Министерство образования и науки Российской Федерации

Санкт-Петербургский Политехнический Университет Петра Великого

—

Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 3**

«VSIX-плагин C#»

по дисциплине «Технологии разработки современного программного обеспечения»

Выполнил:

студент гр. 4851004/00101       А. Н. Хасанов

                                                         <*подпись*>

Проверил:   
преподаватель   Г. С. Кубрин

                                            <*подпись*>

Санкт-Петербург

2023

**Цель работы:**

Цель работы – разработать плагин к программе Microsoft Visual Studio на языке C#.

Задача – плагин должен создать дополнительную панель в Visual Studio на которую по нажатию на кнопку на панели плагин должен выводить статистику для текущего открытого файла в Visual Studio (минимально поддержка только языков C и C++). Статистику следует представлять с помощью таблицы (Grid View). В статистике должно присутствовать для каждой функции или метода (C++) в коде открытого документа следующие параметры: название, количество строк, количество строк без учета пустых и с комментарием, количество ключевых слов. Помимо таблицы на панели должна быть кнопка на Toolbar, по нажатию на которую статистика будет обновлена. И таблица и toolbar должны автоматически подстраиваться (layout) под размер панели. Панель должна появляться с помощью горячей клавиши (по вашему выбору) и через пункт меню Tools.

**Ход работы:**

В ходе выполнения лабораторной работы было реализовано расширение для Visual Studio, помогающий в подсчете строк с комментариями и без, и в подсчете ключевых слов.

Основной файл программы - это файл .xaml.cs, где я реализовал логику моего расширения.

Я создал класс FunctionStatistics, в котором определил поля Name, LineCount, NonEmptyLineCount и KeywordCount для хранения статистики функций.

Также я создал класс ActiveFile, в котором определил поля countWords, ActiveText и lines. Они предназначены для хранения информации о текущем активном файле.

Класс Parser содержит методы GetActiveFile() и Parse(). GetActiveFile() получает содержимое активного файла, а Parse() анализирует этот текст.

В классе MyToolWindowControl я создал экземпляры классов ActiveFile и Parser. Также я реализовал методы AddingRowToStats(), AddRowStats(), RefreshButton\_Click(), CheckResults(), InfoButton\_Click() и FunctionStatsDataGrid\_SelectionChanged() для обработки событий и логики моего расширения.

**Разработка бэкэнда расширения:**

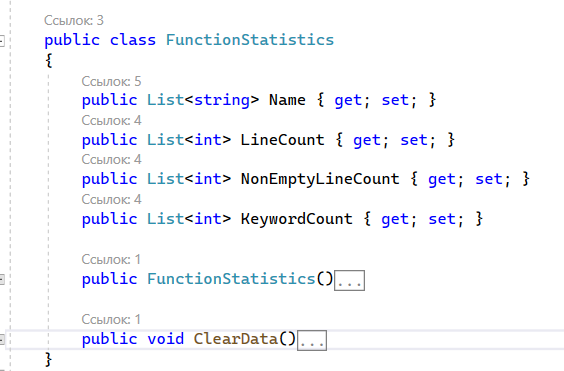
Для создания плагина я использовал шаблон vsix проекта. В этом шаблоне были предусмотрены файлы с расширениями .vsct, .xaml и .xaml.cs, которые я использовал для разработки.

В файле .vsct я предусмотрел возможность вызывать мое расширение из меню "Tools". Я указал родителя "MyCommand" и добавил кнопку с помощью тега <Buttons>. Также я добавил возможность открывать расширение с помощью горячих клавиш "P" + "Shift". Позже в плагине была реализована возможность использование шортката для обновления статистики по файлу клавишами: Ctrl + Alt.

Изначально была реализована возможность считывания данных активного документа с IDE, это было сделано при помощи библиотеки EnvDTE. Сначала была создана тестовая функция TouchFile(), которая позже была переработана в Parse в классе Parser. А часть с активным файлом перенесена в класс, для этого был создан класс ActiveFile, представляющего активный файл, который открыт в IDE:

* Определение свойств для хранения количества слов и текста активного файла.
* Инициализация массива строк lines.
* Реализация конструктора.

Далее я приступил к разработке класса FunctionStatistics для того, чтобы хранить собранные данные с активного файла в модели. Это упрощает дальнейшую разработку и расширение функционала.



Перед добавлением данных через эту модель, были проведены тест-запуски добавление данных в DataGrid, через:

Stats.Items.Add(new { Name = func, Lines = lines, NonEmptyLines = NELines, Keywords = Key });

После было принято решение о доработки автоматического заполнения таблицы через функцию AddingRowStats(). Самая основная работа пришлась на разработку класса Parser. Для реализации парсинга была использована фича Regex – регулярные выражения в C#. Были разработан конструктор для этого класса, получения активного файла средствами EnvDTE и создан массив ключевых слов, которые в последствии используется в функции Parse().

