1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 3**

1. «Создание расширения для Microsoft Visual Studio»
2. по дисциплине «Технологии разработки современного программного обеспечения»
3. Выполнил
4. студент гр. 4851001/80201 Агеева А. И.

<*подпись*>

1. Преподаватель Чернов А. Ю.

<*подпись*>

1. Санкт-Петербург
2. 2021

# **Формулировка задания**

Цель: познакомиться с объектной моделью автоматизации Microsoft Visual Studio, получить навык разработки плагина под Microsoft Visual Studio на языке C#, практическое применение объектов и событий модели автоматизации для расширения функциональности IDE, освоение основ механизма регулярных выражений.

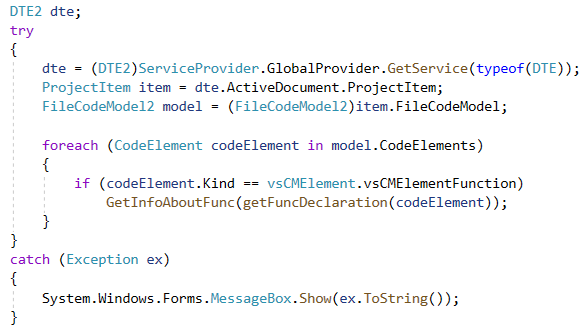
Задачи:

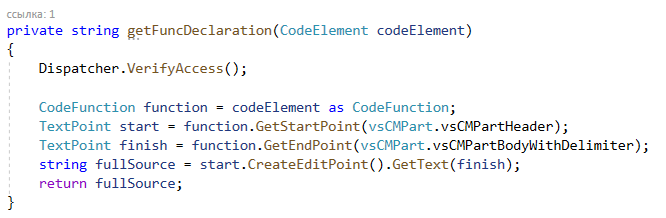
1. Разработать плагин к программе Microsoft Visual Studio на языке C#, удовлетворяющий следующим условиям:
   1. Плагин должен создавать дополнительную панель в Visual Studio на которую по нажатию на кнопку на панели плагин должен выводить статистику для текущего открытого файла;
   2. Статистику следует представлять с помощью таблицы;
   3. В статистике должно присутствовать для каждой функции или метода: название, количество строк, количество строк без пустых и комментариев, количество ключевых слов;
   4. На панели должна быть кнопка, по нажатию на которую статистика обновляется;
   5. Таблица автоматически подстраивается под размер окна;
   6. Панель должна автоматически вызываться с помощь горячей клавиши.
2. **Ход работы**

Для создания плагина использовался шаблон проекта VSIX Project.

После создания в проекте содержатся файлы source.extension.vsixmanifest и <VSPackage\_name>.cs. Первый представляет собой манифест VSIX контейнера, который позволяет упростить развёртывание модуля расширения. VSIX контейнер представляет из себя стандартный OPC (Open Packaging Conventions) архив, содержащий бинарные файлы расширения и вспомогательные файлы, необходимые внедрения плагина. Второй файл содержит основной класс пакета-расширения, унаследованный от класса Microsoft.VisualStudio.Shell.Package, который предоставляет функционал, необходимый полноценному пакету расширения Visual Studio. Однако, этих файлов недостаточно для создания плагина.

В проект также был добавлен элемент ToolWindow.cs, при его добавлении также появляются файлы ToolWindowCommand.cs и ToolWindowControl.xaml и ToolWindowControl.xaml.cs. Основные файлы, которые подверглись изменениям - ToolWindowControl.xaml и ToolWindowControl.xaml.cs. В первом настраивается интерфейс плагина, а во втором его функциональность.

Первым делом нужно получить содержимое текущего документа. Для удобства обработки из открытого документа достаётся не весь код сразу, а только коды функций. Это выполняется при помощи средств Visual Studio. Объекты организованы в иерархию, на вершине которой — глобальный интерфейс DTE, имея ссылку на который, возможно получить в свое распоряжение любой объект модели автоматизации, в том числе и отдельные функции. Таким образом, сначала нужно получить ссылку на DTE, для этого вызывается функция GetService(). Затем из полученного объекта DTE извлекается активный документ, а из него извлекается объект функций и с помощью вспомогательной функции getFuncDeclaration() происходит получение строкового представления тела функции, из которого вычленяется сигнатура функции:



Затем были реализованы функции для получения информации о функциях из открытого документа. Порядок получения информации, следующий:

* Узнать исходное количество строк в функции;
* Удалить комментарии;
* Удалить пустые строки;
* Узнать количество строк в функции без комментариев и пустых строк;
* Удалить содержимое кавычек;
* Узнать количество ключевых слов;
* Узнать название функции.

