Designer for WiX Toolset

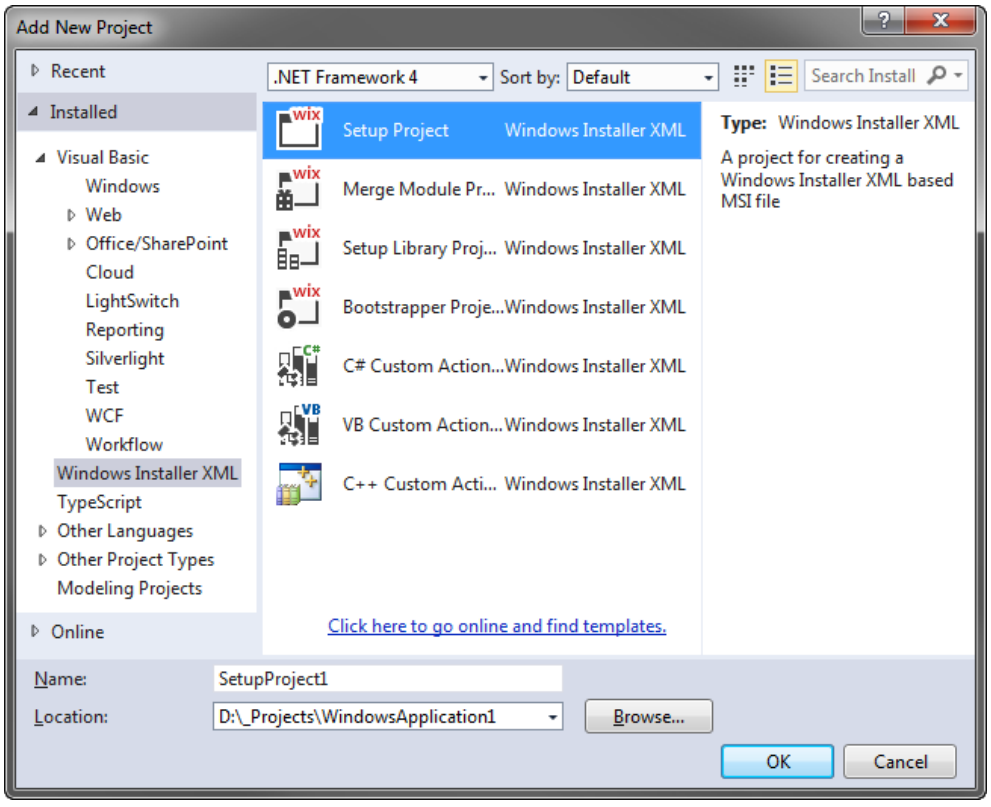
Creating a standard .msi package

This page explains how to create an .msi installation package on an example of an existing Windows Forms application.

> *Make sure WiX Toolset is installed on your machine and registered with your Visual Studio. If WiX is not listed in the About dialog of Visual Studio, install the WiX version supporting the Visual Studio version that you use.*

Step 1. Adding a WiX setup project

In Visual Studio, open your solution, and add a WiX project to it: go to the Visual Studio main menu and click *File -> Add -> New Project* to open the Add New Project dialog.



Choose the *Setup Project* item in the *Windows Installer XML* node, specify the project name and click *OK*. Visual Studio will add the setup project to the solution and open the *Product.wxs* file.

Setup project file

is the source file of your setup project. A newly created project contains the XML code shown below

```
<?xml version="1.0" encoding="utf-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" Name="SetupProject1" Language="1033" Version="1.0.0.0" Manufacturer=""
           UpgradeCode="aa918e08-bbdd-49fc-a9f2-fef05b153069">
    <Package InstallerVersion="200" Compressed="yes" InstallScope="perMachine" />

    <MajorUpgrade
        DowngradeErrorMessage="A newer version of [ProductName] is already installed." />
    <MediaTemplate />

    <Feature Id="ProductFeature" Title="SetupProject1" Level="1">
      <ComponentGroupRef Id="ProductComponents" />
    </Feature>
  </Product>

  <Fragment>
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="INSTALLFOLDER" Name="SetupProject1" />
      </Directory>
    </Directory>
  </Fragment>

  <Fragment>
    <ComponentGroup Id="ProductComponents" Directory="INSTALLFOLDER">
      <!-- TODO: Remove the comments around this Component element and the
                 ComponentRef below in order to addresources to this installer. -->
      <!-- <Component Id="ProductComponent"> -->
      <!-- TODO: Insert files, registry keys, and other resources here. -->
      <!-- </Component> -->
    </ComponentGroup>
  </Fragment>
</Wix>
```

