УДК 004

Создание графического приложения на языке C# для сбора информации о конфигурации системы с помощью библиотеки WMI

Эрдман Александр Алексеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассмотрен процесс создания графического приложения для сбора данных конфигурации системы компьютера. Приложение написано на языке программирования C# с использованием библиотек WinForms и WMI. Результатом исследования будет являться скомпилированное приложения для сбора информации о системе компьютера.
Ключевые слова: C#, WMI, WinForms

Creating a graphical application in C# to collect information about the system configuration using the WMI library

Erdman Alexander Alekseevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article describes the process of creating a graphical application for collecting computer system configuration data. The application is written in the C# programming language using the WinForms and WMI libraries. The result of the study will be a compiled application for collecting information about the computer system.
Keywords: C#, WMI, WinForms

1 Введение

1.1 Актуальность

В настоящее время, когда информационные технологии прочно вошли в повседневную жизнь, создание программ, способных получать подробную информацию о компонентах компьютерной системы, становится актуальным и востребованным направлением в разработке программного обеспечения. Эта задача не только предоставляет пользователю возможность более глубокого понимания технических характеристик своего устройства, но и имеет важное значение для системных администраторов, разработчиков и технических специалистов. Создание данного ПО позволяет эффективно управлять ресурсами, оптимизировать работу устройств, обнаруживать и устранять потенциальные проблемы, а также проводить диагностику и анализ состояния системы. Это особенно важно в условиях быстрого развития аппаратных

технологий, когда постоянно появляются новые устройства и компоненты. Подобные программы могут предоставлять информацию о процессоре, оперативной памяти, графической карте, хранилище данных, сетевых интерфейсах и других ключевых элементах системы. В данной статье будет рассмотрен пример создания такой системы в виде настольного приложения.

1.2 Обзор исследований

Д.Ю. Опенкин рассмотрел в своей статье основные особенности использования инструментария WMI запросов при разработке программного обеспечения для сбора информации о различных аппаратных компонентах персонального компьютера [1]. А.Ю. Короленко привёл способ взаимодействия с Windows Management Instrumentation в удобной, объектно-ориентированной форме [2]. Д.Ю. Опенкин охарактеризовал подход к разработке программы для мониторинга состояния аппаратной части персонального компьютера [3]. А.А. Южаков и Е.А. Верещагина исследовали библиотеку WMI для получения прямого доступа к периферийным USB устройствам [4]. В.А. Белов и Р.А. Шакуров описали процесс разработки системы мониторинга работы и состояния компьютера [5].

1.3 Цель исследования

Целью исследование является создание настольного приложения для сбора сведений о системе компьютера с помощью языка программирования C# и библиотеки WMI.

2 Материалы и методы

Для создания программы используется язык программирования C#, библиотека для создания графических приложений WinForms и модуль для WMI для управления компьютерной инфраструктурой. В качестве IDE используется MS Visual Studio 2022.

3 Результаты и обсуждения

Реализация приложения начинается с создания проекта WinForms, в который подключается модуль WMI в разделе ссылок проекта. После подключения ссылки данный модуль объявляется в коде с помощью команды using System.Management. На данном этапе настройка самого проекта завершается и следует непосредственная реализация графического интерфейса (рис. 1).

С помощью конструктора на форму добавляются такие элементы, как ToolStrip – выпадающий список; ListView – поля вывода текстовой информации. В элемент ToolStrip, а именно в его свойство Items добавляется коллекция названий компонентов компьютера: процессор, видеокарта, чипсет, батарея, BIOS, оперативная память, кэш, USB, диски, логические диски, клавиатура, сеть и пользователи.