Для нахождения комментариев, кавычек, пустых строк и ключевых слов были использованы регулярные выражения.

Таблица 1 - Используемые регулярные выражения

|  |  |
| --- | --- |
| Назначение | Регулярное выражение |
| Нахождение однострочных комментариев | (/{2}((.\*\\\r\n)\*.\*)) |
| Нахождение многострочных комментариев | (/\\*(.|(\r\n))\*?\\*/) |
| Нахождение двойных кавычек | (""([^\\""\r\n]\*(\\""|\\\r\n)\*)\*(""|(\r\n))) |
| Нахождение одинарных кавычек | ('([^\\'\r\n]\*(\\'|\\\r\n)\*)\*('|(\r\n))) |
| Нахождение пустых строк | [\r\n]\s\*[\r\n] |
| Нахождение ключевых слов | alignas|alignof|and|and\_eq|asm|auto|bitand|bitor|bool|  break|case|catch|char|char16\_t|char32\_t|class|compl|  const|constexpr|const\_cast|continue|decltype|default|  delete|do|double|dynamic\_cast|else|enum|explicit|export|  extern|false|float|for|friend|goto|if|inline|int|long|  mutable|namespace|new|noexcept|not|not\_eq|nullptr|  operator|or|or\_eq|private|protected|public|register|  reinterpret\_cast|return|short|signed|sizeof|static|  static\_assert|static\_cast|struct|switch|template|this|  thread\_local|throw|true|try|typedef|typeid|typename|  union|unsigned|using|virtual|void|volatile|wchar\_t|  while|xor|xor\_eq |

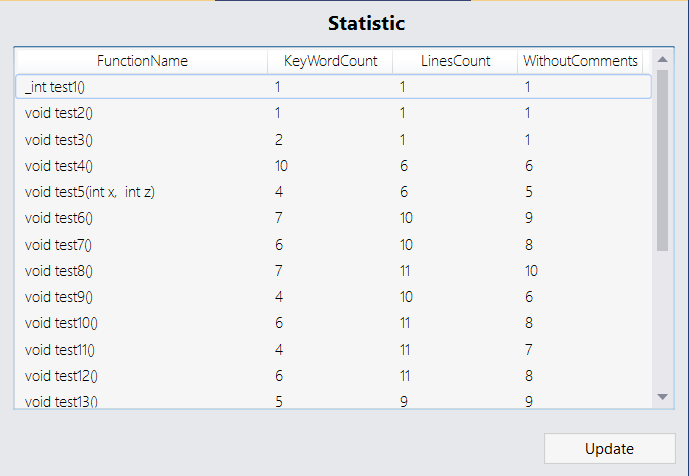
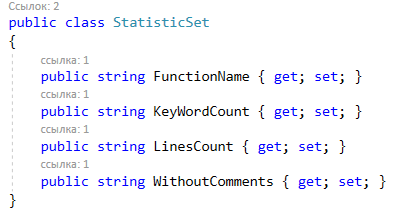
Интерфейс окна с информацией показан на рисунке 1.

Рисунок 1 - Интерфейс окна

Для того, чтобы заполнять таблицу данными, нужно создать класс, который инкапсулирует свойства, соответствующие столбцам таблицы:

Коллекция элементов типа StatisticSet была объявлена как член класса, отвечающего за управление окном плагина. Это было нужно для того, чтобы обращаться к ней из разных функций.

Для того, чтобы связать поле элемента коллекции с определённым столбцом, был использовано свойство GridViewColumn.DisplayMemberBinding.

