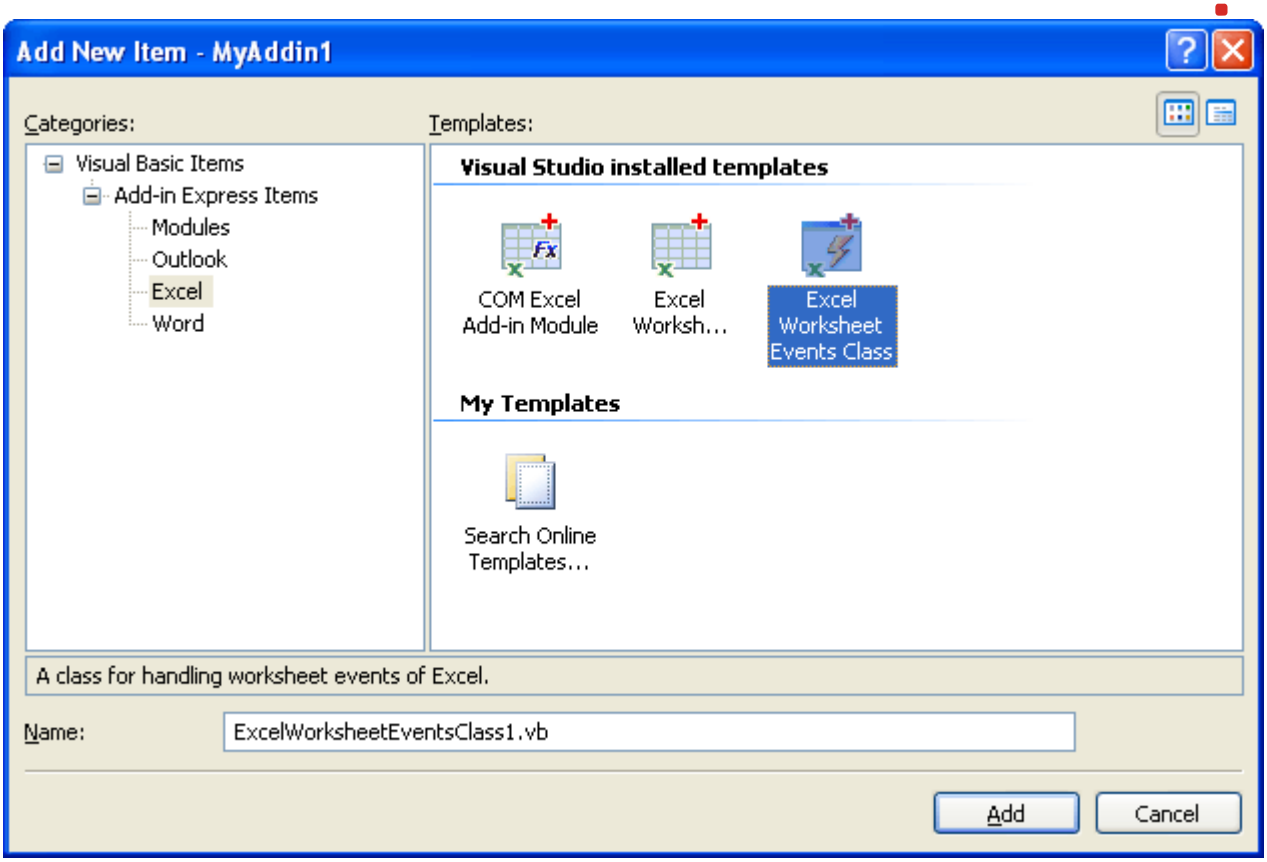
The code is as follows:

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

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

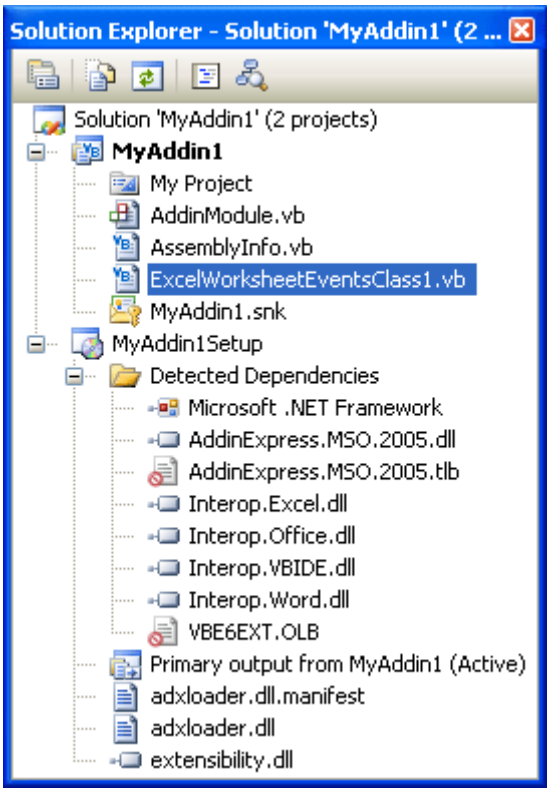
See also - How Do I Find the Source Code of This Sample?

### Step #8 – Handling Excel Worksheet Events

Add-in Express provides the Excel Worksheet event class that allows implementing a set of business rules for an Excel worksheet by handling its events. You add an event class to your project using the Add New Item dialog:




This will add an event class to your project (see Add-in Express Event Classes).



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

```
    Public Overrides Sub ProcessBeforeRightClick(ByVal Target As Object, _
       ByVal E As AddinExpress.MSO.ADXCancelEventArgs)
       Dim R As Excel.Range = CType(Target, 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
```

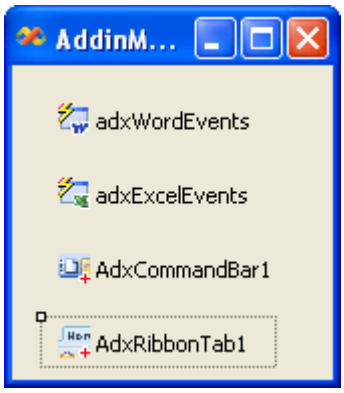
Also, you modify the Activate and Deactivate procedures as follows:

```
    Dim MyEventClass As ExcelWorksheetEventsClass1 = _
        New ExcelWorksheetEventsClass1(Me)
    Dim CurrentSheet As Excel.Worksheet
...
    Private Sub Deactivate(ByVal sender As Object, ByVal hostObj As Object, _
       ByVal window As Object) _
       Handles adxWordEvents.WindowDeactivate, _
           adxExcelEvents.WindowDeactivate
       Me.AdxCommandBarButton1.Enabled = False
       Select Case Me.HostName
          Case "Excel"
             If Not CurrentSheet Is Nothing Then
                MyEventClass.RemoveConnection()
                CurrentSheet = Nothing
             End If
          Case "Word"
          Case Else
             MsgBox(Me.AddinName + " doesn't support " + Me.HostName)
       End Select
    End Sub

    Private Sub Activate(ByVal sender As Object, ByVal hostObj As Object, _
       ByVal window As Object) _
       Handles adxWordEvents.WindowActivate, _
           adxExcelEvents.WindowActivate
       Me.AdxCommandBarButton1.Enabled = True
       Select Case Me.HostName
          Case "Excel"
             CurrentSheet = Me.ExcelApp.ActiveSheet
             MyEventClass.ConnectTo(CurrentSheet, True)
```

```
            Case "Word"
            Case Else
               MsgBox(Me.AddinName + " doesn't support " + Me.HostName)
         End Select
      End Sub
```

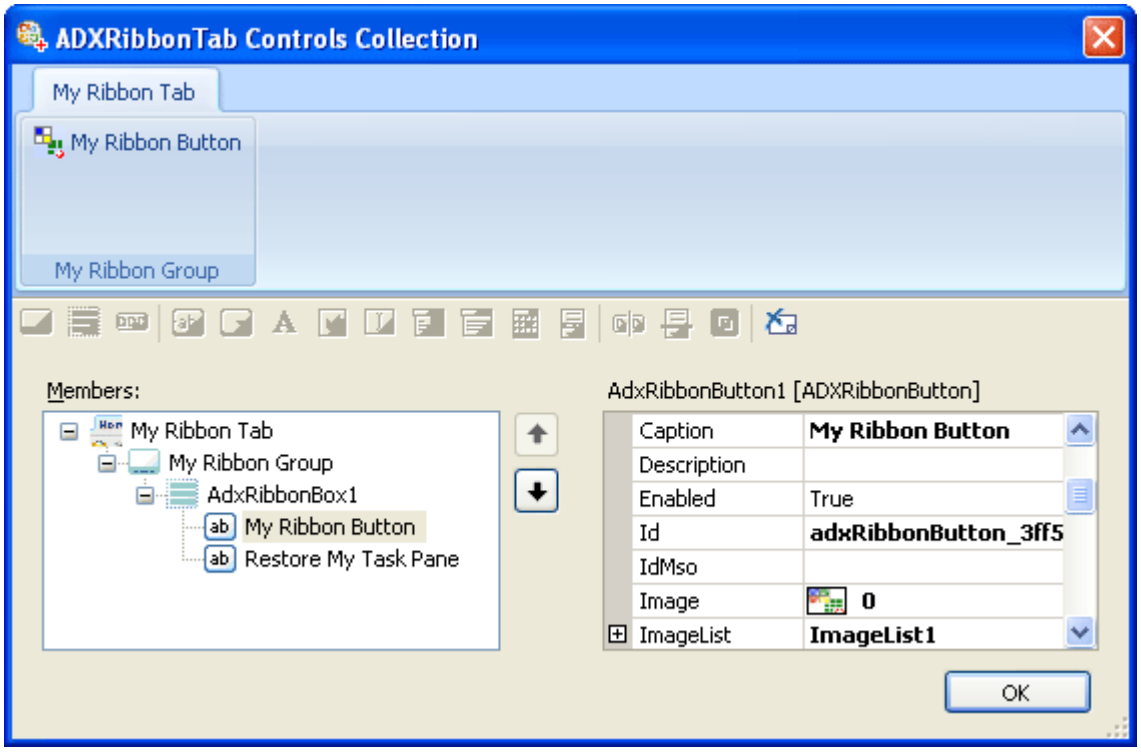
See also - How Do I Find the Source Code of This Sample?.

## 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 add a Ribbon group, and change its caption to My Ribbon Group. Next, you add a button group. Finally, you add a button. Set the button caption to My Ribbon Button. Use the ImageList and Image properties to set the image 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.MSO.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.

In the source code of the sample add-in described here, you can find how you can customize the Office 2007 menu. See How Do I Find the Source Code of This Sample?.

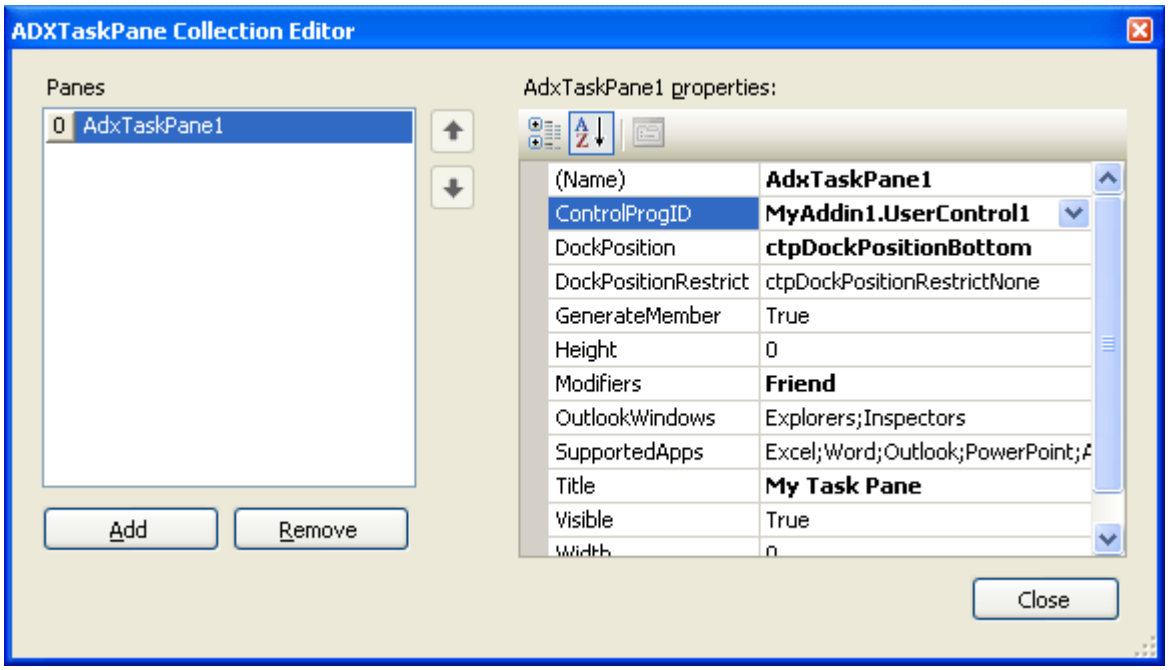
See also Office 2007 Ribbon Components.

## Step #10 - Adding a Custom Task Pane in Office 2007

To add a new task pane, you add a UserControl to your project and flourish it up with your controls. Then you add an item to the TaskPanes collection of the Add-in Module and specify its properties:

- Caption – the caption of your task pane
- DockPosition – you can dock your task pane to the left, top, right, or bottom edges of the host application window
- ControlProgID – the UserControl just added

The sample add-in described in this chapter has the following task pane settings:



Add-in Express you should understand the difference between the task pane component and task pane instances. The TaskPanes collection of the add-in module contains task pane components. When you set, say, the height or dock position of the component, these properties apply to every task pane instance that the host application shows. To modify the height of a task pane instance, you should get the instance itself. This can be done through the Item property of the component (in C# this property is the indexer for the ADXTaskPane class). The property accepts the parameter, which is the host application's window object that displays the task pane. For instance, the following private methods in the samples' add-in module find the currently active instance of the task pane and refresh it. For the task pane to be refreshed in a consistent manner, these methods are called in appropriate event handlers.

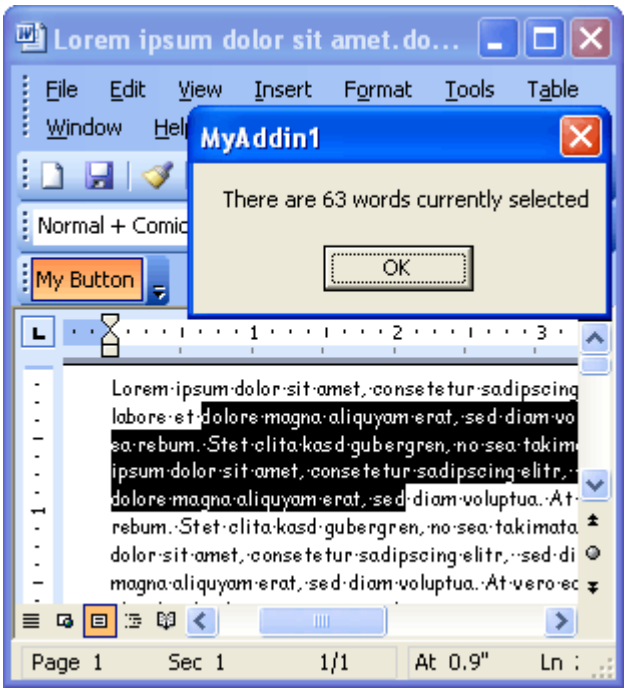
```
Imports AddinExpress.MSO
...
   Private Sub RefreshTaskPane()
      If Version = "12.0" Then
         Dim Window As Object = Me.HostApplication.ActiveWindow
         If Not Window Is Nothing Then
            RefreshTaskPane(AdxTaskPane1.Item(Window))
            Marshal.ReleaseComObject(Window)
         End If
      End If
   End Sub

   Private Sub RefreshTaskPane(ByVal TaskPaneInstance As _
      ADXTaskPane.ADXCustomTaskPaneInstance)
      If Not TaskPaneInstance Is Nothing Then
         Dim uc As UserControl1 = TaskPaneInstance.Control
         If uc IsNot Nothing And TaskPaneInstance.Window IsNot Nothing Then
            uc.InfoString = GetInfoString(TaskPaneInstance.Window)
         End If
      End If
   End Sub
```

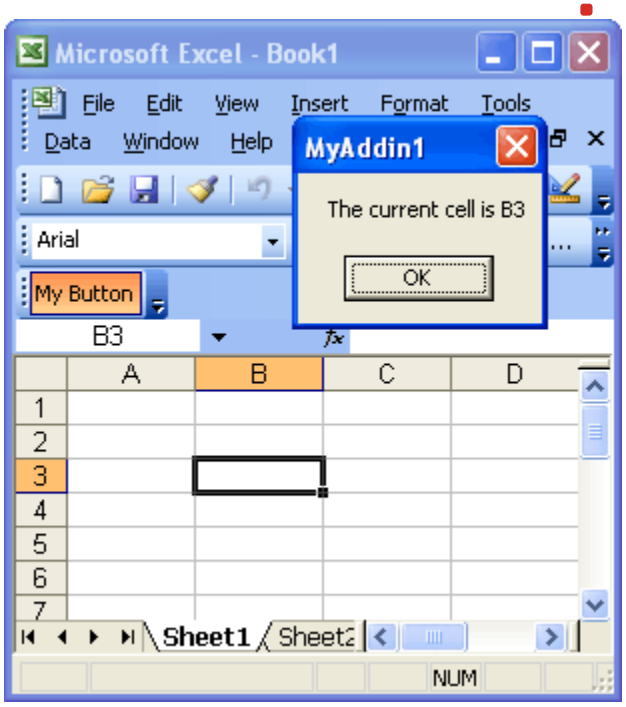
The InfoString property mentioned above just updates the text of the label located on the UserControl.

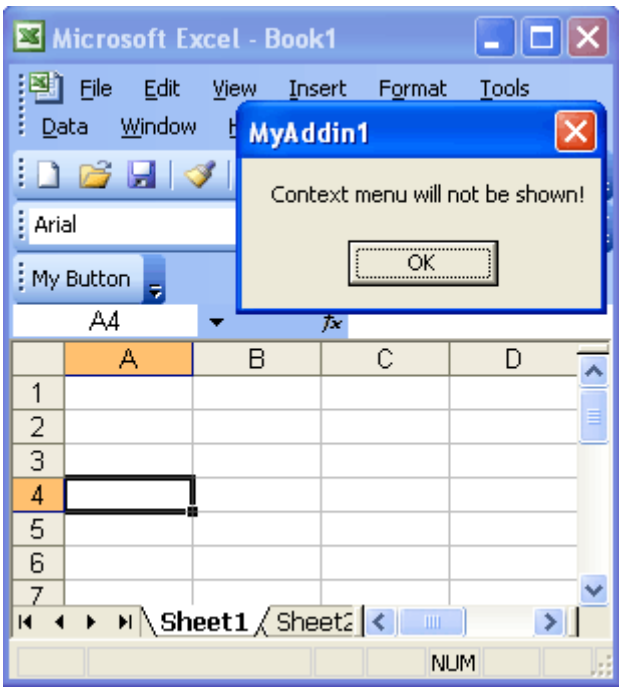
See also Office 2007 Custom Task Panes, How Do I Find the Source Code of This Sample?, Releasing COM objects

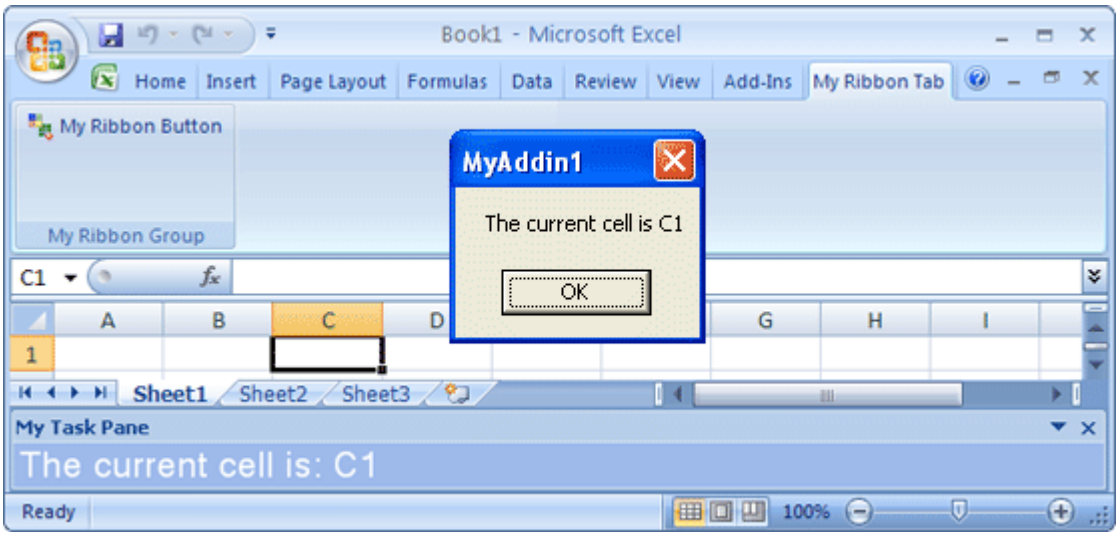
### Step #11 – Running the COM Add-in

Choose the Register Add-in Express Project item in the Build menu, restart the host application(s) you selected, find your toolbar and click the button. You find your add-in in the COM Add-ins dialog (see also Add the COM Add-ins Command to a Toolbar or Menu).


Lorem ipsum dolor sit amet.do...
File Edit View Insert Format Tools Table
Window Help
MyAddin1
There are 63 words currently selected
OK
Normal + Comic
My Button
Page 1 Sec 1 1/1 At 0.9" Ln :

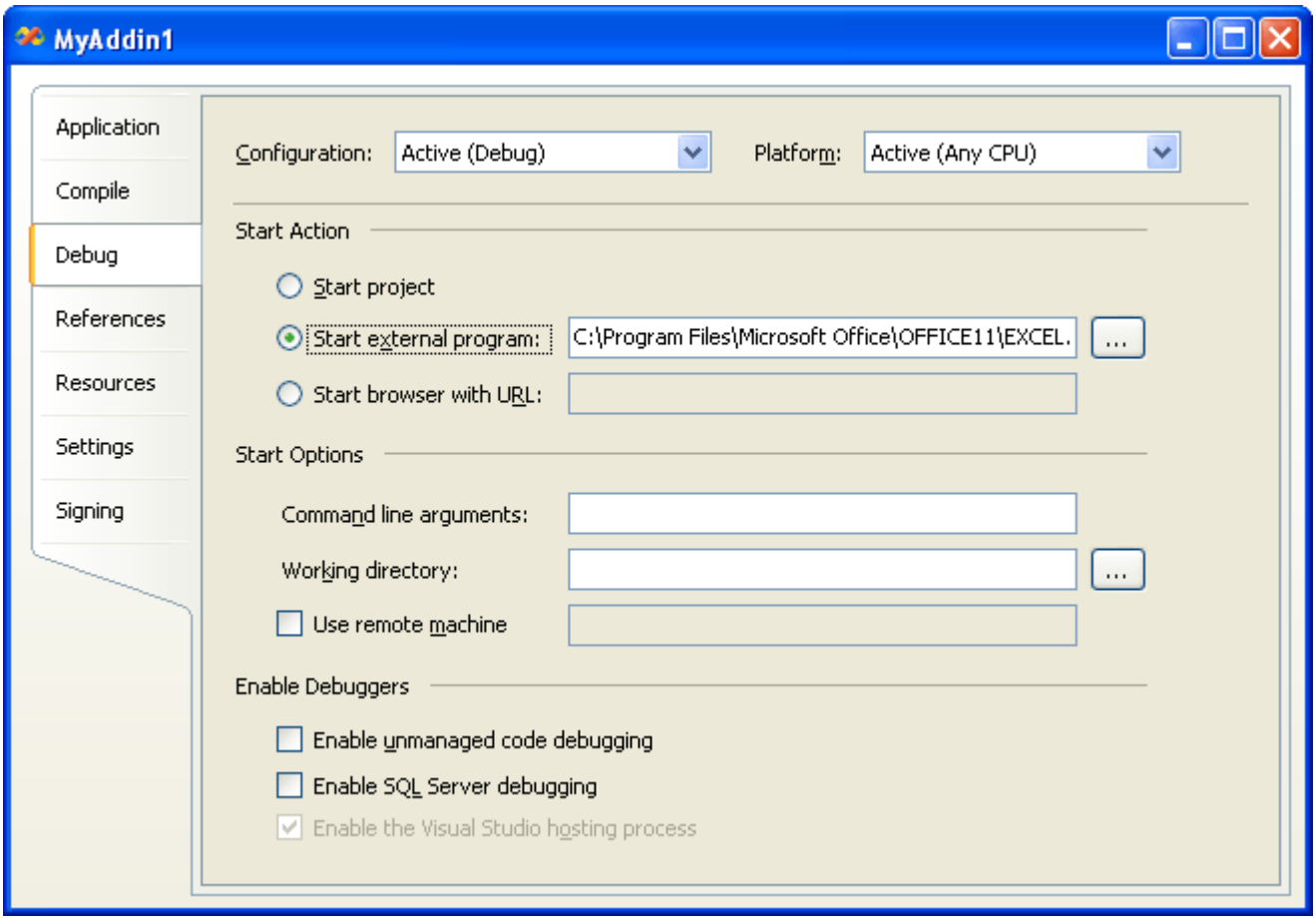

Microsoft Excel - Book1
File Edit View Insert Format Tools
Data Window Help
MyAddin1
The current cell is B3
OK
Arial
My Button
B3
A B C D
1 2 3 4 5 6 7
Sheet1 Sheet2
NUM


Microsoft Excel - Book1
File Edit View Insert Format Tools
Data Window
MyAddin1
Context menu will not be shown!
OK
Arial
My Button
A4
A B C D
1 2 3 4 5 6 7
Sheet1 Sheet2
NUM




## Step #12 – Debugging the COM Add-in

To debug your add-in, in the Project Options window, indicate the add-in host application in Start External Program and run the project.



However, there is a problem here. When debugging an add-in or a smart tag on Visual Studio 2003 and Office XP you cannot see your add-in or smart tag on the COM add-ins or AutoCorrect dialog box. This is a “feature” of Office XP "added" by MS people.

## Step #13 – Deploying the COM Add-in

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the add-in. See also Deploying Add-in Express Projects and Deploying Office Add-ins.

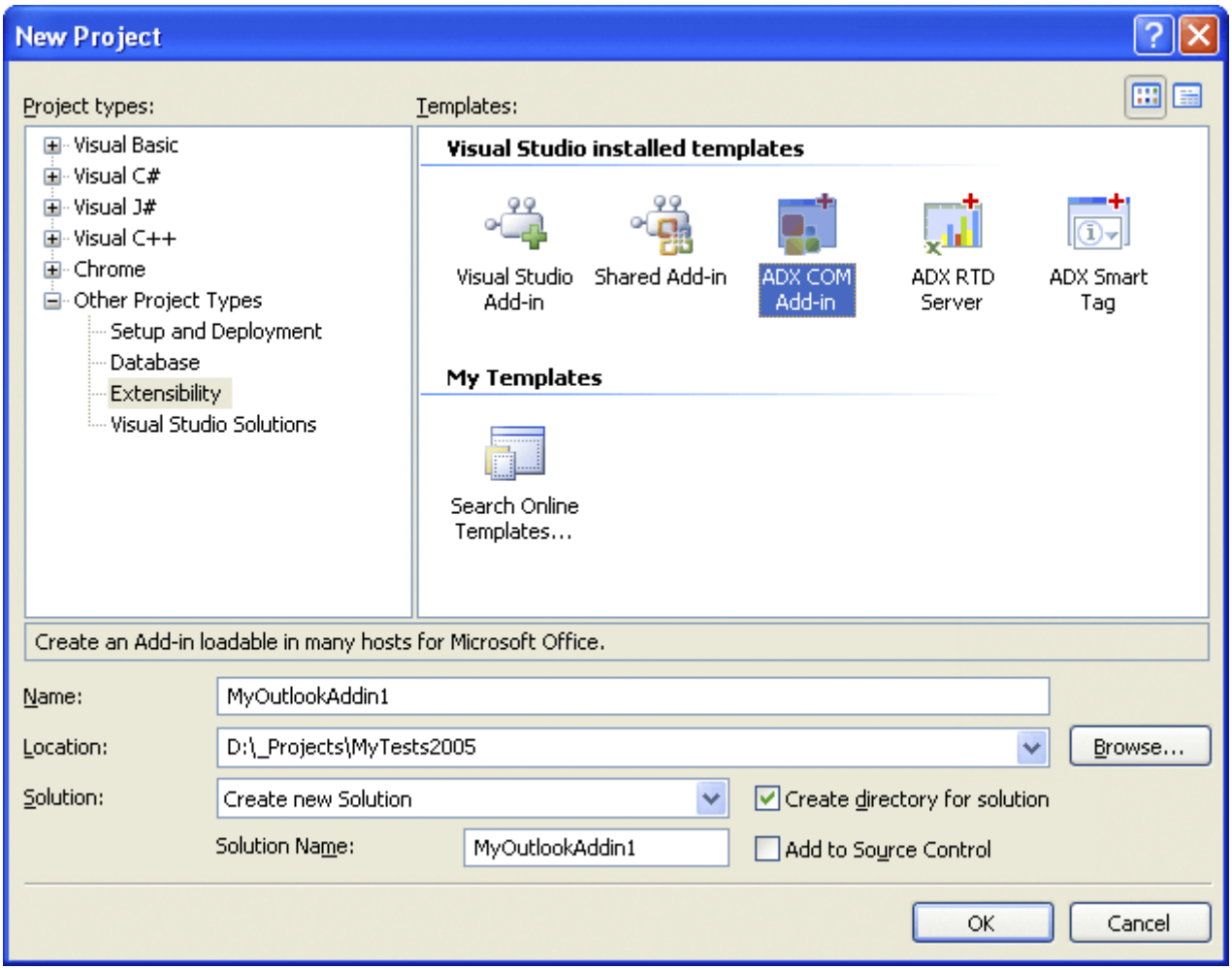
# Your First Microsoft Outlook COM Add-in

Add-in Express provides two Outlook specific command bars: ADXOlExplorerCommandBar and ADXOlInspectorCommandBar. 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 ADXOlExplorerCommandBar and ADXOlInspectorCommandBar 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.

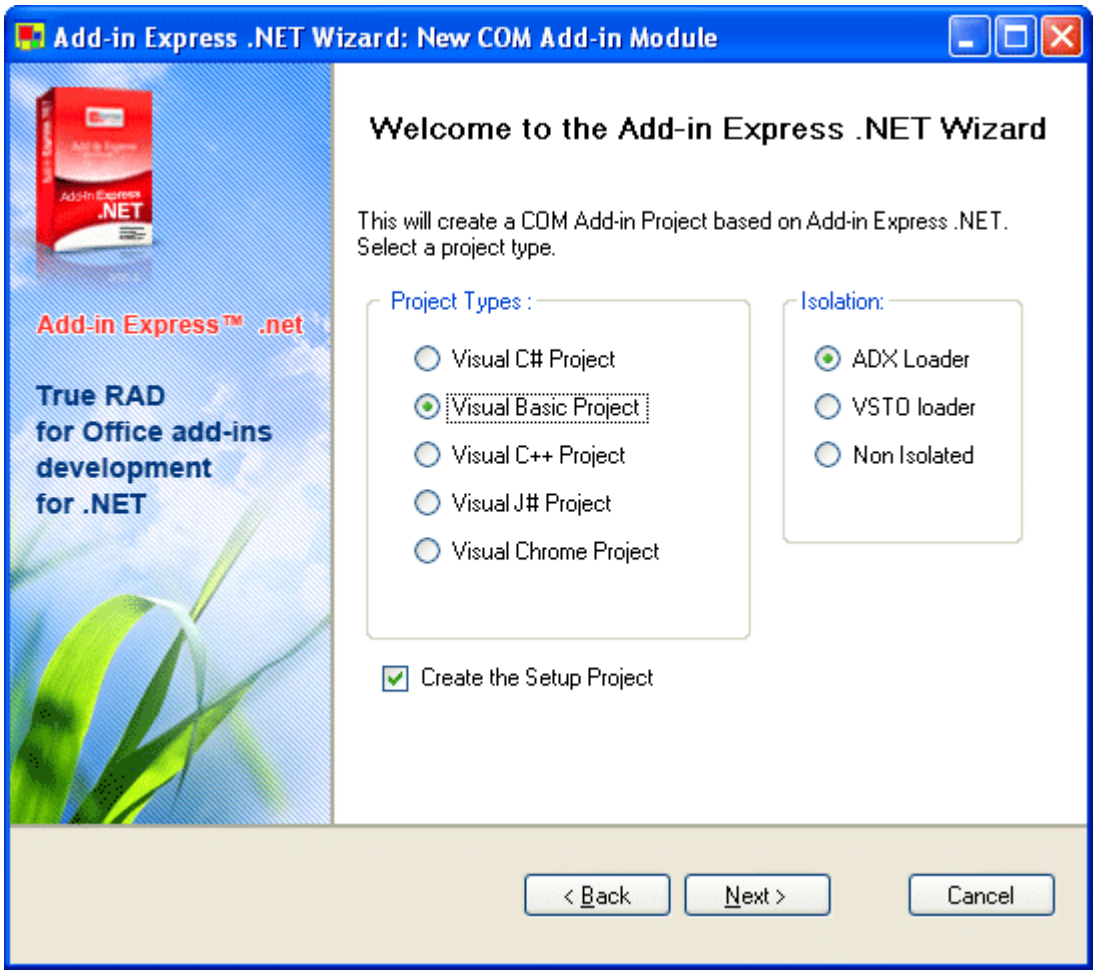
Additionally, Add-in Express supplies the Outlook Property Page component that helps you to add your pages to the Options (Tools | Options menu) and Folder Properties dialogs.

## Step #1 – Creating an Add-in Express COM Add-in Project

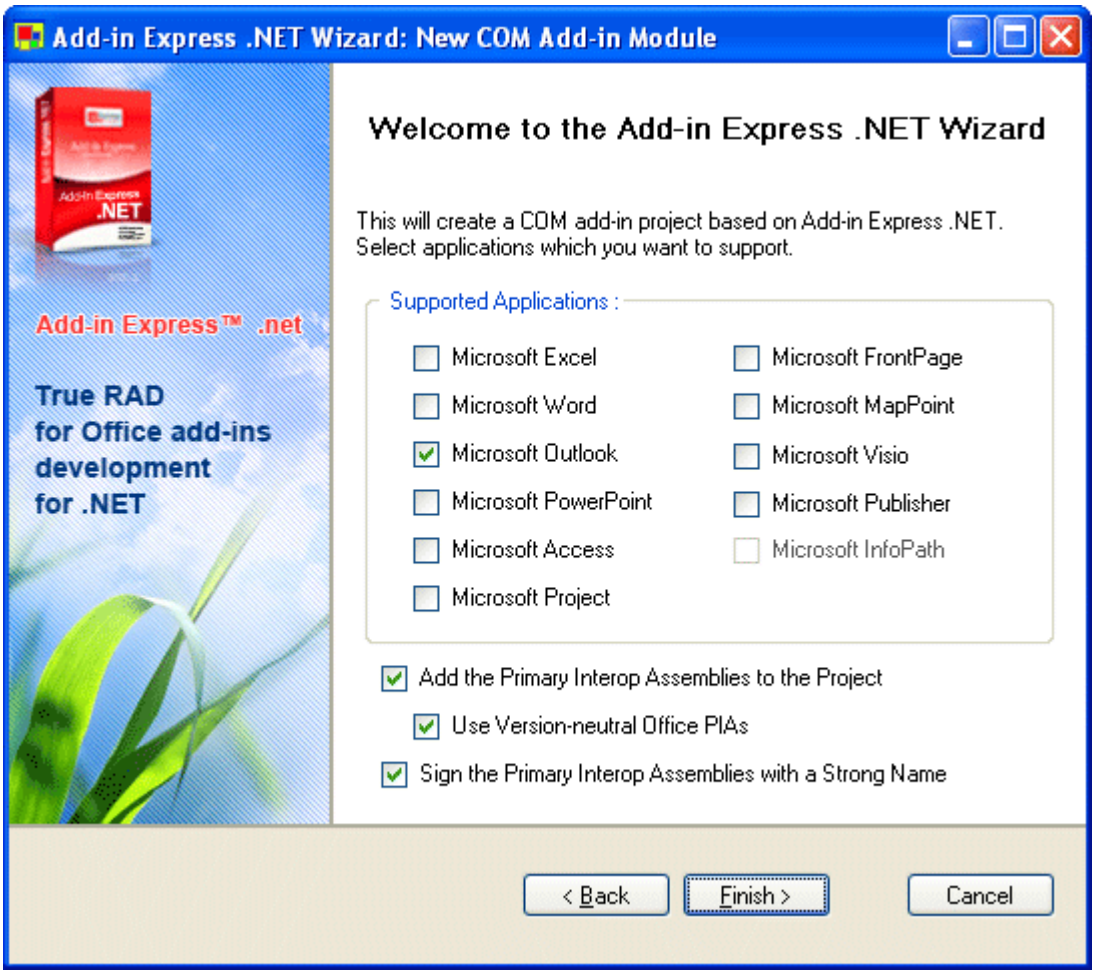
Add-in Express adds the Add-in Express COM Add-in project template to the Visual Studio IDE.



When you select the template and click OK, the Add-in Express COM Add-in project wizard starts. In the wizard windows, you choose the programming language, setup project options, applications supported by your add-in (as you may guess it is Outlook), and PIAs options. See Project Wizard Options for more details.



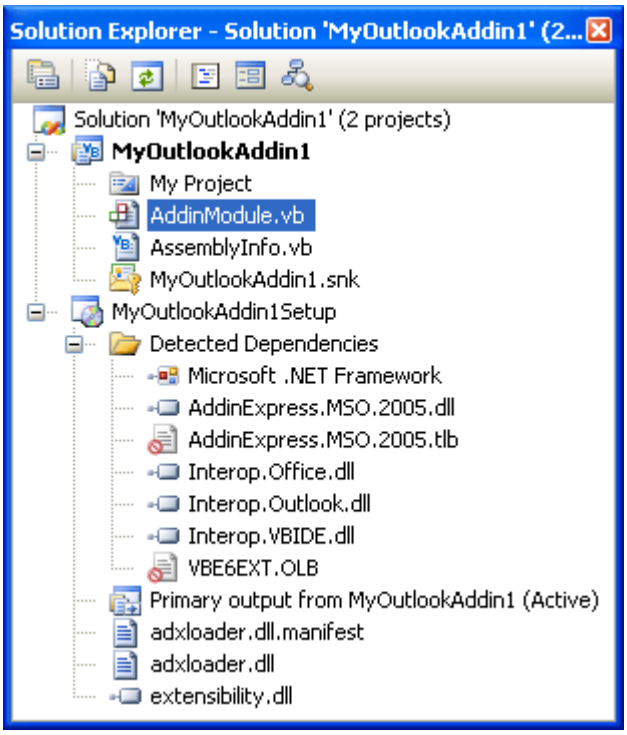

This VB.NET sample shows an Add-in Express COM Add-in project implementing an Outlook COM add-in with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.




