# Visual Studio 2022扩展开发入门

## 课前准备（1）

1. 认识VS扩展

|  |  |
| --- | --- |
| <Grid>  <Button  Width="85"  **Background="Azure"**  **Foreground="Black"**  Height="42"  Margin="95,102,0,0"  HorizontalAlignment="Left"  VerticalAlignment="Top"  Content="Button" />  </Grid> | <Grid>  <Button  Width="85"  Height="42"  Margin="95,102,0,0"  HorizontalAlignment="Left"  VerticalAlignment="Top"  **Background="Azure"**  Content="Button"  **Foreground="Black"** />  </Grid> |

<https://github.com/Xavalon/XamlStyler/>

1. 扩展开发环境搭建

C:\Users\Zmrbak\.nuget\packages\microsoft.vssdk.buildtools\15.9.3032\tools\VSSDK\Microsoft.VsSDK.targets(752,5): error : Extension '0c572937-0d4a-48cc-aa3f-875ffa9805ba' is of the same or lower version than the installed extension. Please make sure that the extension version is higher than that of the installed extension.

========== 全部重新生成: 成功 2 个，失败 1 个，跳过 0 个 ==========

* 卸载已安装的扩展
* **提供新版本号**

**source.extension.vsixmanifest**

<Identity Id="**8C9E25DA-8380-484D-8BDC-5D8120A07E19**" Version="3.2.0.0" Language="en-US" Publisher="Xavalon" />

* 不予理会

1. **Visual Studio 2022** <http://www.microsoft.com>

工作负荷：**Visual Studio 扩展开发**

5>C:\Users\Zmrbak\.nuget\packages\microsoft.vssdk.buildtools\15.9.3032\tools\VSSDK\Microsoft.VsSDK.targets(87,5): error MSB4062: 未能从程序集 C:\Users\Zmrbak\.nuget\packages\microsoft.vssdk.buildtools\15.9.3032\tools\VSSDK\Microsoft.VisualStudio.Sdk.BuildTasks.15.0.dll 加载任务“CompareBuildTaskVersion”。

Could not load file or assembly 'file:///C:\Users\Zmrbak\.nuget\packages\microsoft.vssdk.buildtools\15.9.3032\tools\VSSDK\Microsoft.VisualStudio.Sdk.BuildTasks.15.0.dll' or one of its dependencies.

试图加载格式不正确的程序。 请确认 <UsingTask> 声明正确，该程序集及其所有依赖项都可用，并且该任务包含实现 Microsoft.Build.Framework.ITask 的公共类。

1. 本课程目的：移植XamlStyler扩展到Visual Studio 2022
2. 课程源代码：<https://github.com/zmrbak/VSIX2022>
3. 如何下载课程源代码

## 课前准备（2）

## VSIX项目文件结构（1）

目标框架：.NET Framework 4.7.2

<Assets>

<Asset

Type="**Microsoft.VisualStudio.VsPackage**"

d:Source="Project"

d:ProjectName="%CurrentProject%"

Path="|%CurrentProject%;PkgdefProjectOutputGroup|" />

</Assets>

C:\Users\lib\**.nuget**\packages\microsoft.visualstudio.shell.15.0\17.0.0-previews-1-31410-258\lib\net472\Microsoft.VisualStudio.**Shell**.15.0.dll

## VSIX项目文件结构（2）

## 向VS菜单中添加菜单项（1）

Microsoft.VisualStudio.Interop.dll

namespace Microsoft.VisualStudio.Shell.Interop

{

public enum OLEMSG**ICON**

{

OLEMSGICON\_NOICON,

OLEMSGICON\_CRITICAL,

OLEMSGICON\_QUERY,

OLEMSGICON\_WARNING,

OLEMSGICON\_INFO

}

}

namespace Microsoft.VisualStudio.Shell.Interop

{

public enum OLEMSG**BUTTON**

{

OLEMSGBUTTON\_OK,

OLEMSGBUTTON\_OKCANCEL,

OLEMSGBUTTON\_ABORTRETRYIGNORE,

OLEMSGBUTTON\_YESNOCANCEL,

OLEMSGBUTTON\_YESNO,

OLEMSGBUTTON\_RETRYCANCEL,

OLEMSGBUTTON\_YES**ALL**NOCANCEL

}

}

namespace Microsoft.VisualStudio.Shell.Interop

{

public enum OLEMSG**DEF**BUTTON

{

OLEMSGDEFBUTTON\_FIRST,

OLEMSGDEFBUTTON\_SECOND,

OLEMSGDEFBUTTON\_THIRD,

OLEMSGDEFBUTTON\_FOURTH

}

}

C:\Program Files\Microsoft Visual Studio\2022\Preview\**VS**SDK\VisualStudioIntegration\Common\Inc