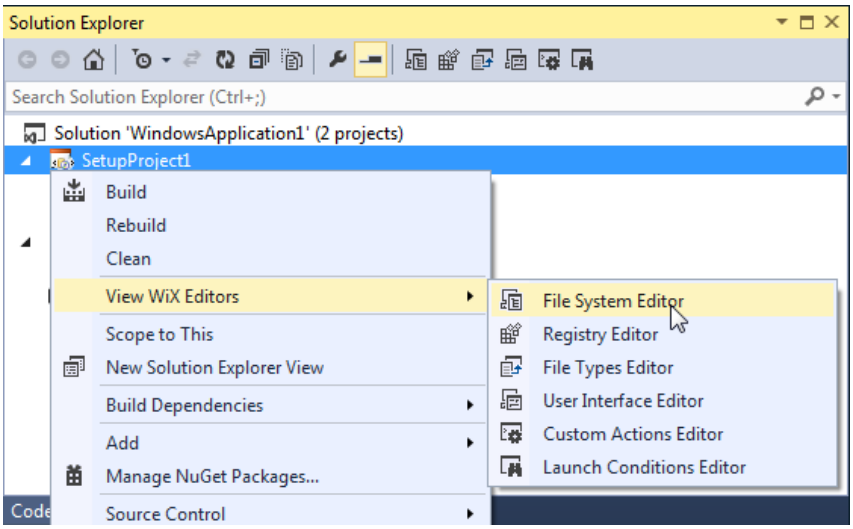
We suggest that you modify the file as described below.

As you can see in the code snippet above, the *Manufacturer* attribute is empty. Checking your project against the WiX schema will generate an error message so you may want to set this attribute right now.

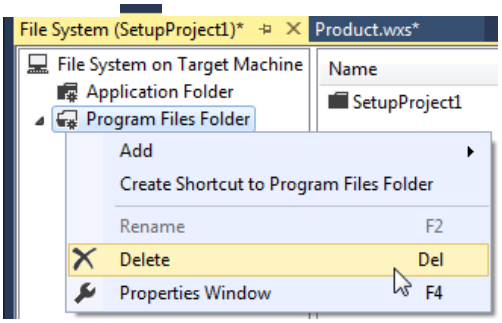
Pay attention to the *MediaTemplate* tag: by adding the *EmbedCab="yes"* attribute, you create a single .MSI file as opposed to getting a CAB file along with the .MSI.