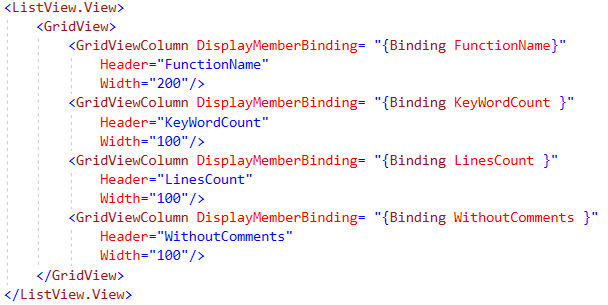
UML диаграмма классов была получена с помощью возможностей Visual Studio:

Рисунок 2 - UML диаграмма классов из Visual Studio SDK

1. **Вывод**

В ходе данной лабораторной работы были получены навыки разработки на языке C#, а также навыки работы с WPF и навыки создания расширений для Visual Studio.

Также удалось познакомиться с механизмом регулярных выражений и составлением собственных регулярных выражений.

**Приложение А. Исходный код интерфейса (.xaml)**

<UserControl x:Class="TRSPO\_3laba.ToolWindow1Control"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:vsshell="clr-namespace:Microsoft.VisualStudio.Shell;assembly=Microsoft.VisualStudio.Shell.15.0"

mc:Ignorable="d"

d:DesignHeight="325"

Name="MyToolWindow" Width="Auto " Background="#FFE7E8EC">

<Grid Width="Auto" Margin="0,0,0,-0.2" Background="#FFE7E8EC">

<ListView Margin="10,36,10.2,53" Name="Statistic" TextBlock.Foreground="#000000" FontWeight="Light" SelectionChanged="Statistic\_SelectionChanged" Background="#FFF6F6F6">

<ListView.View>

<GridView>

<GridViewColumn DisplayMemberBinding= "{Binding FunctionName}"

Header="FunctionName"

Width="200"/>

<GridViewColumn DisplayMemberBinding= "{Binding KeyWordCount }"

Header="KeyWordCount"

Width="100"/>

<GridViewColumn DisplayMemberBinding= "{Binding LinesCount }"

Header="LinesCount"

Width="100"/>

<GridViewColumn DisplayMemberBinding= "{Binding WithoutComments }"

Header="WithoutComments"

Width="100"/>

</GridView>

</ListView.View>

</ListView>

<Button Content="Update" Margin="0,0,10,10" BorderBrush="#FFD0D0D0" FontFamily="Segoe UI" Click="Button\_Click\_1" HorizontalAlignment="Right" Width="105" Height="25" VerticalAlignment="Bottom" Background="#FFF6F6F6"/>

<Label Content="Statistic" Height="31" Margin="198,0,161.6,0" VerticalAlignment="Top" HorizontalContentAlignment="Center" FontFamily="Segoe UI" FontSize="16" FontWeight="Bold"/>

</Grid>

</UserControl>

**Приложение Б. Исходный код .xaml.cs**

namespace TRSPO\_3laba

{

using System.Diagnostics.CodeAnalysis;

using System.Windows;

using System.Windows.Controls;

using System.Text.RegularExpressions;

using Microsoft.VisualStudio.Shell;

using System.Text;

using System.IO;

using System;

using Microsoft.VisualStudio.TextManager.Interop;

using System.Collections.Generic;

using EnvDTE80;

using EnvDTE;

/// <summary>

/// Interaction logic for ToolWindow1Control.

/// </summary>

///

public class StatisticSet

{

public string FunctionName { get; set; }

public string KeyWordCount { get; set; }

public string LinesCount { get; set; }

public string WithoutComments { get; set; }

}

public partial class ToolWindow1Control : UserControl

{

private string patternMultiComm = @"(/\\*(.|(\r\n))\*?\\*/)";

private string patternSingleComm = @"(/{2}((.\*\\\r\n)\*.\*))";

private string patternDoubleQuotes = @"(""([^\\""\r\n]\*(\\""|\\\r\n)\*)\*(""|(\r\n)))";

private string patternSingleQuotes = @"('([^\\'\r\n]\*(\\'|\\\r\n)\*)\*('|(\r\n)))";

private string patternEmptyLine = @"([\n\r]\s\*[\n\r])";

private List<StatisticSet> items;

/// <summary>

/// Initializes a new instance of the <see cref="ToolWindow1Control"/> class.

/// </summary>

public ToolWindow1Control()

{

this.InitializeComponent();

items = new List<StatisticSet>();

Statistic.ItemsSource = items;

GetFunction();

}

/// <summary>

/// Handles click on the button by displaying a message box.

