Host Custom UWP Controls in C++ Win32 Project using XAML Islands

This article explains how to modernize C++ Win32 project with custom UWP Controls through XAML Islands in detail. With custom UWP controls, it allows us to define control layout easily through XAML pages. Not only putting standard UWP controls into the custom control, we can also integrate other custom controls as well, such as latest WinUI controls. We will use a new Xaml Application project to bring the custom UWP controls into our C++ Win32 project. This article has mentioned how to use the new XAML application in Win32 C++ application, we will give more detailed steps.

Overall, in our solution, we will have two parts to demonstrate this method:

- C++ Win32 Project
- A companion UWP app project that defines a XamlApplication object.

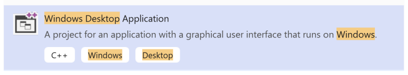
  In this project, we will define a custom UWP control and export it so that C++ Win32 can use the custom UWP control.

Development Environment

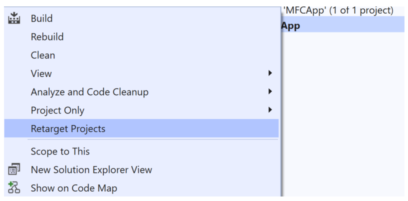
- Visual Studio 2019 (16.3.6)
- Windows 10 1909 (18363.476)
- Windows 10 SDK (10.0.18362.1)

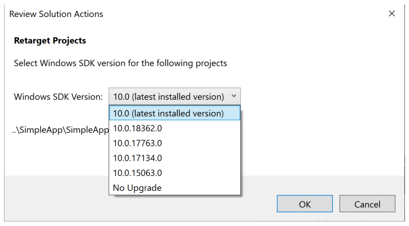
Configure C++ Win32 Project

1. Create C++ Win32 in Visual Studio 2019, will name it SimpleApp


Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.

2. In Solution Explorer, right-click the Simple project node, click **Retarget Project**, select the **10.0.18362.0** or a later SDK release, and then click OK.




Review Solution Actions

Retarget Projects

Select Windows SDK version for the following projects

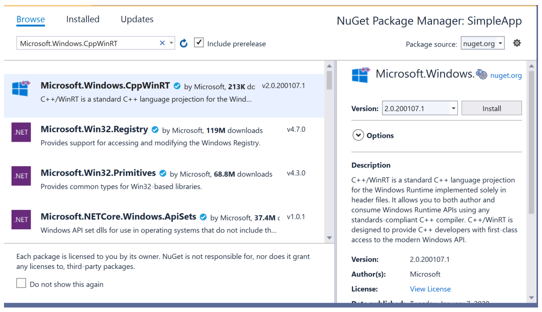
Windows SDK Version: 10.0 (latest installed version)

..\SimpleApp\SimpleApp

3. Install the ***Microsoft.Windows.CppWinRT*** NuGet package:

   a. Right-click the project in Solution Explorer and choose ***Manage NuGet Packages***.

   b. Select the Browse tab, search for the **Microsoft.Windows.CppWinRT** package, and install the latest version of this package.



Microsoft.Windows.CppWinRT

Microsoft.Windows.CppWinRT by Microsoft, 213K dc v2.0.200107.1
C++/WinRT is a standard C++ language projection for the Wind...

Microsoft.Win32.Registry by Microsoft, 119M downloads v4.7.0
Provides support for accessing and modifying the Windows Registry.

Microsoft.Win32.Primitives by Microsoft, 68.8M downloads v4.3.0
Provides common types for Win32-based libraries.

Microsoft.NETCore.Windows.ApiSets by Microsoft, 37.4M v1.0.1
Windows API set dlls for use in operating systems that do not include th...

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

Microsoft.Windows. nuget.org

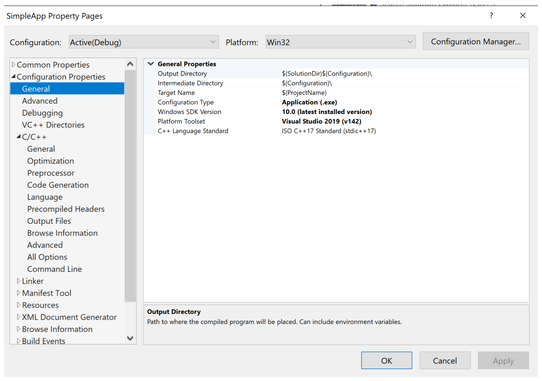
Version: 2.0.200107.1

Description

C++/WinRT is a standard C++ language projection for the Windows Runtime implemented solely in header files. It allows you to both author and consume Windows Runtime APIs using any standards-compliant C++ compiler. C++/WinRT is designed to provide C++ developers with first-class access to the modern Windows API.

Version: 2.0.200107.1
Author(s): Microsoft
License: View License

After install the nuget package, check the SimpleApp project properties, you will notice its C++ version is ISO C++17, which is required by C++/WinRT:



SimpleApp Property Pages

Configuration: Active(Debug) Platform: Win32 Configuration Manager...

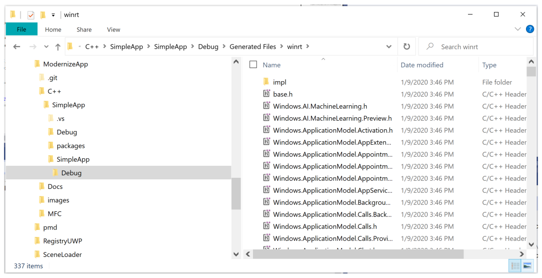
Common Properties
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Headers
Output Files
Browse Information
Advanced
All Options
Command Line
Linker
Manifest Tool
Resources
XML Document Generator
Browse Information
Build Events

General Properties	
Output Directory	$(SolutionDir)$(Configuration)\
Intermediate Directory	$(Configuration)\
Target Name	$(ProjectName)
Configuration Type	**Application (.exe)**
Windows SDK Version	**10.0 (latest installed version)**
Platform Toolset	**Visual Studio 2019 (v142)**
C++ Language Standard	ISO C++17 Standard (std:c++17)

Output Directory
Path to where the compiled program will be placed. Can include environment variables.