The Add-in Express Project Wizard creates and opens the COM Add-in solution in IDE. The solution includes the COM Add-in project and the setup project.

### Your add-ins are version-neutral

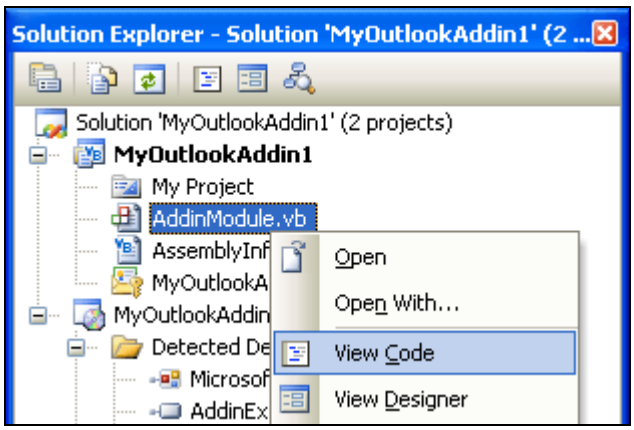
*Add-in Express delivers version-neutral Office interop assemblies that allow develop extensions for Office 2000+. To use them, simply check the Use Version-neutral Office PIAs check box when creating your projects. See also Why Version-Neutral PIAs?*




The COM Add-in project contains the AddinModule.vb (or AddinModule1.cs) file discussed in the next step.

## Step #2 – Add-in Express COM Add-in Module

The AddinModule.vb (or AddinModule1.cs) is a COM Add-in Module that is the core part of the COM add-in project (see COM Add-ins). 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 Solution Explorer, right-click the AddinModule1.vb (or AddinModule1.cs) file and choose the View Code popup menu item.



The code for AddinModule1.vb is as follows:

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express Add-in Module
<GuidAttribute("3BDF26A5-74E4-42CB-A93A-E88435BC0AD3"), _
   ProgIdAttribute("MyOutlookAddin1.AddinModule")> _
Public Class AddinModule
   Inherits AddinExpress.MSO.ADXAddinModule


#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()
      Me.components = New System.ComponentModel.Container
      '
      'AddinModule
      '
      Me.AddinName = "MyOutlookAddin1"
      Me.SupportedApps = AddinExpress.MSO.ADXOfficeHostApp.ohaOutlook

   End Sub

#End Region

#Region " Add-in Express 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

   <ComRegisterFunctionAttribute()> _
   Public Shared Sub AddinRegister(ByVal t As Type)
      AddinExpress.MSO.ADXAddinModule.ADXRegister(t)
```

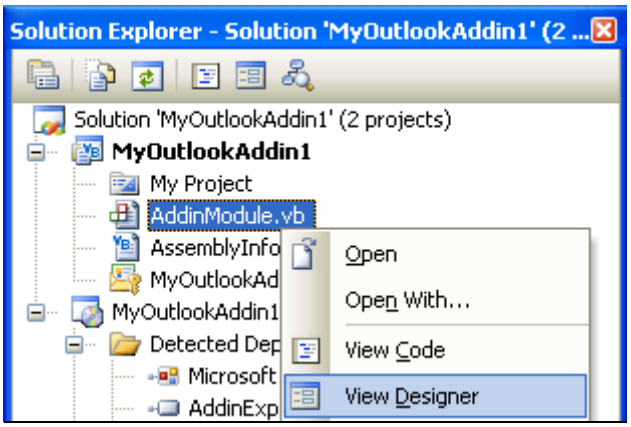
```
    End Sub

    <ComUnregisterFunctionAttribute()> _
    Public Shared Sub AddinUnregister(ByVal t As Type)
        AddinExpress.MSO.ADXAddinModule.ADXUnregister(t)
    End Sub

    Public Overrides Sub UninstallControls()
        MyBase.UninstallControls()
    End Sub

#End Region

    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 CType(HostApplication, Outlook._Application)
        End Get
    End Property

End Class
```

Please, pay attention to the OutlookApp property of the module generated by the Add-in Express Project Wizard. You can use it in your code to get access to Outlook objects.

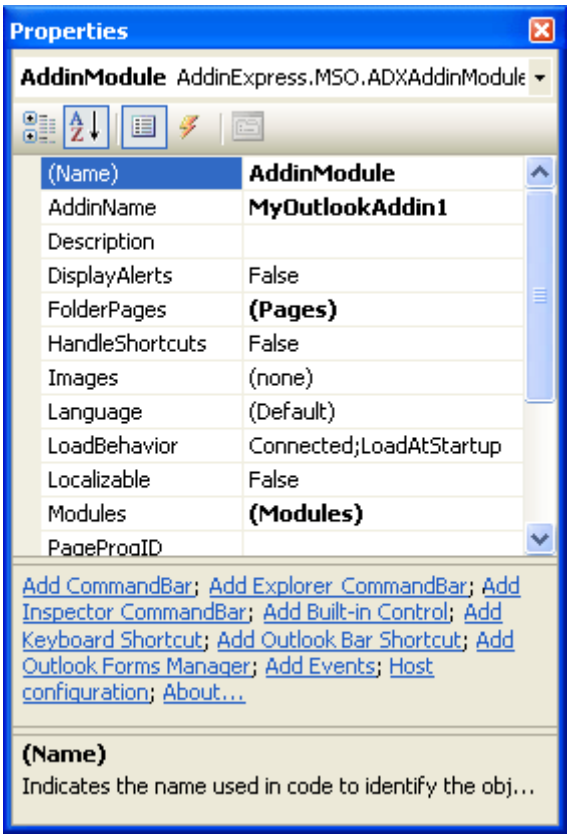
### Step #3 – Add-in Express COM Add-in Designer

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

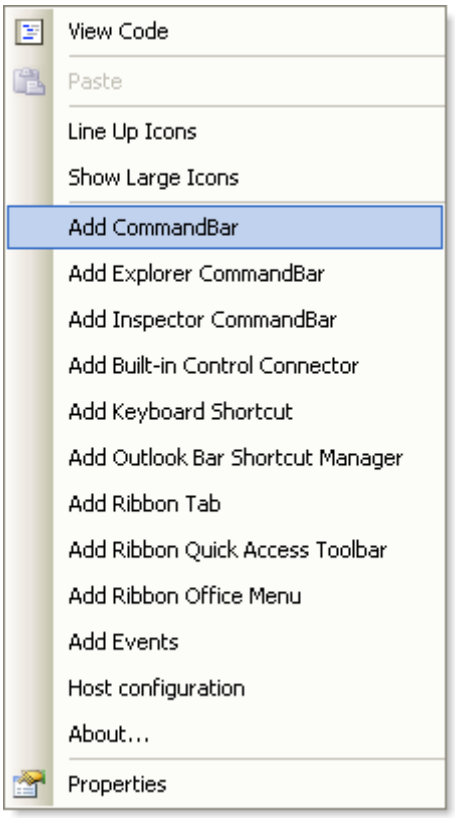
In Solution Explorer, right-click the AddinModule.vb (or AddinModule.cs) file and choose the View Designer popup menu item.




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




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.



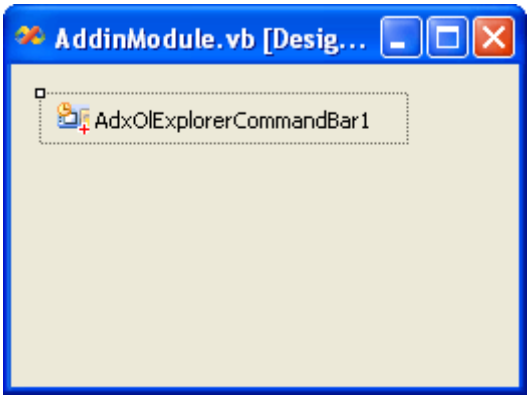

The following commands add the following components to the module:

- 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 Advanced Form Regions in Outlook 2000-200)
- 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)
- Add Events – adds or deletes components that provide access to application-level events of the add-in host applications (see Application-level Events)

## 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 COM 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 provides context-sensitive properties. They are FolderName, FolderNames, and ItemTypes (see Outlook CommandBar Visibility Rules and COM Add-ins for Outlook – Template Characters in FolderName).

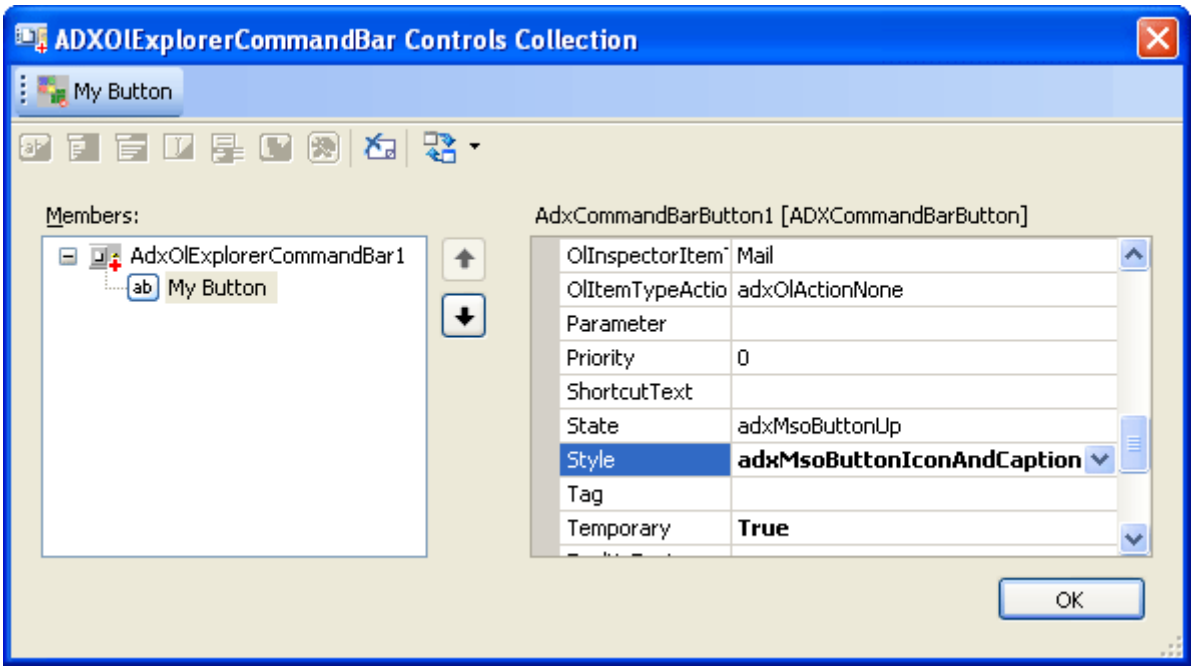

In the screenshot, you see the properties of the Outlook Explorer command bar that will be shown for every Outlook folder (FolderName = "*") the default item types of which are Mail or Task.

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).

This runs the visual designer. Use its toolbar to add or remove Command Bar Controls. Just click the appropriate button and see the result.




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, 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 OutlookApp property in the Add-in Module:

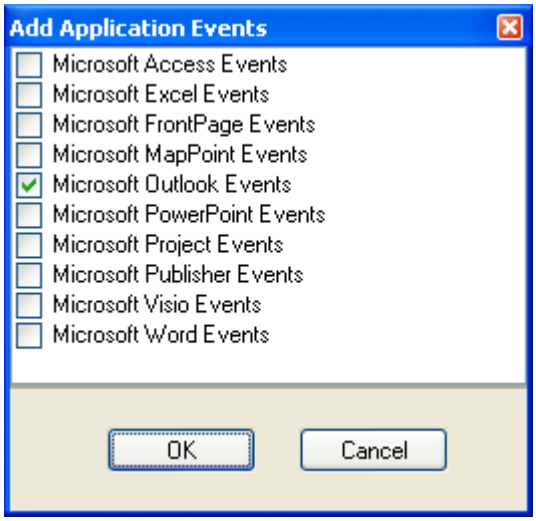
```
Public ReadOnly Property OutlookApp() As Outlook._Application
   Get
      Return CType(HostApplication, Outlook._Application)
   End Get
End Property
```

This allows you to write the following code to the Click event of the button just added:

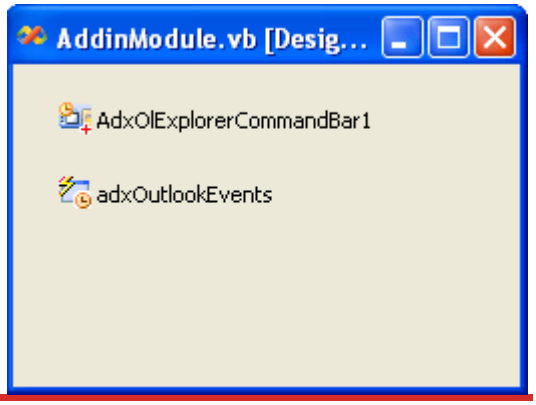
```
    Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
      Handles AdxCommandBarButton1.Click
       Dim explorer As Outlook.Explorer = Nothing
       Dim selection As Outlook.Selection = Nothing
       Dim mailItem As Outlook.MailItem = Nothing
       Try
          explorer = Me.OutlookApp.ActiveExplorer
          selection = explorer.Selection
          mailItem = selection.Item(1)
          MsgBox("The subject is:" _
             + vbCrLf _
             + mailItem.Subject)
       Catch
       Finally
          If explorer IsNot Nothing Then Marshal.ReleaseComObject(explorer)
          If mailItem IsNot Nothing Then Marshal.ReleaseComObject(mailItem)
          If selection IsNot Nothing Then Marshal.ReleaseComObject(selection)
       End Try
    End Sub
```

Note the use of Marshal.ReleaseComObject in the code above. See also Releasing COM objects and How Do I Find the Source Code of This Sample?

## Step #7 – Handling Outlook Events




The COM Add-in Designer provides the Add Events command that adds (and remove) event components that allow handling application-level events. When you choose this command, the Add Application Events dialog box opens allowing you to select the application Events components you need. This allows adding the Outlook Events component to the COM Add-in Module.

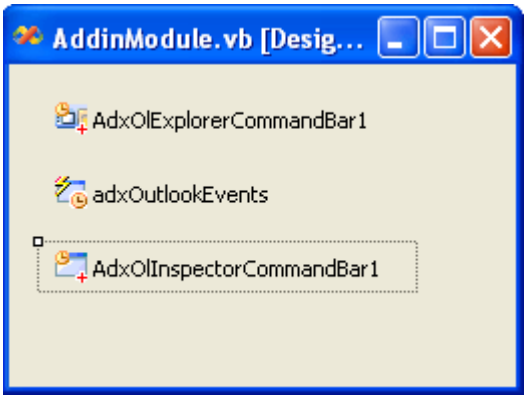


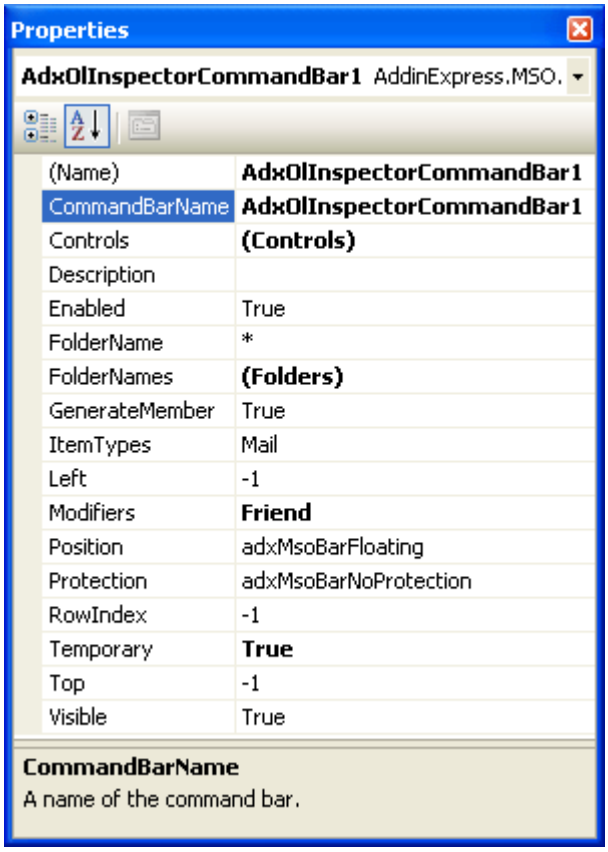

With the Outlook Events component, you handle any application-level events of Outlook. For instance, the following code handles the BeforeFolderSwitch event of the Outlook Explorer class:

```
Private Sub adxOutlookEvents_ExplorerBeforeFolderSwitch _
   (ByVal sender As Object, _
   ByVal e As _
       AddinExpress.MSO.ADXOlExplorerBeforeFolderSwitchEventArgs) _
   Handles adxOutlookEvents.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 COM Add-in Module.




The Inspector command bar component provides the same properties as the Explorer command bar component. In the screenshot, you see the default values for the Inspector command bar.

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 item using the following code that handles the Click event of the button just added:

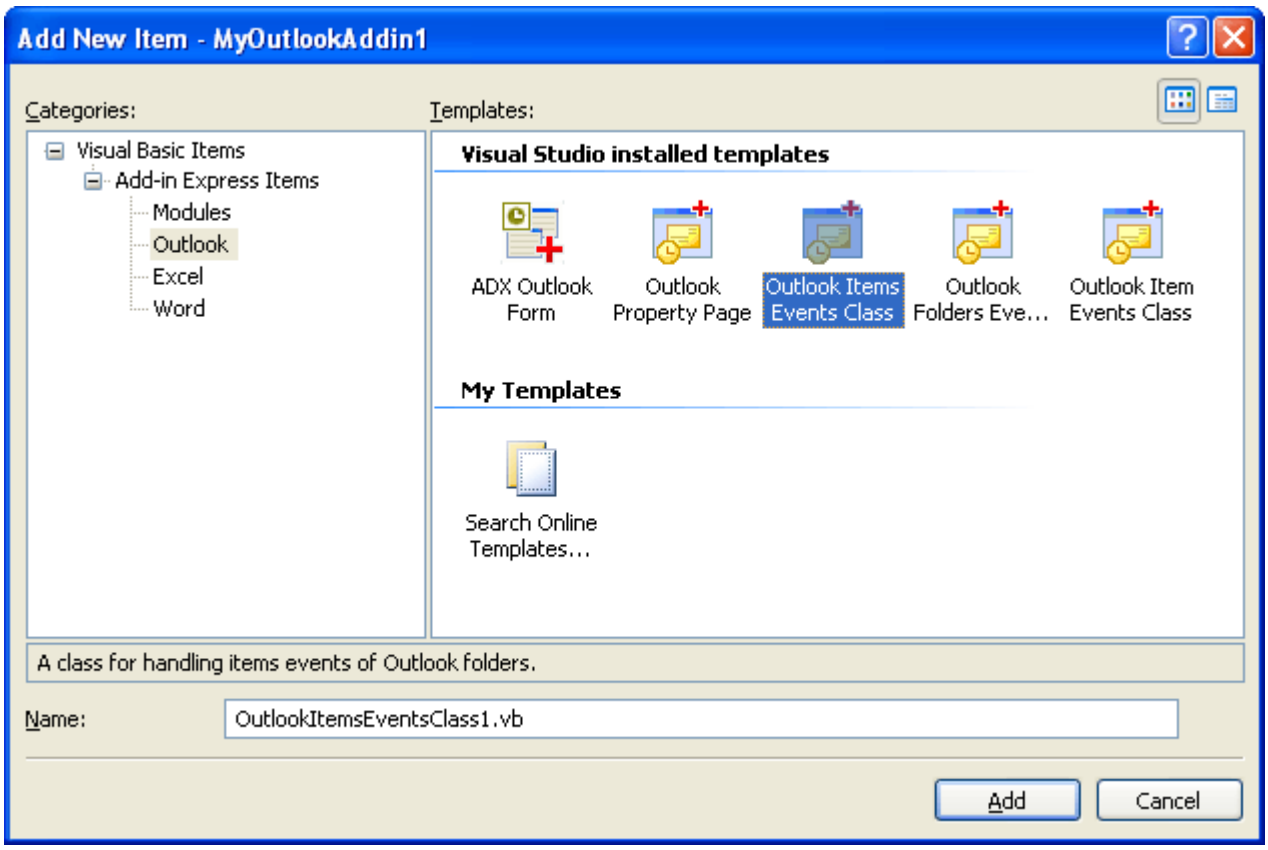
```
Private Sub AdxCommandBarButton2_Click(ByVal sender As System.Object) _
   Handles AdxCommandBarButton2.Click
   Dim inspector As Outlook.Inspector = Nothing
   Dim mailItem As Outlook.MailItem = Nothing
   Try
      inspector = Me.OutlookApp.ActiveInspector
```

```
            mailItem = inspector.CurrentItem
            MsgBox("The subject is:" _
               + vbCrLf _
               + mailItem.Subject)
         Catch
         Finally
            If inspector IsNot Nothing Then Marshal.ReleaseComObject(inspector)
            If mailItem IsNot Nothing Then Marshal.ReleaseComObject(mailItem)
         End Try
      End Sub
```

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

### 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:

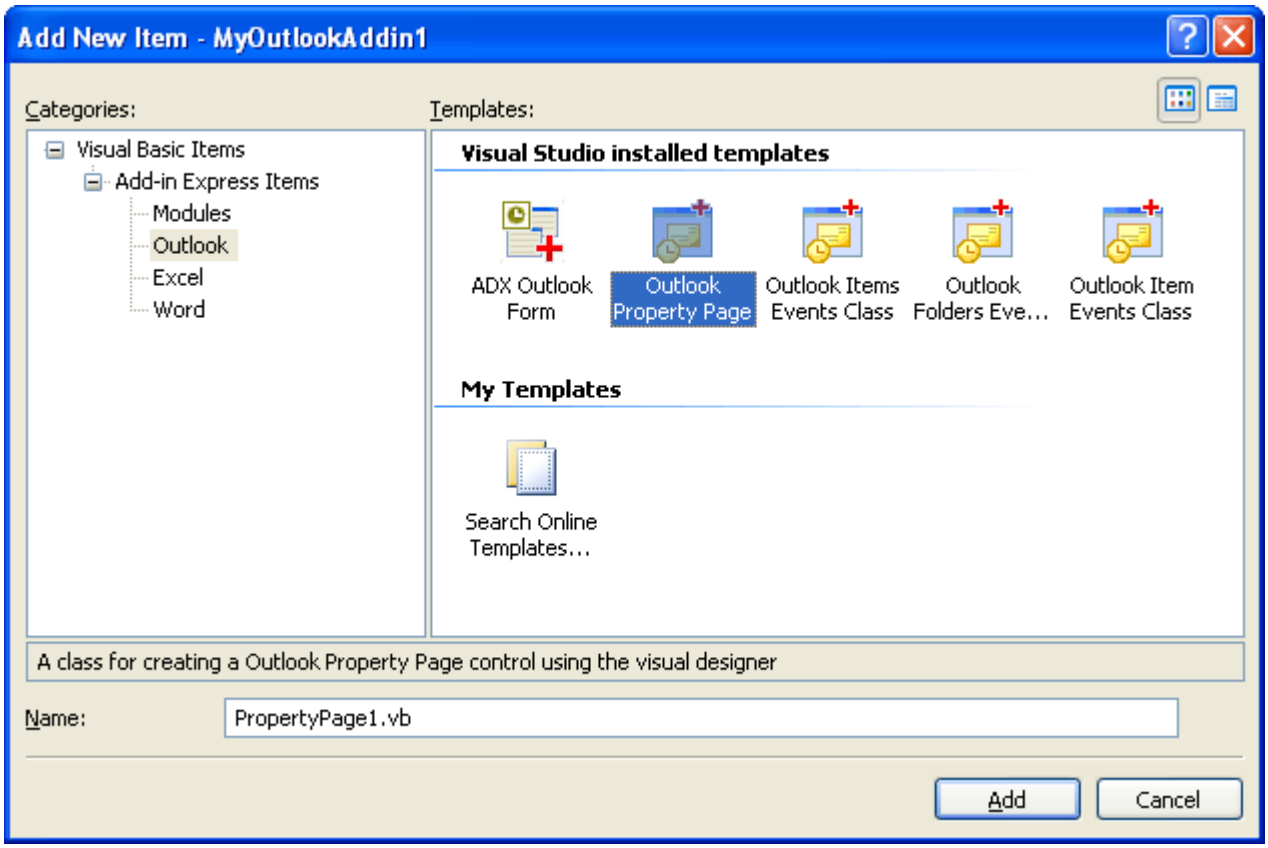
```
Imports System

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

   Public Sub New(ByVal ADXModule As AddinExpress.MSO.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 AddinModule_AddinBeginShutdown(ByVal sender As Object, _
      ByVal e As System.EventArgs) Handles Me.AddinBeginShutdown
      If ItemsEvents IsNot Nothing Then
         ItemsEvents.RemoveConnection()
         ItemsEvents = Nothing
      End If
   End Sub

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

## Step #10 – Adding Property Pages to the Folder Properties Dialog

Outlook allows you to add custom pages (tabs) 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 ADXOlPropertyPage 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:

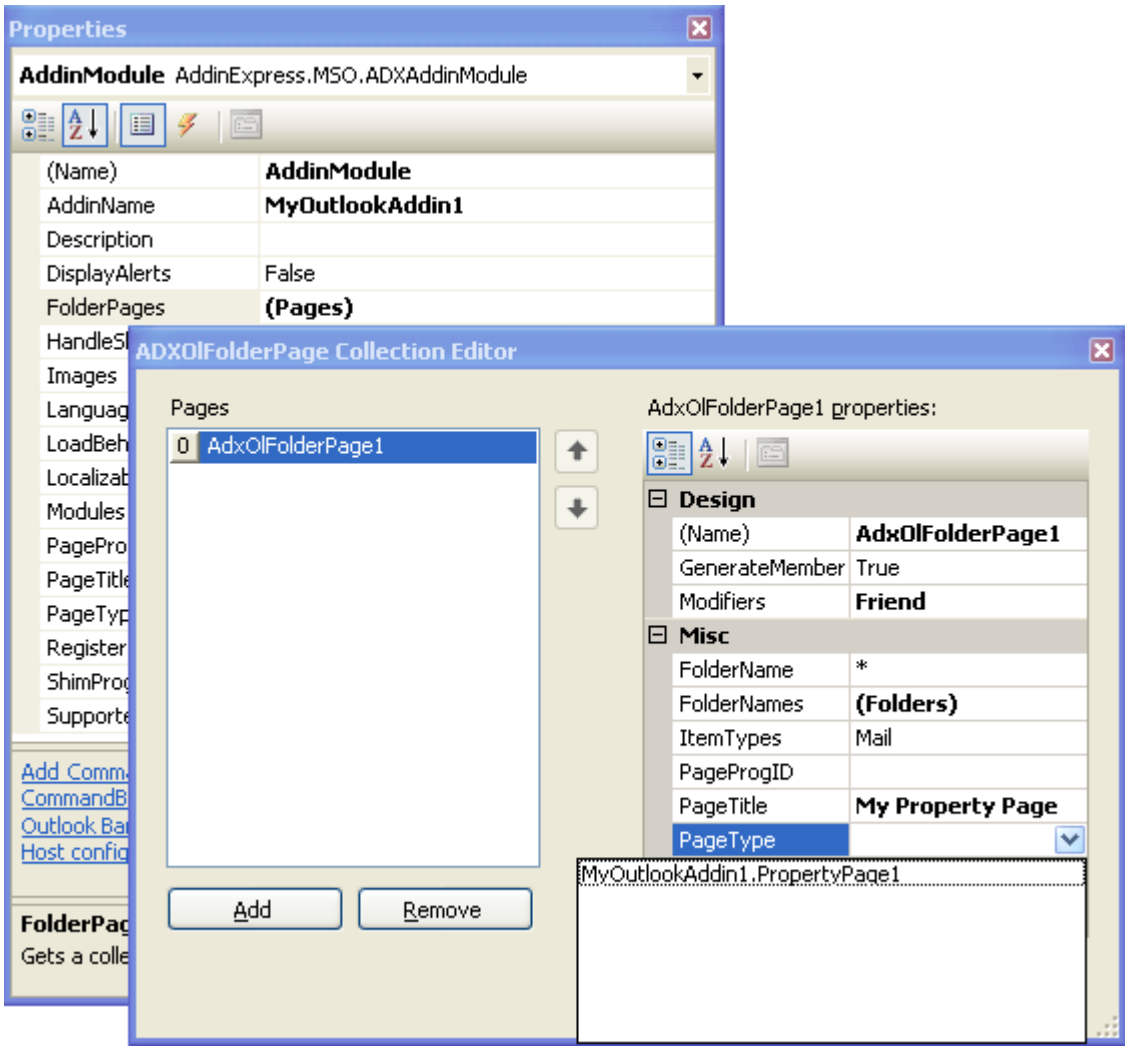
```
Imports System.Runtime.InteropServices

'Add-in Express Outlook Option Page
<GuidAttribute("D8F61EE3-B676-4FFA-9CAF-BAC46C1F0934"), _
    ProgIdAttribute("PropertyPage1.OutlookPage")> _
Public Class PropertyPage1
    Inherits AddinExpress.MSO.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()
        '
        'PropertyPage1
        '
        Me.Name = "PropertyPage1"
        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 (ItemTypes = Mail).

In order to limit the property page to the Inbox folder only, change the code of the AddinStartupComplete event in the AddinModule as follows:

```
    Private Sub AddinModule_AddinStartupComplete(ByVal sender As Object, _
            ByVal e As System.EventArgs) Handles Me.AddinStartupComplete
        ItemsEvents.ConnectTo( _
            AddinExpress.MSO.ADXOlDefaultFolders.olFolderInbox, True)
        Dim ns As Outlook.NameSpace = Me.OutlookApp.GetNamespace("Mapi")
        Dim folder As Outlook.MAPIFolder = _
            ns.GetDefaultFolder(Outlook.OlDefaultFolders.olFolderInbox)
        Me.FolderPages.Item(0).FolderName = GetFolderPath(folder)
        Marshal.ReleaseComObject(folder)
        Marshal.ReleaseComObject(ns)
    End Sub
```

See the code of the GetFolderPath function in FolderPath Property Value in Outlook 2000 and XP.

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 PropertyPage1_Dirty(ByVal sender As System.Object, _
       ByVal e As AddinExpress.MSO.ADXDirtyEventArgs) Handles MyBase.Dirty
       e.Dirty = True
    End Sub

    Private Sub PropertyPage1_Apply(ByVal sender As System.Object, _
       ByVal e As System.EventArgs) Handles MyBase.Apply
       CType(AddinModule.CurrentInstance, _
          MyOutlookAddin1.AddinModule).IsFolderTracked = _
          Me.CheckBox1.Checked
    End Sub

    Private Sub PropertyPage1_Load(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles Me.Load
       TrackStatusChanges = True
       Me.CheckBox1.Checked = _
          CType(AddinModule.CurrentInstance, _
             MyOutlookAddin1.AddinModule).IsFolderTracked
       TrackStatusChanges = False
    End Sub
End Class
```

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(ADXOlDefaultFolders.olFolderInbox, True)
        Else
            ItemsEvents.RemoveConnection()
        End If
    End Set
End Property
```

### 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.*

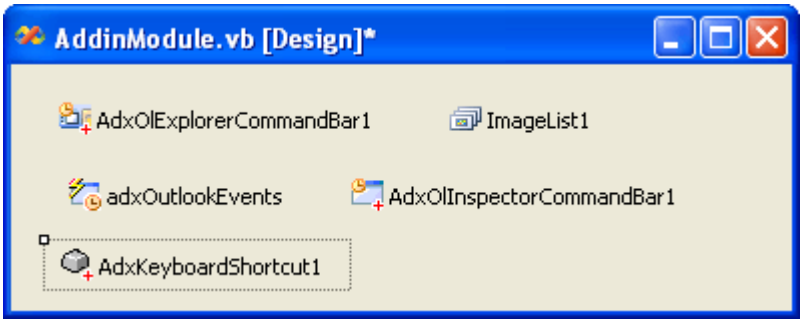
### PageProgId property is obsolete

*Note, whether you add a folder property page or an Outlook Options page, do not use the PageProgId property. It is obsolete; we left it for compatibility only.*

See also Outlook Property Page, How Do I Find the Source Code of This Sample?

## Step #11 - Intercepting Keyboard Shortcut

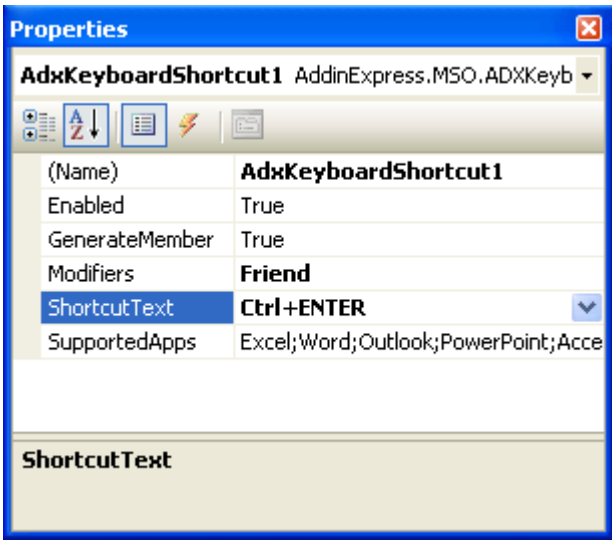
To intercept a keyboard shortcut, you add an ADXKeyboardShortcut component to the COM Add-in Module using the Add Keyboard Shortcut command of the module.




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.*

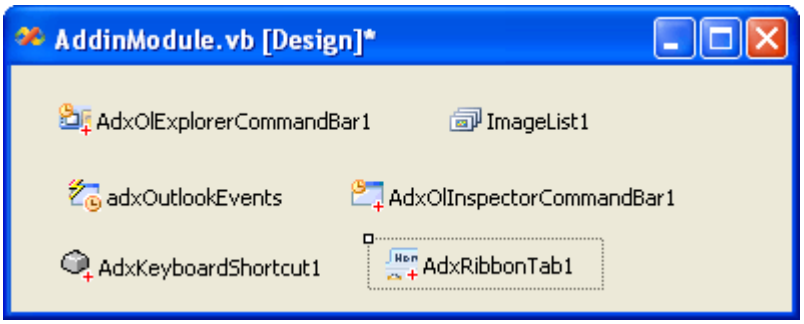



Now you handle the Action event of the component:

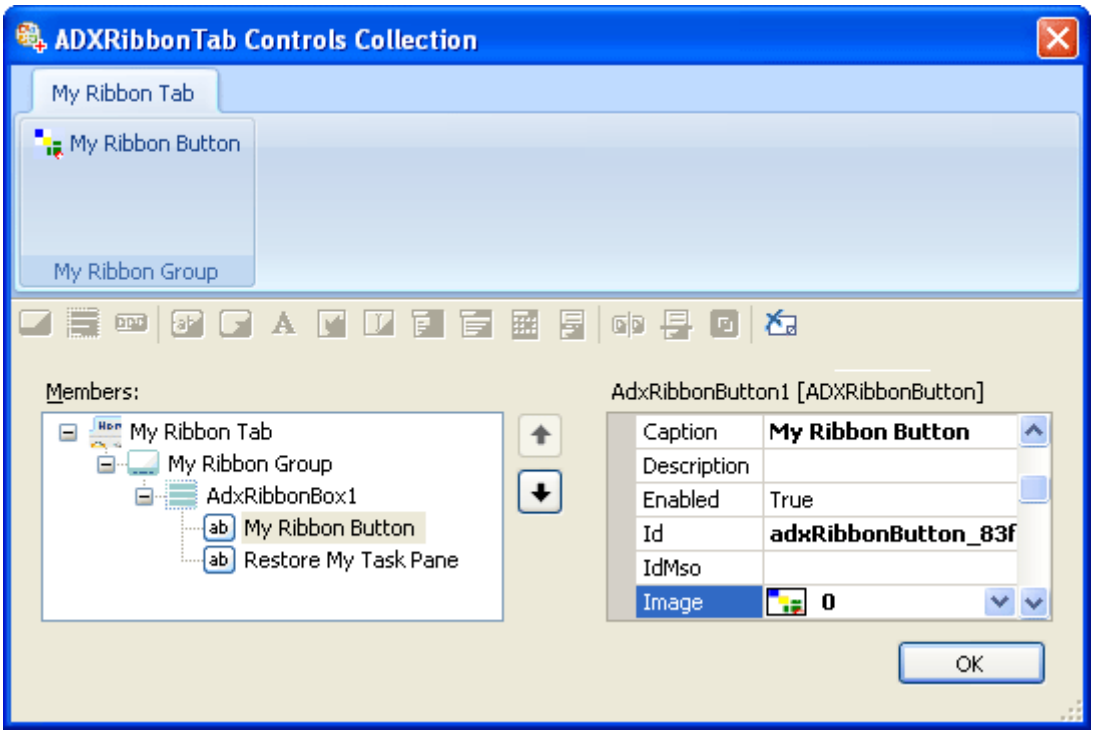
```
Private Sub AdxKeyboardShortcut1_Action(ByVal sender As System.Object) _
   Handles AdxKeyboardShortcut1.Action
   MsgBox("You've pressed " + _
      CType(sender, AddinExpress.MSO.ADXKeyboardShortcut).ShortcutText)
End Sub
```

## 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.MSO.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.

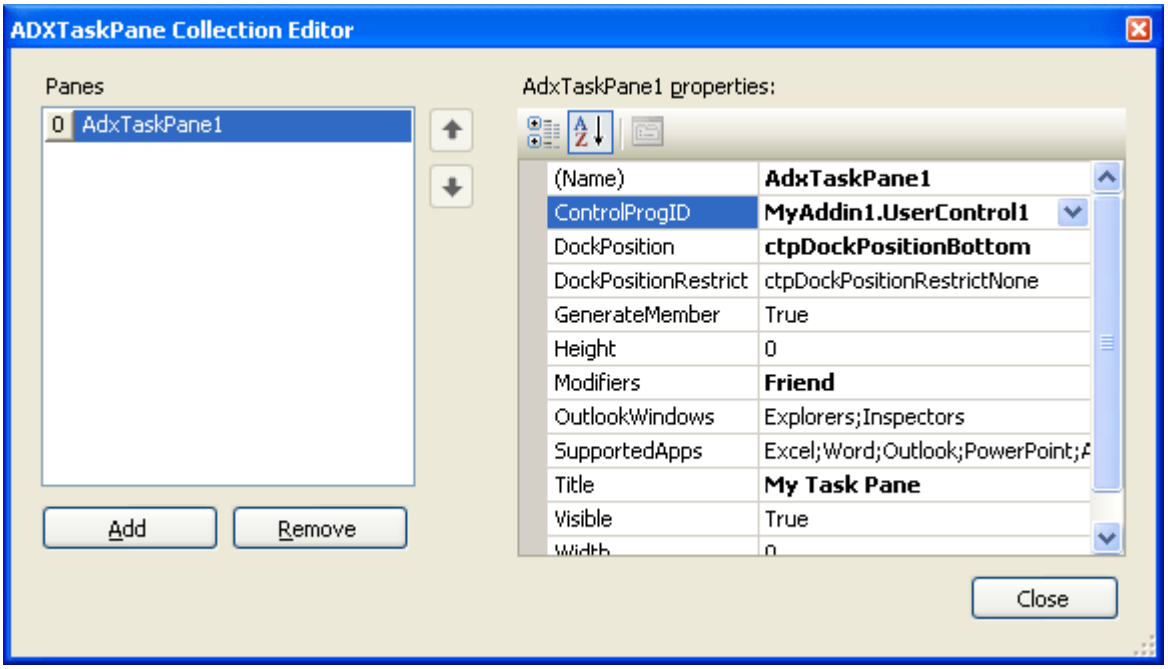
## Step #13 - Adding a Custom Task Pane in Outlook 2007

To add a new task pane, you add a UserControl to your project and add some controls. Then you add an item to the TaskPanes collection of the Add-in Module and specify its properties:

- Caption – the caption of your task pane (required!);

- Height, Width – the height and width of your task pane (applies to horizontal and vertical task panes, correspondingly);
- DockPosition – you can dock your task pane to the left, top, right, or bottom edges of the host application window;
- ControlProgID – the UserControl just added.