## 向VS菜单中添加菜单项（2）

## 向VS菜单中添加菜单项（3）

## 向VS中注册一个事件（1）

Comment Remover

<https://github.com/madskristensen/CommentRemover>

[ProvideAutoLoad(VSIX08Package.UIContextGuidString, PackageAutoLoadFlags.BackgroundLoad)]

[ProvideUIContextRule(VSIX08Package.UIContextGuidString, name: "XAML load", expression: "Dotxaml", termNames: new[] { "Dotxaml" }, termValues: new[] { "HierSingleSelectionName:.xaml$" })]

VSConstants

CommandEvents

DTE

## 向VS中注册一个事件（2）

## 命令菜单组与命令ID

<https://github.com/madskristensen/ExtensibilityTools>

HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\VisualStudio\17.0abc\General

EnableVSIPLogging DWORD with value 1

关闭注册表编辑器

重新启动VS2022

Hold Ctrl + Shift and interact with IDE

<https://github.com/madskristensen/CommandTableInfo/>

<GuidSymbol name="guidVSIX10PackageCmdSet" value="{5f0f5edc-575e-4d1e-8696-af66dc792868}">

<IDSymbol name="MyMenuGroup" value="0x1020" />

<IDSymbol name="Command1Id" value="0x0100" />

</GuidSymbol>

public static readonly Guid CommandSet = new Guid("5f0f5edc-575e-4d1e-8696-af66dc792868");

5f0f5edc-575e-4d1e-8696-af66dc792868

5f0f5edc-575e-4d1e-8696-af66dc792868

public const int CommandId = 0x0100;

<IDSymbol name="Command1Id" value="0x0100" />

## VS扩展开发“扩展套装”：Extensibility Essentials（1）

Extensibility Essentials 2022

1. VSIX Synchronizer
2. Insert Guid
3. Image Optimizer
4. Command Explorer
5. Settings Store Explorer
6. KnownMonikers Explorer

**<Include href="KnownImageIds.vsct"/>**

<Button guid="guidVSIX11PackageCmdSet" id="Command1Id" priority="0x0100" type="Button">

<Parent guid="guidVSIX11PackageCmdSet" id="MyMenuGroup" />

**<Icon guid="ImageCatalogGuid" id="Action" />**

**<CommandFlag>IconIsMoniker</CommandFlag>**

<Strings>

<ButtonText>VSXI11.1 Command</ButtonText>

</Strings>

</Button>

1. Clean MEF Component Cache
2. VSCT IntelliSense
3. Image Manifest Tools
4. **Extensibility Logs**
5. Pkgdef Language
6. Extensibility Template Pack

Comment Remover

## VS扩展开发“扩展套装”：Extensibility Essentials（2）

## VS扩展开发“扩展套装”：Extensibility Essentials（3）

## 向菜单命令绑定键盘快捷键

CTRL+1,CTRL+2

<KeyBindings>

<KeyBinding guid="guidVSIX14Package**CmdSet**" id="**Command1Id**" editor="**guidVSStd97**" key1="1" mod1="CONTROL" key2="2" mod2="CONTROL">

</KeyBinding>

</KeyBindings>

## 向VS菜单添加多条命令(1)

%comspec% /C "C:\Program Files\Microsoft Visual Studio\2022\Preview\VSSDK\VisualStudioIntegration\Tools\Bin\Create**Exp**Instance.exe" /Reset /VSInstance=17.0\_dee98566 /RootSuffix=Exp && PAUSE

* Removed directory C:\Users\lib\AppData\Local\Microsoft\VisualStudio\17.0\_dee98566Exp.
* Created directory C:\Users\lib\AppData\Local\Microsoft\VisualStudio\17.0\_dee98566Exp\Extensions.
* Copied Extensions directory from C:\Users\lib\AppData\Local\Microsoft\VisualStudio\17.0\_dee98566\Extensions to C:\Users\lib\AppData\Local\Microsoft\VisualStudio\17.0\_dee98566Exp\Extensions\Extensions-17.0\_dee98566.