OK Cancel Apply

Build this SimpleApp, we can see winrt projected files are generated in the “Generated Files” folder:



winrt

C++ › SimpleApp › SimpleApp › Debug › Generated Files › winrt ›

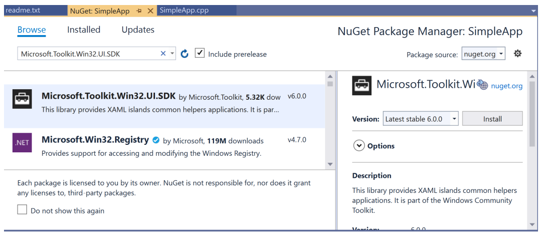
ModernizeApp
.git
C++
SimpleApp
.vs
Debug
packages
SimpleApp
Debug
Docs
images
MFC
pmd
RegistryUWP
SceneLoader

Name	Date modified	Type
impl	1/9/2020 3:46 PM	File folder
base.h	1/9/2020 3:46 PM	C/C++ Header
Windows.AI.MachineLearning.h	1/9/2020 3:46 PM	C/C++ Header
Windows.AI.MachineLearning.Preview.h	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.Activation.h	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.AppExten...	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.Appointm...	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.Appointm...	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.Appointm...	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.AppServic...	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.Backgrou...	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.Calls.Back...	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.Calls.h	1/9/2020 3:46 PM	C/C++ Header
Windows.ApplicationModel.Calls.Provi...	1/9/2020 3:46 PM	C/C++ Header

337 items

4. Install the **Microsoft.Toolkit.Win32.UI.SDK** NuGet package:

   a. In the NuGet Package Manager window, make sure that Include prerelease is selected.
   b. Select the Browse tab, search for the **Microsoft.Toolkit.Win32.UI.SDK** package, and install version v6.0.0 (or Later) of this package.



readme.txt | NuGet: SimpleApp | SimpleApp.cpp

Browse Installed Updates — NuGet Package Manager: SimpleApp

Microsoft.Toolkit.Win32.UI.SDK ☑ Include prerelease — Package source: nuget.org

Microsoft.Toolkit.Win32.UI.SDK by Microsoft.Toolkit, **5.32K** dow v6.0.0
This library provides XAML islands common helpers applications. It is par...

Microsoft.Win32.Registry by Microsoft, **119M** downloads v4.7.0
Provides support for accessing and modifying the Windows Registry.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Do not show this again

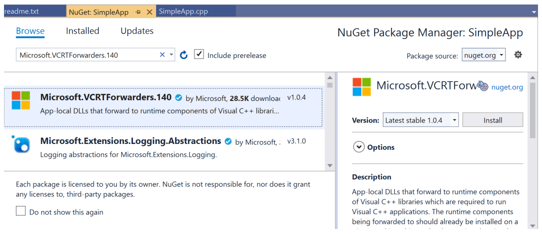
Microsoft.Toolkit.Wi nuget.org

Version: Latest stable 6.0.0 Install

Options

Description
This library provides XAML islands common helpers applications. It is part of the Windows Community Toolkit.

5. Install the ***Microsoft.VCRTForwarders.140*** nuget package as well. Running Custom UWP Control in this project will require VC libs.



readme.txt | NuGet: SimpleApp | SimpleApp.cpp

Browse Installed Updates — NuGet Package Manager: SimpleApp

Microsoft.VCRTForwarders.140 ☑ Include prerelease — Package source: nuget.org

Microsoft.VCRTForwarders.140 by Microsoft, **28.5K** downloa v1.0.4
App-local DLLs that forward to runtime components of Visual C++ librari...

Microsoft.Extensions.Logging.Abstractions by Microsoft, v3.1.0
Logging abstractions for Microsoft.Extensions.Logging.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Do not show this again

Microsoft.VCRTForw nuget.org

Version: Latest stable 1.0.4 Install

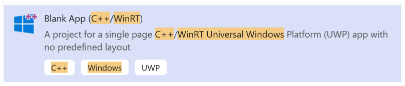
Options

Description
App-local DLLs that forward to runtime components of Visual C++ libraries which are required to run Visual C++ applications. The runtime components being forwarded to should already be installed on a

Configure UWP Project

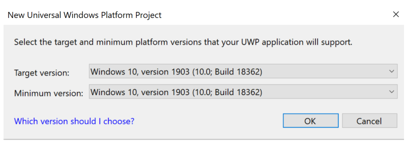
1. In Solution Explorer, right-click the solution node and select Add -> New Project.

2. Add a Blank App (C++/WinRT) project to your solution.



Blank App (C++/WinRT)
A project for a single page C++/WinRT Universal Windows Platform (UWP) app with no predefined layout
C++ Windows UWP

3. Give it a name ***MyApp***, and create it, Make sure the target version and minimum version are ***both*** set to ***Windows 10, version 1903*** or later.



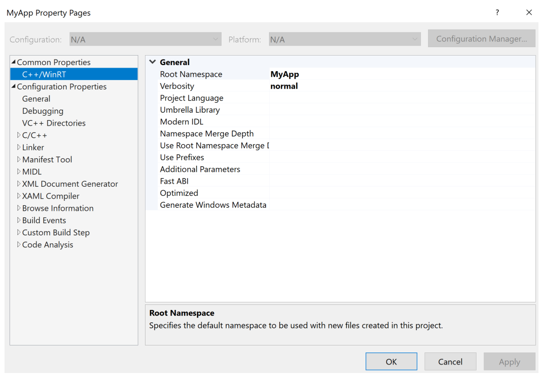
New Universal Windows Platform Project

Select the target and minimum platform versions that your UWP application will support.

Target version: Windows 10, version 1903 (10.0; Build 18362)
Minimum version: Windows 10, version 1903 (10.0; Build 18362)

Which version should I choose? OK Cancel

4. Right click the MyApp and open its properties, make sure its C++/WinRT configuration is as below:


MyApp Property Pages

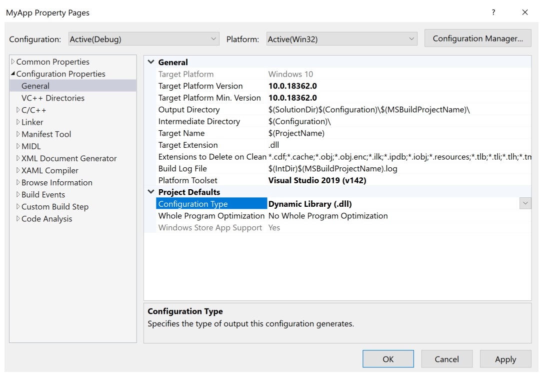
Configuration: N/A Platform: N/A

General

Root Namespace	**MyApp**
Verbosity	**normal**
Project Language	
Umbrella Library	
Modern IDL	
Namespace Merge Depth	
Use Root Namespace Merge I	
Use Prefixes	
Additional Parameters	
Fast ABI	
Optimized	
Generate Windows Metadata	

Root Namespace
Specifies the default namespace to be used with new files created in this project.