Step 3. Specifying the target folder

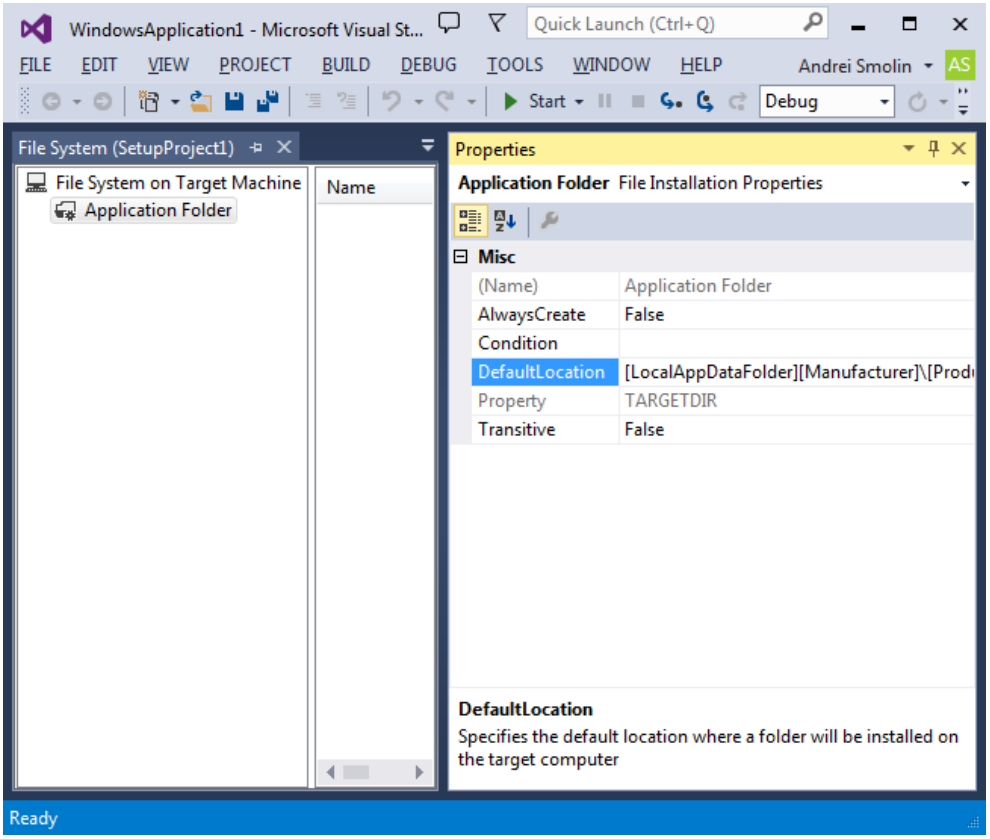
TARGETDIR is a special Windows Installer property (see here) that defines the root destination directory to which the installer delivers the setup project's files on the target machine. In fact, *TARGETDIR* is the main entity of the Windows Installer architecture. To specify it, you open the *File System Editor*.



Essential. The File System Editor reveals the fact that WiX pre-creates two folder entries: Application Folder and [illegible] Folder. You need to **delete the Program Files Folder entry**.



Now specify the *DefaultLocation* property:



The *DefaultLocation* property accepts all Windows Installer properties listed in Windows Installer Property Reference at MSDN. Here is a couple of examples:

[LocalAppDataFolder][Manufacturer]\[ProductName] - typical for installing per-user things
[ProgramFilesFolder][Manufacturer]\[ProductName] - typical for installing 32-bit per-machine programs
[ProgramFiles64Folder][Manufacturer]\[ProductName] - typical for installing 64-bit per-machine programs

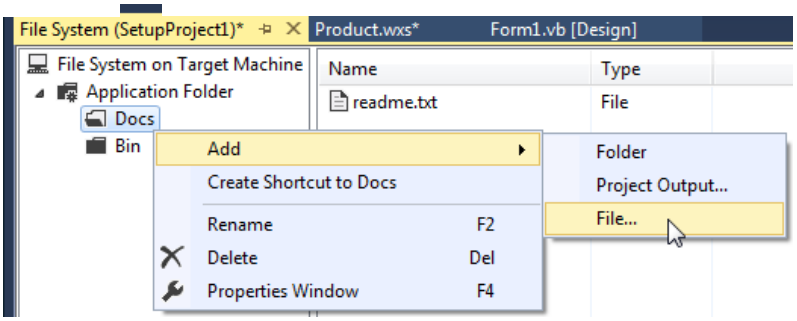
Step 4. Application project outputs

Add all required project outputs to the *Application Folder*. Adding the *Primary Output* is required for delivering the binary form of your code to the target PC.