**VSIX15Package.vsct**

<Symbols>

<GuidSymbol name="guidVSIX15PackageCmdSet" value="{9a9f54d2-caca-4df5-b9c2-afedd6b99890}">

**<IDSymbol name="cmdidCommand3" value="0x1022"/>**

</GuidSymbol>

</Symbols>

<Buttons>

<Button guid="guidVSIX15PackageCmdSet" id="**cmdidCommand3**" priority="0x0100" type="Button">

<Parent guid="guidVSIX15PackageCmdSet" id="**MyMenuGroup**" />

<Strings>

<ButtonText>Invoke Command3</ButtonText>

</Strings>

</Button>

</Buttons>

<Groups>

<Group guid="guidVSIX15PackageCmdSet" id="**MyMenuGroup**" priority="0x0600">

<Parent guid="guidSHLMainMenu" id="IDM\_VS\_MENU\_TOOLS" />

</Group>

</Groups>

Group(id) -> Button(id)

**IDSymbol -> id**

VSIX15Package.cs

protected override async Task InitializeAsync(CancellationToken cancellationToken, IProgress<ServiceProgressData> progress)

{

await this.JoinableTaskFactory.SwitchToMainThreadAsync(cancellationToken);

await Command1.InitializeAsync(this);

await Command2.InitializeAsync(this);

**await Command3.InitializeAsync(this);**

}

Command3.cs

internal sealed class Command3｛

｝

Priority 决定命令的排列顺序，数字越小，则越靠上，数字越大越靠下。

## 向VS菜单添加多条命令(2)

## 向VS菜单添加多级菜单

Group

Button

Button

Group

Menu

Group

Group

Button

## 命令组与命令的重复使用

<CommandPlacements>

<CommandPlacement guid="guidVSIX18PackageCmdSet" id="cmdidCommand3" priority="100">

<Parent guid="guidVSIX18PackageCmdSet" id="**MyMenuGroup1**" />

</CommandPlacement>

</CommandPlacements>

## 向子菜单添加动态菜单（1）

<Button guid="guidVSIX19PackageCmdSet" id="Command1Id" priority="0x0100" type="Button">

<Parent guid="guidVSIX19PackageCmdSet" id="MRUGroup" />

**<CommandFlag>DynamicItemStart</CommandFlag>**

<Strings>

**<ButtonText></ButtonText>**

</Strings>

</Button>

//动态菜单数量

private int mruNumber = 10;

//菜单名称列表

private List<string> mruList;

private void InitMruMenu(OleMenuCommandService commandService)

{

mruList = new List<string>();

for (int i = 0; i < mruNumber; i++)

{

string message = string.Format(CultureInfo.CurrentCulture, "Item {0}", i + 1);

mruList.Add(message);

var menuCommandID = new CommandID(CommandSet, CommandId + i);

var menuItem = new **OleMenuCommand**(this.Execute, menuCommandID);

menuItem.Text = message;

menuItem.**BeforeQueryStatus** += MenuItem\_BeforeQueryStatus;

commandService.AddCommand(menuItem);

}

}

## 向子菜单添加动态菜单（2）

## 更改VS扩展的菜单标题和状态（1）

<Button guid="guidVSIX21PackageCmdSet" id="Command1Id" priority="0x0100" type="Button">

<Parent guid="guidVSIX21PackageCmdSet" id="MyMenuGroup" />

**<CommandFlag>TextChanges</CommandFlag>**

<Strings>

<ButtonText>VSIX21 Invoke Command1</ButtonText>

</Strings>

</Button>

var menuCommandID = new CommandID(CommandSet, CommandId);

var menuItem = new **OleMenuCommand**(this.Execute, menuCommandID);

menuItem.BeforeQueryStatus += MenuItem\_BeforeQueryStatus;

commandService.AddCommand(menuItem);

OleMenuCommand menuCommand = **sender as OleMenuCommand**;

menuCommand.Text = "item" + index;

var menuCommandID = new CommandID(CommandSet, PackageIds.cmdidCommand2);

var commandService = package.**GetService**<IMenuCommandService, OleMenuCommandService>();

var menuCommand1 = commandService.FindCommand(menuCommandID) as **OleMenuCommand**;

menuCommand1.Text = "item" + index;

## 更改VS扩展的菜单标题和状态（1）

## 菜单命令的可见性约束

<VisibilityConstraints>

<VisibilityItem guid="guidVSIX23PackageCmdSet" id="**Command1Id**" context="UICONTEXT\_**SolutionExists**">

</VisibilityItem>

</VisibilityConstraints>

<Button guid="guidVSIX23PackageCmdSet" id="Command1Id" priority="0x0100" type="Button">