The sample add-in described in this chapter has the following task pane settings:




In Add-in Express, you work with the task pane component and task pane instances. The TaskPanes collection of the add-in module contains task pane components. When you set, say, the height or dock position of the component, these properties apply to every task pane instance that the host application shows. To modify a property of a task pane instance, you should get the instance itself. This can be done through the Item property of the component (in C# this property is the indexer for the ADXTaskPane class). The property accepts the Outlook Explorer or Inspector window object that displays the task pane as a parameter. For instance, the RefreshTaskPane method in the samples' add-in module finds the currently active instance of the task pane and refreshes it. For the task pane to be refreshed in a consistent manner, this method is called in appropriate event handlers.

```
Private Sub RefreshTaskPane(ByVal ExplorerOrInspector As Object)
   If Me.HostVersion.Substring(0, 4) = "12.0" Then
      Dim TaskPaneInstance As _
         AddinExpress.MSO.ADXTaskPane.ADXCustomTaskPaneInstance = _
            AdxTaskPane1.Item(ExplorerOrInspector)
      If Not TaskPaneInstance Is Nothing _
         And TaskPaneInstance.Visible Then
         Dim uc As UserControl1 = TaskPaneInstance.Control
```

```
            If Not uc Is Nothing Then _
              uc.InfoString = GetSubject(ExplorerOrInspector)
         End If
      End If
   End Sub
```

The InfoString property just gets or sets the text of the Label located on the UserControl1. The GetSubject method is shown below.

```
   Private Function GetSubject(ByVal ExplorerOrInspector As Object) _
      As String
      Dim mailItem As Outlook.MailItem = Nothing
      Dim selection As Outlook.Selection = Nothing

      If TypeOf ExplorerOrInspector Is Outlook.Explorer Then
         Try
            selection = CType(ExplorerOrInspector, _
               Outlook.Explorer).Selection
            mailItem = selection.Item(1)
         Catch
         Finally
            If Not selection Is Nothing Then _
               Marshal.ReleaseComObject(selection)
         End Try
      ElseIf TypeOf ExplorerOrInspector Is Outlook.Inspector Then
         Try
            mailItem = CType(ExplorerOrInspector, _
               Outlook.Inspector).CurrentItem
         Catch
         End Try
      End If

      If mailItem Is Nothing Then
         Return ""
      Else
         Dim subject As String = "The subject is: " + mailItem.Subject
         Marshal.ReleaseComObject(mailItem)
         Return subject
      End If
   End Function
```

The code of the GetSubject method emphasizes the following:

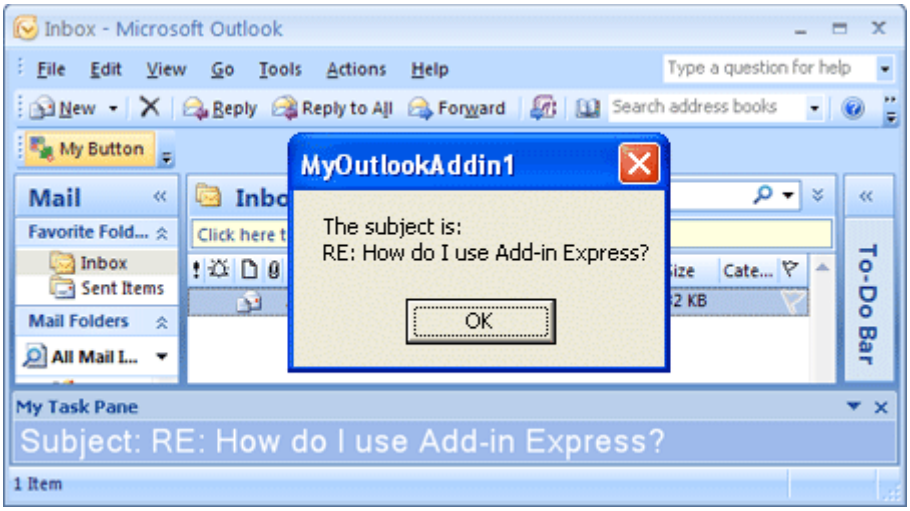
- The ExplorerOrInspector parameter was originally obtained through parameters of Add-in Express event handlers. That's why we don't release it (see Releasing COM objects).
- The selection and mailItem variables were obtained "manually" so they must be released.
- Outlook 2007 fires an exception when you try to obtain the Selection object in some situations.

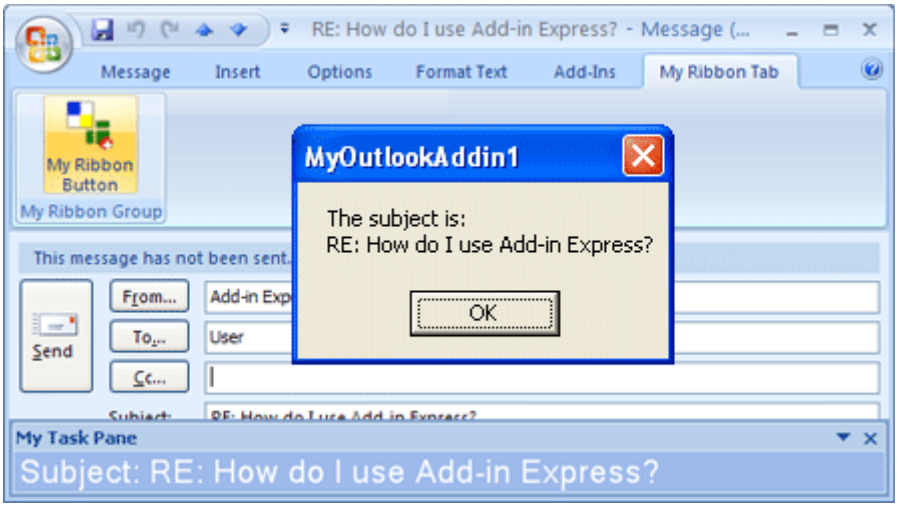
- There can be no items in the Selection object.

You may also want to look into the event handler for the VisibleStateChange event of task pane component in the code of the sample add-in described in this chapter. It works with the CustomTaskPane interface defined in Office 2007. See also - How Do I Find the Source Code of This Sample?

## Step #14 - Running the COM Add-in

Choose the Register Add-in Express Project item in the Build menu, then restart Outlook and find your option page(s), command bars, and controls, Ribbon controls, and custom task panes.

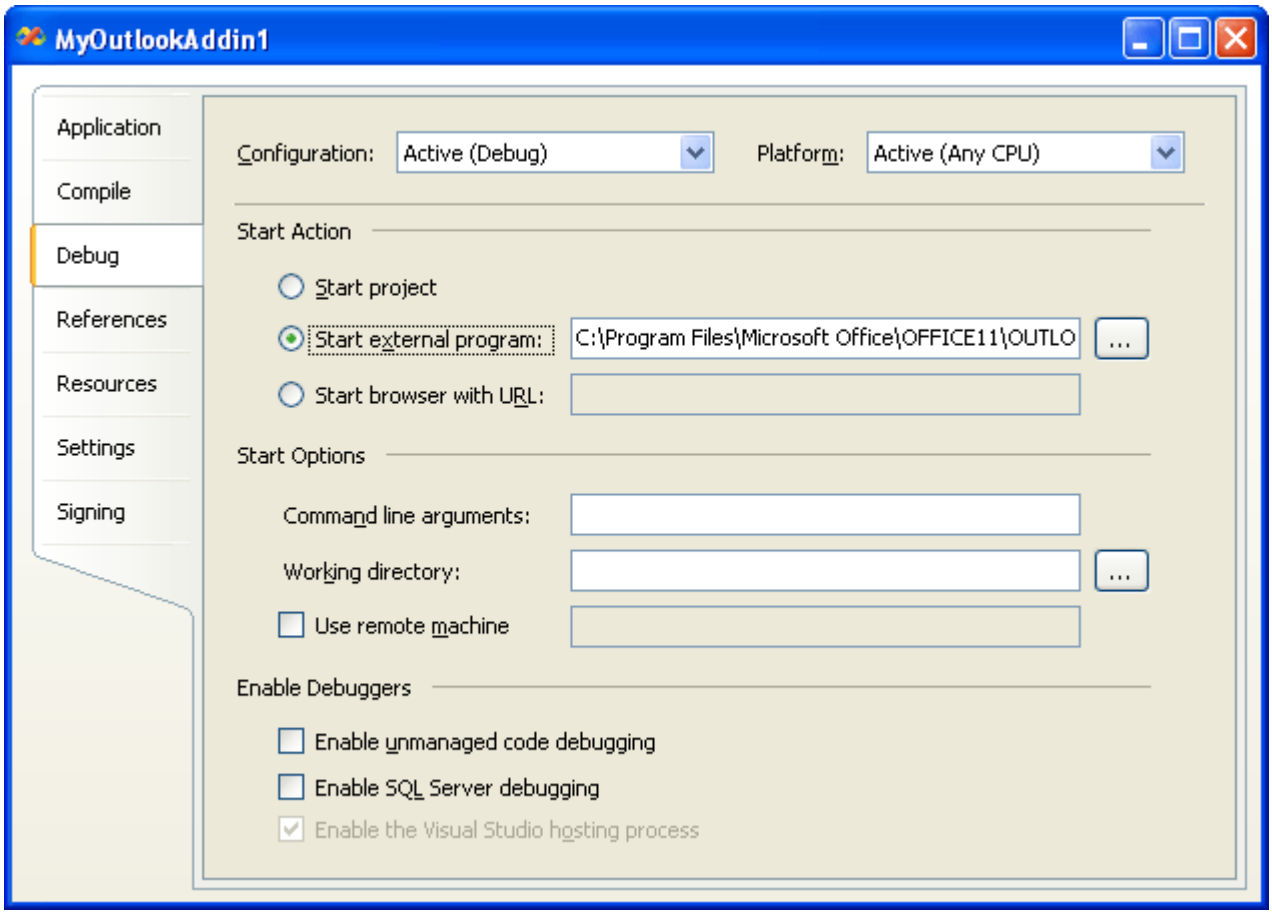



You can find your add-in in the COM Add-ins dialog (see also Add the COM Add-ins Command to a Toolbar or Menu).

## Step #15 – Debugging the COM Add-in

To debug your add-in, just indicate the add-in host application as the Start Program in the Project Options window.




However, there is a problem here. When debugging an add-in or a smart tag on Visual Studio 2003 and Office XP you cannot see your add-in or smart tag on the COM add-ins or AutoCorrect dialog box. This is a “feature” of Office XP "added" by Microsoft.

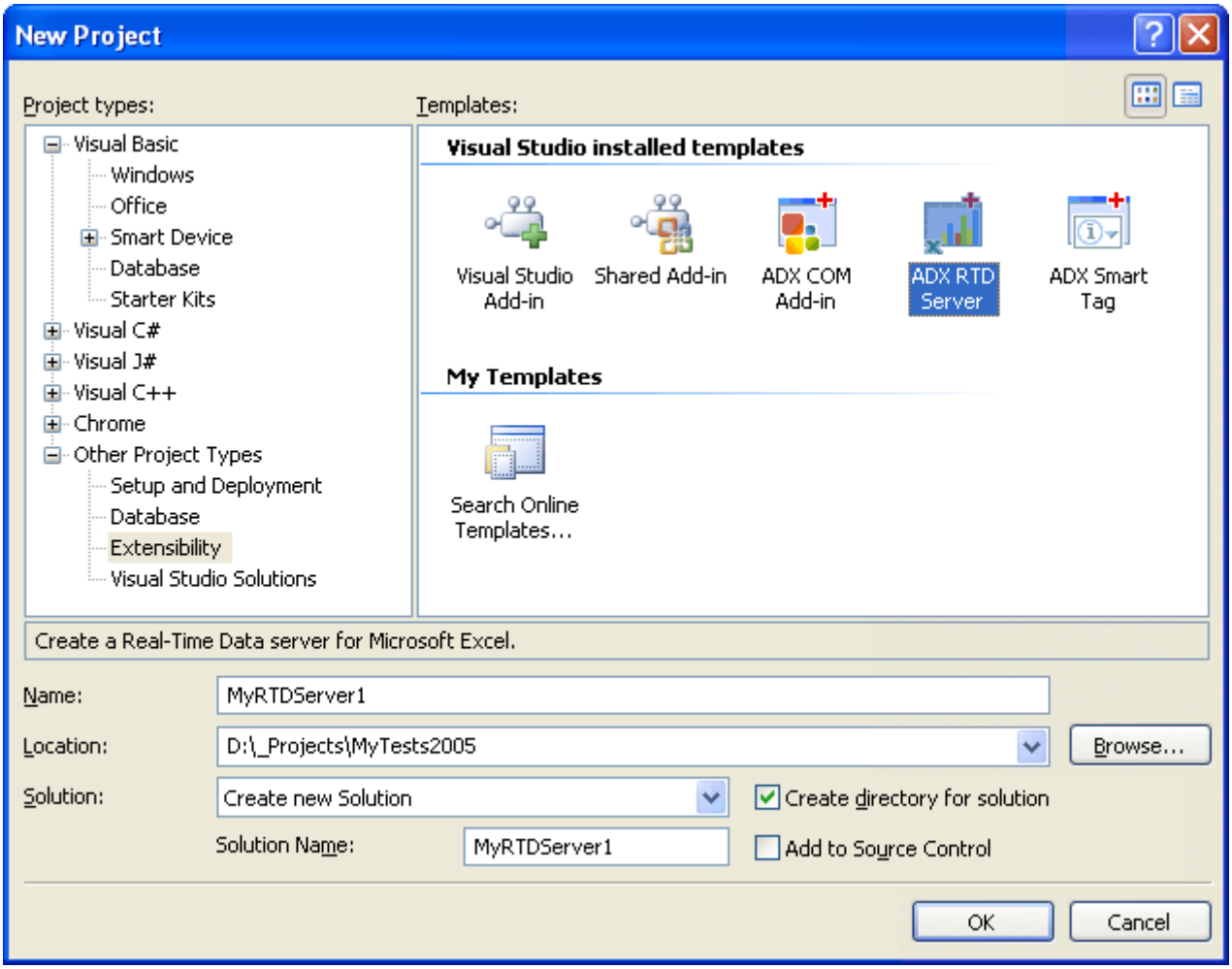
## Step #16 – Deploying the COM Add-in

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the add-in. See also Deploying Add-in Express Projects and Deploying Office Add-ins.

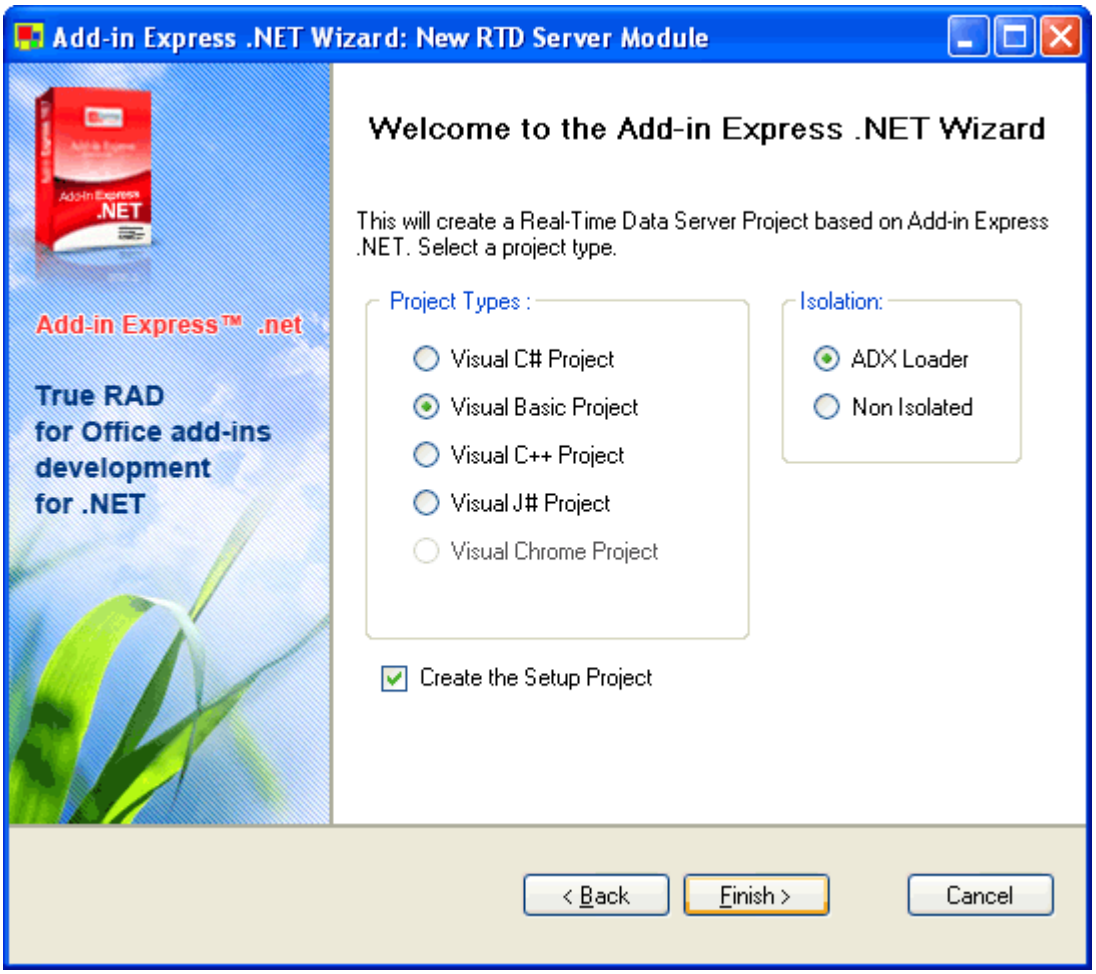
# Your First Excel RTD Server

## Step #1 – Creating a New Add-in Express RTD Server Project

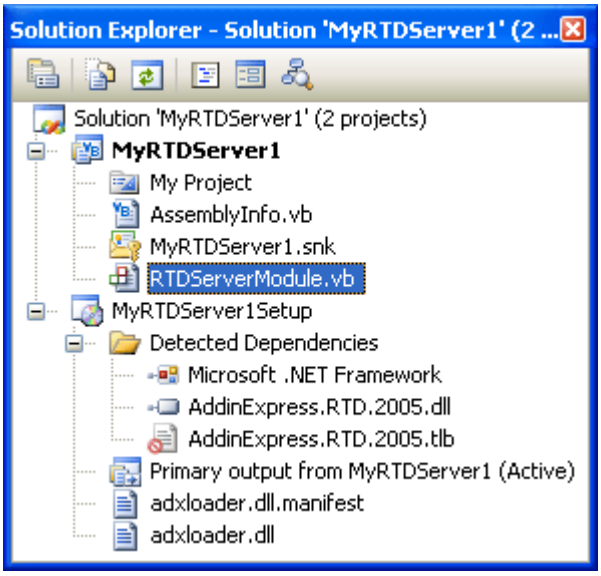
Add-in Express adds the Add-in Express RTD Server project template to the Visual Studio IDE.




When you select the template and click OK, the Add-in Express RTD Server project wizard starts. In the wizard window, you choose the programming language and setup project options.




This VB.NET sample shows an Add-in Express RTD Server project with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.

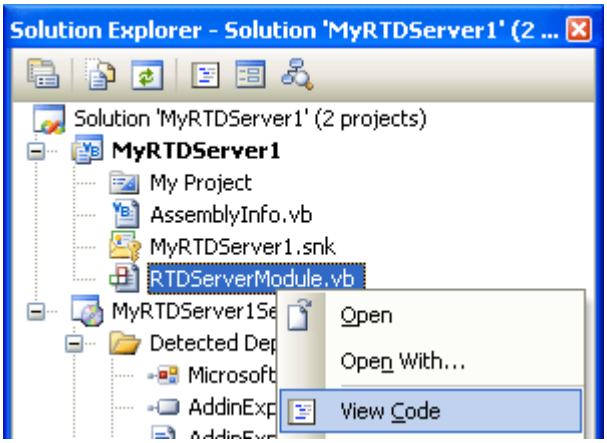



The Add-in Express Project Wizard creates and opens the RTD Server solution in IDE. The solution includes the RTD Server project and the setup project.

The RTD Server project contains the RTDServerModule.vb (or RTDServerModule.cs) file discussed in the next step.

## Step #2 – Add-in Express RTD Server Module

The RTDServerModule.vb (or RTDServerModule.cs) file is an RTD Server Module that is the core part of the RTD Server project. The module is a container for ADXRTDTopic components. It contains the RTDServerModule class, a descendant of the ADXRTDServerModule class that implements the IRtdServer interface and allows you to manage server's topics and their code. To review its source code, in Solution Explorer, right-click the RTDServerModule.vb (or RTDServerModule.cs) file and choose the View Code popup menu item.



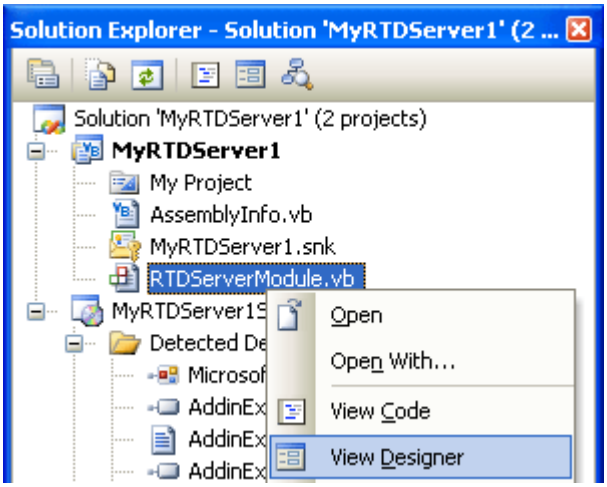

The code of RTDServerModule.vb is as follows:

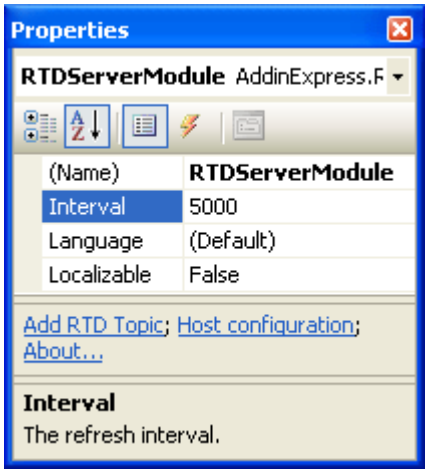
```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express RTD Server Module
<GuidAttribute("ACD23E21-2F2B-4C13-B894-6C74D8AD2EEE"), _
   ProgIdAttribute("MyRTDServer1.RTDServerModule")> _
Public Class RTDServerModule
   Inherits AddinExpress.RTD.ADXRTDServerModule

#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()
      Me.components = New System.ComponentModel.Container
   End Sub

#End Region

#Region " Add-in Express 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

    <ComRegisterFunctionAttribute()> _
    Public Shared Sub RTDServerRegister(ByVal t As Type)
        AddinExpress.RTD.ADXRTDServerModule.ADXRTDServerRegister(t)
    End Sub

    <ComUnregisterFunctionAttribute()> _
    Public Shared Sub RTDServerUnregister(ByVal t As Type)
        AddinExpress.RTD.ADXRTDServerModule.ADXRTDServerUnregister(t)
    End Sub

#End Region

    Public Sub New()
        MyBase.New()
        'This call is required by the Component Designer
        InitializeComponent()

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

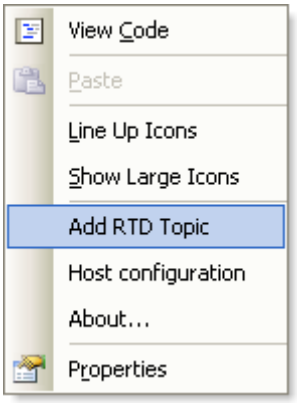
### Step #3 – Add-in Express RTD Server Designer

The Add-in Express RTD Server Designer allows setting RTD Server properties and adding components to the module. In Solution Explorer, right-click the RTDServerModule.vb (or RTDServerModule.cs) file and choose the View Designer popup menu item.




In the Properties window, you set properties of your RTD Server (see RTD Servers).



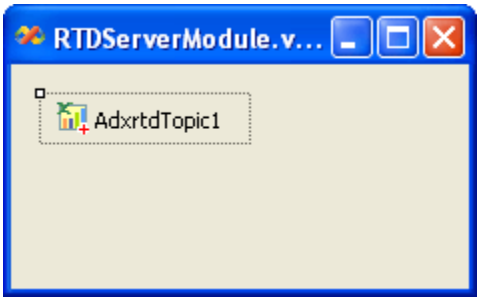

To add an Add-in Express Component to the module, you use an appropriate command in the Properties window, or you can right-click the designer surface and choose the same command in the context menu.

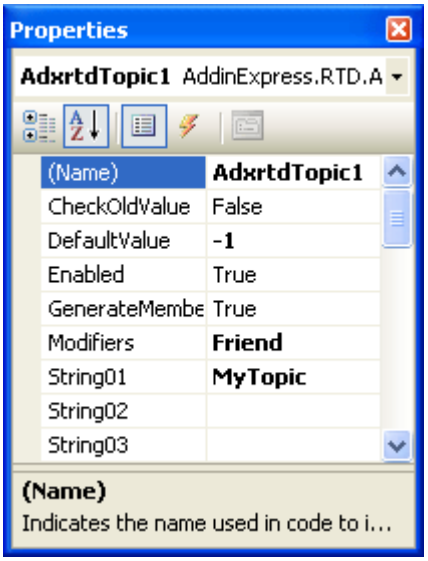
The only command available for the module adds the RTD Topic component to the module (see RTD Topic).

## Step #4 – Adding and Handling a New Topic

To add a new topic to your RTD Server, you use the Add RTD Topic command that adds a new ADXRTDTopic component to the RTD Server Module (see RTD Topic).

Select the newly added component and, in the Properties window, specify the topic using the String## properties.



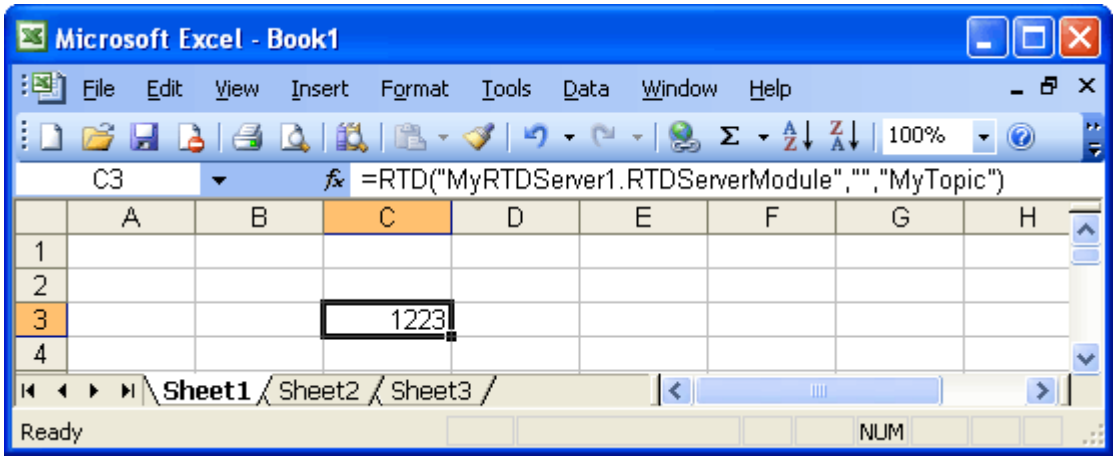

To handle the RefreshData event of the RTD Topic, add the RefreshData event handler and write your code to handle the event:

```
Private Function AdxrtdTopic1_RefreshData( _
   ByVal sender As System.Object) As System.Object _
      Handles AdxrtdTopic1.RefreshData
   Dim Rnd As New System.Random
   Return Rnd.Next(2000)
End Function
```

See also - How Do I Find the Source Code of This Sample?

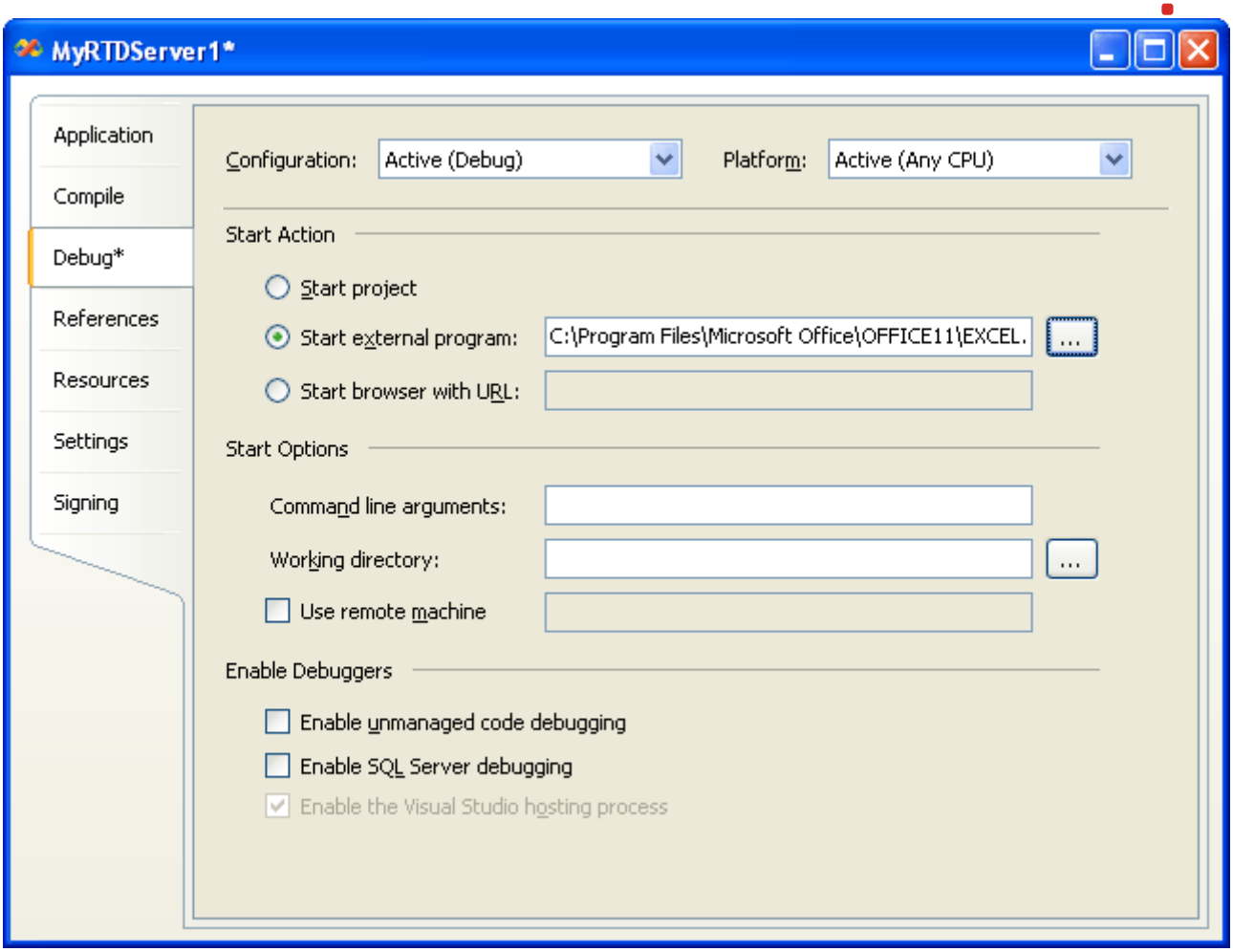
## Step #5 - Running the RTD Server

Choose the Register Add-in Express Project item in the Build menu, restart Excel, and enter the RTD function to a cell.




## Step #6 - Debugging the RTD Server

To debug your RTD Server, just indicate Excel as the Start Program in the Project Options window.



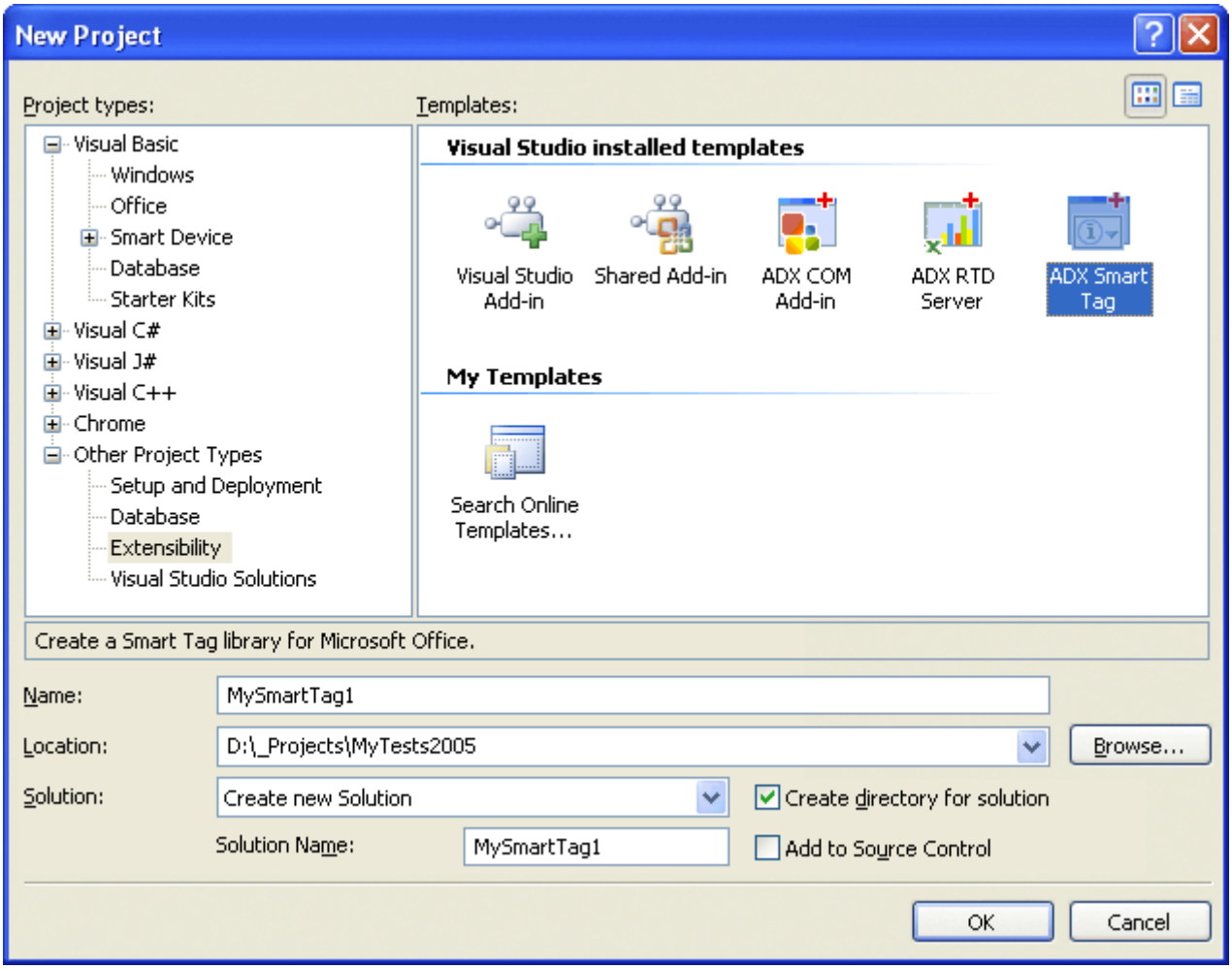

## Step #7 – Deploying the RTD Server

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the RTD Server. See also Deploying Add-in Express Projects and Deploying Office Add-ins.

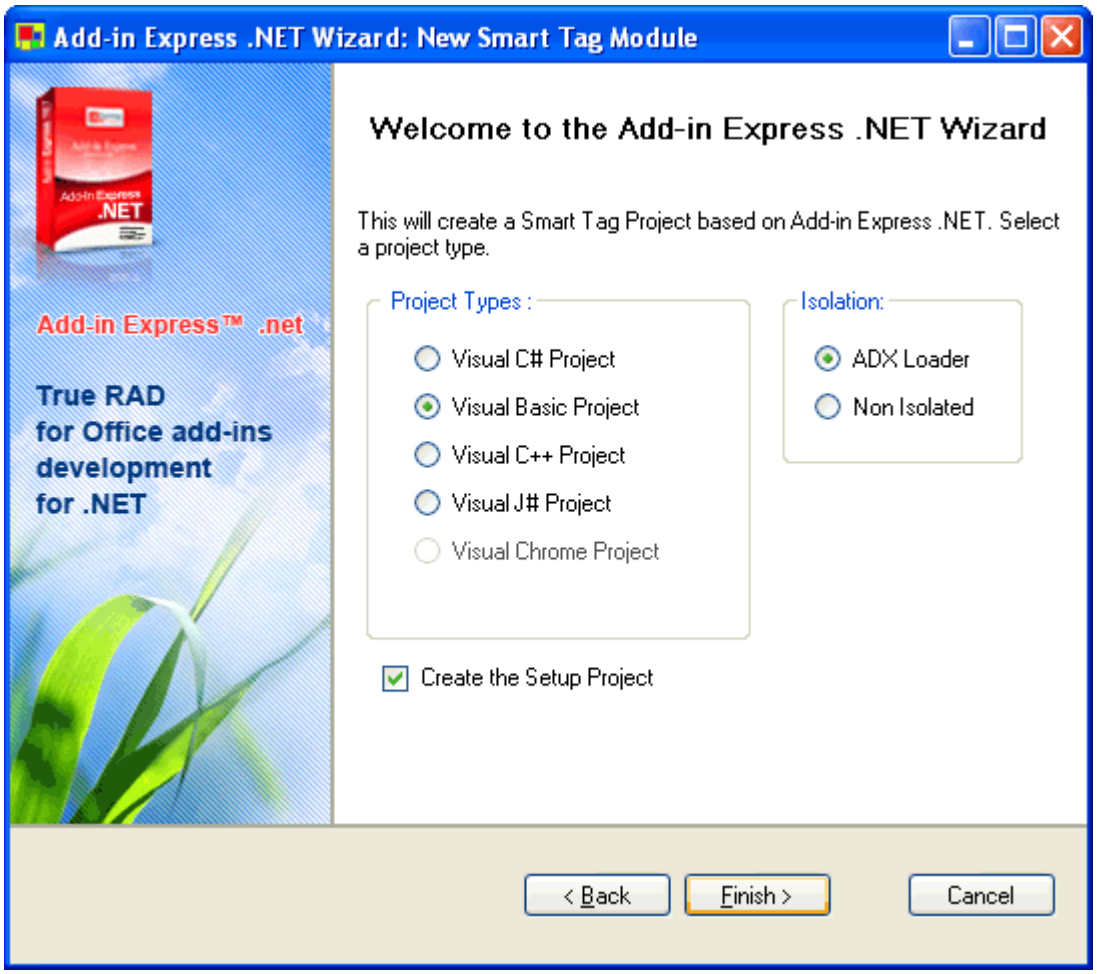
# Your First Smart Tag

## Step #1 – Creating a New Smart Tag Library Project

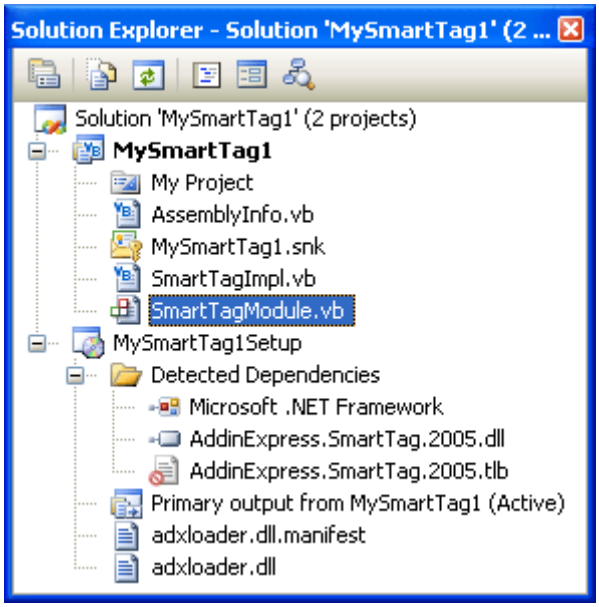
Add-in Express adds the Add-in Express Smart Tag project template to the Visual Studio IDE.




When you select the template and click OK, the Add-in Express Smart Tag project wizard starts. In the wizard window, you choose the programming language and setup project options.




This VB.NET sample shows an Add-in Express SmartTag project with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.



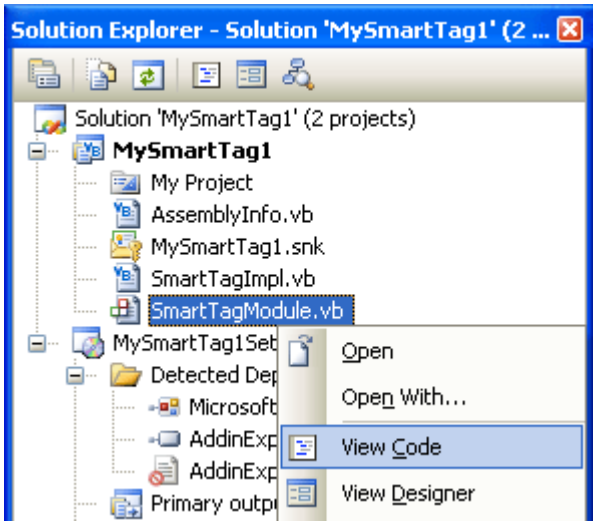

The Add-in Express Project Wizard creates and opens the Smart Tag solution in IDE. The solution includes the SmartTag project and the setup project.

**Note**

*Do not delete the SmartTagImpl.vb file required by the Add-in Express implementation of the Smart Tag technology. Usually, you do not need to modify it.*

The Smart Tag project contains the SmartTagModule.vb (or SmartTagModule.cs) file discussed in the next step.

## Step #2 – Add-in Express Smart Tag Module




The SmartTagModule.vb (or SmartTagModule.cs) file is a Smart Tag Module that is the core part of the Smart Tag project. The Smart Tag module is a container for ADXSmartTag components. It contains the SmartTagModule class, a descendant of the ADXSmartTagModule class that implements the interfaces required by the Smart Tag technology and allows managing smart tags and their code. To review its source code, in Solution Explorer, right-click the SmartTagModule.vb (or SmartTagModule.cs) file and choose the View Code popup menu item.

The smart tag module contains the following code:

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express Smart Tag Module
<GuidAttribute("6FFEFD7D-FD3C-47EE-AE67-A84DE4AD4E7A"),
    ProgIdAttribute("MySmartTag1.SmartTagModule")> _
Public Class SmartTagModule
    Inherits AddinExpress.SmartTag.ADXSmartTagModule

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

   'Required by designer - do not modify
   'the following method
```