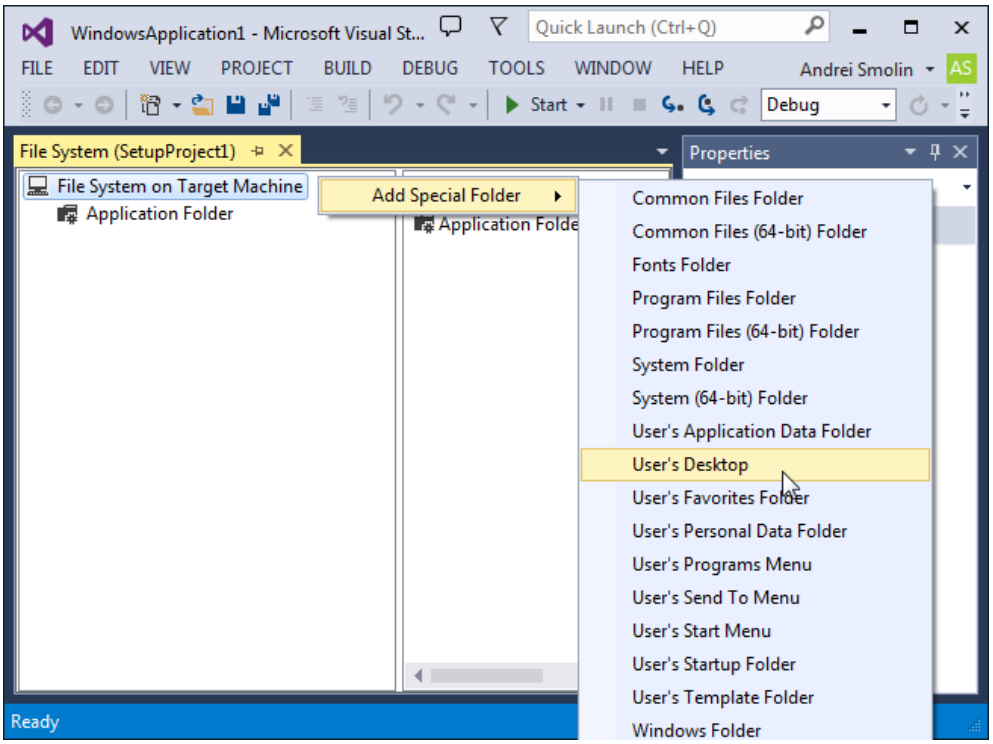
In the current version of WiX designer, adding the *Primary Output* doesn't add dependency assemblies required by the application. You need to add such assemblies manually. Adding an assembly belonging to the .NET Framework version (edition) that you currently use is **not** required.

Step 5. Creating a folder hierarchy and deploying files

You use the *File System Editor* to specify how and where to create folders and files required for your program.

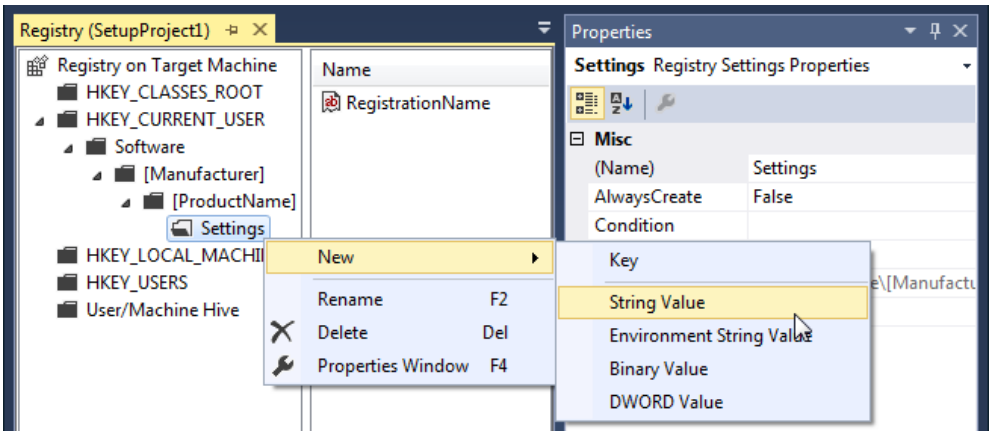


To refer to system folders such as *User's Desktop* or *Global Assembly Cache Folder*, right-click the *File System on Target Machine* node and choose *Add Special Folder* on the context menu:



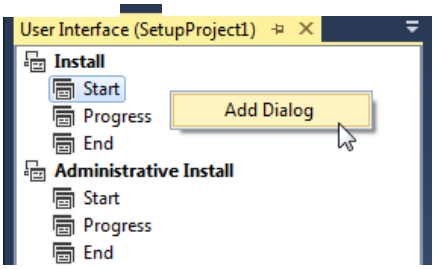
Step 6. Creating registry entries

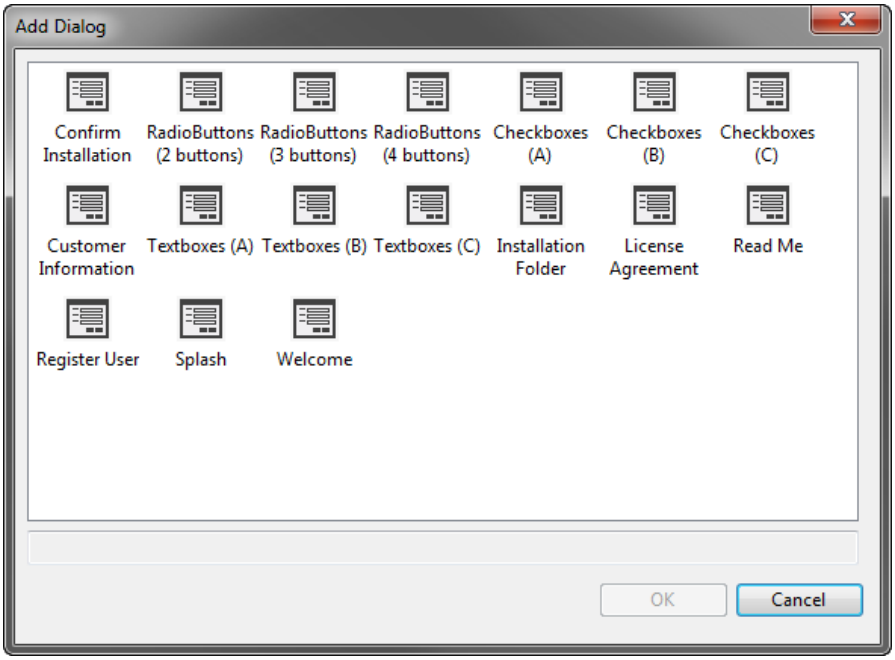
You use the *Registry Editor* to create new and modify existing registry keys and values required for your program.

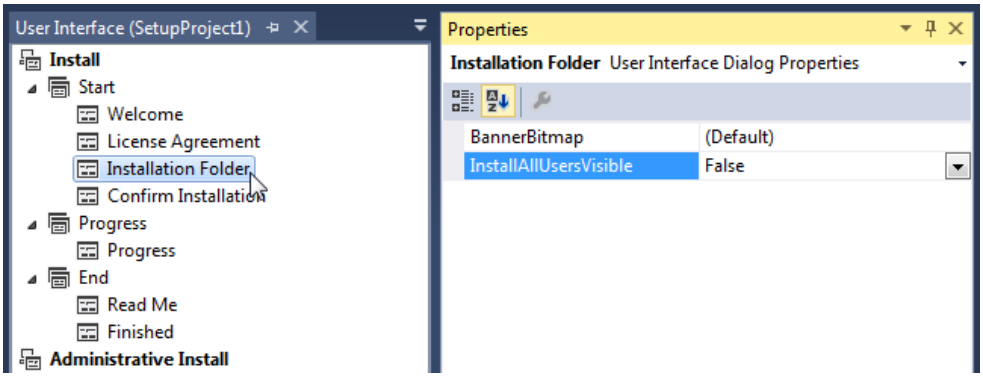


Step 7 Creating the installer UI

provides a number of pre-created dialogs that you use to create the UI of the installer. If you need your
upport multiple languages, see How to make the setup project localizable.