5. Change its output type from .EXE to ***Dynamic Library***


MyApp Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

General

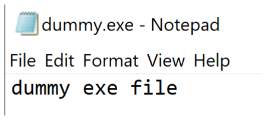
Target Platform	Windows 10
Target Platform Version	**10.0.18362.0**
Target Platform Min. Version	**10.0.18362.0**
Output Directory	$(SolutionDir)$(Configuration)\$(MSBuildProjectName)\
Intermediate Directory	$(Configuration)\
Target Name	$(ProjectName)
Target Extension	.dll
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.tlb;*.tli;*.tlh;*.tn
Build Log File	$(IntDir)$(MSBuildProjectName).log
Platform Toolset	**Visual Studio 2019 (v142)**

Project Defaults

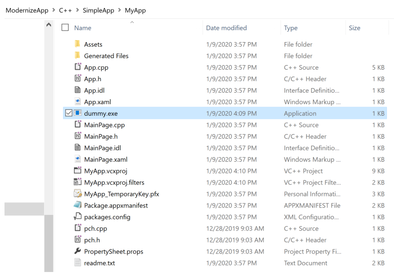
Configuration Type	**Dynamic Library (.dll)**
Whole Program Optimization	No Whole Program Optimization
Windows Store App Support	Yes

Configuration Type
Specifies the type of output this configuration generates.

6. Save a dummy.exe into the ***MyApp*** folder. It doesn't need to be a real exe, just input "dummy exe file" in notepad, and save it as ***dummy.exe***.

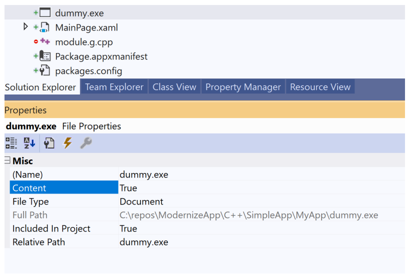

dummy.exe - Notepad

```
dummy exe file
```


ModernizeApp › C++ › SimpleApp › MyApp

Name	Date modified	Type	Size
Assets	1/9/2020 3:57 PM	File folder	
Generated Files	1/9/2020 3:57 PM	File folder	
App.cpp	1/9/2020 3:57 PM	C++ Source	5 KB
App.h	1/9/2020 3:57 PM	C/C++ Header	1 KB
App.idl	1/9/2020 3:57 PM	Interface Definitio...	1 KB
App.xaml	1/9/2020 3:57 PM	Windows Markup ...	1 KB
dummy.exe	1/9/2020 4:09 PM	Application	1 KB
MainPage.cpp	1/9/2020 3:57 PM	C++ Source	1 KB
MainPage.h	1/9/2020 3:57 PM	C/C++ Header	1 KB
MainPage.idl	1/9/2020 3:57 PM	Interface Definitio...	1 KB
MainPage.xaml	1/9/2020 3:57 PM	Windows Markup ...	1 KB
MyApp.vcxproj	1/9/2020 4:10 PM	VC++ Project	9 KB
MyApp.vcxproj.filters	1/9/2020 4:10 PM	VC++ Project Filte...	2 KB
MyApp_TemporaryKey.pfx	1/9/2020 3:57 PM	Personal Informati...	3 KB
Package.appxmanifest	1/9/2020 3:57 PM	APPXMANIFEST File	2 KB
packages.config	1/9/2020 3:57 PM	XML Configuratio...	1 KB
pch.cpp	12/28/2019 9:03 AM	C++ Source	1 KB
pch.h	12/28/2019 9:03 AM	C/C++ Header	1 KB
PropertySheet.props	12/28/2019 9:03 AM	Project Property Fi...	1 KB
readme.txt	1/9/2020 3:57 PM	Text Document	2 KB

Add the dummy.exe into MyApp project, and make sure the Content property of dummy.exe is ***True***.

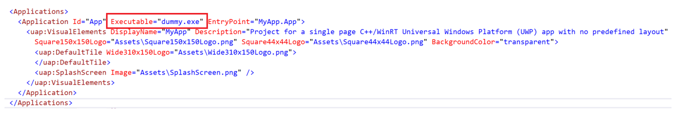


dummy.exe File Properties

Misc

(Name)	dummy.exe
Content	True
File Type	Document
Full Path	C:\repos\ModernizeApp\C++\SimpleApp\MyApp\dummy.exe
Included In Project	True
Relative Path	dummy.exe

7. Now edit Package.appxmanifest, change the Executable attribute to "dummy.exe"



```
<Applications>
  <Application Id="App" Executable="dummy.exe" EntryPoint="MyApp.App">
    <uap:VisualElements DisplayName="MyApp" Description="Project for a single page C++/WinRT Universal Windows Platform (UWP) app with no predefined layout"
      Square150x150Logo="Assets\Square150x150Logo.png" Square44x44Logo="Assets\Square44x44Logo.png" BackgroundColor="transparent">
      <uap:DefaultTile Wide310x150Logo="Assets\Wide310x150Logo.png">
      </uap:DefaultTile>
      <uap:SplashScreen Image="Assets\SplashScreen.png" />
    </uap:VisualElements>
  </Application>
</Applications>
```

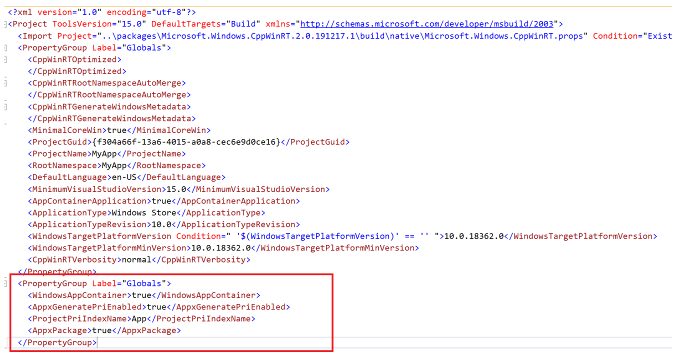
8. Right click the ***MyApp*** project, select ***Unload Project***