<Parent guid="guidVSIX23PackageCmdSet" id="MyMenuGroup" />

<Icon guid="guidImages" id="bmpPic1" />

<CommandFlag>**DynamicVisibility**</CommandFlag>

<Strings>

<ButtonText>VSIX23 Invoke Command1</ButtonText>

</Strings>

</Button>

<Group guid="guidVSIX23PackageCmdSet" id="MyMenuGroup" priority="0x0600">

<Parent guid="guidSHLMainMenu" id="**IDM\_VS\_CTXT\_ITEMNODE**"/>

</Group>

var menuCommandID = new CommandID(CommandSet, CommandId);

var menuItem = new **OleMenuCommand**(this.Execute, menuCommandID) { **Supported** = false };

[ProvideUIContextRule(

contextGuid: VSConstants.UICONTEXT.**SolutionExists\_string**,

name: "CSharp",

expression: "VB | CS",

termNames: new string[] { "VB", "CS" },

termValues: new string[] { "HierSingleSelectionName:.vb$","HierSingleSelectionName:.cs$" }

)]

## 向VS工具栏添加按钮

<Group guid="guidVSIX24PackageCmdSet" id="MyMenuGroup" priority="0x0600">

<Parent guid="**VSMainMenu**" id="**StandardToolbar**" />

</Group>

<Menu guid="guidVSIX24PackageCmdSet" id="**MyMenu**" priority="100" type="**MenuController**">

<Parent guid="guidVSIX24PackageCmdSet" id="**MyMenuGroup**"/>

<Strings>

<ButtonText>string</ButtonText>

</Strings>

</Menu>

<Group guid="guidVSIX24PackageCmdSet" id="**SubMenuGroup**" priority="100">

<Parent guid="guidVSIX24PackageCmdSet" id="**MyMenu**"/>

</Group>

<Button guid="guidVSIX24PackageCmdSet" id="cmdidCommand2" priority="0x0100" type="Button">

<Parent guid="guidVSIX24PackageCmdSet" id="**SubMenuGroup**" />

<Icon guid="guidImages1" id="bmpPic1" />

<CommandFlag>**FixMenuController**</CommandFlag>

<Strings>

<ButtonText>Invoke Command2</ButtonText>

</Strings>

</Button>

## 向VS添加可停靠工具栏

<Menus>

<Menu guid="guidVSIX25PackageCmdSet" id="MyMenu" priority="100" type="**Toolbar**">

<CommandFlag>**DefaultDocked**</CommandFlag>

<Strings>

<ButtonText>测试 ToolBar</ButtonText>

</Strings>

</Menu>

</Menus>

<IDSymbol name="MyMenu" value="0x0101" />

<Group guid="guidVSIX25PackageCmdSet" id="**MyMenuGroup**" priority="0x0600">

<Parent guid="guidVSIX25PackageCmdSet" id="**MyMenu**"/>

</Group>

<Button guid="guidVSIX25PackageCmdSet" id="Command1Id" priority="0x0100" type="Button">

<Parent guid="guidVSIX25PackageCmdSet" id="**MyMenuGroup**" />

<Icon guid="guidImages" id="bmpPic1" />

<Strings>

<ButtonText>Invoke 测试 Command1Id</ButtonText>

</Strings>

</Button>

## 向VS添加工具窗口（1）

<https://www.scootersoftware.com/>

//异步

private void **Execute**(object sender, EventArgs e)

{

this.package.JoinableTaskFactory.**RunAsync**(async delegate

{

ToolWindowPane window = await this.package.ShowToolWindowAsync(typeof(ToolWindow1), 0, true, this.package.DisposalToken);

if ((null == window) || (null == window.Frame))

{

throw new NotSupportedException("Cannot create tool window");

}

});

}

//同步

private void **Execute**(object sender, EventArgs e)

{

ThreadHelper.ThrowIfNotOnUIThread();

ToolWindowPane window = this.package.**FindToolWindow**(typeof(ToolWindow1), 0, true);

if ((null == window) || (null == window.Frame))

{

throw new NotSupportedException("Cannot create tool window");

}

IVsWindowFrame windowFrame = (IVsWindowFrame)window.Frame;

Microsoft.VisualStudio.ErrorHandler.ThrowOnFailure(windowFrame.Show());

}

<Button guid="guidVSIX26PackageCmdSet" id="ToolWindow1CommandId" priority="0x0100" type="Button">