/// </summary>

/// <param name="sender">The event sender.</param>

/// <param name="e">The event args.</param>

[SuppressMessage("Microsoft.Globalization", "CA1300:SpecifyMessageBoxOptions", Justification = "Sample code")]

[SuppressMessage("StyleCop.CSharp.NamingRules", "SA1300:ElementMustBeginWithUpperCaseLetter", Justification = "Default event handler naming pattern")]

private string CompareStartSymbol(Match match)

{

if (match.Value.StartsWith("/\*"))

return "/\*\*/";

else if (match.Value.StartsWith("//"))

return "/\*\*/";

else if (match.Value.StartsWith("\"") || match.Value.StartsWith("'"))

return Regex.Replace(match.Value, @"/\\*\\*/", "");

return match.Value;

}

private string DelComment(string textFunc)

{

textFunc = Regex.Replace(textFunc, patternDoubleQuotes + @"|" + patternSingleQuotes + @"|" + patternSingleComm + @"|" + patternMultiComm, CompareStartSymbol, RegexOptions.Multiline);

return textFunc;

}

private string DelEmptyLine(string textFunc)

{

textFunc = Regex.Replace(textFunc, patternEmptyLine, Environment.NewLine);

return textFunc;

}

private string DelQuotes(string textFunc)

{

textFunc = Regex.Replace(textFunc, patternDoubleQuotes + @"|" + patternSingleQuotes, "\"\"");

return textFunc;

}

private int CountKeyword(string textFunc)

{

string patternKeyword = @"alignas|alignof|and|and\_eq|asm|auto|bitand|bitor|bool|break|case|catch|char|char16\_t|char32\_t|class|compl|const|constexpr|const\_cast|continue|decltype|default|delete|do|double|dynamic\_cast|else|enum|explicit|export|extern|false|float|for|friend|goto|if|inline|int|long|mutable|namespace|new|noexcept|not|not\_eq|nullptr|operator|or|or\_eq|private|protected|public|register|reinterpret\_cast|return|short|signed|sizeof|static|static\_assert|static\_cast|struct|switch|template|this|thread\_local|throw|true|try|typedef|typeid|typename|union|unsigned|using|virtual|void|volatile|wchar\_t|while|xor|xor\_eq";

return Regex.Matches(textFunc, patternKeyword).Count;

}

private void GetInfoAboutFunc(string textFunc)

{

int countLines = textFunc.Split('\n').Length;

string textFuncWithoutComment = DelComment(textFunc);

textFunc = DelEmptyLine(textFuncWithoutComment);

int newCountLines = textFunc.Split('\n').Length - Regex.Matches(textFunc, @"(.\*/\\*\\*/.\*)").Count;

//MessageBox.Show(textFunc);

int openCurlyBracePos = textFunc.IndexOf('{');

string nameFunc = Regex.Replace(Regex.Replace(textFunc.Substring(0, openCurlyBracePos), @"(\s\*[\r\n]\s\*)|(\s+)", " "), @"/\\*\\*/", "");

items.Add(new StatisticSet() { FunctionName = nameFunc, KeyWordCount = CountKeyword(DelQuotes(textFuncWithoutComment)).ToString(), LinesCount = countLines.ToString(), WithoutComments = newCountLines.ToString() });

}

private string getFuncDeclaration(CodeElement codeElement)

{

Dispatcher.VerifyAccess();

CodeFunction function = codeElement as CodeFunction;

TextPoint start = function.GetStartPoint(vsCMPart.vsCMPartHeader);

TextPoint finish = function.GetEndPoint(vsCMPart.vsCMPartBodyWithDelimiter);

string fullSource = start.CreateEditPoint().GetText(finish);

return fullSource;

}

private void GetFunction()

{

Dispatcher.VerifyAccess();

DTE2 dte;

try

{

dte = (DTE2)ServiceProvider.GlobalProvider.GetService(typeof(DTE));

ProjectItem item = dte.ActiveDocument.ProjectItem;

FileCodeModel2 model = (FileCodeModel2)item.FileCodeModel;

foreach (CodeElement codeElement in model.CodeElements)

{

if (codeElement.Kind == vsCMElement.vsCMElementFunction)

GetInfoAboutFunc(getFuncDeclaration(codeElement));

}

}

catch (Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.ToString());

}

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

items.Clear();

GetFunction();

Statistic.Items.Refresh();

}

private void Statistic\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

}

}