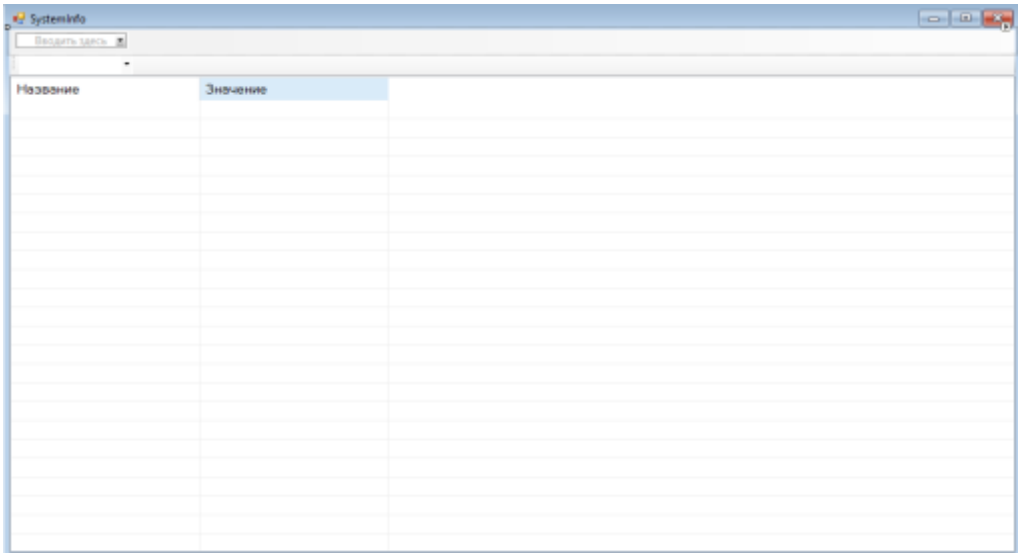

Рисунок 1. Внешний вид приложения

У элемента ListView также меняются параметры колонок (добавляется название колонок «Название» и «Значение»), а также изменяется параметр FullRowSelect на True, который даёт возможность сделать разметку у ListView. Данные действия выполняются для более лучшего восприятия информации с ListView.

По завершению настройки внешнего вида начинается работа над самой программой приложения. Для этого используется класс Form1.cs, который создаётся по умолчанию и является главным классом в проекте.

Первой функцией программы является обработчик события SelectedIndexChanged для объекта toolStripComboBox1 (рис. 2).

Метод toolStripComboBox1_SelectedIndexChanged вызывается при изменении выбора в данном комбинированном элементе управления. В начале метода определяется локальная переменная key типа string, которая будет использоваться для хранения строки-ключа. Затем создаётся конструкция switch, основанная на выбранном элементе в toolStripComboBox1. В каждом случае case определяется строковое значение выбранного элемента, и соответствующий ключ присваивается переменной key. Приведенные варианты включают различные аппаратные компоненты, такие как процессор, видеокарта, чипсет, батарея, BIOS и другие. В случае отсутствия совпадений (оператор default), устанавливается значение по умолчанию, соответствующее процессору. После определения переменной key вызывается метод GetHardWareInfo, который используется для получения информации о аппаратных компонентах. Методу передаются ключ и объект listView1 в качестве параметров.

Общая цель данного обработчика заключается в динамическом изменении выводимой информации о желаемом аппаратном компоненте в listView1 в зависимости от выбранного элемента в toolStripComboBox1.

```
private void toolStripComboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    string key = string.Empty;
    switch (toolStripComboBox1.SelectedItem.ToString())
    {
        case "Процессор":
            key = "Win32_Processor";
            break;
        case "Видеокарта":
            key = "Win32_VideoController";
            break;
        case "Чипсет":
            key = "Win32_IDEController";
            break;
        case "Батарея":
            key = "Win32_Battery";
            break;
        case "Биос":
            key = "Win32_BIOS";
            break;
        case "Оперативная память":
            key = "Win32_PhysicalMemory";
            break;
        case "Кэш":
            key = "Win32_CacheMemory";
            break;
        case "USB":
            key = "Win32_USBController";
            break;
        case "Диск":
            key = "Win32_DiskDrive";
            break;
        case "Логические диски":
            key = "Win32_LogicalDisk";
            break;
        case "Клавиатура":
            key = "Win32_Keyboard";
            break;
        case "Сеть":
            key = "Win32_NetworkAdapter";
            break;
        case "Пользователь":
            key = "Win32_Account";
            break;
        default:
            key = "Win32_Processor";
            break;
    }
    GetHardWareInfo(key, listView1);
}
```

Рисунок 2. Обработчик выбора элемента из спика

В конце обработчика события вызывается метод GetHardWareInfo (рис. 3).

Метод GetHardWareInfo представляет собой процедуру для получения информации об аппаратных компонентах с использованием объекта ManagementObjectSearcher в рамках заданного ключа key и отображения полученных данных в элементе управления ListView (list). В начале метода выполняется очистка элементов в list с помощью метода Clear(). Затем создается объект ManagementObjectSearcher, который используется для выполнения запроса к WMI (Windows Management Instrumentation) с использованием строки запроса "SELECT * FROM " и переданного ключа key. В блоке try, метод перебирает каждый объект ManagementObject возвращенный searcher.Get(). Для каждого объекта, проверяется количество свойств (Properties). Если объект не содержит свойств, отображается сообщение об ошибке, и метод завершает свою работу.

Далее, для каждого объекта создается ListViewGroup на основе значения свойства "Name". В случае возникновения исключения при этом, создается группа с использованием строкового представления объекта. Затем, для каждого свойства объекта, создается ListViewItem внутри ранее созданной

группы. Если количество элементов в list четное, устанавливается фоновый цвет для визуального разделения. Для каждого значения свойства выполняется проверка на null и пустую строку. В случае наличия значения, создается строка resStr, в которую добавляются значения в зависимости от типа данных. После, созданный элемент ListViewItem добавляется в list. В блоке catch перехватываются возможные исключения, и в случае их появления, отображается сообщение об ошибке с информацией об исключении.