<Parent guid="guidSHLMainMenu" id="**IDG\_VS\_WNDO\_OTRWNDWS1**"/>

<Icon guid="guidImages" id="bmpPic1" />

<Strings>

<ButtonText>VSIX26 ToolWindow1</ButtonText>

</Strings>

</Button>

Button-> Execute-> ToolWindow1-> ToolWindow1Control->XAML/CS

## 向VS添加工具窗口（2）

## 为新添加的工具窗口添加工具栏（1）

<Menus>

<Menu guid="guidVSIX28PackageCmdSet" id="**MyMenu**" priority="100" type="**ToolWindowToolbar**">

<CommandFlag>**DefaultDocked**</CommandFlag>

<Strings>

<ButtonText>测试 Toolbar</ButtonText>

<CommandName>测试 Toolbar</CommandName>

</Strings>

</Menu>

</Menus>

<Groups>

<Group guid="guidVSIX28PackageCmdSet" id="MyMenuGroup" priority="0x0600">

<Parent guid="guidVSIX28PackageCmdSet" id="**MyMenu**" />

</Group>

</Groups>

<Button guid="guidVSIX28PackageCmdSet" id="Command1Id" priority="0x0100" type="Button">

<Parent guid="guidVSIX28PackageCmdSet" id="**MyMenuGroup**" />

<Icon guid="guidImages" id="bmpPicX" />

<Strings>

<ButtonText>VSIX28 Invoke Command1</ButtonText>

</Strings>

</Button>

<IDSymbol name="MyMenu" value="0x0101" />

<IDSymbol name="Command1Id2" value="0x0102" />

public ToolWindow1() : base(null)

{

this.Caption = "ToolWindow1";

this.Content = new ToolWindow1Control();

**this.ToolBar = new System.ComponentModel.Design.CommandID(new Guid("4ba44c83-504d-4b54-bfd3-27f478562ab3"), 0x0101);**

}

## 为新添加的工具窗口添加工具栏（2）

## 为新添加的工具窗口添加菜单控制器

## 为VS添加多实例工具窗口

[ProvideToolWindow(typeof(ToolWindow1), **MultiInstances** = true)]

private void Execute(object sender, EventArgs e)

{

**\_ =** this.package.JoinableTaskFactory.RunAsync(async delegate

{

for (int i = 0; **i** < 10; i++)

{

ToolWindowPane window = await this.package.ShowToolWindowAsync(typeof(ToolWindow1), **i**, **false**, this.package.DisposalToken);

if(window==null)

{

window = await this.package.ShowToolWindowAsync(typeof(ToolWindow1), **i**, **true**, this.package.DisposalToken);

if ((null == window) || (null == window.Frame))

{

throw new NotSupportedException("Cannot create tool window");

}

window.Caption = "Window: " + i;

break;

}

}

});

}

## 向为VS添加的工具窗口中添加右键菜单

1. 创建ToolWindow
2. 修改VSIX32Package.vsct

<Menus>

<Menu guid="guidVSIX32PackageCmdSet" id="MyMenu" priority="100" type="**Context**">

<Strings>

<ButtonText> VSIX32 MyMenu</ButtonText>

</Strings>

</Menu>

</Menus>

<Groups>

<Group guid="guidVSIX32PackageCmdSet" id="MyGroup" priority="100">

<Parent guid="guidVSIX32PackageCmdSet" id="MyMenu"/>

</Group>

</Groups>

<Button guid="guidVSIX32PackageCmdSet" id="Button2" priority="100" type="Button">

<Parent guid="guidVSIX32PackageCmdSet" id="MyGroup"/>

<Icon guid="guidImages" id="bmpPicArrows"/>

<Strings>

<ButtonText>Button1 Yellow</ButtonText>

</Strings>

</Button>

1. ToolWindow1.cs

public ToolWindow1() : base(null)

{

this.Caption = "ToolWindow1";

**OleMenuCommandService oleMenuCommandService = GetService(typeof(IMenuCommandService)) as OleMenuCommandService;**

this.Content = new ToolWindow1Control(**oleMenuCommandService**);

}

1. ToolWindow1Control.xaml.cs

private void MyToolWindow\_MouseRightButtonUp(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

if(OleMenuCommandService!=null)

{

var menuCommandID = new CommandID(PackageGuids.guidVSIX32PackageCmdSet, PackageIds.MyMenu);

**Point p=this.PointToScreen(e.GetPosition(this));**

OleMenuCommandService.**ShowContextMenu**(menuCommandID,(int)p.X,(int)p.Y);