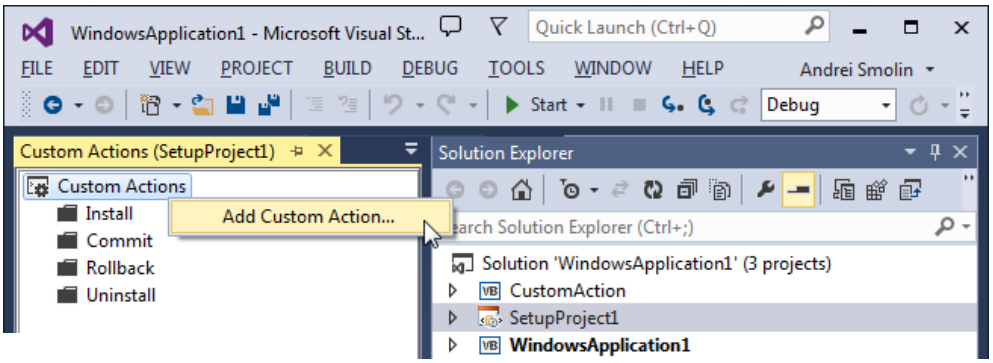


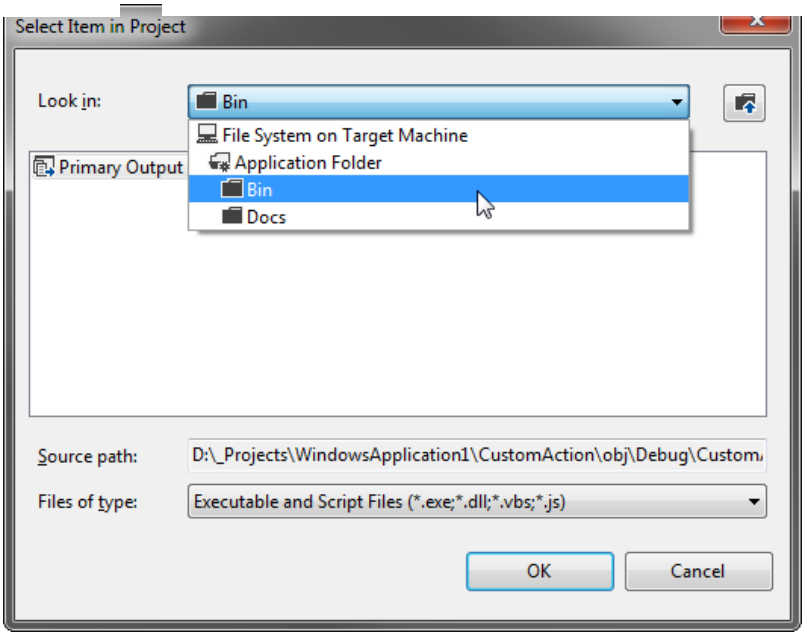


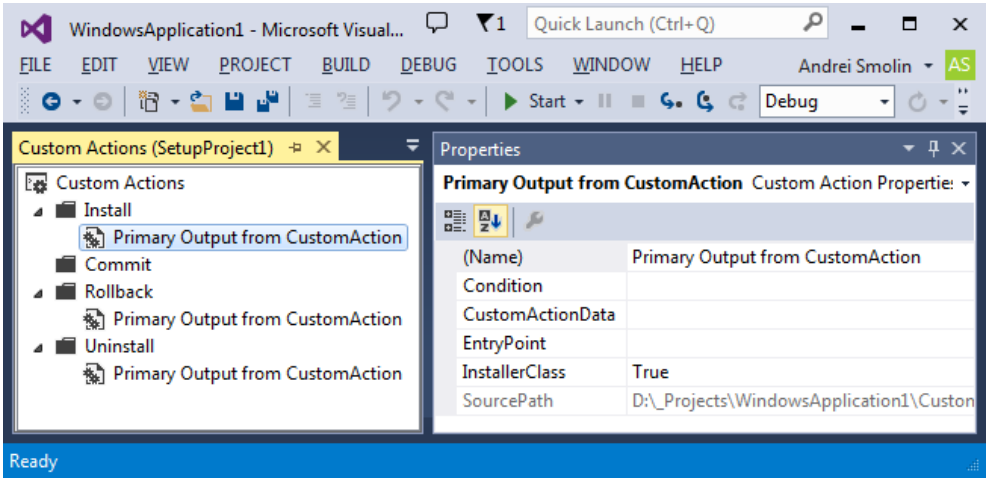


Step 8. Specifying custom actions

Use the *Custom Actions Editor* to specify the custom actions for your setup package.







Designer for WiX Toolset <<

>> Creating a web setup project

Have any questions? Ask us right now!

Add-in Express Feedback

If you need quick assistance or want to share your concerns, ideas or suggestions, please write to us using this form.