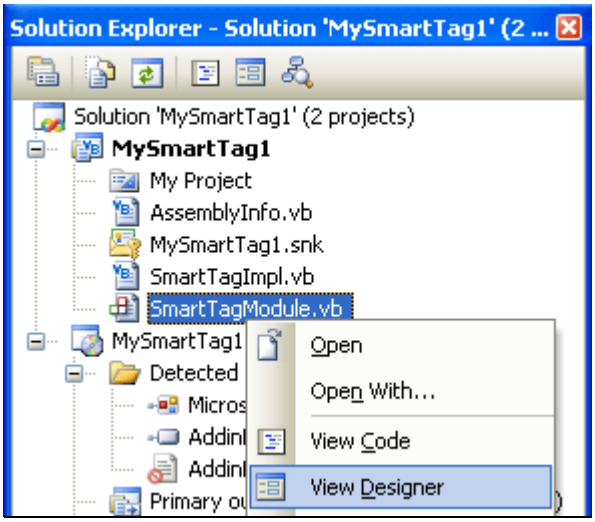
```
    Private Sub InitializeComponent()
        Me.components = New System.ComponentModel.Container
        '
        'SmartTagModule
        '
        Me.NamespaceURI = "mysmarttag1/smarttagmodule"
        Me.LibraryName = New AddinExpress.SmartTag.ADXLocalizedList
        Me.LibraryName.Add(0, "MySmartTag1")
        Me.Description = New AddinExpress.SmartTag.ADXLocalizedList
        Me.Description.Add(0, "This is a description")

    End Sub

#End Region

#Region " Add-in Express 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

    <ComRegisterFunctionAttribute()> _
    Public Shared Sub SmartTagRegister(ByVal t As Type)
        AddinExpress.SmartTag.ADXSmartTagModule.ADXSmartTagRegister(t, _
            System.Type.GetType("MySmartTag1.SmartTagRecognizerImpl"), _
            System.Type.GetType("MySmartTag1.SmartTagActionImpl"))
    End Sub

    <ComUnregisterFunctionAttribute()> _
    Public Shared Sub SmartTagUnregister(ByVal t As Type)
        AddinExpress.SmartTag.ADXSmartTagModule.ADXSmartTagUnregister(t, _
            System.Type.GetType("MySmartTag1.SmartTagRecognizerImpl"), _
            System.Type.GetType("MySmartTag1.SmartTagActionImpl"))
    End Sub

#End Region

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

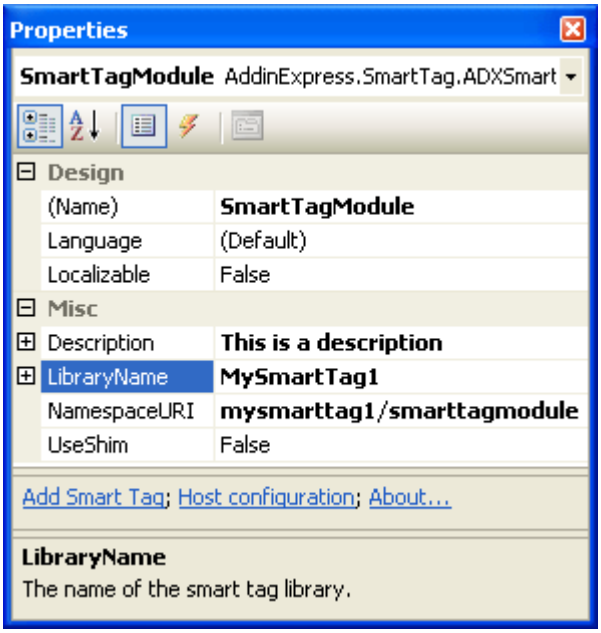
```
        'This call is required by the Component Designer
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub
End Class
```

### Step #3 – Add-in Express Smart Tag Designer

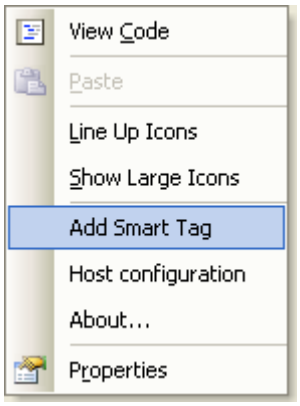
The Add-in Express Smart Tag Designer allows setting Smart Tag properties and adding components to the module.

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




In the Properties window, you set properties of your Smart Tag module (see Smart Tags).

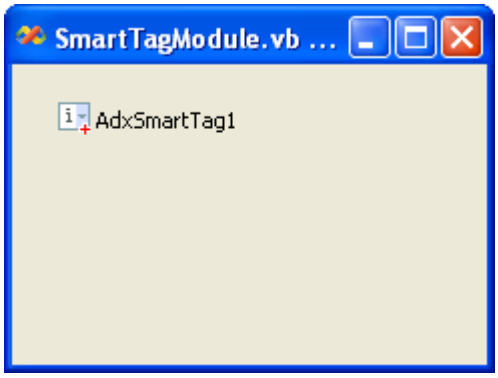



To add an Add-in Express Component to the module, you use an appropriate command in the Properties window, or you can right-click the designer surface and choose the same command in the context menu.

The only command available for the module adds the Smart Tag component to the

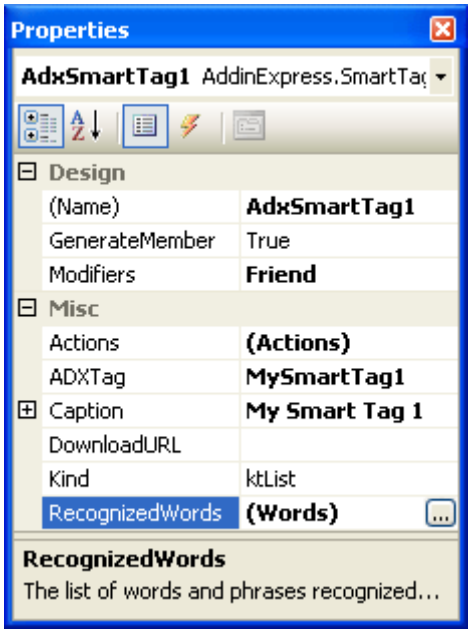
module.

## Step #4 – Adding a New Smart Tag

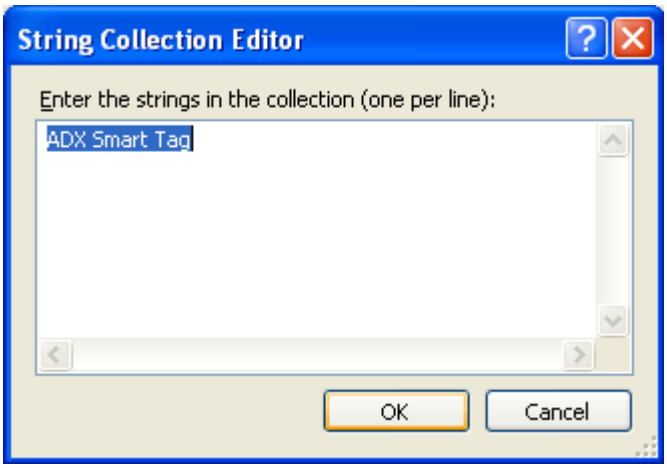



To add a new Smart Tag to your library, use the Add Smart Tag command that adds a new ADXSmartTag component to the Smart Tag Module (see Smart Tag).

Select the newly added component and, in the Properties window, specify the caption for the added smart tag in the Caption property. The value of this property will become a caption of the smart tag context menu. Also, specify the phrase(s) recognizable by the smart tag in the RecognizedWords string collection.

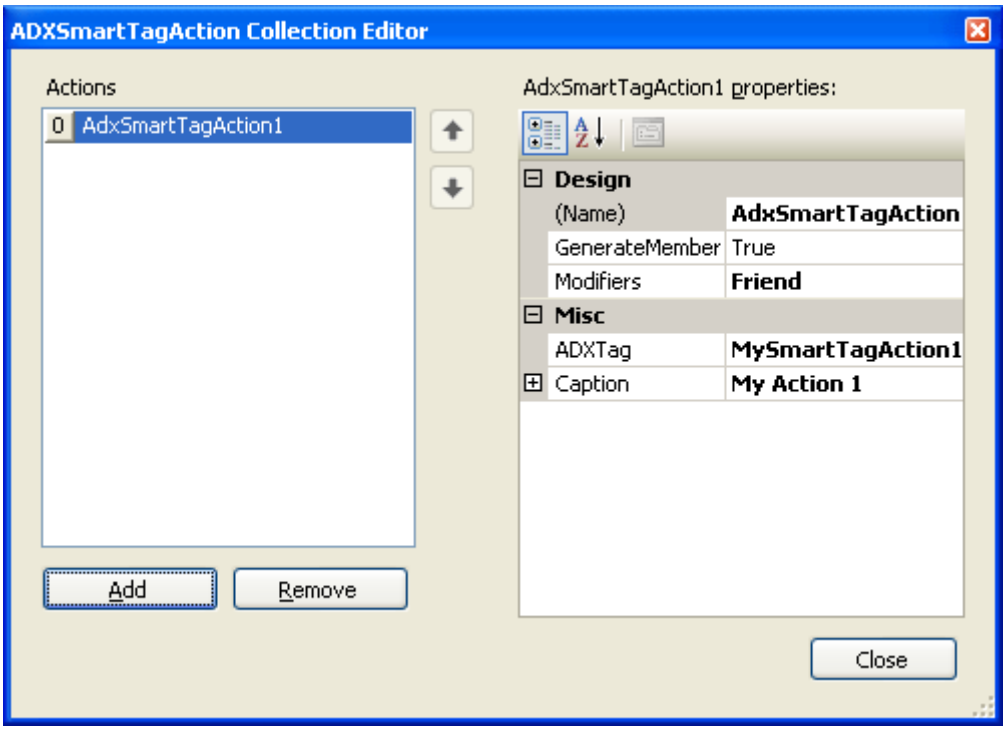



Say, in this sample Smart Tag, the words are as follows:




## Step #5 – Adding and Handling Smart Tag Actions

Now you add smart tag actions to your smart tag context menu. To add a new smart tag action, add an item to the Actions collection of the smart tag.




The value of the action's Caption property will become the caption of the appropriate item in the smart tag context menu.

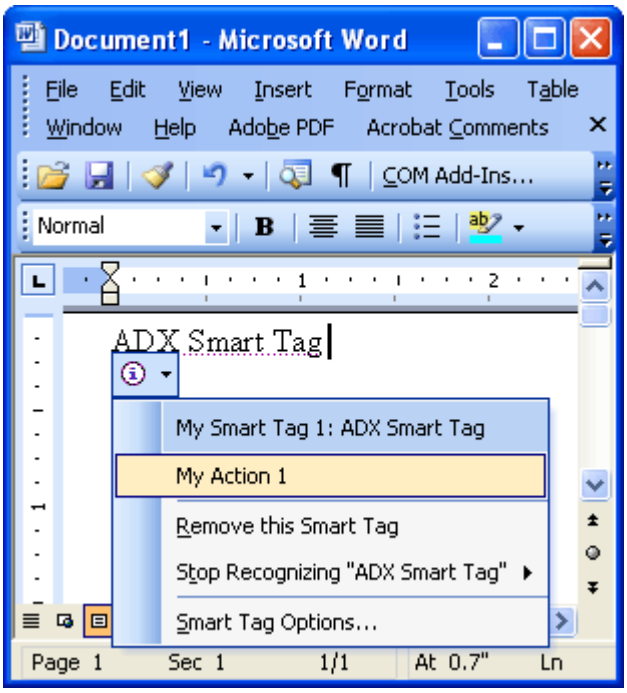
To handle the Click event of the action, close the Actions collection editor, and, in the Properties window, select the added action. Then add the Click event handler and write your code to handle the event:

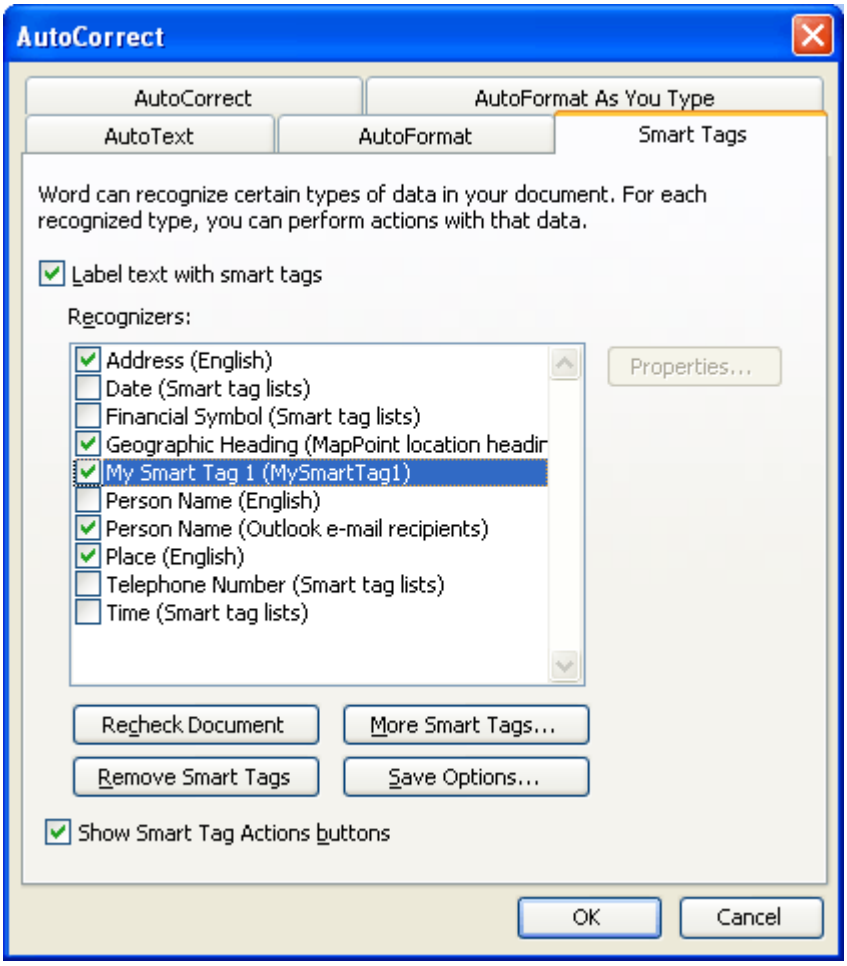
```
Private Sub AdxSmartTagAction1_Click(ByVal sender As System.Object, _
   ByVal e As AddinExpress.SmartTag.ADXSmartTagActionEventArgs) _
      Handles AdxSmartTagAction1.Click
   MsgBox("Recognized text is '" + e.Text + "'!")
End Sub
```

See also - How Do I Find the Source Code of This Sample?

## Step #6 - Running Your Smart Tag

Choose the Register Add-in Express Project item in the Build menu. Restart Word, put the words recognizable by your smart tag into a document, and see if the smart tag works.

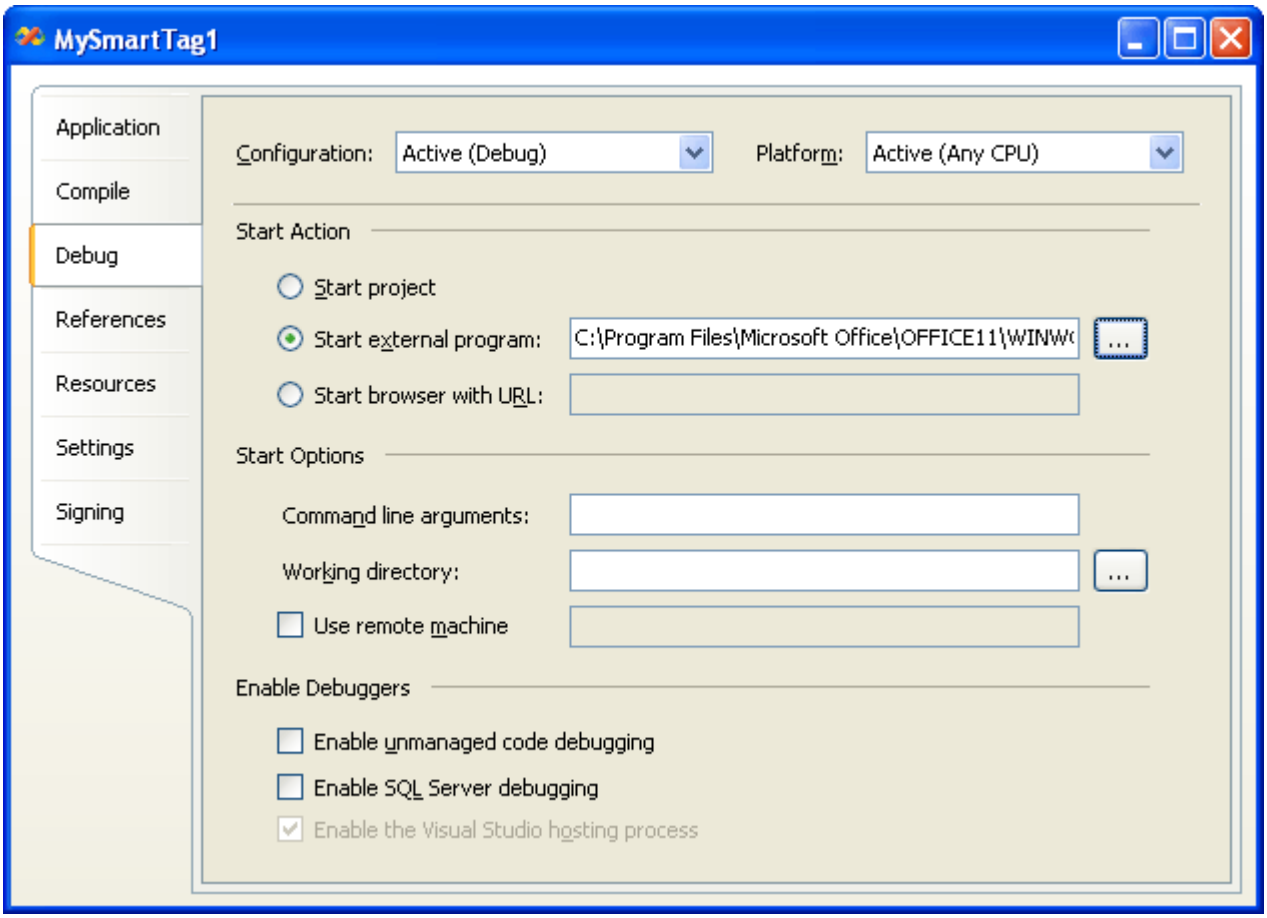



In Office 2003-2003, choose the Tools | AutoCorrect menu item and find your smart tag on the Smart Tags tab. In Office2007, the path to this dialog is as follows: Office button | Word Options | Add-ins | Mange | Manage Smart Tags | Go.

## Step #7 - Debugging the Smart Tag

To debug your Smart Tag, just indicate the add-in host application as the Start Program in the Project Options window.




However, there is a problem here. When debugging an add-in or a smart tag on Visual Studio 2003 and Office XP you cannot see your add-in or smart tag on the COM add-ins or AutoCorrect dialog box. This is a “feature” of Office XP "added" by MS people.

## Step #8 - Deploying the Smart Tag

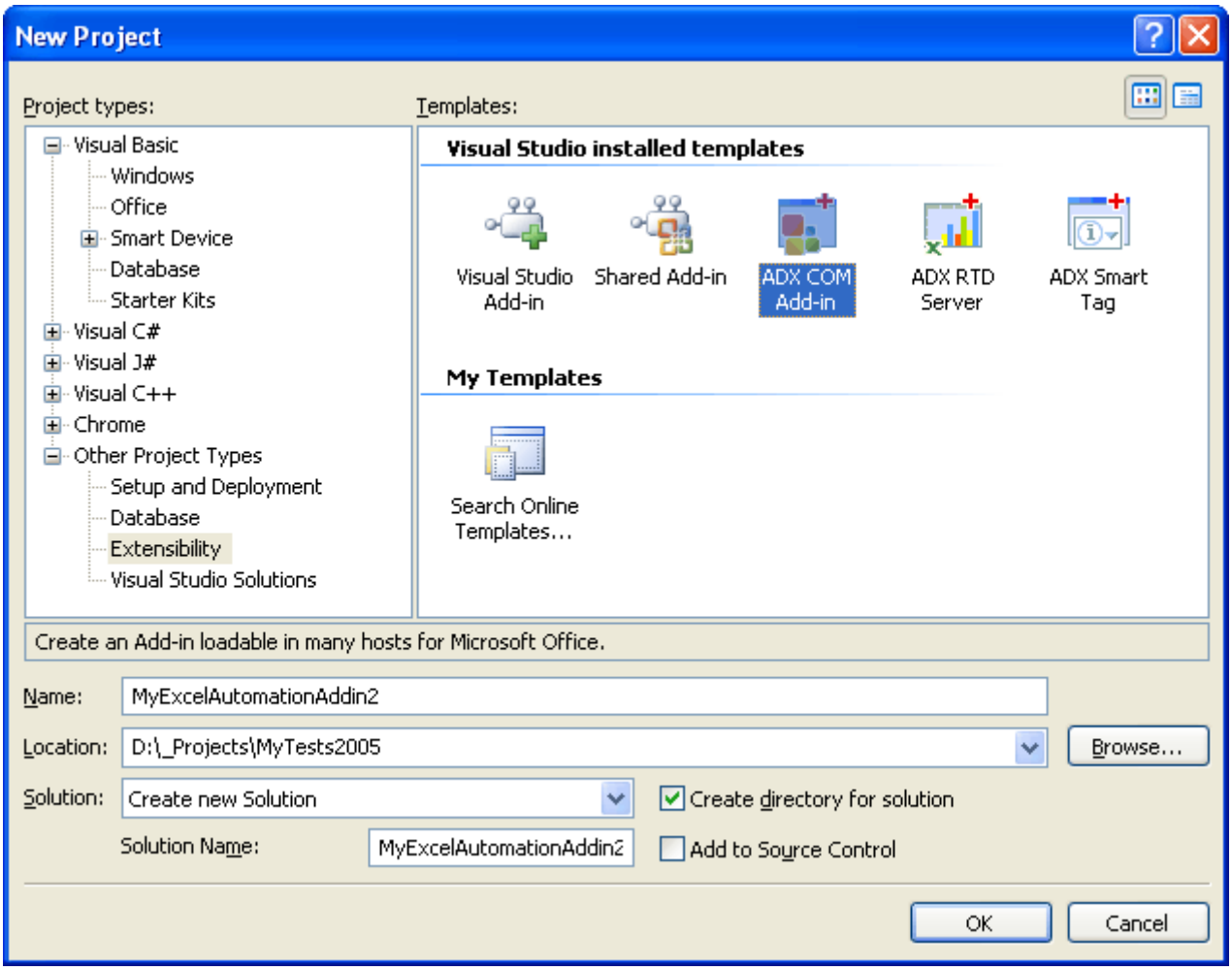
Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the Smart Tag. See also Deploying Add-in Express Projects and Deploying Office Add-ins.

# Your First Excel Automation Add-in

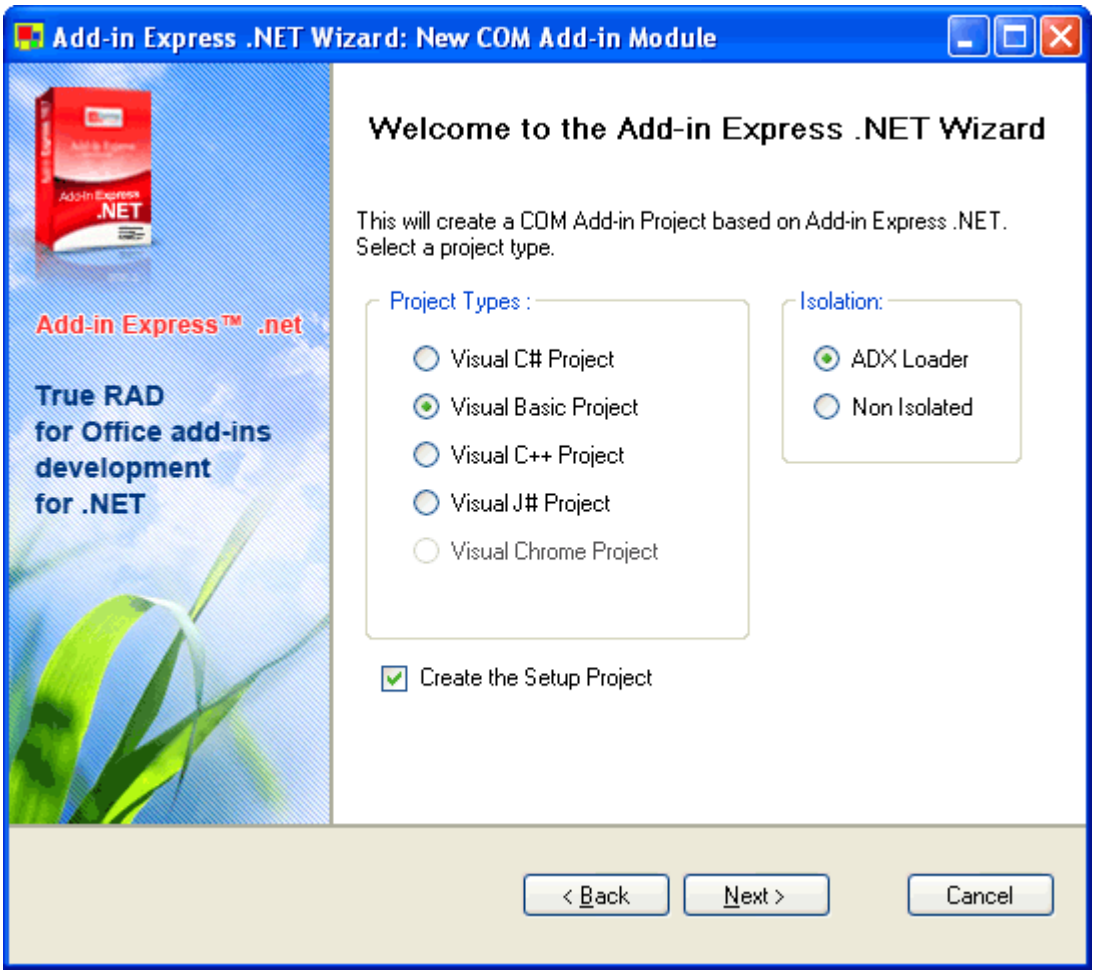
The fact is that Excel Automation Add-ins do not differ from COM Add-ins except for the registration in the registry. That's why Add-n Express bases Excel Automation Add-in projects on Add-in Express COM Add-in projects.

## Step #1 – Creating a New COM Add-in Project

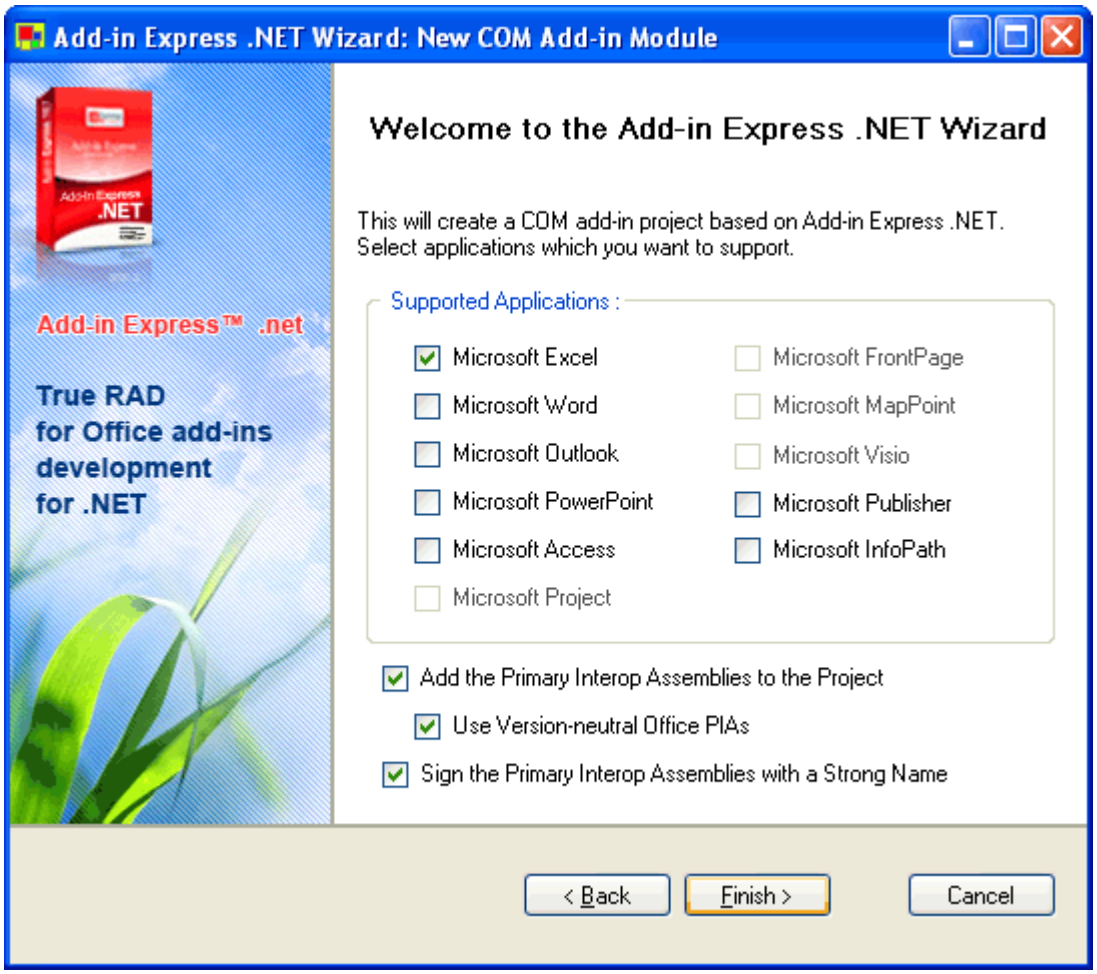
Add-in Express adds the Add-in Express COM Add-in project template to the Visual Studio IDE.

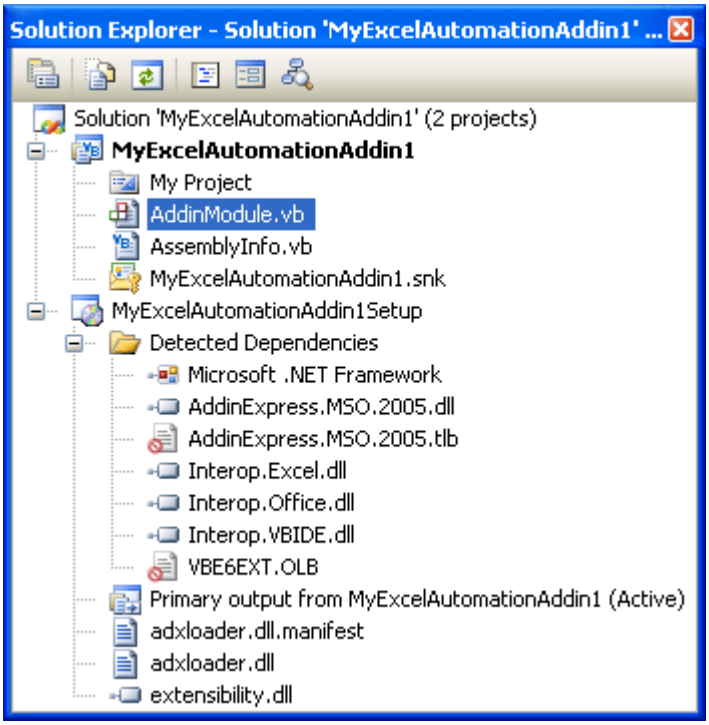



When you select the template and click OK, the Add-in Express COM Add-in project wizard starts. In the wizard windows, you choose the programming language, setup project options, and supported applications of your add-in.




This VB.NET sample shows an Add-in Express COM Add-in project implementing a COM add-in for Excel with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.



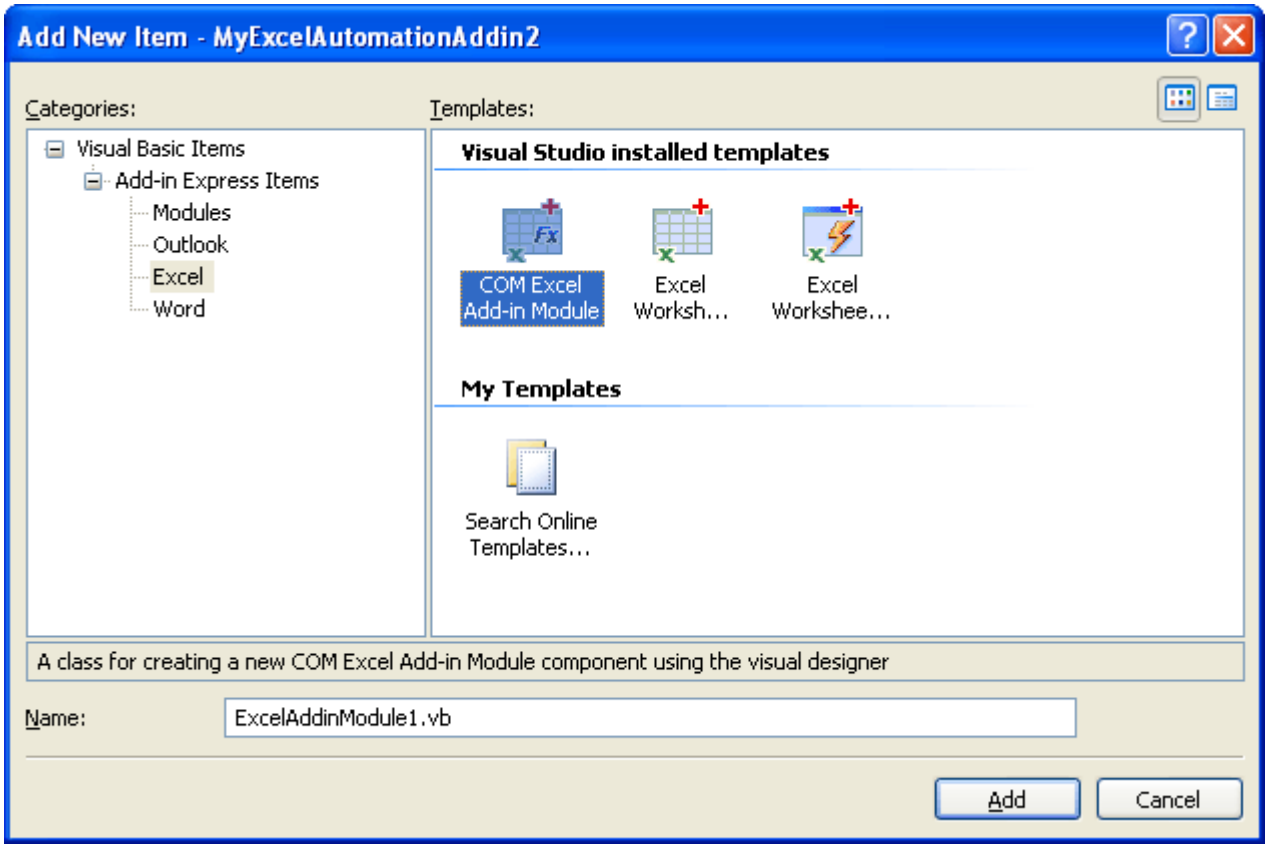




The Add-in Express Project Wizard creates and opens the COM Add-in solution in IDE. The solution includes the COM Add-in project and the setup project.

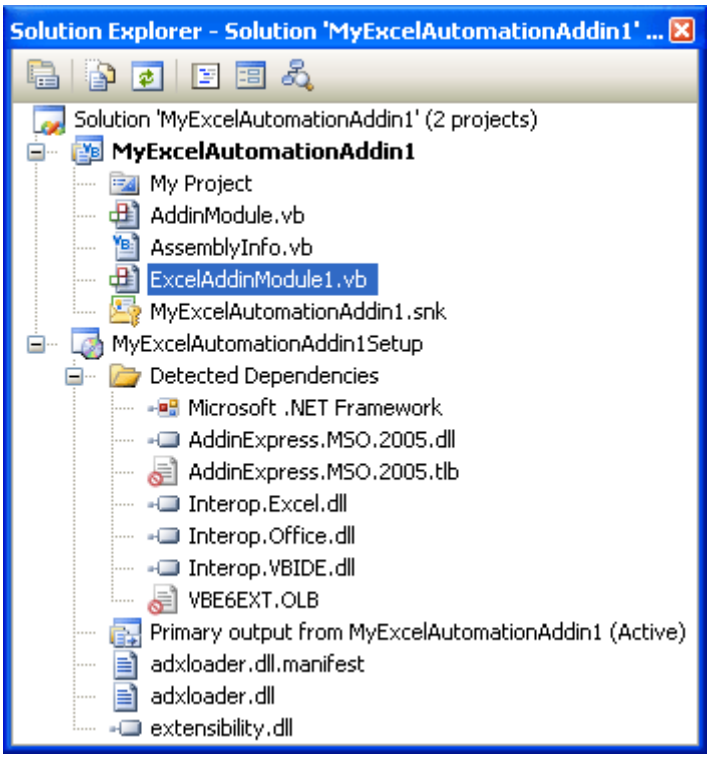
The COM Add-in project contains the AddinModule.vb (or AddinModule1.cs) file discussed in Your First Microsoft Office COM Add-in.

## Step #2 – Adding a New COM Excel Add-in Module

In order to add Excel user-defined functions to the COM Add-in, you add the COM Excel Add-in Module to the Add-in Express COM Add-in project using the Add New Item dialog.

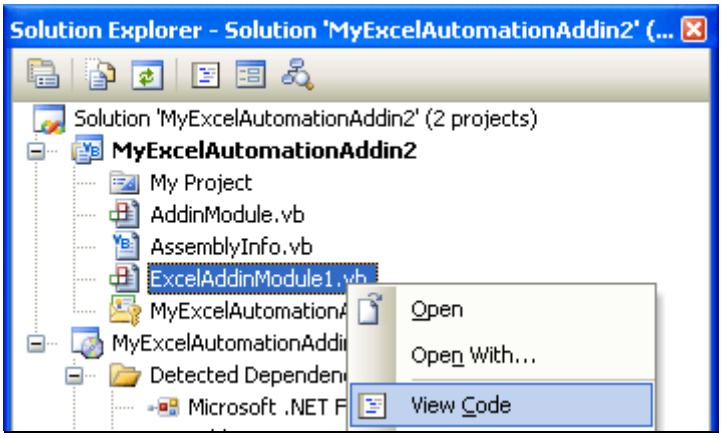



This adds the ExcelAddinModule1.vb (or ExcelAddinModule1.cs) file to your COM Add-in project.




## Step #3- Writing a User-Defined Function

In Solution Explorer, right-click the ExcelAddinModule.vb (or ExcelAddinModule.cs) file and choose the View Code item in the context menu.




The module contains the following code:

```
Imports System.Runtime.InteropServices

'Add-in Express Excel Add-in Module
<GuidAttribute("287D044F-D233-47E6-BB48-35999635BAD3"), _
```

```
   ProgIdAttribute("MyExcelAutomationAddin2.ExcelAddinModule1"), _
      ClassInterface(ClassInterfaceType.AutoDual)> _
   Public Class ExcelAddinModule1
      Inherits AddinExpress.MSO.ADXExcelAddinModule

   #Region " Add-in Express automatic code "

      <ComRegisterFunctionAttribute()> _
      Public Shared Sub AddinRegister(ByVal t As Type)
         AddinExpress.MSO.ADXExcelAddinModule.ADXExcelAddinRegister(t)
      End Sub

      <ComUnregisterFunctionAttribute()> _
      Public Shared Sub AddinUnregister(ByVal t As Type)
         AddinExpress.MSO.ADXExcelAddinModule.ADXExcelAddinUnregister(t)
      End Sub

   #End Region

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

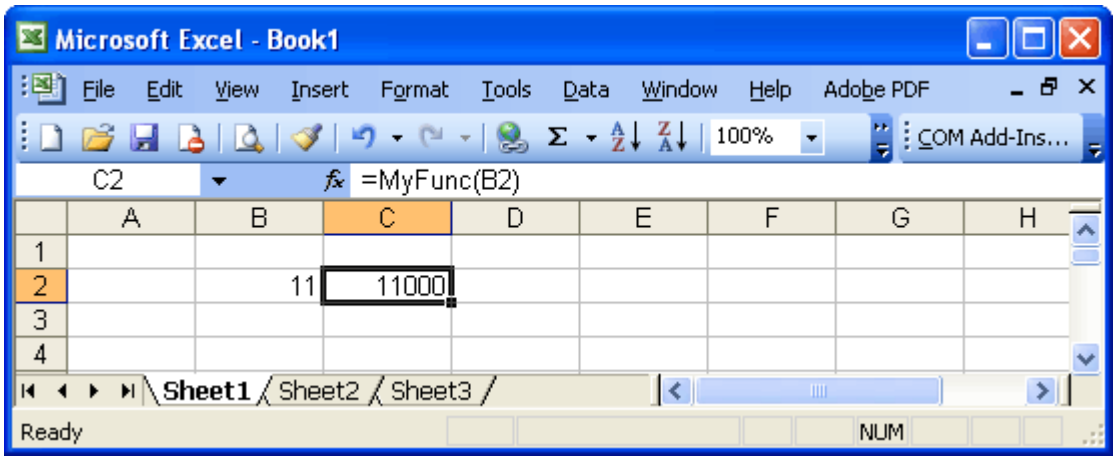
Just add a new function to the module. Say, the following one:

```
      Public Function MyFunc(ByVal Range As Object) As Object
         MyFunc = CType(Range, Excel.Range).Value * 1000
      End Function
```

See also - How Do I Find the Source Code of This Sample?

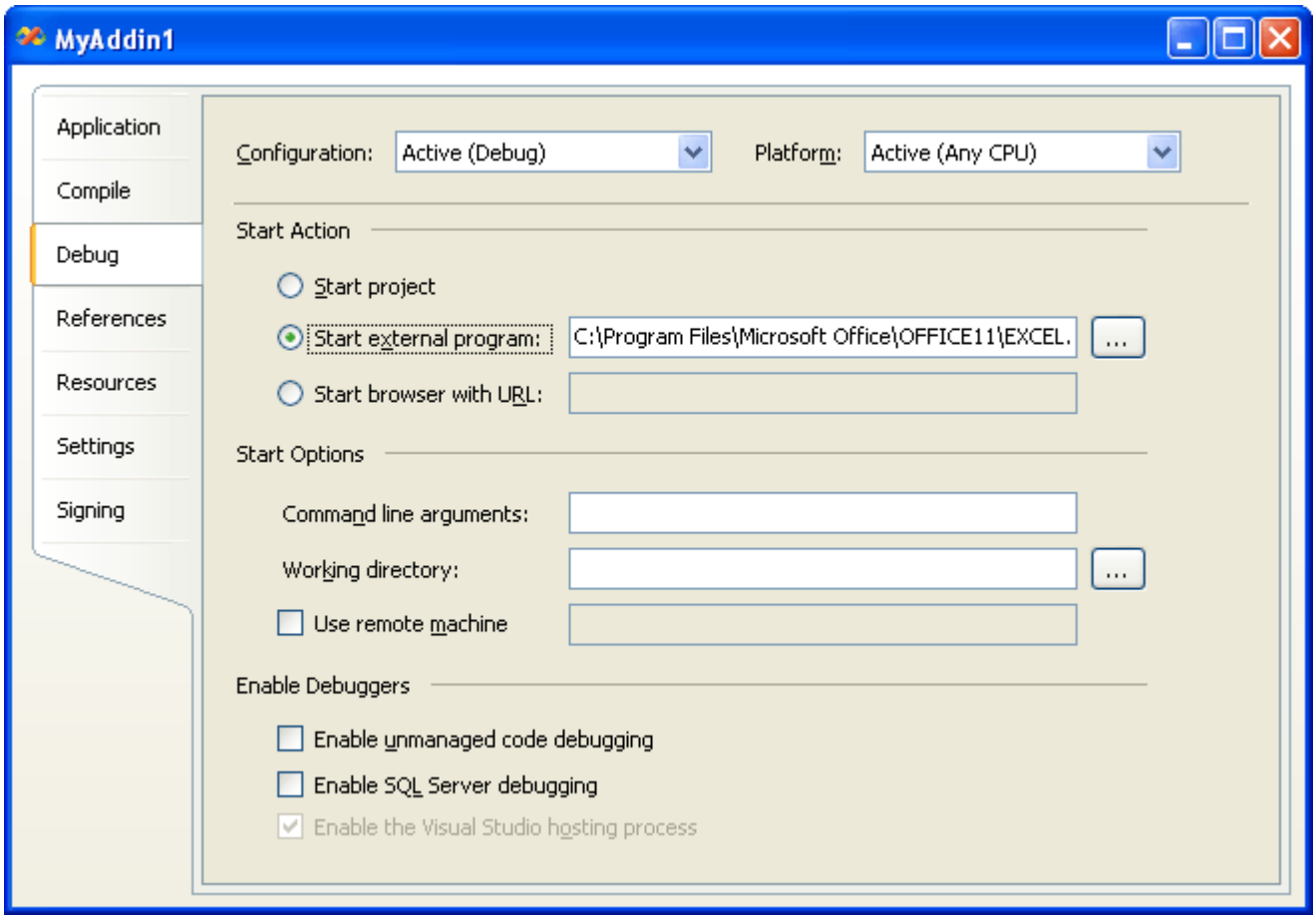
### Step #4 – Running the Excel Automation Add-in

Choose the Register Add-in Express Project item in the Build menu, restart Excel, and check if your add-in works.




## Step #5 – Debugging the Excel Automation Add-in

To debug your add-in, just indicate the add-in host application as the Start Program in the Project Options window.



However, there is a problem here. When debugging an add-in or a smart tag on Visual Studio 2003 and Office XP you cannot see your add-in or smart tag on the COM add-ins or AutoCorrect dialog box. This is a “feature” of Office XP "added" by MS people.

## Step #6 – Deploying the Excel Automation Add-in

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the add-in. See also Deploying Add-in Express Projects and Deploying Office Add-ins.

# Your First XLL add-in

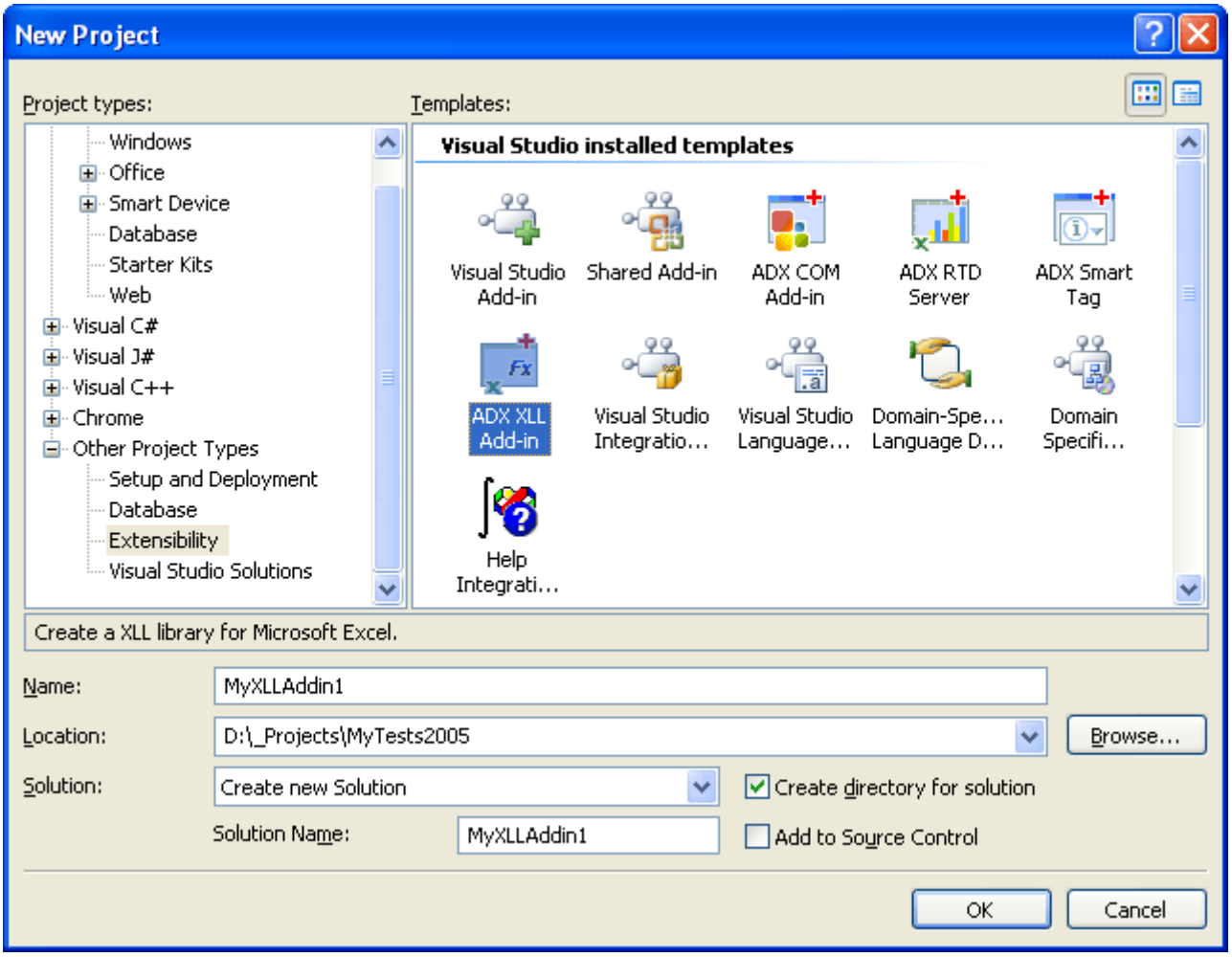
Please note that the current Add-in Express version doesn't support Excel 2007 specific features.

### XLL and Visual Studio 2003

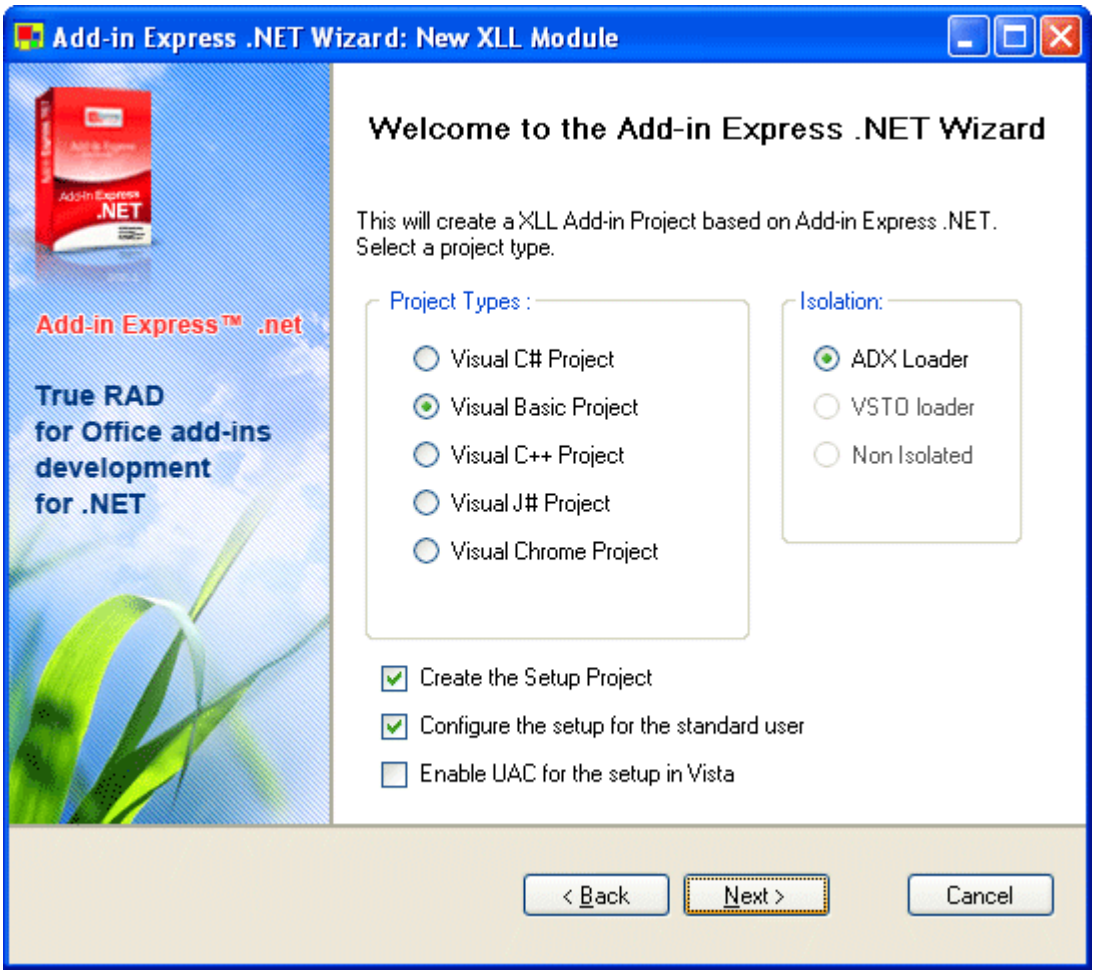
*Add-in Express doesn't support creating XLL add-ins in Visual Studio 2003 (.NET Framework 1.1).*

## Step #1 - Creating a New Add-in Express XLL Add-in Project

Add-in Express adds the Add-in Express XLL Add-in project template to the Visual Studio IDE.



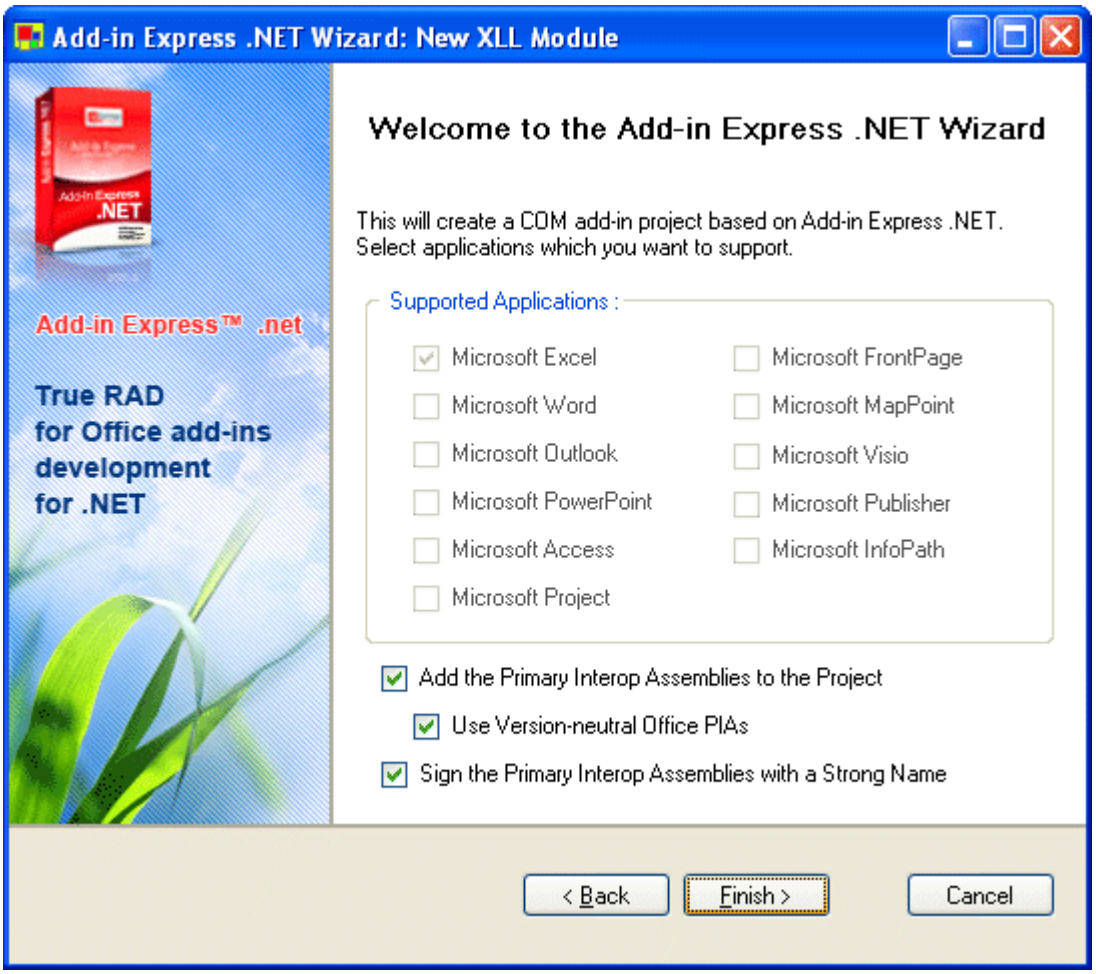

When you select the template and click OK, the Add-in Express XLL Add-in project wizard starts. In the wizard window, you choose the programming language and setup project options.

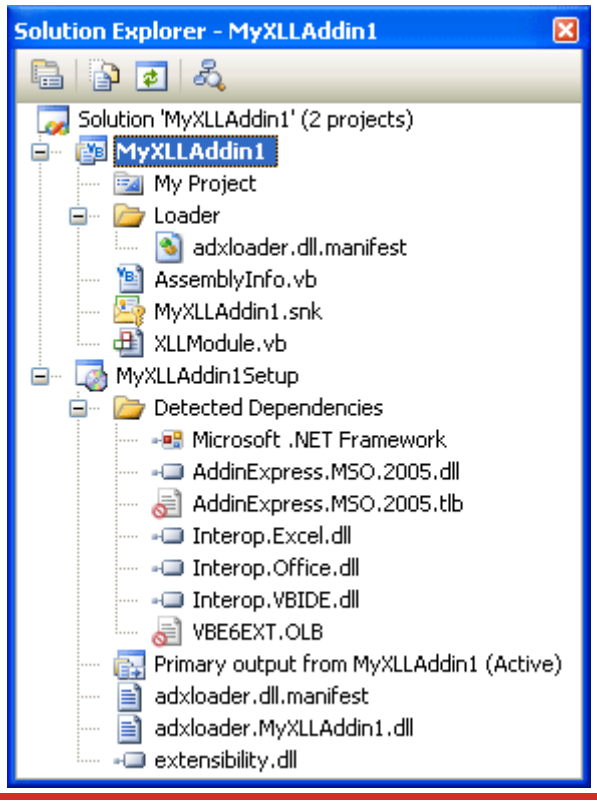



This VB.NET sample shows an Add-in Express XLL Add-in project with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.

**Note**

*Currently, the sample described here isn't available in the Add-in Express setup package.*




The Add-in Express Project Wizard creates and opens the XLL Add-in solution in IDE. The solution includes the XLL Add-in project and the setup project.

The XLL Add-in project contains the XLLModule.vb (or XLLModule.cs) file discussed in the next step.

### Step #2 – Add-in Express XLL Module

The XLLModule.vb (or XLLModule.cs) file is the core part of the XLL Add-in project. The XLL module is a container for Category components. It contains the XLLModule class, a descendant of the ADXXLLModule class that implements the interfaces required by the XLL technology and allows creating and configuring custom used defined functions (UDF). To review its source code, in Solution Explorer, right-click the XLLModule.vb (or XLLModule.cs) file and choose the View Code popup menu item.

Please note the XLLContainer class in the code below. We describe its use in the next steps.

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express XLL Add-in Module
<ComVisible(True)> _
Public Class XLLModule
    Inherits AddinExpress.MSO.ADXXLLModule

#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()
      '
      'XLLModule
      '
      Me.AddinName = "MyXLLAddin1"

   End Sub

#End Region

#Region " Add-in Express 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
```

```
    <ComRegisterFunctionAttribute()> _
    Public Shared Sub RegisterXLL(ByVal t As Type)
        AddinExpress.MSO.ADXXLLModule.RegisterXLLInternal(t)
    End Sub

    <ComUnregisterFunctionAttribute()> _
    Public Shared Sub UnregisterXLL(ByVal t As Type)
        AddinExpress.MSO.ADXXLLModule.UnregisterXLLInternal(t)
    End Sub

#End Region

#Region " Define your UDFs in this section "

    Friend Class XLLContainer

        Friend Shared ReadOnly Property _Module() As MyXLLAddin1.XLLModule
            Get
                Return CType(AddinExpress.MSO.ADXXLLModule. _
                    CurrentInstance, MyXLLAddin1.XLLModule)
            End Get
        End Property

    End Class

#End Region

    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 CType(HostApplication, Excel._Application)
        End Get
    End Property

End Class
```

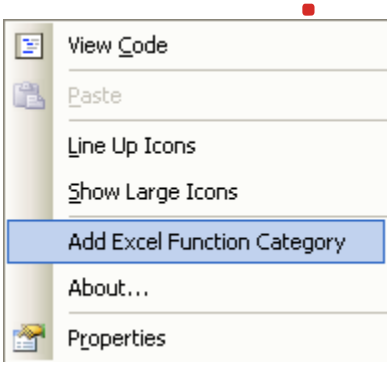
### Step #3 – Creating a New User-Defined Function

Just add a new public Shared (static in C#) function to the XLLContainer class. Say, we add the following code to demonstrate all the types that Excel can pass to user-defined functions.

```
    Friend Class XLLContainer

...

      Public Shared Function AllSupportedExcelTypes(ByVal arg As Object) _
         As String
         If (TypeOf arg Is Double) Then
            Return "Double: " + arg.ToString()
         ElseIf (TypeOf arg Is String) Then
            Return "String: " + arg
         ElseIf (TypeOf arg Is Boolean) Then
            Return "Boolean: " + arg.ToString()
         ElseIf (TypeOf arg Is AddinExpress.MSO.ADXExcelError) Then
            Return "ExcelError: " + arg.ToString()
         ElseIf (TypeOf arg Is Object(,)) Then
            Return String.Format("Array[{0},{1}]", arg.GetLength(0), _
                arg.GetLength(1))
         ElseIf (TypeOf arg Is System.Reflection.Missing) Then
            Return "Missing"
         ElseIf (arg Is Nothing) Then
            Return "Empty"
         ElseIf (TypeOf arg Is AddinExpress.MSO.ADXExcelRef) Then
            Return "Reference: " + arg.ConvertToA1Style()
         ElseIf (TypeOf arg Is AddinExpress.MSO.ADXExcelRef) Then
            Return String.Format("Reference [{0},{1},{2},{3}]", _
                arg.ColumnFirst, arg.RowFirst, arg.ColumnLast, arg.RowLast)
         Else
            Return "Unknown Type"
         End If
      End Function

    End Class
```

### Step #4 – Configuring UDFs

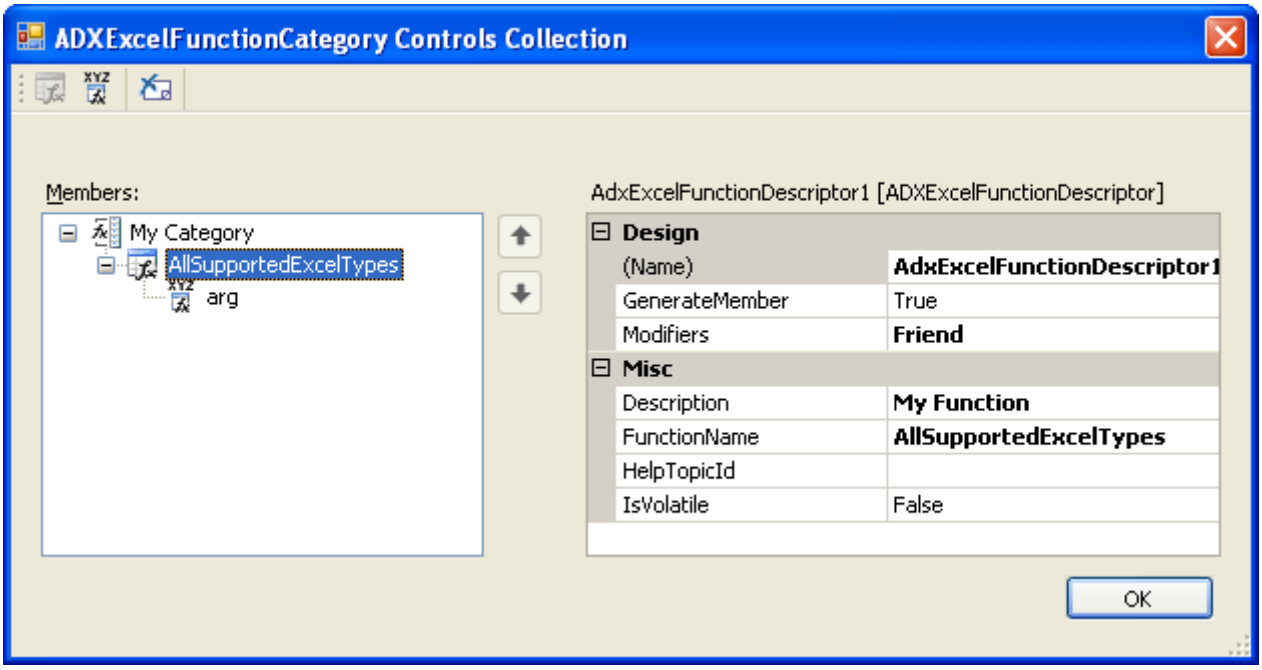
To integrate the XLL add-in with Excel, you have to supply Excel with user-friendly add-in name, function names, parameter names, help topics, etc. When using Add-in Express XLL add-ins, you start with categories. For a category, you add function descriptors. For a function descriptor, you add parameter descriptors. Let's go.




In the Solution Explorer window, right-click the XLL module (XLLModule.vb or XLLModule.cs) and choose View Designer in the popup menu. You can specify the add-in name in the Properties window.

Now you right-click the designer surface and, in the popup menu, choose the Add Excel function Category item.

For the category, you specify its name. Then, you need to add a function descriptor and bind it to the UDF you created in the XLLContainer class. Just click the Functions property of the Category component – this runs the visual designer.




For function descriptors, the properties of interest are FunctionName and IsVolatile. The former is a combo box that allows choosing a function from the list of functions defined in the XLLContainer class. As to the latter, it is set to True, Excel will call the appropriate function whenever it recalculates the worksheet.

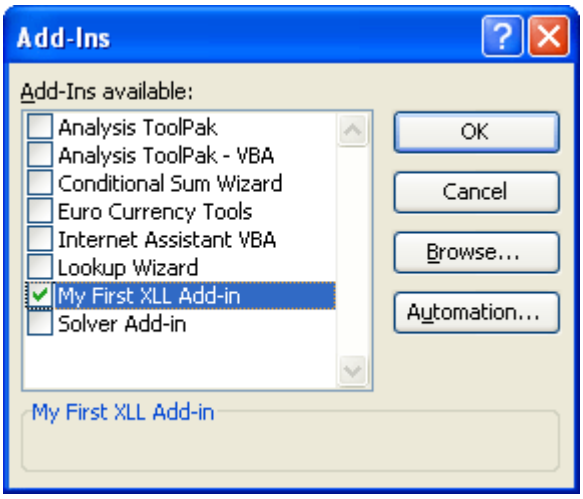
In the same way, you describe the arguments of the function: just select the parameter name. The property is a combo box that allows choosing a parameter from the list of parameters of a given function.

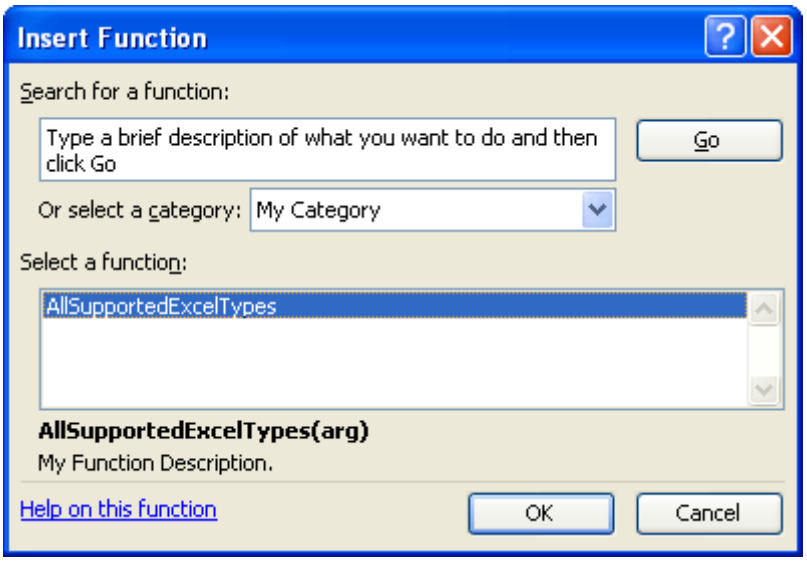
**Warning!**

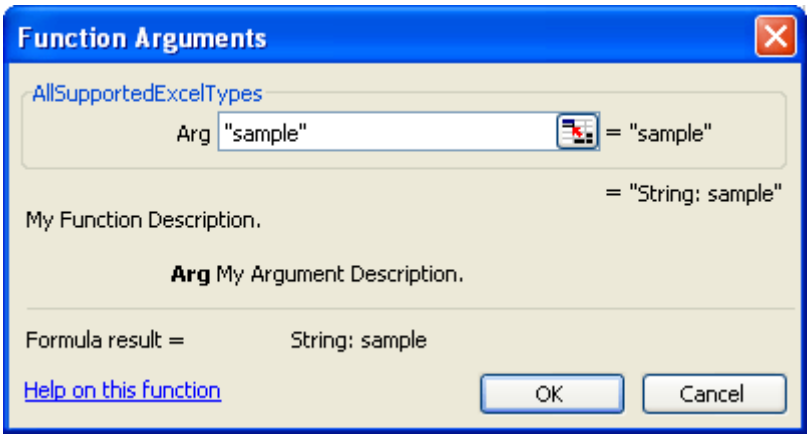
*When renaming functions and arguments, you have to reflect these changes in appropriate descriptors. In the opposite case, Excel will not receive the information required.*

## Step #5 – Running Your XLL Add-in

Build and register your XLL add-in (choose the Register Add-in Express Project item in the Build menu), restart Excel, and check if your add-in works.

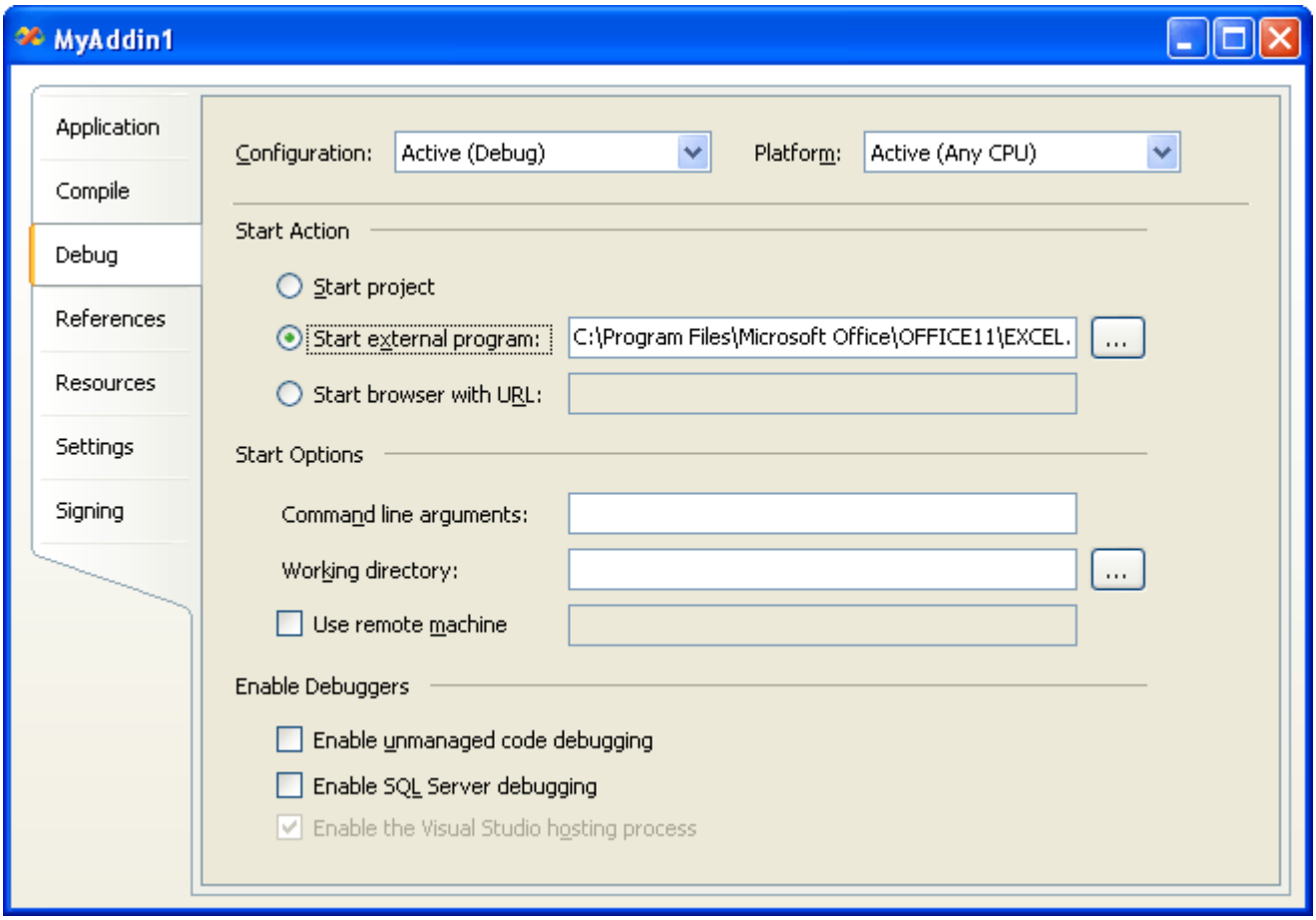


## Step #6 – Debugging the XLL Add-in

To debug your add-in, in the Project Options window, indicate Excel in the Start External Program and run the project.




## Step #7 – Deploying the XLL Add-in

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the add-in. See also Deploying Add-in Express Projects and Deploying Office Add-ins.

# Deploying Add-in Express Projects

## Shims in Add-in Express Projects

You have three isolation options when creating Add-in Express .NET projects. You can see them in the Add-in Express Project Wizard window. Every isolation option uniquely determines the shim (see What are Shims? below) and this, in its turn, influences on the deployment of your project.

### What are Shims?

All Office applications are unmanaged while all Add-in Express .NET based add-ins are managed class libraries. Therefore, there must be some software located between Office applications and your add-ins. Otherwise, Office applications will not know of your .NET add-ins and other Office extensions. That software is called a shim. Shims are unmanaged DLLs that isolate your add-ins in a separate application domain.

When you install your add-in, the registry settings for the add-in will point to the shim. And the shim will be the first DLL examined by the host application when it runs your add-in or smart tag.

### Add-in Express Loader

If, in the project wizard windows, you choose the Add-in Express Loader and automatic setup project generation options, the wizard goes through the steps listed in the Creating Setup Projects Manually and generates the solution that includes the Add-in Express Loader based add-in project and appropriate setup project.

Add-in Express Loader (adxloader.dll) is a compiled shim (What are Shims?) not bound to any certain Add-in Express project. Instead, the loader uses the adxloader.dll.manifest file containing a list of .NET assemblies that must be registered. The loader's files (adxloader.dll and adxloader.dll.manifest) must always be located in the Loader subdirectory of the main Add-in Express project directory. When an Add-in Express based project is being built, the loader files are copied to the project's output directory. You can sign the loader with a digital signature and, in this way, create trusted COM extensions for Office. For Add-in Express loader based setup projects, the loader provides ready-to-use Install, Uninstall, and Rollback custom actions.

The manifest is the source of configuration information for the loader. Below, you see the contents of a sample manifest file.