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание  
 **Функция Parse** – самая основная часть кода, отвечает за анализ текста активного файла и извлечение информации о функциях. Она принимает текст в качестве входного параметра. В начале функции мы создаем шаблон для поиска ключевых слов. Затем мы создаем шаблон для поиска функций, которые содержат эти ключевые слова. Мы используем стек bracesStack для отслеживания открывающих и закрывающих скобок. Мы также используем StringBuilder currentFunction для сохранения текущей обрабатываемой функции. Затем мы проходим по каждому символу в тексте. Если символ является открывающей скобкой '{', мы добавляем его в стек bracesStack. Если символ является закрывающей скобкой '}', мы удаляем соответствующую открывающую скобку из стека. Если стек bracesStack становится пустым, это означает, что мы достигли конца функции, и мы обрабатываем содержимое функции. Мы извлекаем имя функции, количество строк, количество ключевых слов и количество пустых и непустых строк внутри функции. Для извлечения имени функции мы используем регулярное выражение. Затем мы исключаем строки в кавычках из поиска ключевых слов, чтобы не учитывать ключевые слова, которые могут быть в строках. Мы также проверяем последнюю строку на наличие комментария после закрывающей скобки функции. Мы добавляем полученные данные о функции в соответствующие списки в объекте FuncStat типа FunctionStatistics, которые хранят статистику функций. Если количество ключевых слов в функции больше нуля, мы сохраняем ключевые слова в список keywords и выводим сообщение с информацией о найденных ключевых словах и имени функции. В конце мы очищаем currentFunction, чтобы готовить его для следующей функции. Таким образом, функция Parse проходит по тексту активного файла, находит функции, извлекает информацию о них и сохраняет ее в объекте FuncStat.

**Разработка класса MyToolWindowControl** основывалась на модели MVC, моделью в этом случае выступает класс FunctionStatistics, Control и View между собой делят .xaml разметка и описываемый класс MTWC (говоря о View, стоит сказать, что логику поведения программы реализует класс Parser):

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

**Класс окна MyToolWindowControl** представляет пользовательский элемент управления (UserControl), который используется в качестве контента для инструментального окна. Оно содержит элементы управления и логику, связанную с обработкой событий и взаимодействием с пользователем. В конструкторе MyToolWindowControl происходит инициализация элементов управления и привязка обработчика события KeyUp к методу MyToolWindowControl\_KeyUp. Это позволяет обрабатывать событие нажатия клавиш на клавиатуре. Метод PreClearData используется для очистки данных перед обновлением статистики. Он очищает список Stats.Items и вызывает метод ClearData у объекта FuncStat в экземпляре parser. Метод AddRowStats используется для добавления строки со статистикой в список Stats.Items. Он принимает параметры, такие как имя функции (func), количество строк (lines), количество непустых строк (NELines) и количество ключевых слов (Key). Создается новый объект анонимного типа, содержащий эти значения, и добавляется в список Stats.Items. Метод AddingRowToStats используется для добавления всех строк статистики в список Stats.Items. Он проходит по спискам Name, LineCount, NonEmptyLineCount и KeywordCount объекта FuncStat и вызывает метод AddRowStats для добавления каждой строки. Обработчик события RefreshButton\_Click вызывается при нажатии кнопки "Refresh". Внутри этого метода сначала вызывается метод PreClearData для очистки данных. Затем получается активный текст файла с помощью метода GetActiveFile у объекта parser.CurrentFile. После этого вызывается метод Parse у объекта parser, чтобы проанализировать полученный текст и получить статистику функций. Затем вызывается метод AddingRowToStats для добавления полученных строк статистики в список Stats.Items. Обработчик события MyToolWindowControl\_KeyUp вызывается при нажатии клавиши на клавиатуре. Если нажатая клавиша - Alt или левый Ctrl, то происходит вызов метода RefreshButton\_Click для обновления статистики. Обработчик события InfoButton\_Click вызывается при нажатии кнопки "Info". В этом методе отображается сообщение, содержащее информацию о разработчике (Developed by Arthur Khasanov aka thurrsense. 2023.) и инструкции о том, как обновить статистику файла. Также указаны значения, которые отображаются в статистике функций (Name of function, Number of lines in this function, Number of lines without comms and space, Number of keywords).

**Фронтэнд плагина:**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Окно MyToolWindowControl представляет собой пользовательский элемент управления, реализованный с использованием языка разметки XAML. Внутри окна есть две кнопки и таблица данных.

Первая кнопка с именем "RefreshButton" имеет иконку обновления и надпись "Refresh". Она используется для обновления статистики. При наведении курсора мыши на кнопку ее фон меняется на серый цвет.

Вторая кнопка с именем "InfoButton" имеет иконку информации и надпись "About". Она используется для отображения информации о приложении. При наведении курсора мыши на кнопку ее фон также меняется на серый цвет.

Ниже кнопок располагается таблица данных типа DataGrid с именем "Stats". Эта таблица отображает статистику функций, включающую имя функции, количество строк, количество непустых строк и количество ключевых слов. Столбцы таблицы имеют соответствующие заголовки: "Name", "Lines", "NonEmpty Lines" и "Keywords". Стиль для заголовков столбцов определен для изменения внешнего вида заголовков.

Внешний вид окна выполнен в темных тонах. Фон окна и элементов управления имеет серый цвет. Текст на кнопках и в таблице отображается в светлом цвете.Обработка нажатия кнопок и обновление статистики выполняются в коде класса MyToolWindowControl. Дизайн окна MyToolWindowControl вдохновлен стилем и внешним видом самой интегрированной среды разработки (IDE). Он имеет схожие цветовые схемы, типографику и расположение элементов управления, чтобы обеспечить единообразный и знакомый пользовательский интерфейс.

Рисунок 1 - UML диаграмма классов

**Вывод:**

В ходе выполнения лабораторной работы были получены знания по разработке расширения для Microsoft Visual Studio на языке С#, по работе с классом Regex для поиска и замены выражений в строках, а также по доступу к исходному коду документа, открытого в Visual Studio. Полученные знания были применены для разработки расширения для Visual Studio, получающего различную статистику о функциях открытого в Visual Studio файла: имя функции, число строк функции, число полных строк функции и количество ключевых строк функции.