Designing and Creating Office Solutions

Visual Studio 2013 1 out of 1 rated this helpful

Visual Studio provides project templates that you can use to create several different types of Office solutions. This section of the documentation describes the project templates and provides guidance about creating Office projects. For information about how to implement code and user interface customizations after you have created your project, see Developing Office Solutions.

Applies to: The information in this topic applies to document-level projects and application-level projects for Office 2013 and Office 2010. See Features Available by Office Application and Project Type.

Creating Office Projects

Before you begin, you should determine your requirements and discover the type of solution that offers the best fit. For example, if your Office solution must run every time the application is used, an application-level add-in best fits your requirements. If the code is tightly integrated with a single document, create a document-level customization. These project types are available as Visual Studio project templates. For more information about the Office project templates that are included with Visual Studio, see Office Project Templates Overview. For more information about how to create Office projects, see How to: Create Office Projects in Visual Studio.

Office projects have features and project items that are different from other types of projects in Visual Studio. For example, when you create a document-level project, the document or workbook in your project can be opened and edited inside Visual Studio. For more information, see Office Projects in the Visual Studio Environment.

Choosing a .NET Framework Version

After selecting the project type that best fits your requirements, you can choose which version of the .NET Framework to use in your development process. You can target the following .NET Framework versions in Office projects:

- .NET Framework 4
- .NET Framework 4 Client Profile
- .NET Framework 4.5

The .NET Framework version that you choose for your project is required on end user computers for your solution to run. For example, if your project targets the .NET Framework 4, the .NET Framework 4 is required on end user computers. In this example, your solution will not run if only the .NET Framework 3.5 is installed on end user computers.

If you migrate an application-level add-in project that targets the .NET Framework 3.5, Visual Studio changes the target framework of your project to .NET Framework 4 or the.NET Framework 4.5 depending on the version of Office that you have installed.

However, after Visual Studio changes the target framework, you might need to modify some of the code in your project if it uses certain features. For more information about how to change the target framework, see How to: Target a Version of the .NET Framework. For more information about changes you might need to make in your project, see Migrating Office Solutions to the .NET Framework 4 or the .NET Framework 4.5.

If Visual Studio changes the target .NET Framework for your project and you are using ClickOnce to deploy your solution, make sure that you also select the corresponding version of the .NET Framework in the **Prerequisites** dialog box. This selection does not change automatically when you change the target framework for your project. For more information, see How to: Install Prerequisites on End User Computers to Run Office Solutions.

Note

You cannot target the .NET Framework 3.5 or earlier in Office projects that you create by using Visual Studio 2013. Office projects that you create by using Visual Studio 2013 require features that were first introduced in the .NET Framework 4 Client Profile

Understanding When the Office PIAs Are Required on End User Computers

By default, Office primary interop assemblies (PIAs) do not need to be installed on end user computers if the **Embed Interop Types** property of each Office PIA reference in the project is set to **True**, which is the default value. In this scenario, the type information for the PIA types that are used by your solution is embedded into the solution assembly when you build the project. At run time, the embedded type information is used instead of the PIAs to call into the Office application's COM-based object model. For more information about how types from PIAs are embedded into your solution, see Type Equivalence and Embedded Interop Types.

If the **Embed Interop Types** property of each Office PIA reference in the project is set to **False**, Office PIAs must be installed and registered in the global assembly cache on each end user computer that runs the solution. In most cases, the PIAs are installed by default with Office, but you can also include the PIA redistributable as a prerequisite for your solution. For more information, see Office Solution Prerequisites for Deployment.

Understanding the Client Profile

The .NET Framework Client Profile is a subset of the full .NET Framework. You can target the .NET Framework Client Profile if you need to use only the client features in the .NET Framework and you want to provide the fastest possible deployment experience for your Office solution. For more information, see .NET Framework Client Profile.

When you create an Office project that targets the .NET Framework 4, the .NET Framework 4 Client Profile is targeted by default. If you want to develop for the full .NET Framework 4, you must set this option after the project is created. For more information, see How to: Target a Version of the .NET Framework.

Creating Solutions for the 64-bit Edition of Microsoft Office

Microsoft Office 2013 and Office 2010 are available in 64-bit and 32-bit editions. To create Office solutions that can run in either edition, the platform target setting for your project must be set to **Any CPU**. This is the default value for Office projects. For more information, see Building Office Solutions.

There are separate 64-bit and 32-bit versions of the Visual Studio Tools for Office runtime that are used by the 64-bit and 32-bit editions of Microsoft Office 2013 and Office 2010. For more information, see Visual Studio Tools for Office Runtime Overview.

Assemblies in Office Solutions

When you create an Office project by using the Office development tools in Visual Studio, the code that you write is eventually compiled into an assembly. The assembly is usually deployed to a shared server or to a directory on the client computer.

Assemblies in Office solutions are loaded by an Office application. After the assembly is loaded, code in the assembly can respond to events that are raised in the application, for example, when a user clicks a menu item. Code in the assembly can also call into the object model to automate and extend the application, and it can use any of the classes in the .NET Framework. For more information, see Architecture of Document-Level Customizations and Architecture of Application-Level Add-Ins.

Office solutions use deployment manifests and application manifests to identify the assembly. The manifests contain information about the assembly's name, version, and location, so that the application can find, link to, and run the correct

assembly. For more information, see Application and Deployment Manifests in Office Solutions.

Document-level projects include a document in addition to an assembly. The document acts as the front end of the application and is where all user interaction takes place. Each document can have only one main project assembly associated with it; however, multiple documents can point to the same assembly.

Assemblies in document-level projects are not embedded in the document; instead, they are stored elsewhere and are identified by the document's application manifest.

Security Considerations for Assemblies

For an Office solution to run on a computer, the assemblies used by the solution must be trusted to run. For more information about security, see Securing Office Solutions.

By default, the solution assembly and any referenced assemblies that are in your project's output folder are trusted to run on the development computer when you build the project. For more information, see Building Office Solutions.

For security reasons, it is best to create projects on your local computer, rather than developing on a shared location. For more information, see Collaborative Development of Office Solutions.

Referenced Assemblies

The assembly can reference other assemblies, which are listed in the project's references. However, one document-level project assembly cannot reference another document-level project assembly.

See Also

Tasks
How to: Create Office Projects in Visual Studio
How to: Target Office Applications Through Primary Interop Assemblies
How to: Set Up Configuration Information for an Office Solution
Concepts
Office Projects in the Visual Studio Environment
Properties in Office Projects
Running Solutions in Different Versions of Microsoft Office
Application and Deployment Manifests in Office Solutions
Common Tasks in Office Programming
Other Resources
Office Project Templates Overview
Using Office Functionality Inside of Visual Studio
Deploying an Office Solution
Developing Office Solutions
Architecture of Office Solutions in Visual Studio