```
   <?xml version="1.0" encoding="utf-8"?>
   <configuration>
     <assemblyIdentity name="MyAddin14, PublicKeyToken=f9f39773da5c568a" />
     <loaderSettings generateLogFile="true" shadowCopyEnabled="true"
privileges="user" />
   </configuration>
```

The manifest file allows generating the log file containing useful information about errors on the add-n loading stage. The log file is located here: My Documents\Add-in Express\adxloader.log. The manifest file allows you to disable the Shadow Copy feature of Add-in Express Loader, which is enabled by default (see Deploying – Shadow Copy). The **privileges** attribute accepts the "user" string indicating that the Add-in Express based setup projects can be run with non-administrator privileges. Please, note, any other value will require administrator privileges to install your project. You should be aware that the value of this attribute is controlled by the RegisterForAllUsers property value of the add-in and RTD module.

Also, you can run regsvr32 against the adxloader.dll. If the correct manifest file is located in the same folder, this will register all Add-in Express projects listed in the loader manifest.

## VSTO Loader

VSTO Loader is a compiled shim that must be pre-installed on the PC as a part of the VSTO Run-time installation. The loader uses the vstoloader.manifest file that supplies VSTO loader with all the information it needs. The manifest must always be located in the Loader subdirectory of the main Add-in Express project directory. For VSTO loader based setup projects, Add-in Express supports the Install, Ununstall, and Rollback custom actions provided by a special .NET assembly called Add-in Registrator (adxregaddin.exe). It is located in the Redistributables folder of Add-in Express setup folder.

Starting from version 3.5, Add-in Express project wizard provides the VSTO Loader option. When using this option you should be aware of the following points.

**Creating an Add-in Project**

- VSTO Loader supports Office 2003 and 2007 only.
- You cannot use Version-Neutral PIAs in this case: they are not recognized by VSTO Loader;
- The AddinExpress.MSO.OfficeTools namespace is located in the AddinExpress.MSO.OfficeTools.dll; it is added to the setup project automatically;

**Developing the Add-in**

- The Loader folder contains the only file vstoloader.manifest generated automatically; it could resemble the following one:

```
   <?xml version="1.0" encoding="utf-8"?>
   <assembly xmlns="urn:schemas-microsoft-com:asm.v1" xmlns:asmv2="urn:schemas-
microsoft-com:asm.v2" manifestVersion="1.0">
     <assemblyIdentity name="MyAddin12.dll" version="1.0.0.0" />
     <asmv2:entryPoint name="Startup" dependencyName="dependency0">
       <asmv2:clrClassInvocation class="MyAddin12.ADXAddinEntryPoint" />
     </asmv2:entryPoint>
     <asmv2:dependency asmv2:name="dependency0">
       <asmv2:dependentAssembly>
         <assemblyIdentity name="MyAddin12" version="1.0.0.0" />
```

```
        <asmv2:installFrom codebase="MyAddin12.dll" />
      </asmv2:dependentAssembly>
    </asmv2:dependency>
  </assembly>
```

- The add-in module in this case inherits from AddinExpress.MSO.OfficeTools.ADXAddin and not from AddinExpress.MSO.ADXAddinModule;

**Registering/unregistering the Add-in**

- When you run Build / Register Add-in Express Project menu command, Add-in Express creates the following code group: User / Code groups / All code / <Add-in Module GUID> <ProgId>, and grants it the "Full Trust" permission. You can find the group in Control Panel / Administrative Tools / .NET Framework 2.0 configuration.
- In addition, the Add-in Registrator creates all the registry records required by Office add-ins.

**Deploying the Add-in**

- For the setup project, the Add-in Registrator (adxregaddin.exe) is used as a custom action to provide the functionality above: see the Custom Actions view of any setup project created using the VSTO Loader and Create the Setup Project options.

## Non-isolated Add-in Express Projects

If you choose the Non Isolated option in the Add-in Express Project wizard, your add-ins are loaded into the host application with a universal shim called MSCoree.DLL. This shim is pre-installed with the version of .NET Framework required by your add-in. That is, you don't need to include it into your setup project.

Typically, you use this shim when your add-in doesn't have any special requirements to security. The main advantage of using this shim is its simplicity. However, you cannot sign it, and this is its main drawback. If the host application security is set to Medium, High or Very High (Tools | Macro | Security), MSCoree based add-ins will not start even if you sign them.

There is another drawback: if an MSCoree based add-in fires an unhandled exception, the user can opt to stop the add-in from loading for the next time and this will effectively stop all MSCoree based add-ins from loading.

## Shim Comparison Table

| | Add-in Express Loader | VSTO Loader | Non-Isolated (MSCOREE.DLL) |
|---|---|---|---|
| Which Visual Studio versions are supported by the shim? | VS 2003, VS 2005, VS 2008 | VS 2005, VS 2008 | VS 2003, VS 2005, VS 2008 |
| Which Office versions are supported by the shim? | Office 2000, 2002, 2003, 2007 | Office 2003, 2007 | Office 2000, 2002, 2003, 2007 |
| Is the add-in isolated in a separate application domain? | Yes | Yes | No, your add-in shares the same application domain with other non-isolated add-ins. |
| Does the shim require adding to the setup project? | Add-in Express Project Wizard adds it automatically | No, it is pre-installed with VSTO run-time | No, it is pre-installed with .NET Framework 1.1 and 2.0 |
| Which assembly provides custom actions for setup projects? | adxloader.dll | adxregaddin.exe | AddinExpress.Install.dll<br>AddinExpress.Install.2005.dll |
| Can the shim be signed? | Yes | No | No |
| Does the shim allow updating add-in DLLs when add-in is running? | Yes. See Deploying – Shadow Copy | No | No |
| Is the shim configurable? | Yes, you can edit the manifest file. See Add-in Express Loader | Yes, you can edit the manifest file. See VSTO Loader. | No |
| Can regsvr32 be run against the shim? | Yes | No | No |
| Does the shim allow many add-ins in one assembly? | Yes | No | Yes |
| Is the shim supported by Add-in Express ClickOnce Solution? | Yes | Yes | No |

# Add-in Express ClickOnce Projects

## ClickOnce Overview

What follows below is a brief compilation of the following Internet resources:

- ClickOnce article from Wikipedia
- ClickOnce FAQ on windowsclient.net
- Introduction to ClickOnce deployment on msdn2.microsoft.com (also compares ClickOnce and MSI)
- ClickOnce Deployment in .NET Framework 2.0 on 15seconds.com

ClickOnce is a deployment technology introduced in .NET Framework 2.0. Targeted to non-administrator-privileges installations it also allows updating your applications. Subject to many restrictions, it isn't a panacea in no way. Say, if your prerequisites include .NET Framework 2.0 and the user doesn't have it installed, your application (as well as an add-in) will not be installed without administrator privileges. In addition, ClickOnce will not allow installing shared components, such as custom libraries. It is quite natural, though.

When applied to a Windows forms application, ClickOnce deployment implies the following steps:

- Publishing an application

You deploy the application to any of the following locations: File System (CD/DVD included) or Web Site. The files include all application files as well as application manifest and deployment manifest. The application manifest describes the application itself, including the assemblies, the dependencies and files that make up the application, the required permissions, and the location where updates will be available. The deployment manifest describes how the application is deployed, including the location of the application manifest, and the version of the application that the user should run. The deployment manifest also contains an update location (a Web page or network file share) where the application checks for updated versions. ClickOnce Publish properties are used to specify when and how often the application should check for updates. Update behavior can be specified in the deployment manifest, or it can be presented as user choices in the application's user interface by means of the ClickOnce APIs. In addition, Publish properties can be employed to make updates mandatory or to roll back to an earlier version.

- Installing the application

The user clicks a link to the deployment manifest on a web page, or double-clicks the deployment manifest file in Windows Explorer. In most cases, the end user is presented with a simple dialog box asking the user to confirm installation, after which installation proceeds and the application is launched without further intervention. In cases where the application requires elevated permissions, the dialog box also asks the user to grant permission before the installation can continue. This adds a shortcut icon to the Start menu and lists the application in the Control Panel/Add Remove Programs. Note, it doesn't add anything to the registry, the

desktop, or to Program Files. Note also that the application is installed into the ClickOnce Application Cache (per user).

- Updating the application

When the application developer creates an updated version of the application, he or she also generates a new application manifest and copies files to a deployment location—usually a sibling folder to the original application deployment folder. The administrator updates the deployment manifest to point to the location of the new version of the application. When the user opens the deployment manifest, it is run by the ClickOnce loader and in this way updates the application.

## Add-in Express ClickOnce Solution

Starting from version 3.3, Add-in Express adds a new item to the Build menu in Visual Studio 2005: Publish Add-in Express Project. When you choose this item, Add-in Express shows the Publish dialog that generates the deployment manifest and places it into the Publish subfolder of the solution folder. In addition, the dialog generates the application manifest and places it to the Publish / <AssemblyVersion> folder. Then the dialog copies the add-in files and dependencies (as well as the Add-in Express Loader and its manifest) to the same folder.

One more file copied to the Publish / <AssemblyVersion> folder is called the Add-in Express Launcher for ClickOnce Applications or the Launcher. Its file name is adxlauncher.exe. It is located in the Redistributables folder of Add-in Express setup folder. This file is the heart of the Add-in Express ClickOnce Solution. The Launcher is the application that will be installed on the user's PC. It is listed in the Start menu and Add Remove Programs; it registers and unregisters your add-in, and it provides a form that allows the user to register, unregister, and update your add-in. It also allows the user to switch between two latest versions of your add-in. Overall, the Launcher takes upon itself the task of communicating with ClickOnce API.

### Notes

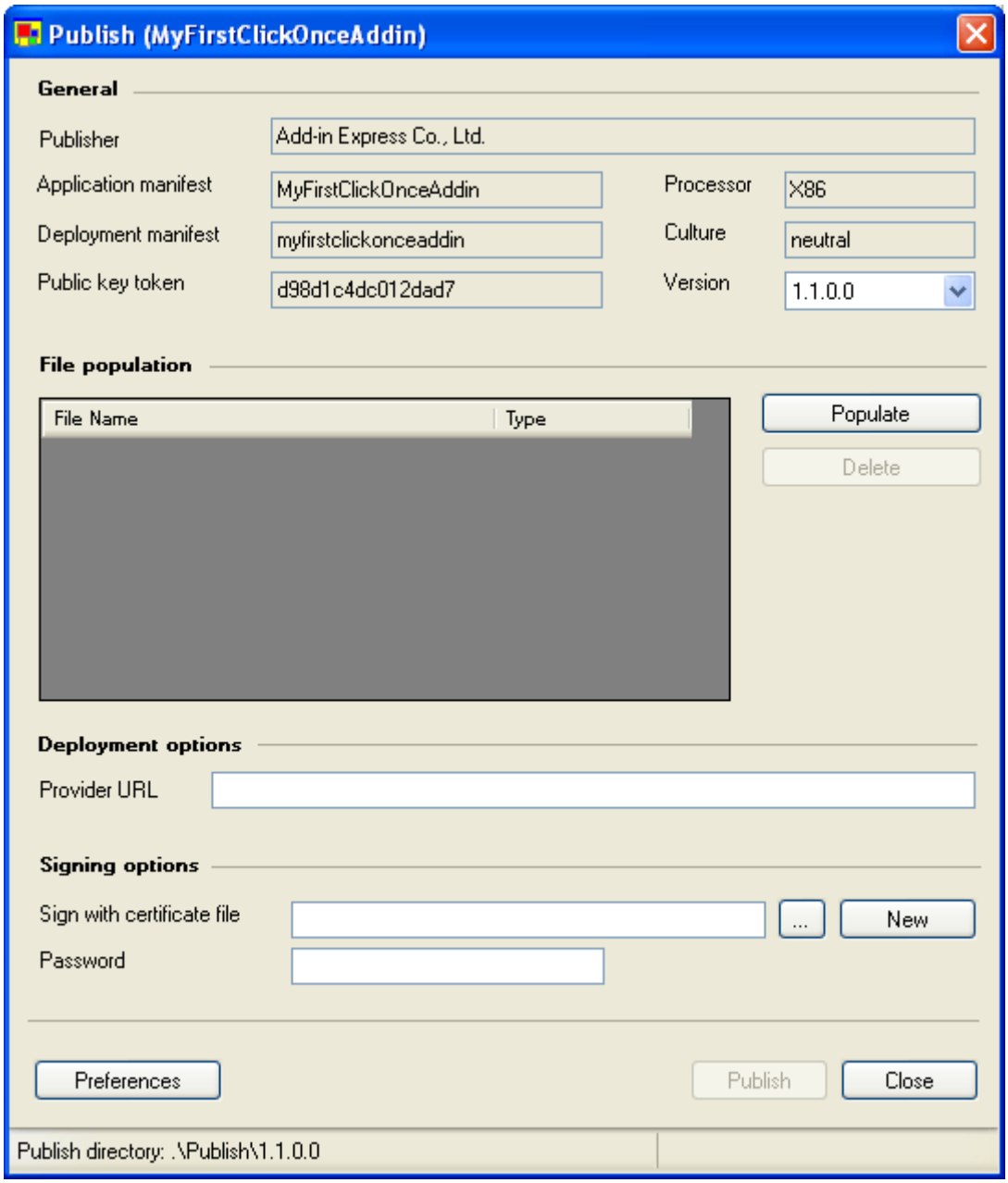
*1. The Launcher (adxlauncher.exe) is located in the Redistributables folder of Add-in Express setup folder. You can check its properties (name, version, etc) in Windows Explorer. Subsequent Add-in Express releases will replace this file with its newer versions. And this may require you to copy a new Launcher version to your Publish / <AssemblyVersion> folder.*

*2. For your convenience, we recommend avoiding using the asterisk in the <AssemblyVersion> tag. Also, you should set the <AssemblyCompany> and <AssemblyProduct> AssemblyInfo tags before publishing the add-in.*

All this will be done when you publish the add-in. However, let's click the Publish Add-in Express Project menu item to see the Publish dialog.

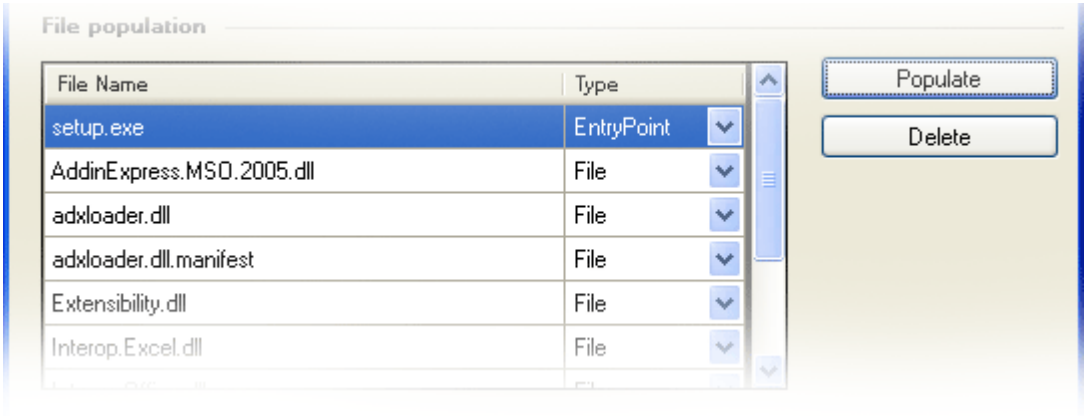
## On the Development PC

The Add-in Express Publish dialog helps you to create application and deployment manifests. In the current release, it shows the following form:




### Step #1 – Populating the Application Manifest

Just click the Populate button. This is the moment when all the above-mentioned folders are created and files are copied.




To set a custom icon for the Launcher, you can add an .ico file and mark it as Icon File in the Type column of the File Population list box.

### How do I add additional files to the application manifest?

*The current release doesn't provide the user interface for adding additional files and/or folders. However, you can copy the files and/or folders required by your add-in to the Publish / <AssemblyVersion> folder and click the Populate button again.*

#### Step #2 – Specifying the Deployment / Update Location

You fill the Provider URL textbox with the URL of the deployment manifest (remember, it is located in the Publish folder). For Web-site based deployment, the format of the URL string is as follows:

```
http://<web-site path>/<deployment manifest name>.application
```

### Case-dependent

*Please note that <deployment manifest name> must be entered in lower case. You can copy it from the Deployment manifest textbox in the Publish dialog window.*

When debugging, you can create a Virtual Directory on your IIS server and bind it to the folder where your deployment manifest is located (the Publish folder is the easiest choice). In this case, the Provider URL could be like this:

```
http://localhost/clickoncetest/myclickonceaddin1.application
```

When releasing a real add-in, the Provider URL must specify the location of the next update for the current add-in version. You can upload version 1.0 of your add-in to any web or LAN location and specify the update location for this version. In subsequent add-in versions, you can use the same or any other update location.

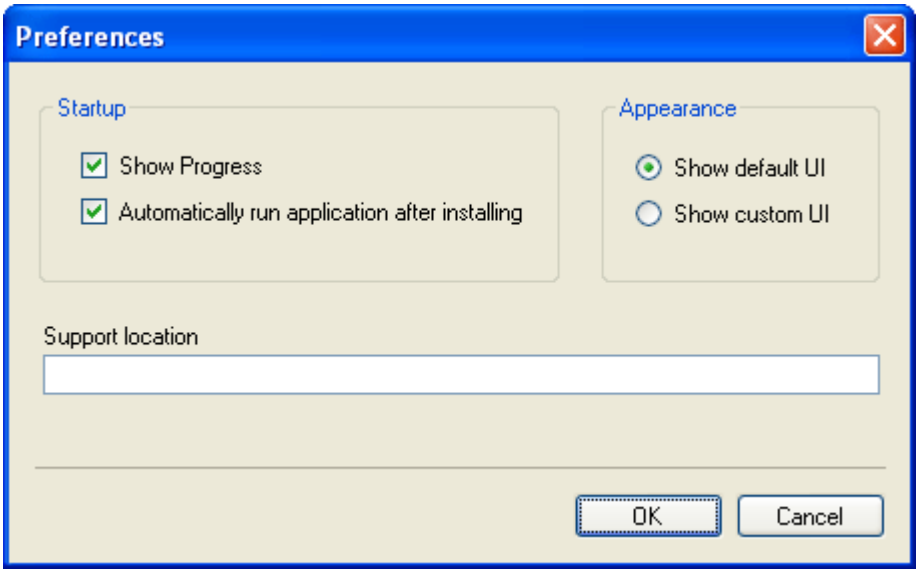
For instance, you can use the same Provider URL in order to look for versions 1.0, 1.1, and 1.2 in one location and, when publishing version 1.3, specify another update location. Please note, that when the user updates the current version, he or she will get the most fresh add-in version existing in the location. That is, it is possible that the user updates from version 1.0 to version 1.3. The opposite is possible , too: this scenario requires the developer to publish v.1.3 and then re-publish v.1.0.

### Step #3 – Signing the Manifests

Browse the existing certificate file or click New to create a new one. Enter the password for the certificate (optional).

### Step #4 – Preferences

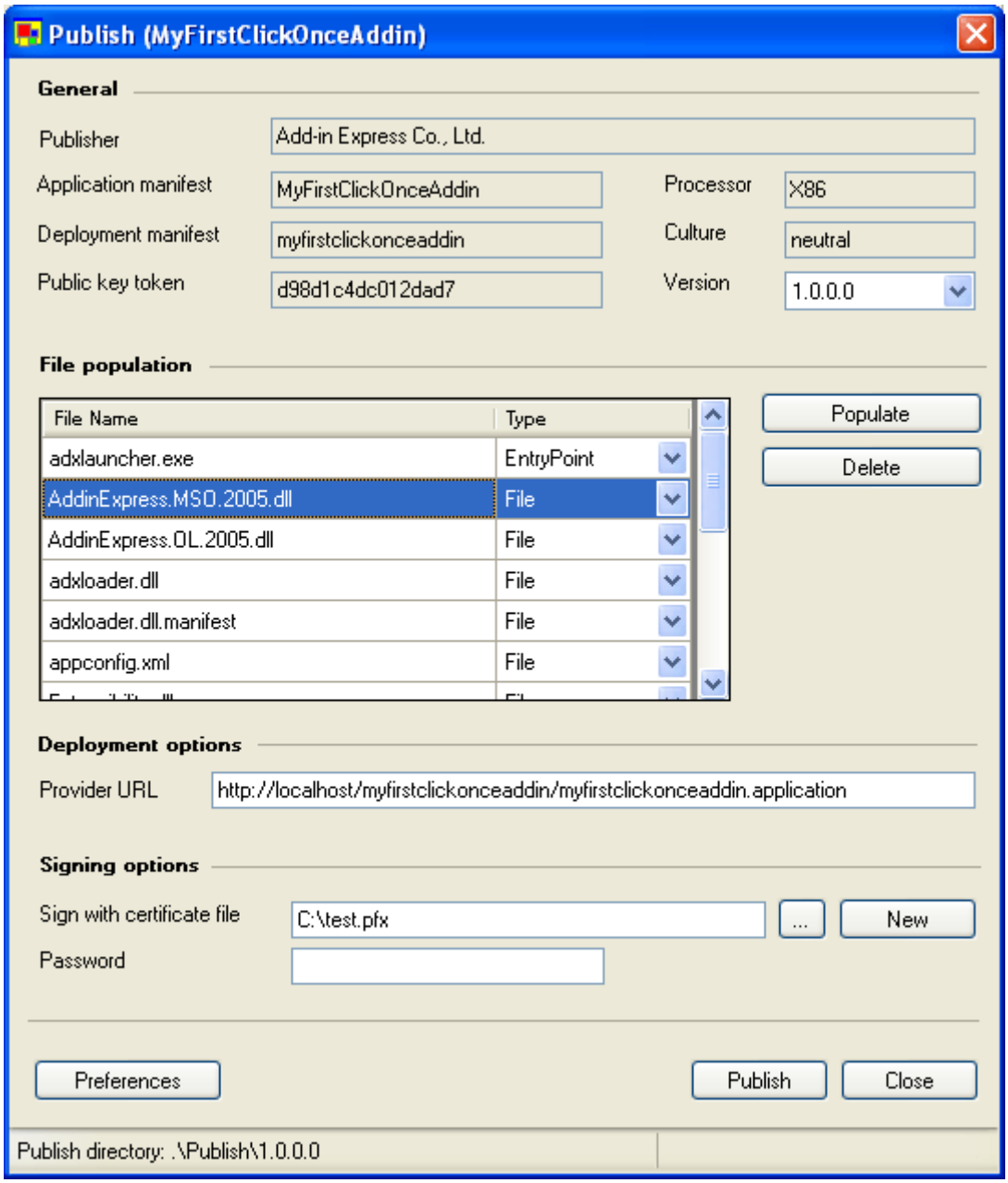
Click the Preferences button to open the following dialog window:




- Show Progress – allows the user to see that something is going on.
- Automatically run application after installing – this option is a standard one for ClickOnce applications. However, for add-in developers this provides a rather hard choice. If you set this option on, the Add-in Express Launcher will be run when the user install the add-in. If you clear this check box, the user will get the following message when restoring a previous version of the add-in:
- If you choose the Show Custom UI option, then the ClickOnce module will get the OnShowCustomUI event. This allows you to show a custom form in this moment.

### Step #5 – Publishing the Add-in

Now you are ready to click the Publish button.




When you click it, Add-in Express generates (updates) the manifests.

### Manifest file names and locations

*Deployment manifest – <SolutionFolder>/Publish/<projectname>.application*

*Application manifest - <SolutionFolder>/Publish/<ProjectVersion>/<ProjectName>.exe.manifest*

Now you are able to copy files and folders of the Publish folder to a deployment location, say a web server. For testing purposes, you can just double-click the deployment manifest in Windows Explorer.

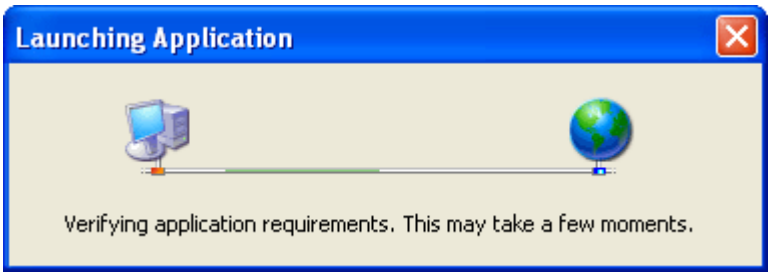
#### Step #6 – Publishing a New Add-in Version

In AssemblyInfo, change the version number and build the project. Click Publish and add the add-in files (button Populate). Fill in all the other fields. You can use the Version check box to switch to the data associated with any previous version.

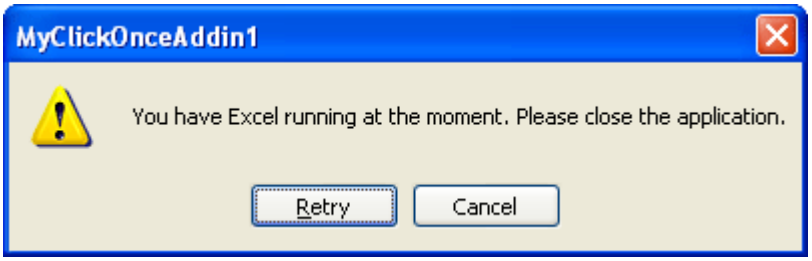
## On the Target PC

#### Installing: User Perspective

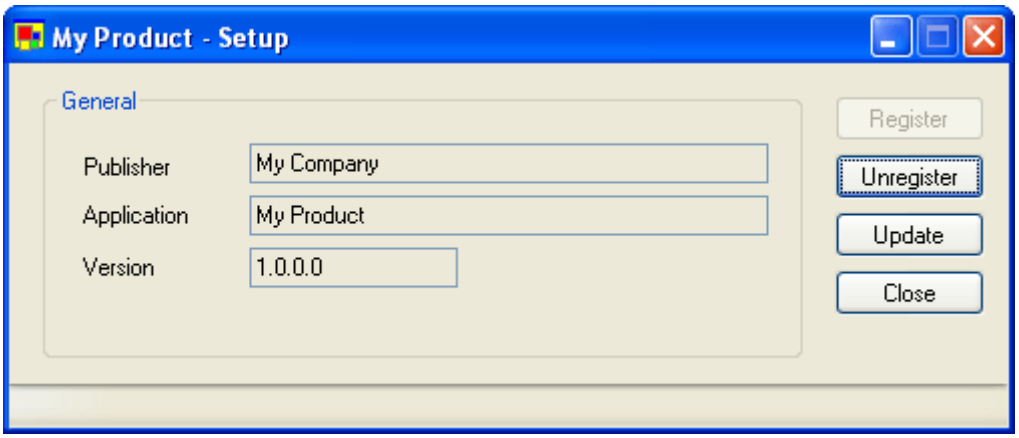
The user browses the deployment manifest (<projectname>.application) in either Internet Explorer or Windows Explorer and runs it. The following window is shown:




In accordance with the manifests, the ClickOnce loader will download the files to the ClickOnce cache and run the Launcher application. When run in this mode, it registers the add-in. If the add-in's host applications are running at this moment, the user will be prompted to close them.




If the user clicks Cancel, the Launcher will be installed, but the add-in will not be registered. However, in any appropriate moment, the user can click the Launcher entry in the Start menu to run the Launcher and register/unregister the add-in through the Launcher GUI.




**Note**

*The current Add-in Express version relies on the name and location of the product entry in the Start Menu. Please, add this information to your user's guide.*

**Installing: Developer Perspective**

If a ClickOnce module is added to your add-in project, you are able to handle all the actions applicable to add-ins: install, uninstall, register, unregister, update to a newer version, and revert to the previous version. For instance, you can easily imagine a form or wizard allowing the user to tune up the settings of your add-in. The ClickOnce module also allows you to show a custom GUI whenever the Launcher Application is required to show its GUI. Please note that if you don't process the corresponding event, the standard GUI of the Add-in Express ClickOnce application will be shown.

You can also make use of the ComRegisterFunction and ComUnRegisterFunction attributes in any assembly listed in the loader manifest (see assemblyIdentity tags). The methods marked with the ComRegisterFunction attribute will run when the add-in is registered. See MSDN for the description of the attributes.

**Updating: User Perspective**

The user can check for add-in updates in the Launcher GUI (or in the GUI you supply). To run it, the user clicks the entry in the Start Menu. If there is no update in the update location specified in the deployment manifest, an information message box is shown. If there is an update, the Launcher requests the user to confirm his/her choice. If the answer is positive, the ClickOnce Loader downloads new and updated files to the ClickOnce Cache, the Launcher unregisters the current add-in version, restarts itself (this will run the Launcher application supplied in the update files), and registers the add-in.

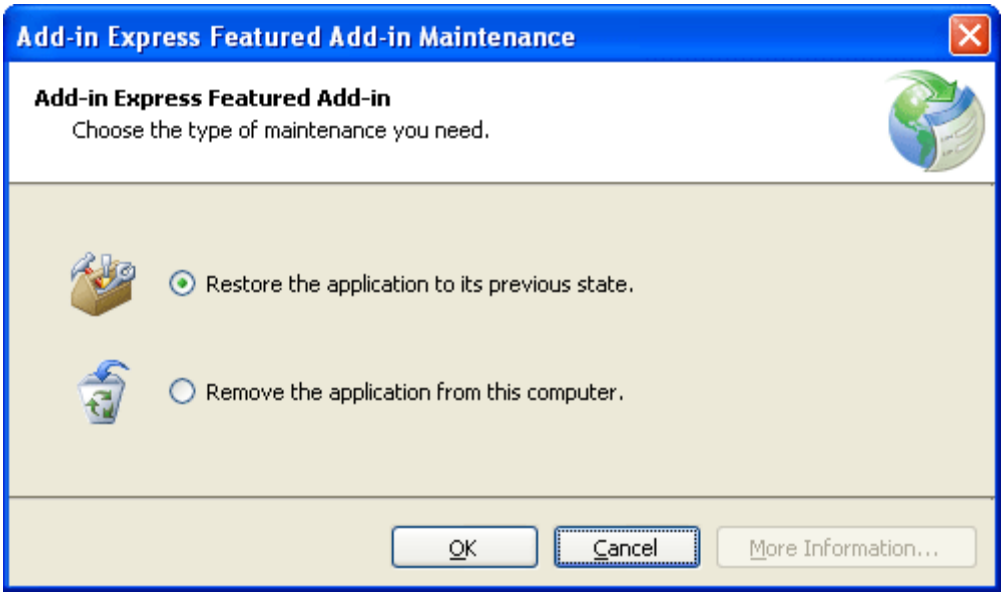
**Updating: Developer Perspective**

The add-in module provides you with the CheckForUpdates method. This method can result in one of the following ways:

- the add-in becomes updated;
- the ClickOnce module invokes the OnError event handler.

**Uninstalling: User Perspective**

To uninstall the add-in, the user goes to Add Remove Programs and clicks on the product name entry. This opens the following dialog.




- **Restore the application to its previous state.**

This option is disabled, if the add-in was never updated. If the user choose this option, the Launcher will run, require the user to close the host applications of your add-in, unregister the add-in, requests ClickOnce API to start the Launcher application of the previous add-in version, and quits. Then the Launcher application of the previous add-in version registers the add-in.

- **Remove the application from this computer**

This runs the Launcher that will require the user to close the host applications of your add-in. Then the Launcher will unregister the add-in and request ClickOnce API to delete both the add-in and the Launcher files.

**Uninstalling: Developer Perspective**

Handle the corresponding event of the ClickOnce module or use the ComUnRegisterFunction attribute to run your actions when the add-in is unregistered.

## Restrictions of Add-in Express ClickOnce Solution

- With the Add-in Express ClickOnce Solution, you can deploy only Add-in Express add-ins based on .NET Framework 2.0 and later. Add-ins developed with .NET Framework 1.1 (Visual Studio 2003) will never be able to use this feature.
- In the Web-based deployment scenario, the user can install such add-ins using Internet Explorer only. The ClickOnce article from Wikipedia states that FireFox allows ClickOnce-based installations too, but this was neither tested nor even verified.
- ClickOnce deployment is supported if your add-in is shimmed with either Add-in Express Loader or VSTO Loader.

# Creating Setup Projects Manually

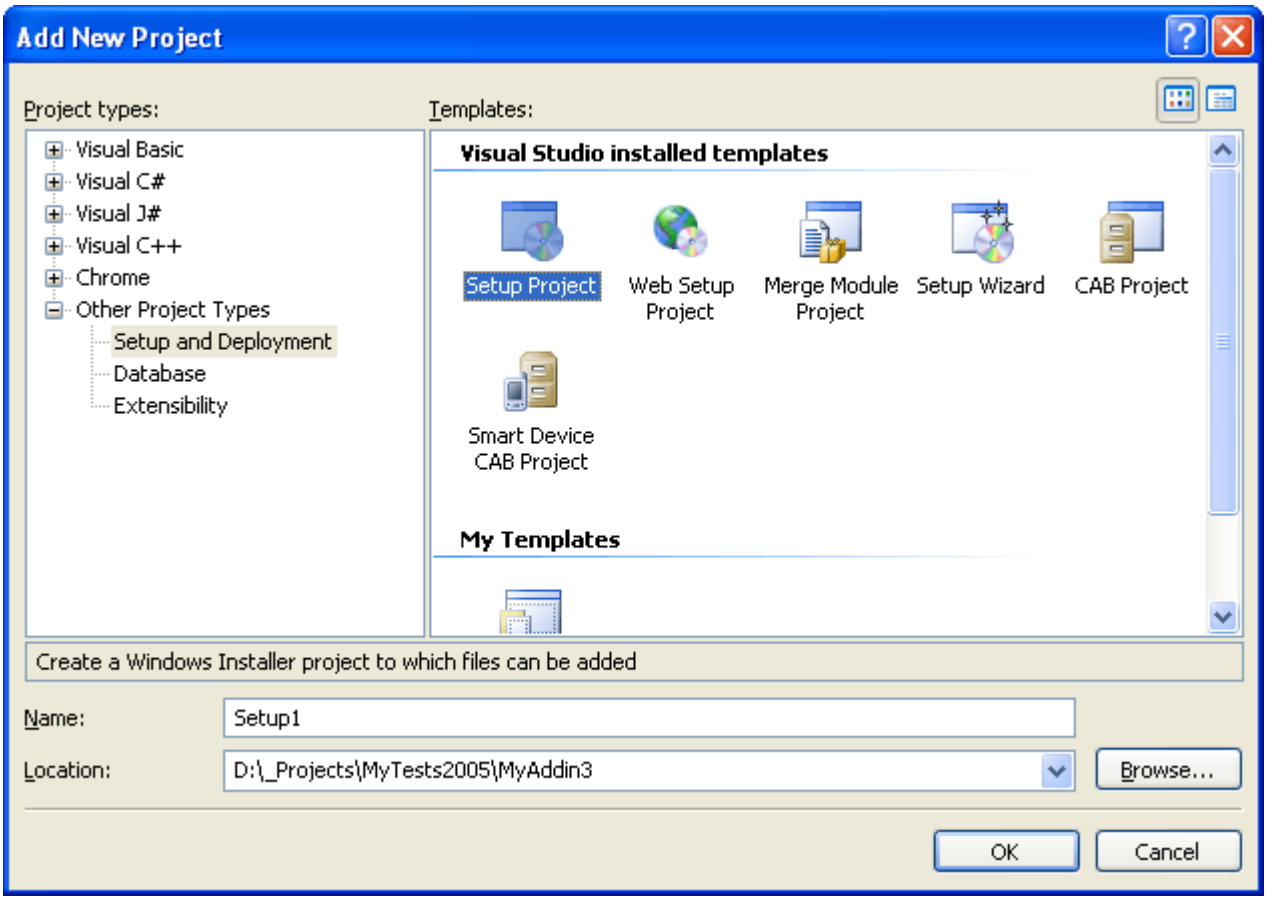
We don't recommend that you create setup project manually because you always can create them with the Add-in Express Project Wizard when starting a new project. Nevertheless, if you need to create a setup project manually, use the following step-by-step instructions.

## Add-in Express Loader

To create the setup project manually, please follow the steps below.

### Add a New Setup Project

Right-click the solution item and choose Add | New Project.



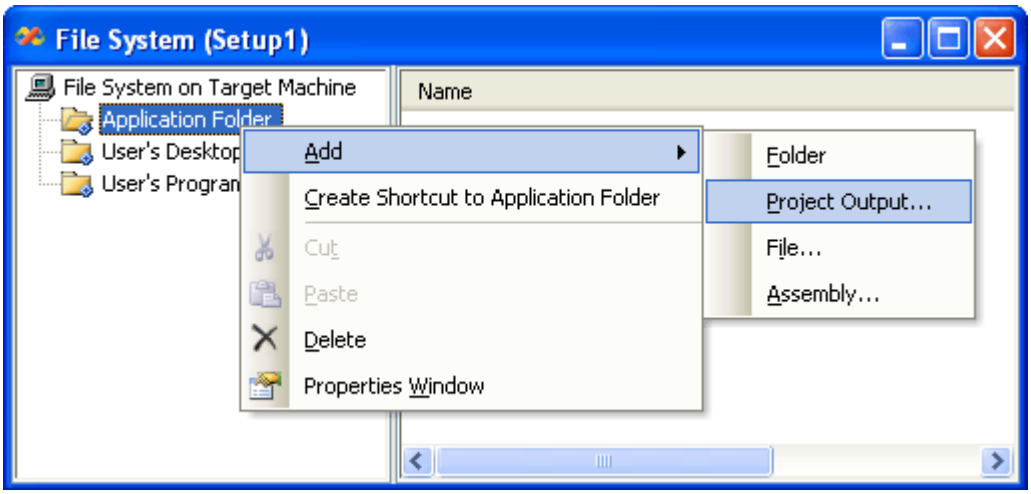

In the Add New Project dialog, select the Setup Project item and click OK. This will add the setup project to your solution.

### File System Editor

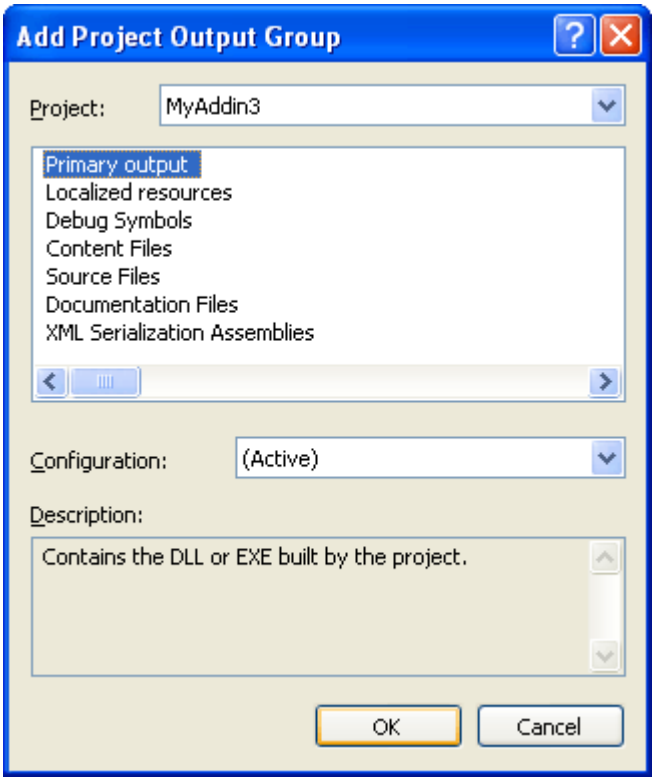
Right-click the setup project item (Setup1 in the screenshot) and choose View | File System.

**Primary Output**

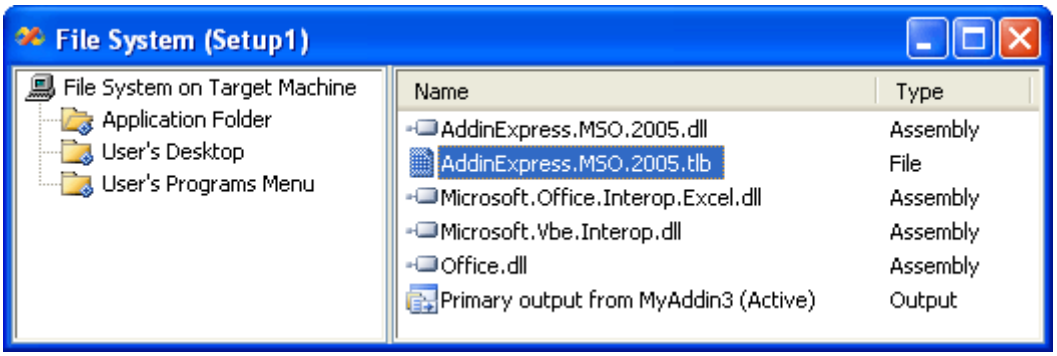
Right-click the Application Folder item and choose Add | Project Output

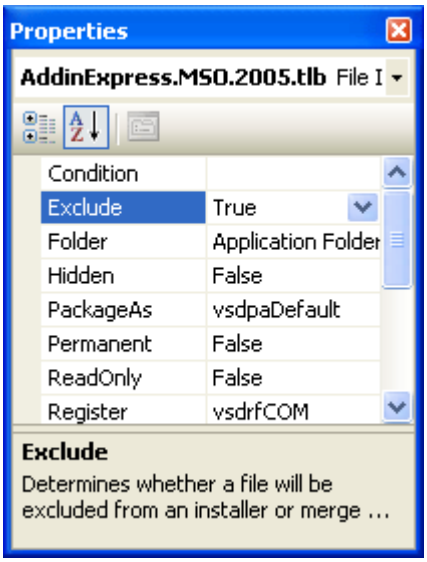



In the Add Project Output Group dialog, select the Primary Output Item of your Add-in/RTD Server/Smart Tag project and click OK.




This adds the following entries to the Application Folder of your setup project.



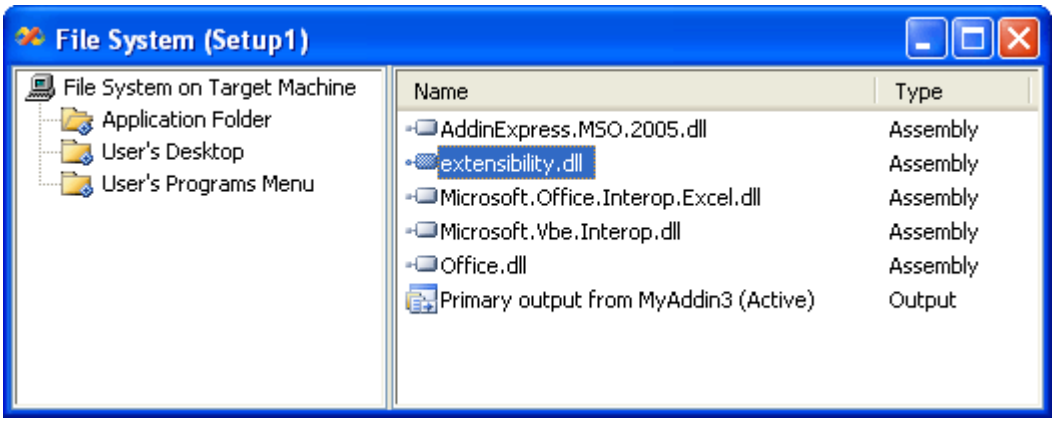

Select AddinExpress.MSO.2005.tlb (or AddinExpress.MSO.2003.tlb in Visual Studio 2003) and, in the Properties window, set the Exclude property to True. If you use version-neutral PIAs, please exclude the VB6EXT.OLB file in the same way.

### Exclude all TLBs and OLBs

*Always exclude all TLB and OLB files from the setup project except for TLBs that you create yourself.*

### Extensibility.dll

Add the Extensibility.dll assembly to the Application Folder if it doesn't exist in the Detected Dependencies section of the setup project.

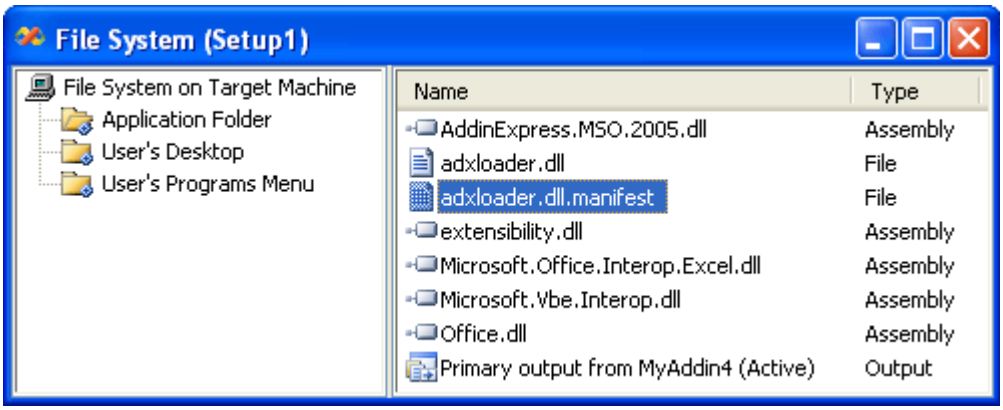



### Project-depended Resources

Now you add all resources (e.g. assemblies, dlls or any resources) required for your project.

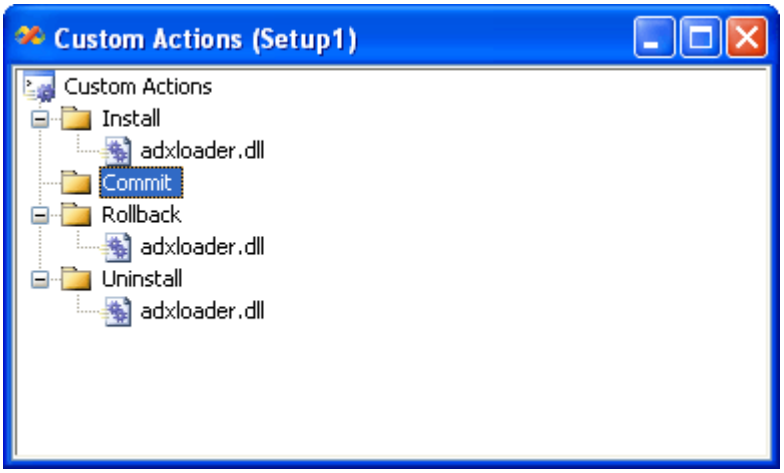
### Add-in Express Loader and Manifest

Add the adxloader.dll and adxloader.dll.manifest files from the 'Loader' subfolder of the add-in project directory to the 'Application Folder' of the setup project.

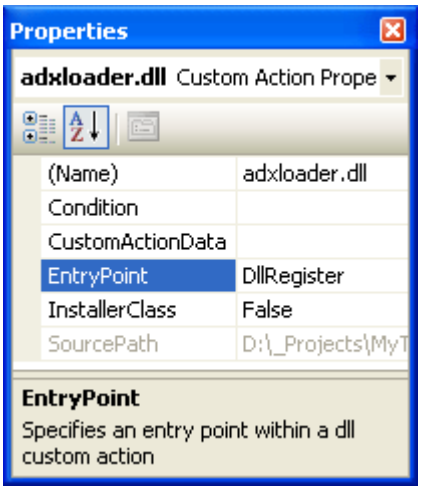



### Custom Actions

Open the Custom Actions editor and add a new action to the Install, Rollback, Uninstall sections. Use the adxloader.dll file as an item for the custom actions.




### EntryPoint




Add the following parameter to the EntryPoint property of the following custom actions:

- **Install**