9. Right click the ***MyApp (Unloaded)*** project, select ***Edit MyApp.vcxproj***

10. Add below properties to the ***MyApp.vcxproj*** project file:

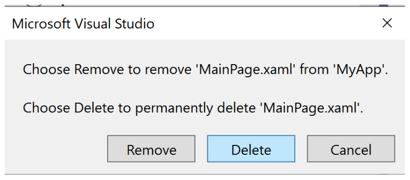
```
<PropertyGroup Label="Globals">
    <WindowsAppContainer>true</WindowsAppContainer>
    <AppxGeneratePriEnabled>true</AppxGeneratePriEnabled>
    <ProjectPriIndexName>App</ProjectPriIndexName>
    <AppxPackage>true</AppxPackage>
  </PropertyGroup>
```

For example:



```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="15.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="..\packages\Microsoft.Windows.CppWinRT.2.0.191217.1\build\native\Microsoft.Windows.CppWinRT.props" Condition="Exist
  <PropertyGroup Label="Globals">
    <CppWinRTOptimized>
    </CppWinRTOptimized>
    <CppWinRTRootNamespaceAutoMerge>
    </CppWinRTRootNamespaceAutoMerge>
    <CppWinRTGenerateWindowsMetadata>
    </CppWinRTGenerateWindowsMetadata>
    <MinimalCoreWin>true</MinimalCoreWin>
    <ProjectGuid>{f304a66f-13a6-4015-a0a8-cec6e9d0ce16}</ProjectGuid>
    <ProjectName>MyApp</ProjectName>
    <RootNamespace>MyApp</RootNamespace>
    <DefaultLanguage>en-US</DefaultLanguage>
    <MinimumVisualStudioVersion>15.0</MinimumVisualStudioVersion>
    <AppContainerApplication>true</AppContainerApplication>
    <ApplicationType>Windows Store</ApplicationType>
    <ApplicationTypeRevision>10.0</ApplicationTypeRevision>
    <WindowsTargetPlatformVersion Condition=" '$(WindowsTargetPlatformVersion)' == '' ">10.0.18362.0</WindowsTargetPlatformVersion>
    <WindowsTargetPlatformMinVersion>10.0.18362.0</WindowsTargetPlatformMinVersion>
    <CppWinRTVerbosity>normal</CppWinRTVerbosity>
  </PropertyGroup>
  <PropertyGroup Label="Globals">
    <WindowsAppContainer>true</WindowsAppContainer>
    <AppxGeneratePriEnabled>true</AppxGeneratePriEnabled>
    <ProjectPriIndexName>App</ProjectPriIndexName>
    <AppxPackage>true</AppxPackage>
  </PropertyGroup>
```

11. Right click the ***MyApp (Unloaded)*** project, select ***Reload Project***.
12. Right click ***Mainpage.xml***, select ***Remove***. And then click ***Delete***



Microsoft Visual Studio

Choose Remove to remove 'MainPage.xaml' from 'MyApp'.

Choose Delete to permanently delete 'MainPage.xaml'.

13. Copy App.Xaml, App.cpp, App.h, App.idl contents to overwrite current ones:

App.Xaml

```
<Toolkit:XamlApplication
    x:Class="MyApp.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:MyApp"
    xmlns:Toolkit="using:Microsoft.Toolkit.Win32.UI.XamlHost"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    RequestedTheme="Light"
    mc:Ignorable="d">
</Toolkit:XamlApplication>
```

App.cpp

```
#include "pch.h"
#include "App.h"

using namespace winrt;
using namespace Windows::UI::Xaml;

namespace winrt::MyApp::implementation
{
    App::App()
    {
        Initialize();

        AddRef();
        m_inner.as<::IUnknown>()->Release();
    }

    App::~App()
    {
        Close();
    }
}
```

App.h

```
//
// Declaration of the App class.
//

#pragma once

#include "App.g.h"
#include "App.base.h"

namespace winrt::MyApp::implementation
{
    class App : public AppT2<App>
    {
    public:
        App();
        ~App();
    };
}

namespace winrt::MyApp::factory_implementation
{
    class App : public AppT<App, implementation::App>
    {
    };
}
```

App.idl

```
namespace MyApp
{
    [default_interface]
    runtimeclass App: Microsoft.Toolkit.Win32.UI.XamlHost.XamlApplication
    {
        App();
    }
}
```

14. Create app.base.h in this MyApp project, and use below content:

    app.base.h

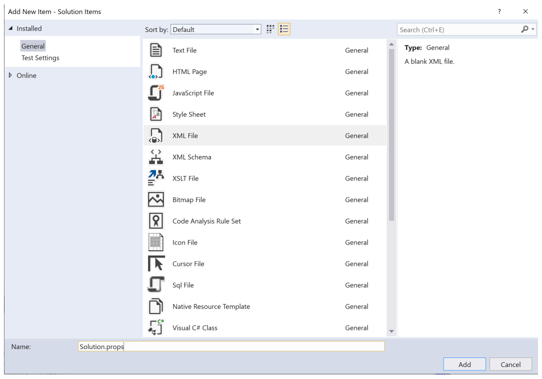
```
#pragma once

namespace winrt::MyApp::implementation
{
    template <typename D, typename... I>
    struct App_baseWithProvider : public App_base<D, ::winrt::Windows::UI::Xaml::Markup::IXamlMetadataProvider>
    {
        using IXamlType = ::winrt::Windows::UI::Xaml::Markup::IXamlType;

        IXamlType GetXamlType(::winrt::Windows::UI::Xaml::Interop::TypeName const& type)
        {
            return AppProvider()->GetXamlType(type);
        }

        IXamlType GetXamlType(::winrt::hstring const& fullName)
        {
            return AppProvider()->GetXamlType(fullName);
        }

        ::winrt::com_array<::winrt::Windows::UI::Xaml::Markup::XmlnsDefinition> GetXmlnsDefinitions()
        {
            return AppProvider()->GetXmlnsDefinitions();
        }

    private:
        bool _contentLoaded{ false };
        std::shared_ptr<XamlMetaDataProvider> _appProvider;
        std::shared_ptr<XamlMetaDataProvider> AppProvider()
        {
            if (!_appProvider)
            {
                _appProvider = std::make_shared<XamlMetaDataProvider>();
            }
            return _appProvider;
        }
    };

    template <typename D, typename... I>
    using AppT2 = App_baseWithProvider<D, I...>;
}
```

15. Install Microsoft.Toolkit.Win32.UI.XamlApplication Nuget package.
16. If you build ***MyApp*** now, it should create MyApp.dll without any error.