}

}

## 为VS工具窗口添加搜索功能（1）

## 为VS工具窗口添加搜索功能（2）

## 增强搜索功能——搜索的选项

## 向添加的VS工具窗口添加订阅事件

public class ToolWindow1 : ToolWindowPane**,IVsRunningDocTableEvents**

{}

protected override void Initialize()

{

ThreadHelper.ThrowIfNotOnUIThread();

IVsRunningDocumentTable rdt=**this.GetService**(typeof(SVsRunningDocumentTable)) as IVsRunningDocumentTable;

if (rdt != null)

{

rdt.AdviseRunningDocTableEvents(this,out pdwCookie);

}

base.Initialize();

}

protected override void Dispose(bool disposing)

{

ThreadHelper.ThrowIfNotOnUIThread();

IVsRunningDocumentTable rdt = **Microsoft.VisualStudio.Shell.Package.GetGlobalService**(typeof(SVsRunningDocumentTable)) as IVsRunningDocumentTable;

if (rdt != null)

{

rdt.UnadviseRunningDocTableEvents(pdwCookie);

}

base.Dispose(disposing);

}

## 向VS“输出”窗口中输出信息（1）

DTE2 dTE= (DTE2)Package.GetGlobalService(typeof(DTE));

OutputWindowPanes panes = dTE.ToolWindows.**OutputWindow.OutputWindowPanes**;

IVsOutputWindow outputWindow=await **ServiceProvider.GetServiceAsync**(typeof(SVsOutputWindow)) as IVsOutputWindow;

**Assumes.Present**(outputWindow);

outputWindow.CreatePane(ref guid, paneTitle, Convert.ToInt32(true), Convert.ToInt32(false));

outputWindow.GetPane(ref guid, out outputWindowPane);

outputWindow.DeletePane(guid);

**DTE2 接口**

Visual Studio 自动化对象模型中的顶级对象。

myUIShell = myPackage.GetService(System.typeof(IVsUIShell)) as IVsUIShell;

//await this.**GetServiceAsync**(typeof(DTE));

**DTE 接口**

Visual Studio 自动化对象模型中的顶级对象。

**\_DTE 接口**

Visual Studio 自动化对象模型中的顶级对象。 有关此功能，请参考 \_DTE。 **不要从此类实例化**。

**Package 类**

提供创建完全正常运行的 VSPackage 所需的接口的托管实现。

**OutputWindowPanes 接口**

包含集成开发环境中的所有 " **输出** " 窗口窗格 (IDE)

**IVsOutputWindow 接口**

管理和控制特定于包含多个窗格的输出工具窗口的函数。

## 向VS“输出”窗口中输出信息（2）

## 向VS“输出”窗口中输出信息（3）

## 向VS“选项”窗口中添加扩展

internal class Class1:**DialogPage**

{

private int addDays;

public int **AddDays**

{

get { return addDays; }

set { addDays = value; }

}

}

[ProvideOptionPage(typeof(Class1),"My Options","My Page",101,106,true)]

public ToolWindow1() : base(null)

{

this.Caption = "ToolWindow1";

this.Content = new ToolWindow1Control(**this**);

}

package.**GetAutomationObject**("My Options"+"."+"My Page",out obj);

## 向VS添加编辑器分类器扩展（1）

IClassifier 接口

将 IClassificationType 对象分配给中的文本 ITextBuffer

IClassificationType 接口

文本范围的逻辑分类类型

IClassificationTypeRegistryService 接口

维护所有已知分类类型的集合的服务

ClassificationChangedEventArgs 类

提供有关 ClassificationChanged 事件的信息

ClassificationSpan 类

描述文本区域 IClassificationType

ClassificationTypeDefinition 类

描述用于声明分类类型的，仅用于数据的导出

EditorFormatDefinition 类

提供特定项的格式信息

ClassificationFormatDefinition 类

提供特定分类类型的分类格式信息

ExportAttribute 类

指定某个类型、属性、字段或方法提供特定的导出

ImportAttribute 类

指定属性、字段或参数值应由 CompositionContainer 对象提供。

特性化编程模型

<https://docs.microsoft.com/zh-cn/dotnet/framework/mef/>

ClassificationTypeAttribute 类

用于声明逻辑分类类型的名称以及从中派生的分类类型的名称。

NameAttribute 类

将名称与编辑器扩展部件关联。

UserVisibleAttribute 类

确定导出对用户是否可见

OrderAttribute 类

对扩展部件的多个实例进行排序

Priority 类

表示分类格式的内置优先级

EventHandler<TEventArgs> 委托

表示当事件提供数据时将处理该事件的方法

SnapshotSpan 结构

特定文本快照中的不可变文本跨度

Span 结构

一个不可变整数间隔，用于描述从 Start 到 End 左侧关闭并在右侧打开的的值范围： [Start.结束) 。 跨度通常应用于 ITextSnapshot 来表示文本跨度，但它独立于任何特定的文本缓冲区或快照