```
DllRegister
```

- **Rollback**

```
DllUnregister
```

- **Uninstall**

```
DllUnregister
```

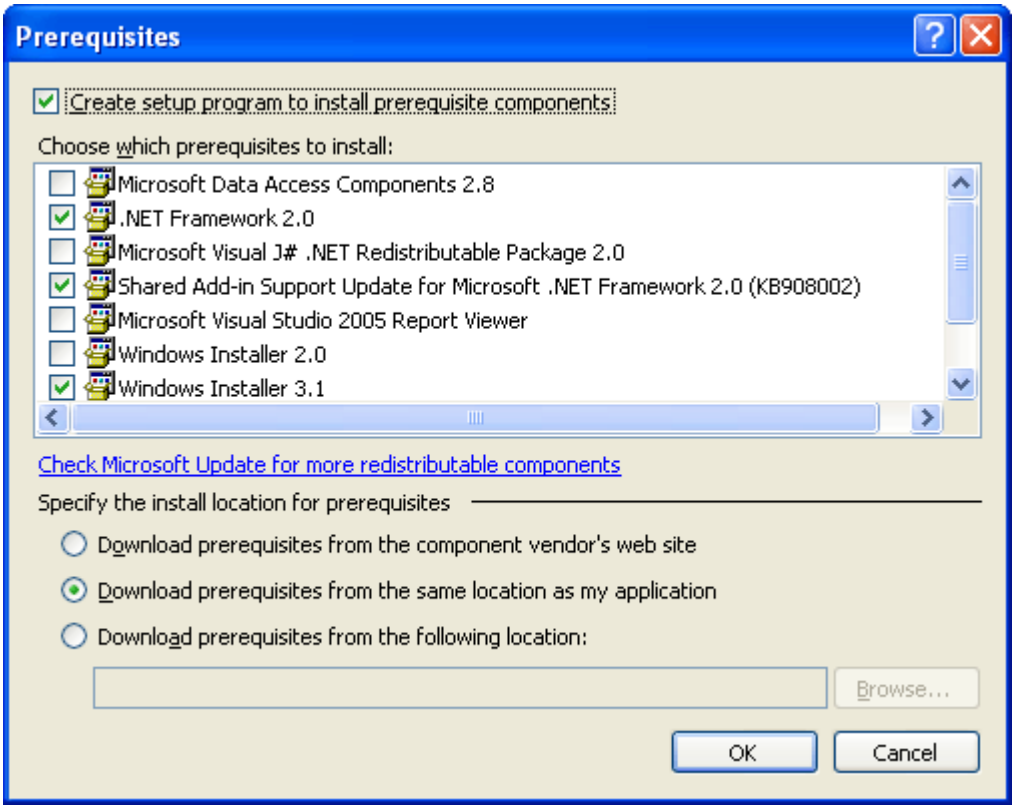
### Dependencies

Right click on the Detected Dependencies section of the setup project and choose the Refresh Dependencies option. Also, exclude all dependencies that are not required for your setup.

**VS 2005 only**

Right click on the setup project and open the Properties dialog.

Click on the Prerequisites button and, in the Prerequisites dialog, check all prerequisites you need.




You can choose the 'Download prerequisites from the same location as my application' option to distribute all prerequisites with the add-in installation package.

However, note that if you include them to the setup, a non-admin on Vista will get the elevation dialog and this can end with installing the add-in to the admin profile. In such a case, the add-in will not be available for the standard user.

**Deploy**

Rebuild the setup project. You may want to run one of the following command lines:

```
%AddinExpressInstallFodler%\Bin\DisableUAC.exe %BuiltOuputPath% /UAC=On
%AddinExpressInstallFodler%\Bin\DisableUAC.exe %BuiltOuputPath% /UAC=Off
```

See Project Wizard Options for more details.

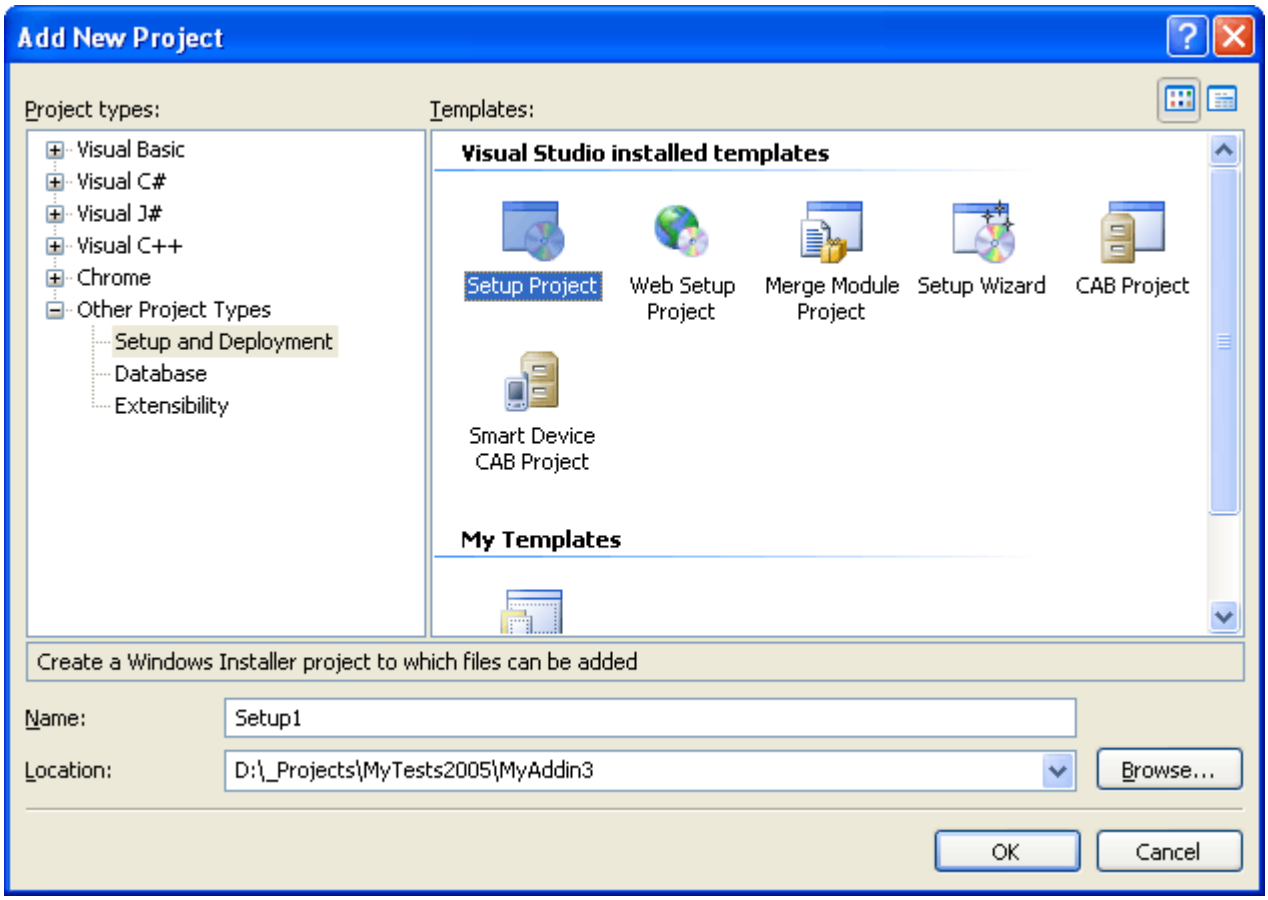
Copy all setup files to the target PC and run the msi file to install the add-in. However, to install Prerequisites, you will need to run setup.exe.

## VSTO Loader

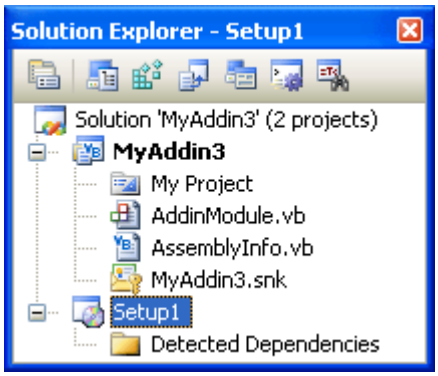
To create the setup project manually, please follow the steps below.

### Add a New Setup Project

Right-click the solution item and choose Add | New Project.




In the Add New Project dialog, select the Setup Project item and click OK. This will add the setup project to your solution.

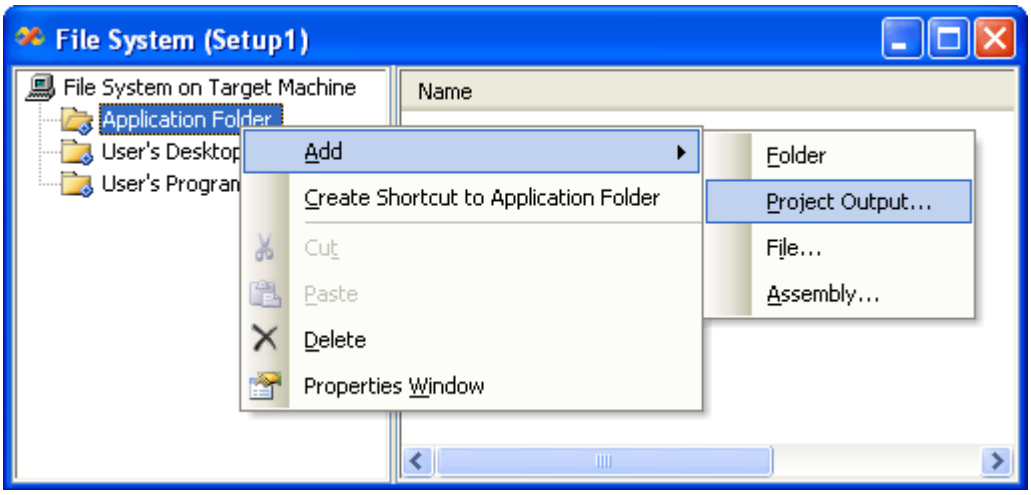


### File System Editor

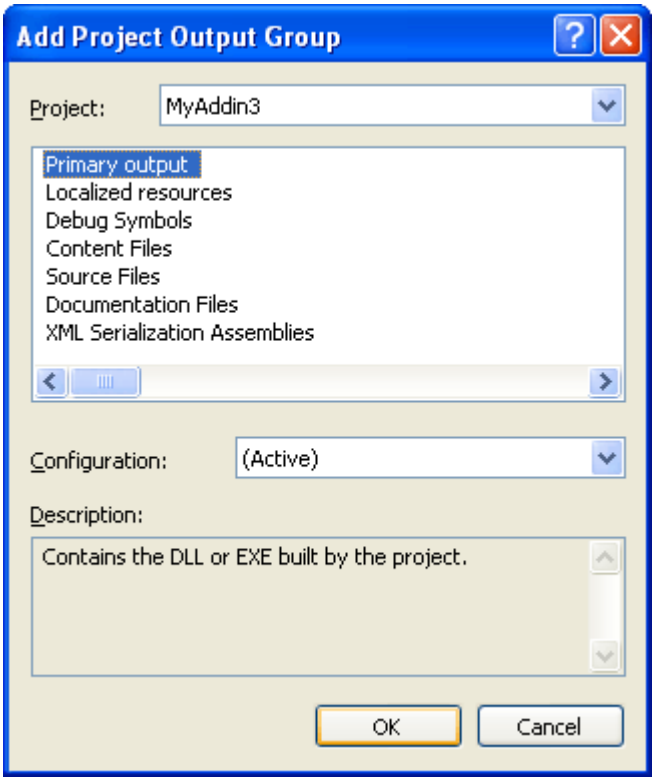
Right-click the setup project item (Setup1 in the screenshot) and choose View | File System.

### Primary Output

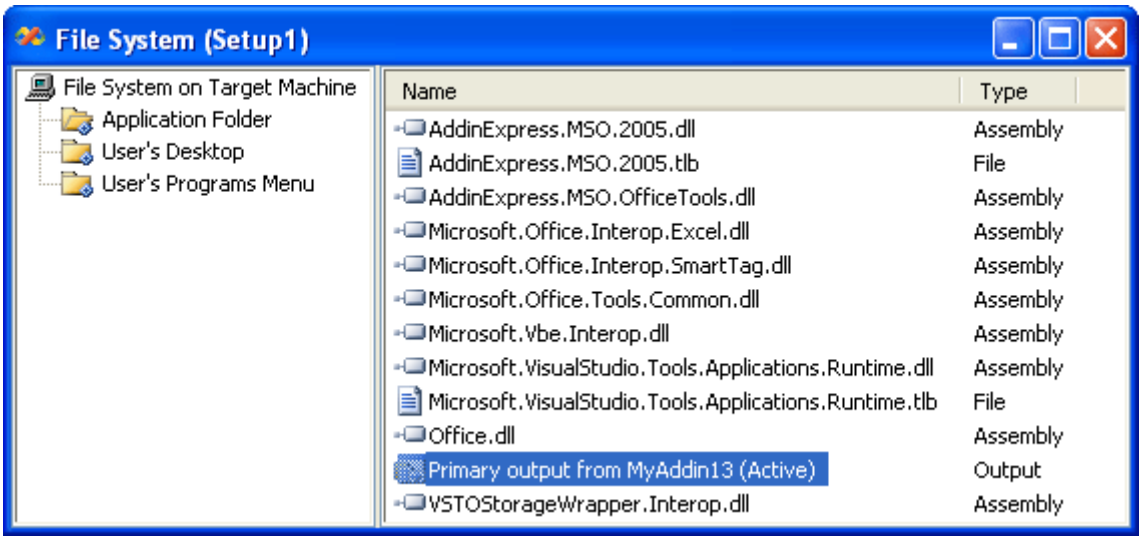
Right-click the Application Folder item and choose Add | Project Output

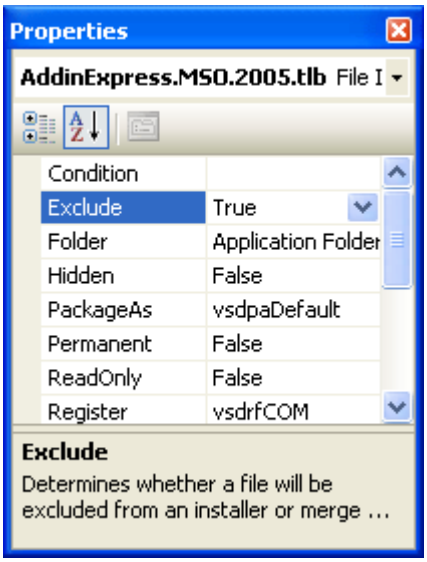



In the Add Project Output Group dialog, select the Primary Output Item of your Add-in/RTD Server/Smart Tag project and click OK.



This adds the following entries to the Application Folder of your setup project.




Select AddinExpress.MSO.2005.tlb and, in the Properties window, set the Exclude property to True. In the same way, you exclude the following files:

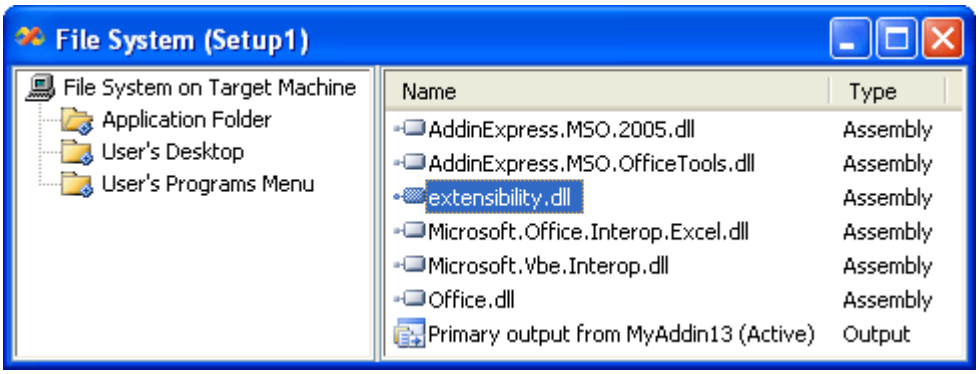
- Microsoft.VisualStudio.Tools.Applications.Runtime.dll
- Microsoft.VisualStudio.Tools.Applications.Runtime.tlb
- VSTOStorageWrapper.Interop.dll
- Microsoft.Office.Interop.SmartTag
- Microsoft.Office.Tools.Common.dll

### Exclude all TLBs

*Always exclude all TLB files from the setup project except for TLBs that you create yourself.*

### Extensibility.dll

Add the Extensibility.dll assembly to the Application Folder if it doesn't exist in the Detected Dependencies section of the setup project.

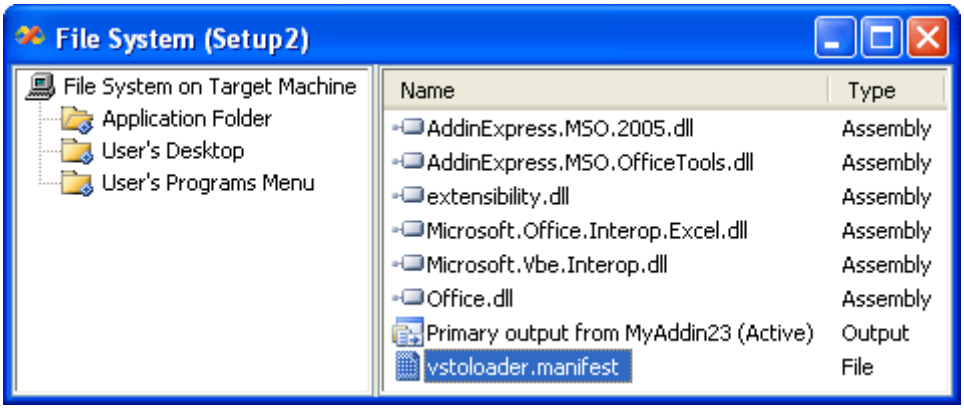



**Project-depended Resources**

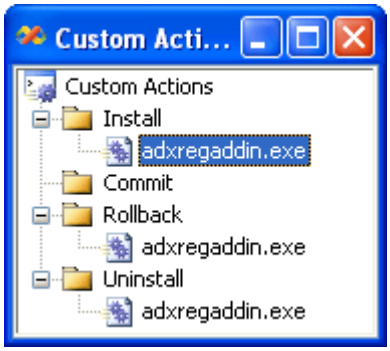
Now you add all resources (e.g. assemblies, dlls or any resources) required for your project.

**VSTO Loader Manifest**

Add the <projectname>.manifest located in the Loader folder of your add-in project to the 'Application Folder' of the setup project.




**Custom Actions**




Add the adxregaddin.dll file located in the Redistributables folder of the Add-in Express install folder to the 'Application Folder' of the setup project.

Open the Custom Actions editor and add a new action to the Install, Rollback, and Uninstall sections. Use the adxregaddin.dll file as an item for the custom actions.

Add the following strings to the Arguments property of the following custom actions:

- **Install**

```
/install="[TARGETDIR]\vstoloader.manifest" /displayerrors=1
```

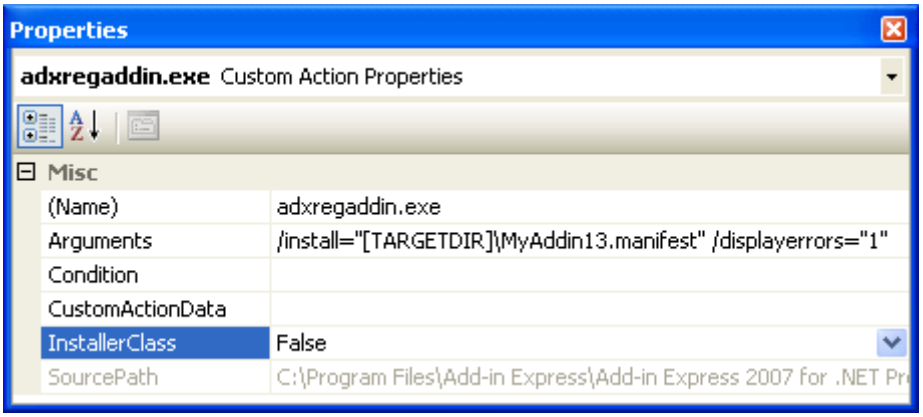
- **Rollback**

```
/uninstall="[TARGETDIR]\vstoloader.manifest"
```

- **Uninstall**

```
/uninstall="[TARGETDIR]\vstoloader.manifest"
```

Please, don't forget to set the InstallerClass property of the custom actions to **False**.



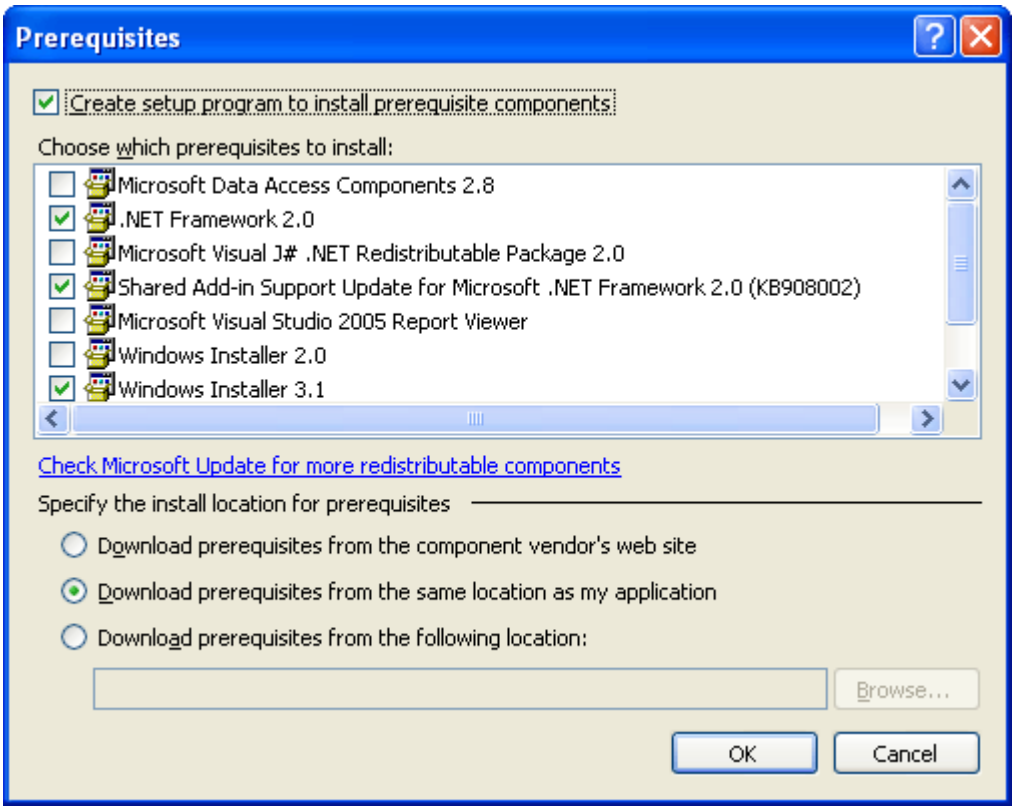

### Dependencies

Right click on the Detected Dependencies section of the setup project and choose the Refresh Dependencies option. Also, exclude all dependencies that are not required for your setup.

### Prerequisites

Right click on the setup project and open the Properties dialog.

Click on the Prerequisites button and, in the Prerequisites dialog, check all prerequisites you need.




You can choose the 'Download prerequisites from the same location as my application' option to distribute all prerequisites with the add-in installation package.

However, note that if you include them to the setup, a non-admin on Vista will get the elevation dialog and this can end with installing the add-in to the admin profile. In such a case, the add-in will not be available for the standard user.

**Deploy**

Rebuild the setup project. You may want to run one of the following command lines:

```
%AddinExpressInstallFodler%\Bin\DisableUAC.exe %BuiltOuputPath% /UAC=On
%AddinExpressInstallFodler%\Bin\DisableUAC.exe %BuiltOuputPath% /UAC=Off
```

See Project Wizard Options for more details.

Copy all setup files to the target PC and run the msi file to install the add-in. However, to install Prerequisites, you will need to run setup.exe.

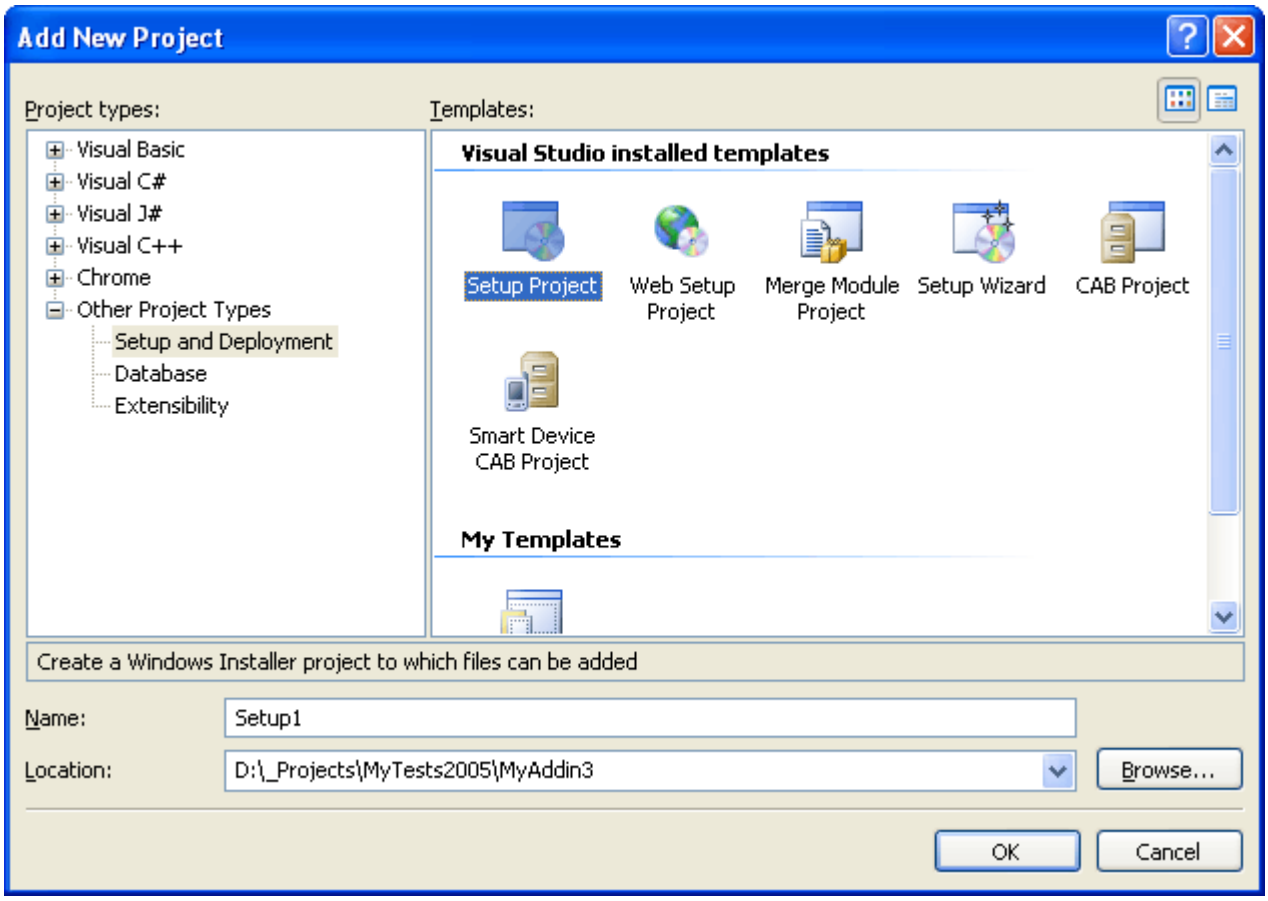
## MSCOREE.DLL

If you choose the Create the Setup Project option of the Add-in Express Project Wizard, the wizard will create the setup project automatically.

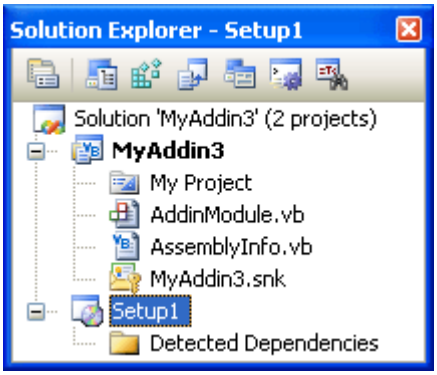
To create setup project manually, please follow the steps below.

### Add a New Setup Project

Right-click the solution item and choose Add | New Project.




In the Add New Project dialog, select the Setup Project item and click OK. This will add the setup project to your solution.

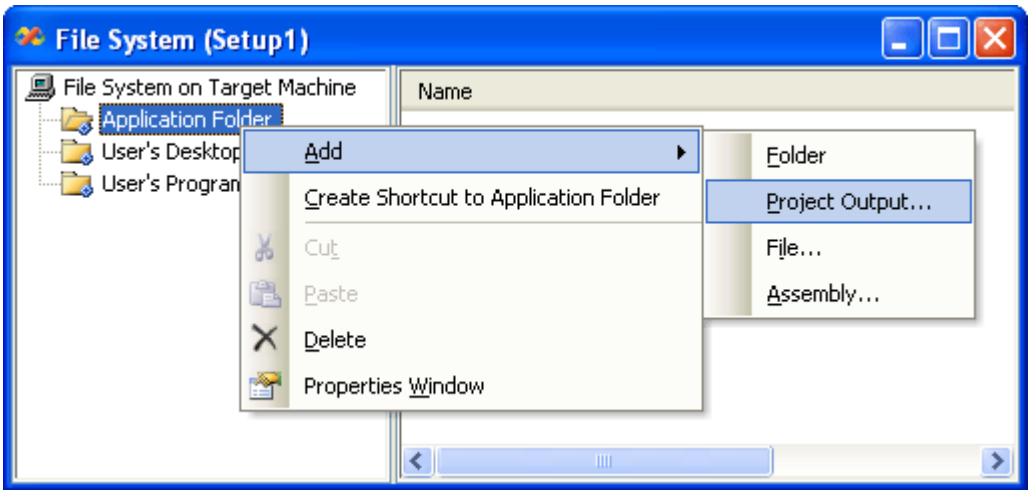


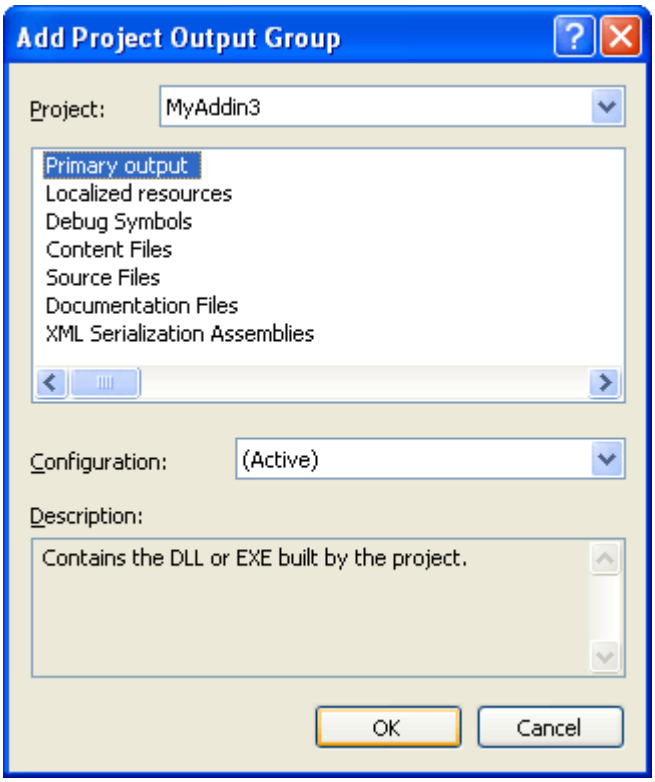

### File System Editor

Right-click the setup project item (Setup1 in the screenshot) and choose View | File System.

### Primary Output

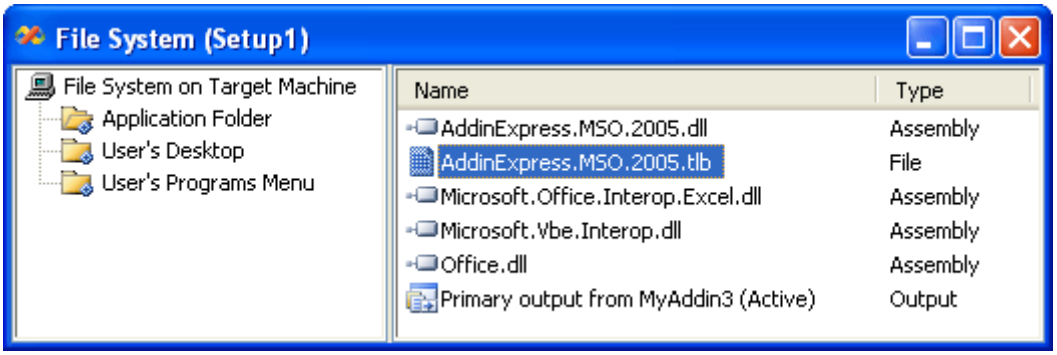
Right-click the Application Folder item and choose Add | Project Output



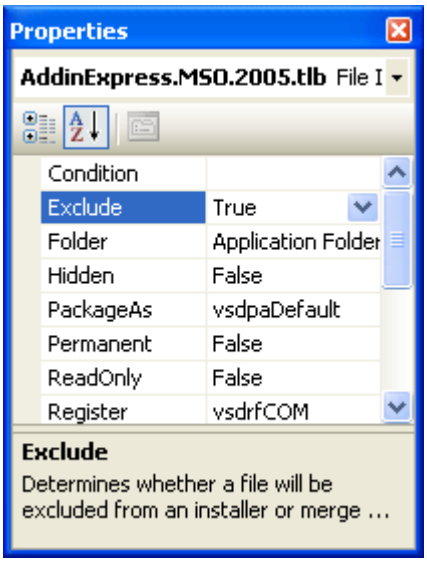




In the Add Project Output Group dialog, select the Primary Output Item of your Add-in/RTD Server/Smart Tag project and click OK.

This adds the following entries to the Application Folder of your setup project.



Select AddinExpress.MSO.2005.tlb (or AddinExpress.MSO.2003.tlb in Visual Studio 2003) and, in the Properties window, set the Exclude property to True. If you use version-neutral PIAs, please exclude the VB6EXT.OLB file in the same way.

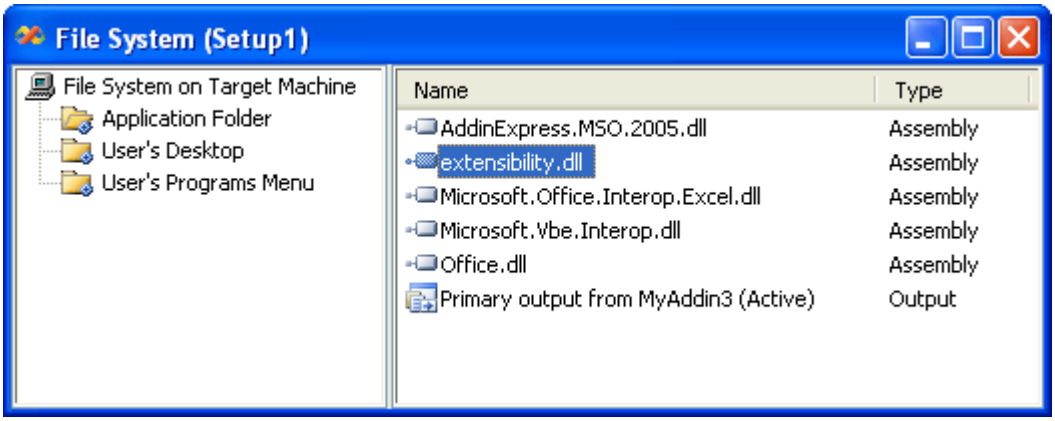



**Exclude all TLBs**

*Please exclude all TLB files from the setup project except for TLBs that you create yourself.*

**Extensibility.dll**

Add the Extensibility.dll assembly to the Application Folder if it doesn't exist in the Detected Dependencies section of the setup project

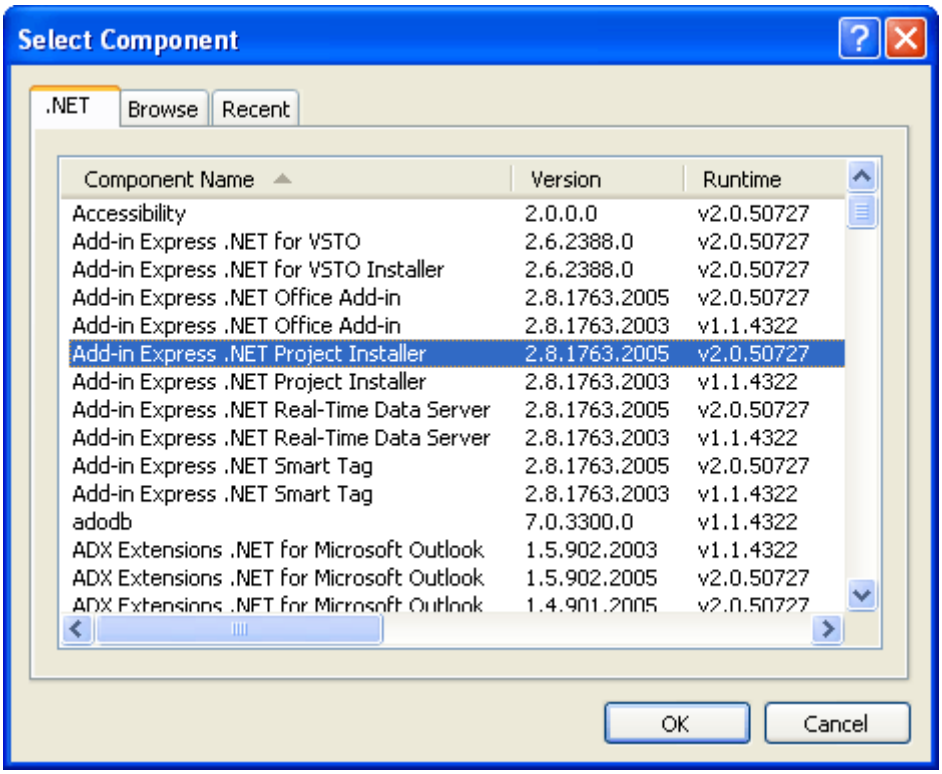


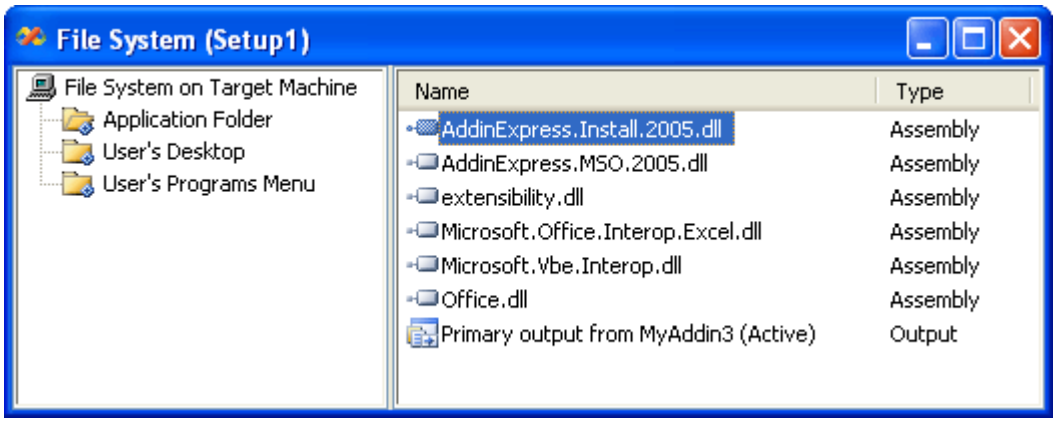

**Project-depended Resources**

Now you add all resources (e.g. assemblies, dlls or any resources) required for your project.

**AddinExpress.Install.2005.dll / AddinExpress.Install.dll**

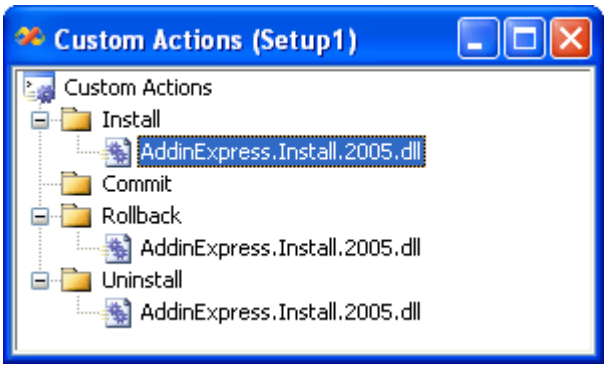
Add the AddinExpress.Install2005.dll / AddinExpress.Install.dll (for VS 2003) assembly to the Application Folder.




**Custom Actions**

Open the Custom Actions editor and add a new action to the Install, Rollback, Uninstall sections. Use the AddinExpress.Install.dll / AddinExpress.Install.2005.dll (for VS 2005) assembly as an item for the custom actions.




**CustomActionData**

Add the following parameter to the CustomActionData property of all the custom actions mentioned above depending on your project type.

- **For COM Add-ins**

```
/Addin="[TARGETDIR]\<the add-in assembly name>.dll"
```

- **For RTD Servers**

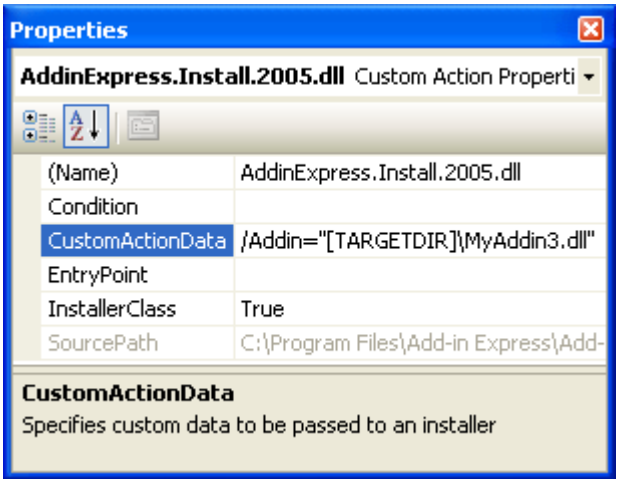
```
/RTD="[TARGETDIR]\<the RTD server assembly name>.dll"
```

- **For Smart Tags**

```
/SmartTag="[TARGETDIR]\<the smart tag assembly name>.dll"
```

- **For Excel Automation Add-ins**

```
/Addin="[TARGETDIR]\<the add-in assembly name>.dll"
```




**Dependencies**

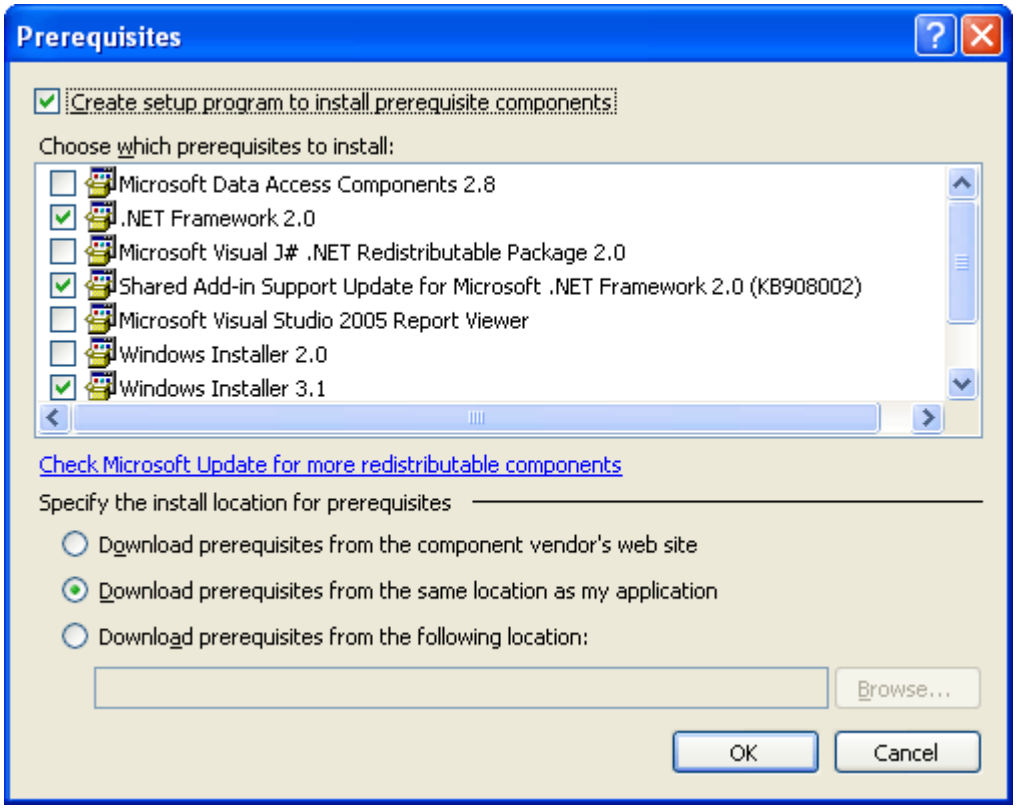
Right click on the Detected Dependencies section of the setup project and choose the Refresh Dependencies option. Also, exclude all dependencies that are not required for your setup.

**VS 2005 only**

Right click on the setup project and open the Properties dialog.

Click on the Prerequisites button and, in the Prerequisites dialog, check all prerequisites you need.

You can choose the 'Download prerequisites from the same location as my application' option to distribute all prerequisites with the add-in installation package.




You can choose the 'Download prerequisites from the same location as my application' option to distribute all prerequisites with the add-in installation package.

However, note that if you include them to the setup, a non-admin on Vista will get the elevation dialog and this can end with installing the add-in to the admin profile. In such a case, the add-in will not be available for the standard user.

**Deploy**

Rebuild the setup project. You may want to run one of the following command lines:

```
%AddinExpressInstallFodler%\Bin\DisableUAC.exe %BuiltOuputPath% /UAC=On
%AddinExpressInstallFodler%\Bin\DisableUAC.exe %BuiltOuputPath% /UAC=Off
```

See Project Wizard Options for more details.

Copy all setup files to the target PC and run the msi file to install the add-in. However, to install Prerequisites, you will need to run setup.exe.

# Add-in Express Tips and Notes

You might have an impression that creating add-ins is a very simple task. Please don't get too enthusiastic. 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 intricate than a single call of MessageBox.

## Common stuff

### 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.

### Project Wizard Options

The Isolation group box in the Add-in Express project wizard allows choosing the shim to be used with your add-in. To understand shims, see Deploying Add-in Express Projects.

Check the "Configure the setup project for the standard user" checkbox, for your add-in to be installed by a non-admin user – it disables the "Everyone / Just Me" choice in the setup UI and targets the DefaultLocation property of the setup project to [AppDataFolder] instead of [ProgramFilesFolder]. If you check the "Enable UAC for the setup on Vista" checkbox, the setup will require administrative privileges to be asked, when run by a non-admin user on Vista. For VS2003 developers, these features are available with the following command lines:

```
%AddinExpressInstallFodler%\Bin\DisableUAC.exe %BuiltOuputPath% /UAC=On
%AddinExpressInstallFodler%\Bin\DisableUAC.exe %BuiltOuputPath% /UAC=Off
```

They emulate checking the "Configure for the standard user" and checking/clearing the "Enable UAC" checkboxes. % BuiltOuputPath% specifies the path and file name of the MSI.

"Add the Primary Interop Assemblies to the Project" adds currently installed PIAs (see What are PIAs?) to the project. Combined with "Use Version-neutral Office PIAs", it adds the PIAs supplied with Add-in Express installation package (see Why Version-Neutral PIAs?). If you sign your add-in, you need to sign the interops: this is what the "Sign ... with a strong Name" check box is designed to do.

## How Do I Find the Source Code of This Sample?

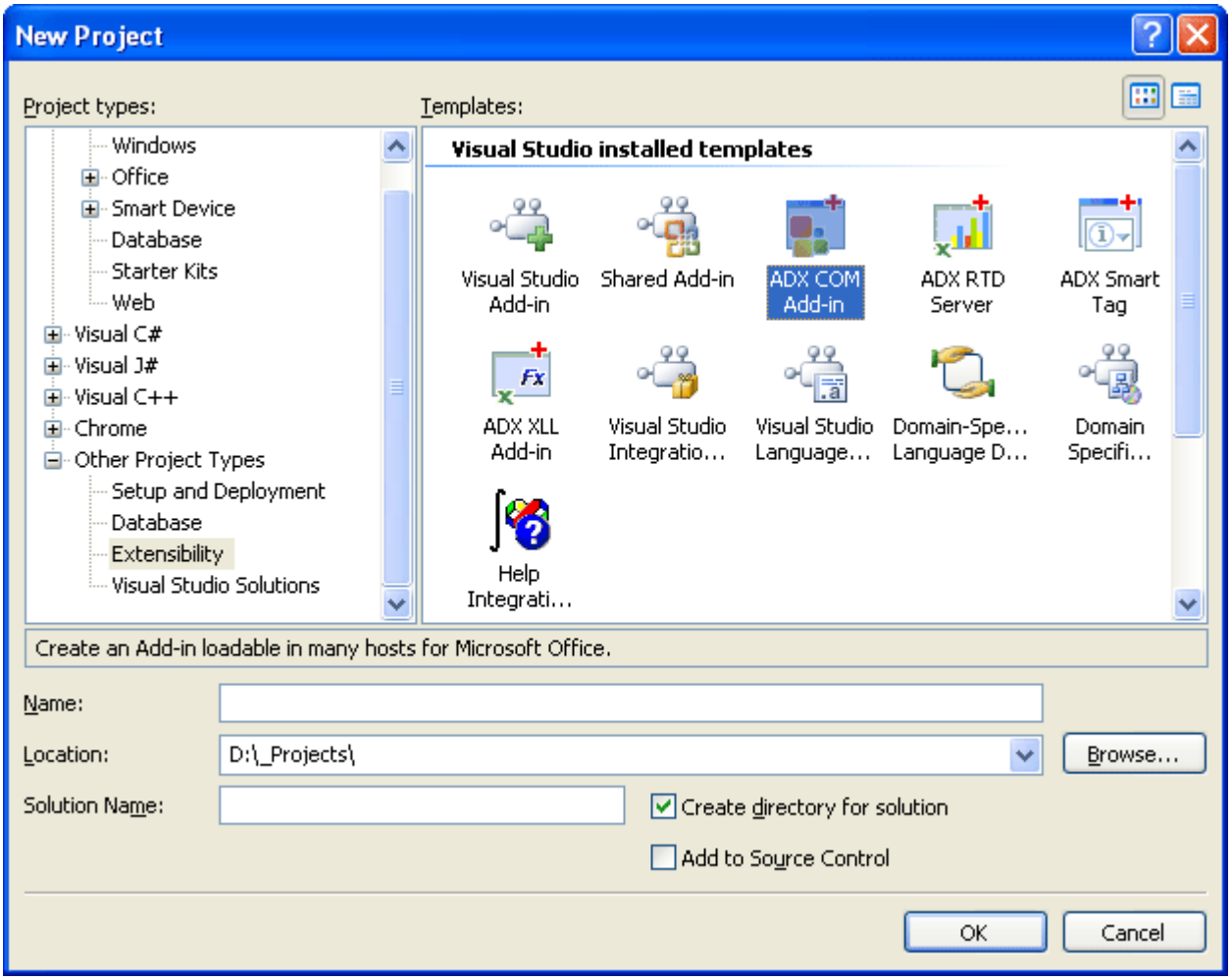
You find all the samples described in this document in the Docs / Samples folder of the Add-in Express install folder. The samples are available in both Visual Basic and C# for Visual Studio 2005. The samples use Add-in Express Loader and provide the setup projects. Please note that the setup projects include the KB908002 in their prerequisites. See http://support.microsoft.com/kb/908002.

## .NET Framework 1.1 and 2.0 Installed on the Development PC

In a mixed .NET Framework environment, you may need to use the Host Configuration command of an Add-in Express Module. This command [creates and] changes the configuration file for your host application. The examples of configuration file names are outlook.exe.config and excel.exe.config. The file is located in C:\Program Files\Microsoft Office\OFFICE11.

## New Project dialog

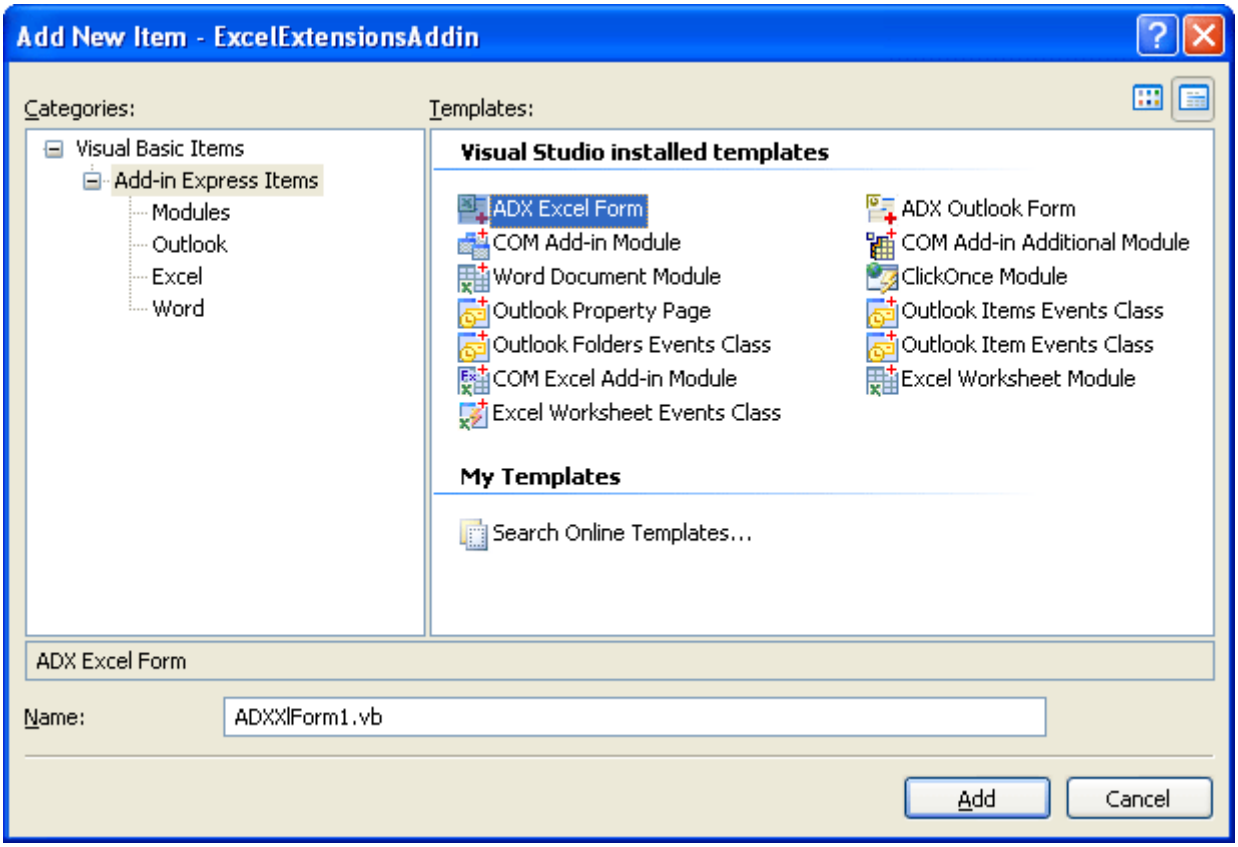
Add-in Express installs the following items to the New Project dialog.



- Add-in Express COM Add-in – a project wizard for creating COM Add-ins.
- Add-in Express RTD Server – this is a project wizard for creating RTD Servers.
- Add-in Express Smart Tag – allows creating Smart Tags.
- Add-in Express XLL Add-in – use this wizard to create Excel XLL Add-ins.

## Add New Item Dialog

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




- Add-in Express Excel Form – a form designed for embedding into Excel windows. To use this functionality, you will have to install an appropriate Add-in Express .NET package. This item is available in Standard and higher Add-in Express .NET product packages.
- 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 an appropriate Add-in Express .NET package. This item is available in Standard and higher Add-in Express .NET product packages.
- COM Add-in Module – the core of any Add-in Express COM add-in. See COM Add-ins.
- COM Add-in Additional Module – it is an additional add-in module. Indeed. It will supplement your main module in case it has grown up too big in size.

- Word Document Module – allows handling events of any MS Forms controls placed on a specified Word document. See Word Documents.
- ClickOnce Module – allows accessing ClickOnce-related features in Add-in Express ClickOnce Projects.
- Outlook Property Page – the form designed for extending Outlook Options and Folder Properties dialogs with custom pages. See Outlook Property Page and Your First Microsoft Outlook COM Add-in.
- Outlook Items Event Class – provides easy access to the events of the Items class of Outlook (see Add-in Express Event Classes).
- Outlook Folders Event Class – provides easy access to the events of the Folders class of Outlook (see Add-in Express Event Classes).
- Outlook Item Event Class – provides easy access to the events of the MailItem, TaskItem, ContactItem, etc classes of Outlook (see Add-in Express Event Classes).
- COM Excel Add-in Module – this module allows implementing the functionality when developing Excel Automation Add-ins. You may need it only if you want to add new functions that can be used in Excel formulas.
- Excel Worksheet Module – allows handling events of any MS Forms controls placed on a specified Excel worksheet. See Excel Workbooks.
- Excel Worksheet Event Class – provides easy access to the events of the Worksheet class.

## What are PIAs?

PIAs (Primary Interop Assemblies) provide the compiler with early-binding information on the host application objects, their properties, and methods. In Visual Basic 6 terms, PIAs correspond to references you set using the Project / References dialog. Office applications are almost 100% backward compatible and you can use PIAs from an earlier Office version to work with a newer one. Say, you can develop your add-in using Outlook 2000 PIA and it will work with Outlook 2007. Note, in this case, you can access Outlook 2007 features using late binding (see the InvokeMember method in MSDN). Obviously, you cannot expect the add-in developed with Outlook 2007 PIA to work with Outlook 2000: say Outlook 2000 (as well as Outlook XP and Outlook 2003) doesn't have the Explorer.Search event specified in Outlook 2007 PIA.

## Why Version-Neutral PIAs?

Add-in Express includes version-neutral PIAs for all Office applications: just check the Version-neutral PIAs checkbox in the Add-in Express project wizard. This will add PIAs for the earliest versions of your add-in's host applications. Say, if you host your add-in in Outlook, this will be Outlook 2000 PIA. And for InfoPath, this will be InfoPath 2007 PIA.

### 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. Select a class/property/method and press F1 to get the help topic that relates to the object.

## COM Add-ins

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

To add the COM Add-ins command to a toolbar or menu 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

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, such as OutlookApp and ExcelApp. To identify the host application, you can also use the HostName, HostType, and HostVersion properties of the module.

### Releasing COM objects

The list of rules is very short:

- You **must never** release COM objects obtained through the parameters of events provided by Add-in Express.
- You **must always** release COM objects retrieved by you ("manually") from any COM object.

Note, just set a variable of say, Outlook.MailItem type to null (Nothing in VB) has nothing to do with releasing its underlying COM object. To release it, you call the.Marshal.ReleaseComObject method (System.Runtime.InteropServices namespace) and pass the variable as a parameter.

You may also want to read the following article - Why Outlook is not closing when I run my add-in?.

### What is ProgId?

ProgID = Program Identifier. This is a textual name representing a server object. It consists of the project name and the class name, like MyServer.MyClass.

You find it in ProgIDAttribute of an Add-in Express Module. For instance:

```
...
'Add-in Express Add-in Module
<GuidAttribute("43F48D82-7C6F-4705-96BB-03859E881E2C"), _
    ProgIdAttribute("MyAddin1.AddinModule")> _
Public Class AddinModule
    Inherits AddinExpress.MSO.ADXAddinModule
...
```

**Note**

*We found the definition of ProgId in The COM / DCOM Glossary. On that page, you can find other COM-related terms and their definitions.*

### 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>
```

### FolderPath Property Value in Outlook 2000 and XP

The function returns the same value as the MAPIFolder.FolderPath property available in Outlook 2003+.

```
    Private Function GetFolderPath(ByVal folder As Outlook.MAPIFolder) _
        As String

        Dim path As String = ""
        Dim toBeReleased As Boolean = False
        Dim tempObj As Object = Nothing

        While folder IsNot Nothing
            path = "\" + folder.Name + path
            Try
                tempObj = folder.Parent
```

```
            Catch
               'permissions aren't set
               tempObj = Nothing
            Finally
               If toBeReleased Then
                  Marshal.ReleaseComObject(folder)
               Else
                  'the caller will release the folder passed
                  toBeReleased = True
               End If
               folder = Nothing
            End Try

            'The parent of a root folder is of the Outlook.Namespace type
            If TypeOf tempObj Is Outlook.MAPIFolder Then
               folder = CType(tempObj, Outlook.MAPIFolder)
            End If
         End While

         If tempObj IsNot Nothing Then Marshal.ReleaseComObject(tempObj)
         If path <> "" Then path = Mid$(path, 2)
         Return path
      End Function
```

## Command Bars and Controls

### ControlTag vs. Tag Property

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 a pop-up as using the Controls property of a command bar and then add any control to the pop-up via the Controls property of the pop-up.

However, pop-ups have an 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 Add-in Express. It seems to be intended by MS.

## Edits and Combo Boxes and the Change Event

The Change event occurs only when the value was changed **AND** the focus is shifted. This is also not our bug but MS guys' "trick".

## 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.

You can add any built-in control to your command bar. To do this, 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. To find out the built-in control IDs, use the free Built-in Controls Scanner utility (http://www.add-in-express.com/downloads/controls-scanner.php)

## CommandBar.SupportedApps

Use this property to specify if the command bar will appear in some or all host applications supported by the add-in.

## 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.

## COM 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.

## 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.

### 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 command bar types.

## Ribbon

### Being Ribbonned

Find IDs of built-in Ribbon controls in 2007 Office System Document: Lists of Control IDs at http://www.microsoft.com/downloads/details.aspx?FamilyID=4329d9e9-4d11-46a5-898d-23e4f331e9ae&DisplayLang=en.

### Sharing Ribbon Controls Across Multiple Add-ins

First off, you assign a string value to the Namespace property of AddinModule. This makes Add-in Express add two xmlns attributes to the customUI tag in the resulting Xml markup:

- xmlns:default="%ProgId of your add-in, see the ProgID attribute of the AddinModule class%",
- xmlns:shared="%the value of the AddinModule.Namespace property%".

Originally, all the Ribbon controls are located in the default namespace (id="%Ribbon control's id%" or idQ="default:%Ribbon control's id%") and you have full control over them via the callbacks provided by Add-in Express. When you specify the Namespace property, Add-in Express changes the markup to use idQ's instead of id's.

Then, in all the add-ins that are to share a Ribbon control, for the control with the same Id (you can change the Id's to match), you set the Shared property to True. For the Ribbon control whose Shared property is True, Add-in Express changes its idQ to use the shared namespace (idQ="shared:%Ribbon control's id%") instead of the default one. Also, for such Ribbon controls, Add-in Express cuts out all the callbacks and replaces them with "static" versions of the attributes. Say, getVisible="getVisible_CallBack" will be replaced with visible="%value%".

The shareable Ribbon controls are the following Ribbon container controls:

- Ribbon Tab - ADXRibbonTab
- Ribbon Box - ADXRibbonBox

- Ribbon Group - ADXRibbonGroup
- Ribbon Button Group - ADXRibbonButtonGroup

When referring to a shared Ribbon control in the BeforeId and AfterId properties of another Ribbon control, you use the shared controls' idQ: %namespace abbreviation% + ":" + %control id%. The abbreviations of these namespaces are "default" and "shared" string values.

Resulting Xml markup may look like this:

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"
          xmlns:default="MyOutlookAddin1.AddinModule"
          xmlns:shared="MyNameSpace" [callbacks omitted]>
 <ribbon>
  <tabs>
   <tab idQ=" shared:adxRibbonTab1" visible="true" label="My Tab">
    <group idQ="default:adxRibbonGroup1" [callbacks omitted]>
     <button idQ="default:adxRibbonButton1" [callbacks omitted]/>
    </group>
   </tab>
  </tabs>
 </ribbon>
</customUI>
```

In the XML-code above, the add-in creates a shared tab, containing a private group, containing a button (private again).

## Deploying and Debugging

### Deploying - Shadow Copy

The Add-in Express Loader uses the ShadowCopy-related properties and methods of the AppDomain class. When you run your Add-in Express add-in, the host application loads the Add-in Express Loader DLL referenced in the registry. The Add-in Express Loader DLL does the following:

- It finds your add-in DLLs in the DLL Cache. If there are no add-in DLLs in the cache it copies all .NET DLLs to the cache (including dependencies). The cache folder is located in C:\Documents and Settings\<user name>\Local Settings\Application Data\assembly\dl<number>. If all add-in DLLs (including dependencies) already exist in the cache, it compares their versions. If the versions are not the same, it copies new DLLs to the cache.
- It loads the add-in DLLs from the cache.

You can see how the add-in versioning influences the add-in loading.

This approach (it is built into .NET, as you can see) allows replacing add-in DLLs when the add-in is loaded. The disadvantage is numerous files located in the cache. As far as we know, MS doesn't provide a solution for this problem. You may think you can remove these files in an add-in's uninstall custom action. However, this will remove the files from the current profile only.

## Deploying – "Everyone" Option in a COM Add-in MSI package

The Everyone option of the msi installer doesn't have any effect on the Add-in Express based COM add-ins and RTD servers.

## How Do I Find the PublicKeyToken of My Add-in?

It can be found in the setup project (that must be already built) - click on your add-in primary output in the setup project and, in the Properties window, expand the KeyOutput property and see the PublicKeyToken property value.

## Deploying Office Add-ins

- Make sure Vista, Windows XP, and Office have all the updates: Microsoft eventually closes their slips and blunders with service packs and other updates. Keep an eye on Visual Studio 2005 updates, too.
- If a non-admin user will install your add-in, use [AppDataFolder] as the default location. Make sure the user is instructed to run MSI. On Vista, if the user runs setup.exe, a non-admin will get the elevation dialog and this can end with installing the add-in to the admin profile. In such a case, the add-in will not be available for the standard user.

## My Add-in Is Always Disconnected

If your add-in fires exceptions at the startup, the host application can block the add-in and move it to the Disabled Items list. To find the list, in the host application, go to "Help", then "About". At the bottom of the About dialog, there is the Disabled Items button. Check it to see if the add-in is listed there (if so, select it and click the enable button).

## ClickOnce Cache

The cache location is visible in the COM Add-ins dialog. It may have the following look:

```
C:\Documents and Settings\user\Local
Settings\Apps\2.0\NCPNO3QK.0KJ\ONNRMXC3.ALM\add-..d-
in_5c073faf40955414_0001.0000_2a2d23ab74b720da
```

Currently, we don't know if there is a decent way to clear the cache.

# RTD

## No RTD Servers in EXE

Add-in Express currently supports RTD Servers in DLLs only.

## Update Speed for an RTD Server

Microsoft limits the minimal interval between updates to 2 seconds. There is a way to change this minimum value but Microsoft doesn't recommend doing this.

# Final Note

If your questions are not answered here, please see the HOWTOs section on www.add-in-express.com. We update these pages regularly.