Centralize Output, Input and C++/WinRT files in Solution

This step is necessary for our next steps because we need to include winrt header files in different projects properly, and MFCApp also needs to reference MyApp resource files.

1. Add a new ***Solution.Props*** file by right clicking the ***solution*** node, and select Add -> New Item:

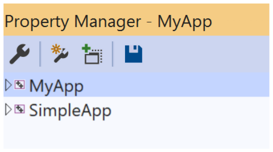


Name: Solution.props

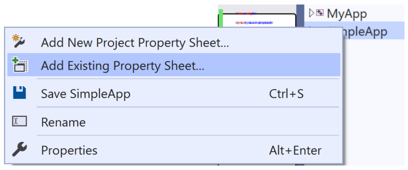
2. Use below content to overwrite the Solution.Props:

```
<?xml version="1.0" encoding="utf-8"?>
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <IntDir>$(SolutionDir)\obj\$(Platform)\$(Configuration)\$(MSBuildProjectName)\</IntDir>
    <OutDir>$(SolutionDir)\bin\$(Platform)\$(Configuration)\$(MSBuildProjectName)\</OutDir>
    <GeneratedFilesDir>$(IntDir)Generated Files\</GeneratedFilesDir>
  </PropertyGroup>
</Project>
```

3. Click Views -> Other Windows -> Property Manager



4. Right click ***SimpleApp***, select ***Add Existing Property Sheet***, add the new ***solution.props*** file


MyApp
pleApp
Add New Project Property Sheet...
Add Existing Property Sheet...
Save SimpleApp Ctrl+S
Rename
Properties Alt+Enter

5. Repeat the step 4. for ***MyApp***. We can close the Property Manager window now.

6. Right click the project node ***MFCApp***, select Properties.

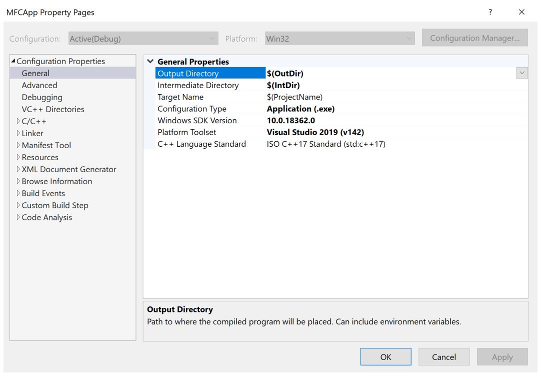
   If you see the Outut and Intermidiate directries are already become as below, then can skip this step:

   $(SolutionDir)$(Platform)$(Configuration)$(MSBuildProjectName)
   and $(SolutionDir)$(Platform)$(Configuration)$(MSBuildProjectName)

   Otherwise please manually set:

   Output Directory: $(OutDir)

   Intermidia Directory: $(IntDir)



Configuration: Active(Debug) Platform: Win32

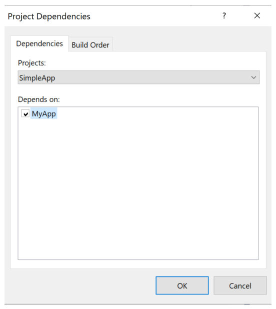
General Properties

Output Directory	$(OutDir)
Intermediate Directory	$(IntDir)
Target Name	$(ProjectName)
Configuration Type	Application (.exe)
Windows SDK Version	10.0.18362.0
Platform Toolset	Visual Studio 2019 (v142)
C++ Language Standard	ISO C++17 Standard (std:c++17)

Output Directory
Path to where the compiled program will be placed. Can include environment variables.

7. Repeat the step 6 for ***MyApp***.

8. Right click the Solution node, and choose ***Project Dependencies***, make sure SimpleApp depends on MyApp:



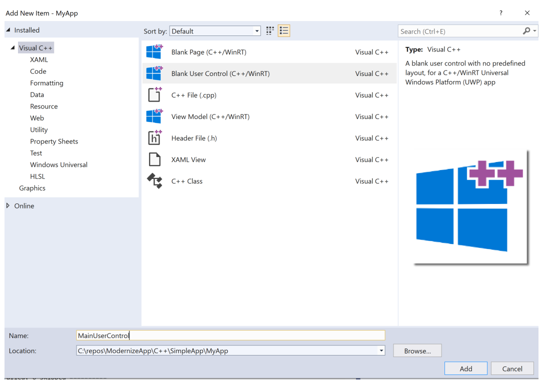
Projects: SimpleApp

Depends on: ✓ MyApp

9. Rebuild the whole solution, it should work without errors.

Add UWP Custom XAML Control to MyApp

1. Right click ***MyApp***, select Add -> New Item
2. Create ***Blank User Conrol (C++/WinRT)***, here we call it MainUserControl:



Type: Visual C++

A blank user control with no predefined layout, for a C++/WinRT Universal Windows Platform (UWP) app

Name: MainUserControl

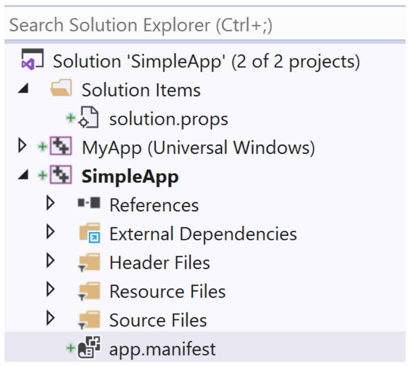
Location: C:\repos\ModernizeApp\C++\SimpleApp\MyApp