ContentTypeAttribute 类

声明扩展部件与特定内容类型之间的关联

IClassifierProvider 接口

为给定ITextBuffer创建分类器

ITextBuffer 接口

使用 UTF-16 编码的 Unicode 字符的可变序列。 缓冲区中的位置被视为字符序列 (从字符零开始) 或从行零) 开始 (行序列。 空缓冲区有一行不包含任何字符。

GetOrCreateSingletonProperty<T>(Func<T>)

从属性集合中获取或创建类型的属性 T 。 如果已存在该类型的属性，它将返回现有属性。 否则，它使用 creator 创建该类型的实例。

PropertyCollection 类

允许属性所有者控制集合中属性的生存期。

## 向VS添加编辑器分类器扩展（2）

## 向VS添加编辑器边框扩展（1）

Canvas 类

定义一个区域，可在其中使用相对于 Canvas 区域的坐标以显式方式来定位子元素。

IWpfTextViewMargin 接口

表示附加到的边缘的边距 IWpfTextView

IWpfTextView 接口

表示 WPF 平台的 Visual Studio ITextView

FrameworkElement 类

提供 Windows Presentation Foundation (WPF) 元素的属性、事件和方法的 WPF 框架级别集。 此类表示所提供的 WPF 框架级别实现基于 UIElement 定义的 WPF 核心级别 API。

ITextViewMargin 接口

表示附加到的边缘的边距 ITextView

GC 类

控制系统垃圾回收器（一种自动回收未使用内存的服务）。

GC.SuppressFinalize(Object) 方法

请求公共语言运行时不要调用指定对象的终结器。

ObjectDisposedException 类

对已释放的对象执行操作时所引发的异常。

IWpfTextViewMarginProvider 接口

通过给定的IWpfTextViewHost创建IWpfTextViewMargin

PredefinedMarginNames 类

指定由 Visual Studio 提供的预定义边距的名称

MarginContainerAttribute 类

指定边距容器的类型。

TextViewRoleAttribute 类

使用此属性可指定扩展适用的 TextViews 类型。

IWpfTextViewHost 接口

包含 IWpfTextView 和环绕它的边距，如滚动条或行号装订线。

## 向VS添加编辑器边框扩展（2）

## 向VS添加编辑器文本装饰

IAdornmentLayer 接口

表示修饰层

IWpfTextView 接口

表示 WPF 平台的 Visual Studio ITextView 。

IWpfTextView.GetAdornmentLayer(String) 方法

获取名为的 IAdornmentLaye

TextViewLayoutChangedEventArgs 类

提供的布局更改事件的信息 ITextView

ITextViewLine 接口

表示已设置为在文本视图中显示的格式的文本。

IWpfTextViewLineCollection 接口

允许 ITextView 访问视图的 ITextViewLine 对象集合

AdornmentPositioningBehavior 枚举

定义修饰的位置。

IAdornmentLayer.AddAdornment 方法

IWpfTextViewLineCollection.GetMarkerGeometry 方法

为提供的创建标记几何 bufferSpan

IWpfTextViewCreationListener 接口

侦听IWpfTextView的创建时间

AdornmentLayerDefinition 类

提供 IAdornmentLayer 导出信息

## 向VS添加编辑器视窗装饰

[Order(After = PredefinedAdornmentLayers.**Caret**)]

this.adornmentLayer.AddAdornment(AdornmentPositioningBehavior.**ViewportRelative**, null, null, this.image, null);