Integrate Custom XAML Control in SimpleApp

1. Add one XML file **app.manifest** in your project with below content to register custom control type:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<assembly
    xmlns="urn:schemas-microsoft-com:asm.v1"
    xmlns:asmv3="urn:schemas-microsoft-com:asm.v3"
    manifestVersion="1.0">
    <asmv3:file name="MyApp.dll">
        <activatableClass
            name="MyApp.App"
            threadingModel="both"
            xmlns="urn:schemas-microsoft-com:winrt.v1" />
        <activatableClass
            name="MyApp.XamlMetadataProvider"
            threadingModel="both"
            xmlns="urn:schemas-microsoft-com:winrt.v1" />
        <activatableClass
            name="MyApp.MainUserControl"
            threadingModel="both"
            xmlns="urn:schemas-microsoft-com:winrt.v1" />
    </asmv3:file>
</assembly>
```

2. Now the project structure is like as below:



3. Right click the Win32 Project ***SimpleApp***, select ***Unload Project***
4. Right click the ***SimpleApp (Unloaded)*** project, select ***Edit SimpleApp.vcxproj***

5. Remove the three lines from the ***SimpleApp.vcxproj*** project file:

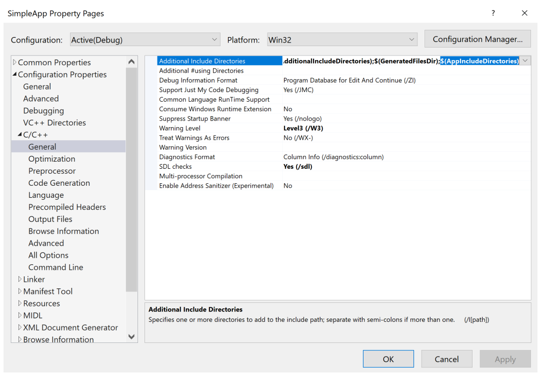
```
<ItemGroup>
    <Manifest Include="app.manifest" />
</ItemGroup>
```

Add below properties to the ***SimpleApp.vcxproj*** project file before the ***"<Import Project=""$(VCTargetsPath).Cpp.targets" />"*** line:

```
  <ItemGroup>
    <Manifest Include="app.manifest" />
    <AppxManifest Include="$(SolutionDir)\bin\$(Platform)\$(Configuration)\$(AppProjectName)\AppxManifest.xml" />
  </ItemGroup>
  <PropertyGroup>
    <AppProjectName>MyApp</AppProjectName>
  </PropertyGroup>
  <PropertyGroup>
    <AppIncludeDirectories>$(SolutionDir)\obj\$(Platform)\$(Configuration)\$(AppProjectName)\;$(SolutionDir)\obj\$(Platform)\$(Configuration)\$(AppPr
  </PropertyGroup>
    <ItemGroup>
    <NativeReferenceFile Include="$(SolutionDir)\bin\$(Platform)\$(Configuration)\$(AppProjectName)\*.xbf">
      <DeploymentContent>true</DeploymentContent>
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </NativeReferenceFile>
    <NativeReferenceFile Include="$(SolutionDir)\bin\$(Platform)\$(Configuration)\$(AppProjectName)\*.dll">
      <DeploymentContent>true</DeploymentContent>
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </NativeReferenceFile>
    <NativeReferenceFile Include="$(SolutionDir)\bin\$(Platform)\$(Configuration)\$(AppProjectName)\resources.pri">
      <DeploymentContent>true</DeploymentContent>
      <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
    </NativeReferenceFile>
  </ItemGroup>
  <------Right Here--------->
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
```

6. Right click the ***SimpleApp (Unloaded)*** project, select ***Reload Project***.

7. Right click ***SimpleApp***, select ***Properties***, setup ***$(AppIncludeDirectories)*** as a part of include file path, this macro has been defined in the above project file:



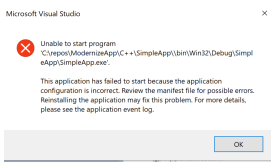
SimpleApp Property Pages

Configuration: Active(Debug) Platform: Win32

```
Additional Include Directories          .dditionalIncludeDirectories);$(GeneratedFilesDir);$(AppIncludeDirectories)
Additional #using Directories
Debug Information Format                Program Database for Edit And Continue (/ZI)
Support Just My Code Debugging          Yes (/JMC)
Common Language RunTime Support
Consume Windows Runtime Extension       No
Suppress Startup Banner                 Yes (/nologo)
Warning Level                           Level3 (/W3)
Treat Warnings As Errors                No (/WX-)
Warning Version
Diagnostics Format                      Column Info (/diagnostics:column)
SDL checks                              Yes (/sdl)
Multi-processor Compilation
Enable Address Sanitizer (Experimental) No
```

Additional Include Directories
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

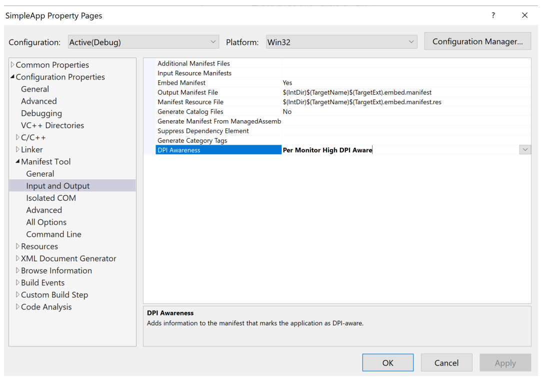
8. Set ***"Per Monitor DPI Aware"*** for ***DPI Awareness*** otherwise you may be not able to start this SimpleApp when it is ***"High DPI Aware"*** and hit configuration error in Manifest:



Microsoft Visual Studio

Unable to start program 'C:\repos\ModernizeApp\C++\SimpleApp\\bin\Win32\Debug\SimpleApp\SimpleApp.exe'.

This application has failed to start because the application configuration is incorrect. Review the manifest file for possible errors. Reinstalling the application may fix this problem. For more details, please see the application event log.



SimpleApp Property Pages

Configuration: Active(Debug) Platform: Win32

```
Additional Manifest Files
Input Resource Manifests
Embed Manifest                      Yes
Output Manifest File                $(IntDir)$(TargetName)$(TargetExt).embed.manifest
Manifest Resource File              $(IntDir)$(TargetName)$(TargetExt).embed.manifest.res
Generate Catalog Files              No
Generate Manifest From ManagedAssemb
Suppress Dependency Element
Generate Category Tags
DPI Awareness                       Per Monitor High DPI Aware
```

DPI Awareness
Adds information to the manifest that marks the application as DPI-aware.

9. Open framework.h, remove this #define:

```
#define WIN32_LEAN_AND_MEAN
```

10. In SimpleApp.h, add below code to include necessary winrt header files:

```
#include <winrt/Windows.Foundation.Collections.h>
#include <winrt/Windows.system.h>
#include <winrt/windows.ui.xaml.hosting.h>
#include <windows.ui.xaml.hosting.desktopwindowxamlsource.h>
#include <winrt/windows.ui.xaml.controls.h>
#include <winrt/Windows.ui.xaml.media.h>
#include <winrt/Windows.UI.Core.h>
#include <winrt/myapp.h>
```

11. Using winrt namespaces in SimpleAPP.cpp

```
using namespace winrt;
using namespace Windows::UI;
using namespace Windows::UI::Composition;
using namespace Windows::UI::Xaml::Hosting;
using namespace Windows::Foundation::Numerics;
using namespace Windows::UI::Xaml::Controls;
```

12. Declare hostApp, _desktopWindowXamlSource and our custom control in ***SimpleApp.cpp***

```
     // Global Variables:
    HINSTANCE hInst;
    // current instance
    WCHAR szTitle[MAX_LOADSTRING];
    // The title bar text
    WCHAR szWindowClass[MAX_LOADSTRING];
    // the main window class name


    winrt::MyApp::App hostApp{ nullptr };
    winrt::Windows::UI::Xaml::Hosting::DesktopWindowXamlSource _desktopWindowXamlSource{ nullptr };
    winrt::MyApp::MainUserControl _mainUserControl{ nullptr };
```

13. Initalize hostApp, _desktopWindowXamlSource in ***wWinMain*** in ***SimpleApp.CPP***

```
    winrt::init_apartment(winrt::apartment_type::single_threaded);
    hostApp = winrt::MyApp::App{};
    _desktopWindowXamlSource = winrt::Windows::UI::Xaml::Hosting::DesktopWindowXamlSource{};
```

14. Add below code in ***InitInstance*** in ***SimpleApp.cpp***

```
     if (_desktopWindowXamlSource != nullptr)
    {
        // Get handle to corewindow
        auto interop = _desktopWindowXamlSource.as<IDesktopWindowXamlSourceNative>();
        // Parent the DesktopWindowXamlSource object to current window
        check_hresult(interop->AttachToWindow(hWnd));

        // This Hwnd will be the window handler for the Xaml Island: A child window that contains Xaml.
        HWND hWndXamlIsland = nullptr;
        // Get the new child window's hwnd
        interop->get_WindowHandle(&hWndXamlIsland);

        RECT windowRect;
        ::GetWindowRect(hWnd, &windowRect);
        ::SetWindowPos(hWndXamlIsland, NULL, 0, 0, windowRect.right - windowRect.left, windowRect.bottom - windowRect.top, SWP_SHOWWINDOW);

        _mainUserControl = winrt::MyApp::MainUserControl();
        _desktopWindowXamlSource.Content(_mainUserControl);
    }
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);
```

15. Clean up resources when the view is disconstructed in MFCAppView.cpp

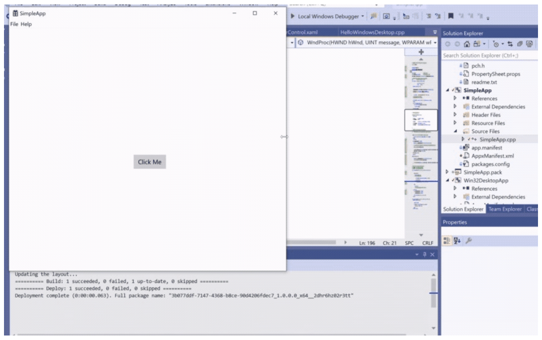
```
 case WM_DESTROY:
    PostQuitMessage(0);
    if (_desktopWindowXamlSource != nullptr)
    {
        _desktopWindowXamlSource.Close();
        _desktopWindowXamlSource = nullptr;
    }
    break;
```

16. Add AdjustLayout function to make XAML content layout properly in SimpleApp.cpp :

```
void AdjustLayout(HWND hWnd)
{
    if (_desktopWindowXamlSource != nullptr)
    {
        auto interop = _desktopWindowXamlSource.as<IDesktopWindowXamlSourceNative>();
        HWND xamlHostHwnd = NULL;
        check_hresult(interop->get_WindowHandle(&xamlHostHwnd));

        RECT windowRect;
        ::GetWindowRect(hWnd, &windowRect);
        ::SetWindowPos(xamlHostHwnd, NULL, 0, 0, windowRect.right - windowRect.left, windowRect.bottom - windowRect.top, SWP_SHOWWINDOW);
    }
}
```

17. In SimpleApp.CPP handle WM_SIZE in the ***WinProc*** function: `C++ case WM_SIZE: AdjustLayout(hWnd);`

18. Now you can build and run this SimpleApp. It should display a button in the central of view window:

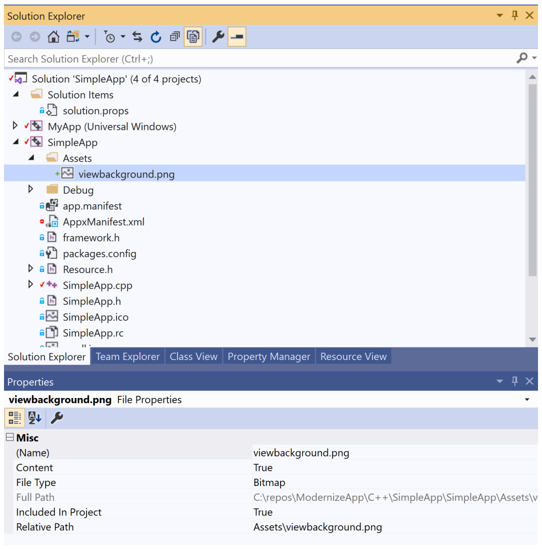


```
Updating the layout...
========== Build: 1 succeeded, 0 failed, 1 up-to-date, 0 skipped ==========
========== Deploy: 1 succeeded, 0 failed, 0 skipped ==========
Deployment complete (0:00:00.063). Full package name: "3b077ddf-7147-4368-b8ce-90d4206fdec7_1.0.0.0_x64__2dhr6hz02r3tt"
```

Using Ink Control in MyApp UWP Project

1. Add an ***Assets*** file folder under ***SimpleApp*** on file system, add viewbackground.png file. And then create a new filter of "Assets", add this existing viewbackground.png into the ***Assets*** filter:

   Set the file's **Content** property as **True**

(Name)	viewbackground.png
Content	True
File Type	Bitmap
Full Path	C:\repos\ModernizeApp\C++\SimpleApp\SimpleApp\Assets\v
Included In Project	True
Relative Path	Assets\viewbackground.png

   If you cannot see the ***viewbackground.png*** file, please click the "Show All Files" tool icon in the solution explorer window.

2. Add the sample png file into ***Assets*** folder of ***MyApp***, and make its ***Content*** property as **True** as well.

3. Modify MainUserControl.Xaml as below in ***MyApp***:

```
<UserControl
    x:Class="MyApp.MainUserControl"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:MyApp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <RelativePanel HorizontalAlignment="Stretch" VerticalAlignment="Stretch">
        <Image Source="assets/viewbackground.png" RelativePanel.AlignLeftWithPanel="True"
               RelativePanel.AlignRightWithPanel="True"
               RelativePanel.AlignHorizontalCenterWithPanel="True"></Image>
        <InkCanvas x:Name="ic" RelativePanel.AlignLeftWithPanel="True"
                   RelativePanel.AlignRightWithPanel="True"
                   RelativePanel.AlignBottomWithPanel="True"
                   RelativePanel.AlignTopWithPanel="True"></InkCanvas>
        <InkToolbar x:Name="it" HorizontalAlignment="Left" VerticalAlignment="Top"></InkToolbar>
    </RelativePanel>
</UserControl>
```

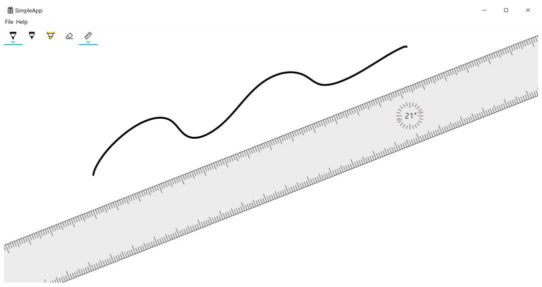
4. Add below code in MainUserControl.CPP, remove the Button ClickHandler code as we don't use it now:

```
    MainUserControl::MainUserControl()
    {
        InitializeComponent();
        ic().InkPresenter().InputDeviceTypes((winrt::Windows::UI::Core::CoreInputDeviceTypes)7);
        it().TargetInkCanvas(ic());
    }
```

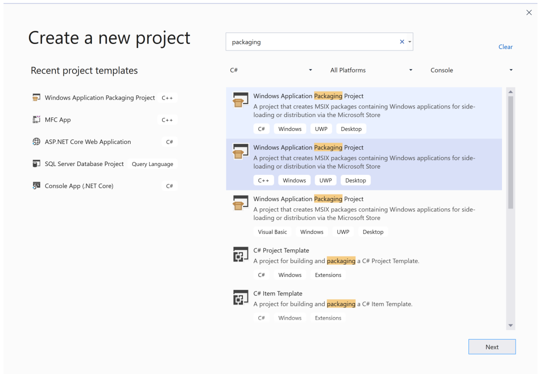
5. Include more winrt header files in MainUserControl.h:

```
#include <winrt/Windows.UI.Input.Inking.h>
#include <winrt/Windows.UI.Xaml.Media.Imaging.h>
```

6. Build and run Simple, if steps have been taken exactly as above, it will show as below:



7. You may notice that it doesn't display the background image, this is because the uri **assets/viewbackground.png** needs to be used UWP package. In Visual Studio, with **"Windows Application Packaging Project (C++)"**, it is easily to packaging our SimpleApp project:



Create a new project

packaging

Clear

Recent project templates

Windows Application Packaging Project	C++
MFC App	C++
ASP.NET Core Web Application	C#
SQL Server Database Project	Query Language
Console App (.NET Core)	C#

C# · All Platforms · Console

Windows Application Packaging Project
A project that creates MSIX packages containing Windows applications for side-loading or distribution via the Microsoft Store
C# Windows UWP Desktop

Windows Application Packaging Project
A project that creates MSIX packages containing Windows applications for side-loading or distribution via the Microsoft Store
C++ Windows UWP Desktop

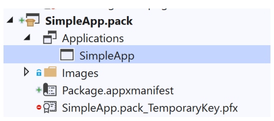
Windows Application Packaging Project
A project that creates MSIX packages containing Windows applications for side-loading or distribution via the Microsoft Store
Visual Basic Windows UWP Desktop

C# Project Template
A project for building and packaging a C# Project Template.
C# Windows Extensions

C# Item Template
A project for building and packaging a C# Item Template.
C# Windows Extensions

Next

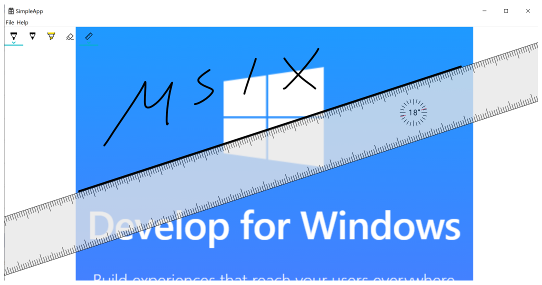
Create the packaging project in the solution, right click the **Application** node, and select **Add Reference**, add SimpleApp. Now the packaging project structure is like:



```
SimpleApp.pack
  Applications
    SimpleApp
  Images
  Package.appxmanifest
  SimpleApp.pack_TemporaryKey.pfx
```

For more information about packaging project, refer to: Package a desktop app from source code using Visual Studio

8. After this, choose the packaging project as Start Up project, Ctrl+F5 to run it. We can see the expected result will show up:



Further more, you can publish this packaging app as MSIX or APPX, and easily deploy it:

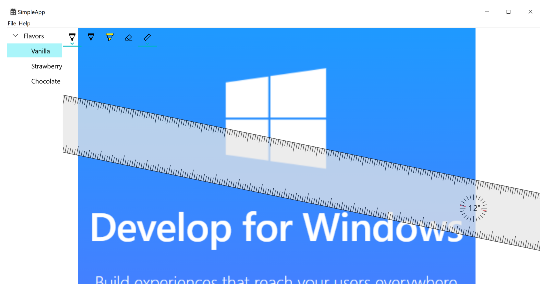
Package a UWP app with Visual Studio

Wrap Up

This article gives detailed steps on how to leverage XamplApplication to host standard XAML control in C++ Win32 project, with this method, it is flexible to leverage Win10 Controls in Win32 project. The whole sample solution can be found from this repo: https://github.com/freistli/ModernizeApp/tree/master/C%2B%2B/SimpleApp

Further Step

To use WinUI 2.x, you can refer to this part and embedded it in your C++ project. The sample solution is also ready from this repo: https://github.com/freistli/ModernizeApp/tree/master/C%2B%2B/SimpleAppWithWinUI/SimpleApp. Its left TreeView is from WinUI 2.4 prerelease:


SimpleApp
File Help
Flavors
Vanilla
Strawberry
Chocolate
12°
Develop for Windows