## VS扩展中的ContentType设置

ContentTypeAttribute(String) 构造函数

内容类型名称。 内容类型名称不区分大小写。

ContentTypeAttribute 类

声明扩展部件与特定内容类型之间的关联。

[1]:htmlLSPClient

code-languageserver-preview

[2]:FSharpInteractive

code

[3]:sighelp-doc

intellisense

[4]:Roslyn Languages

code

[5]:underscore

JavaScript

[6]:snippet picker

text

[7]:LegacyRazorVisualBasic

LegacyRazor

[8]:TestsOutput

Output

[9]:C#\_LSP

Roslyn Languages

[10]:Memory

code

[11]:CSharp Signature Help

sighelp

[12]:Basic Signature Help

sighelp

[13]:code-languageserver-textmate-indentation

[14]:DebugOutput

Output

[15]:Immediate

code

[16]:projection

any

[17]:code++

code

[18]:CSharp

Roslyn Languages

code-languageserver-base

code

[19]:mustache

HTML

[20]:Output

text

[21]:WebForms

code

[22]:VB\_LSP

Roslyn Languages

[23]:JavaScript

[24]:ConsoleOutput

Output

[25]:quickinfo

intellisense

[26]:LegacyRazor

HTML

[27]:Specialized CSharp and VB Interactive Command

code

[28]:HTMLProjection

projection

[29]:Register

code

[30]:LegacyRazorCSharp

LegacyRazor

[31]:Interactive Command

code

[32]:Interactive Content

text

projection

[33]:BuildOrderOutput

Output

[34]:cssLSPClient

code-languageserver-preview

[35]:code-textmate-commentselection

[36]:RoslynPreviewContentType

projection

text

[37]:DatabaseOutput

Output

[38]:code-languageserver-preview

code-languageserver-base

code-languageserver-textmate-color

code-languageserver-textmate-structure

code-languageserver-textmate-brace

code-languageserver-textmate-indentation

code-textmate-commentselection

[39]:code-languageserver-textmate-structure

[40]:F#

Roslyn Languages

code-languageserver-base

code

[41]:sighelp

intellisense

[42]:any

[43]:css

projection

text

code

[44]:htc

HTML

[45]:HQL

code

[46]:wsh

HTML

[47]:pig

text

[48]:srf

HTML

[49]:XML

code

[50]:Basic

Roslyn Languages

code-languageserver-base

code

[51]:inert

[52]:cssLSPServer

[53]:U-SQL

code

[54]:Razor

code-languageserver-preview

code

[55]:UnitTestSummary

text

[56]:code-textmate-ontypeindentation

[57]:Interactive Output

text

[58]:vbscript

text

code

[59]:code-languageserver-textmate-brace

[60]:code-languageserver-textmate-color

[61]:UNKNOWN

[62]:plaintext

text

[63]:handlebars

HTML

[64]:TFSourceControlOutput

Output

[65]:TypeScript

Roslyn Languages

code-languageserver-base

code

[66]:Command

code

[67]:StreamAnalytics

code

[68]:VsTextControl Content

text

[69]:InBoxPowerShell

code++

[70]:code-languageserver-base

code

[71]:WebFormsProjection

projection

[72]:html-delegation

code-languageserver-preview

[73]:SQL Server Tools

code

[74]:T-SQL90

code

[75]:TypeScript Signature Help

sighelp

[76]:F# Signature Help

sighelp

[77]:css.extensions

css

[78]:htmlLSPServer

[79]:BuildOutput

Output

[80]:DockerFileContentType

text

[81]:Dockerfile

code

[82]:LegacyRazorCoreCSharp

LegacyRazorCSharp

[83]:intellisense

text

[84]:FindResults

text

[85]:SourceControlOutput

Output

[86]:code

text

[87]:LESS

css.extensions

code

[88]:JSON

code

[89]:HTML

text

code

[90]:text

any

[91]:SCSS

css.extensions

code

[92]:yaml

code-languageserver-preview

[93]:XAML

code

code-languageserver-base

## VS扩展中的ContentType自定义

ContentTypeDefinition 类

定义内容类型。

FileExtensionToContentTypeDefinition 类

指定内容类型和文件扩展名之间的映射。

internal static class Class1

{

#pragma warning disable 649

[Export]

[Name("zmrbak")]

[BaseDefinition("text")]

internal static ContentTypeDefinition exampleDefinition;

[Export]

[FileExtension(".zmrbak")]

[ContentType("zmrbak")]

internal static FileExtensionToContentTypeDefinition abcFileExtensionDefinition;

#pragma warning restore 649

}

internal /\*static\*/ class Class1

{

#pragma warning disable 649

[Export]

[Name("zmrbak")]

[BaseDefinition("text")]

internal /\*static\*/ ContentTypeDefinition exampleDefinition;

[Export]

[FileExtension(".zmrbak")]

[ContentType("zmrbak")]

internal /\*static\*/ FileExtensionToContentTypeDefinition abcFileExtensionDefinition;

#pragma warning restore 649

}

## 编写扩展为VS编辑器边框添加指示图形