Описанный метод предназначен для заполнения элемента отображения ListView информацией об аппаратных компонентах на основе данных, полученных с использованием WMI.

```
Ссылок: 1
private void GetHardWareInfo(string key, ListView list)
{
    list.Items.Clear();
    ManagementObjectSearcher searcher = new ManagementObjectSearcher("SELECT * FROM " + key);
    try
    {
        foreach (ManagementObject obj in searcher.Get())
        {
            if (obj.Properties.Count == 0)
            {
                MessageBox.Show("Не удалось получить информацию", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            ListViewGroup listViewGroup;
            try
            {
                listViewGroup = list.Groups.Add(obj["Name"].ToString(),
                obj["Name"].ToString());
            }
            catch (Exception ex)
            {
                listViewGroup = list.Groups.Add(obj.ToString(), obj.ToString());
            }
            foreach (PropertyData data in obj.Properties)
            {
                ListViewItem item = new ListViewItem(listViewGroup);
                if (list.Items.Count % 2 == 0)
                {
                    item.BackColor = Color.WhiteSmoke;
                }
                item.Text = data.Name;

                if (data.Value != null && !string.IsNullOrEmpty(data.Value.ToString()))
                {
                    string resStr = string.Empty;
                    switch (data.Value.GetType().ToString())
                    {
                        case "System.String[]":
                            string[] stringData = data.Value as string[];
                            foreach (string s in stringData)
                            {
                                resStr += s + " ";
                            }
                            item.SubItems.Add(resStr);
                            break;
                        case "System.UInt16[]":
                            ushort[] ushortData = data.Value as ushort[];
                            foreach (ushort us in ushortData)
                            {
                                resStr += us.ToString() + " ";
                            }
                            item.SubItems.Add(resStr);
                            break;
                        default:
                            item.SubItems.Add(data.Value.ToString());
                            break;
                    }
                    list.Items.Add(item);
                }
            }
        }
    }

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
```

Рисунок 3. Метод GetHardWareInfo

Графический интерфейс и функционал программы реализован. Перед тестированием программы на практике её нужно скомпилировать – это нужно для того, чтобы проверить работоспособность приложения на разных системах, а также для удобства пользования. Компиляция происходит с помощью инструмента «Собрать решение» среды Visual Studio. После сборки проекта получается исполняемый exe файл. Затем осуществляется тестирование приложения на двух разных компьютерах (рис. 4 и 5).

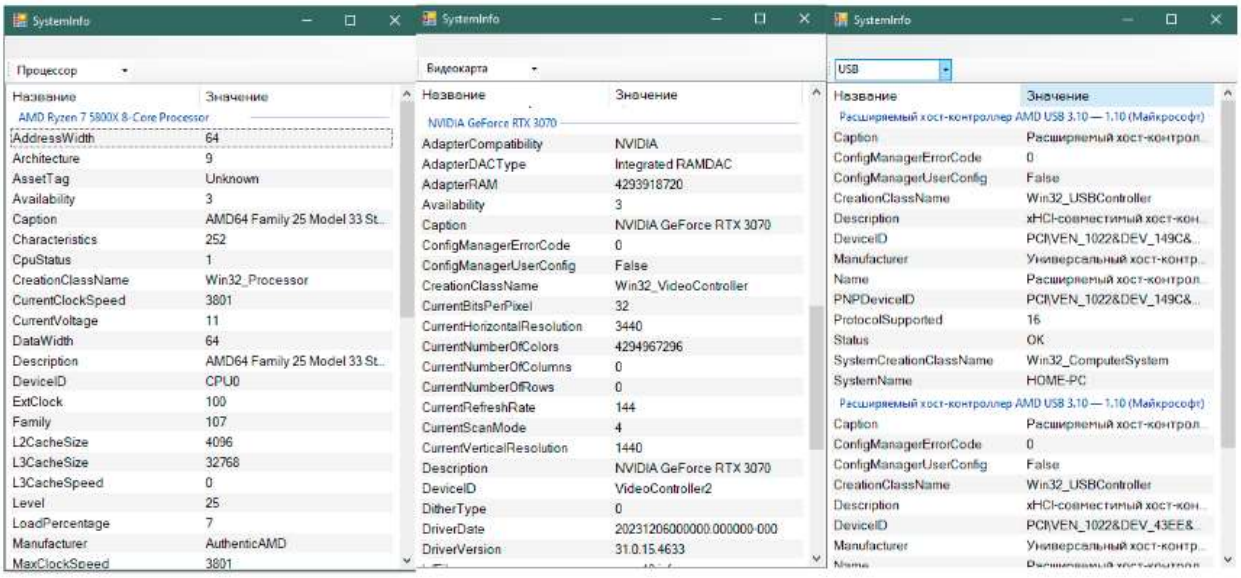

Рисунок 4. Демонстарция работы приложения на ПК1, сведения (слева направо) о процессоре, видеокарте и USB интерфейсах

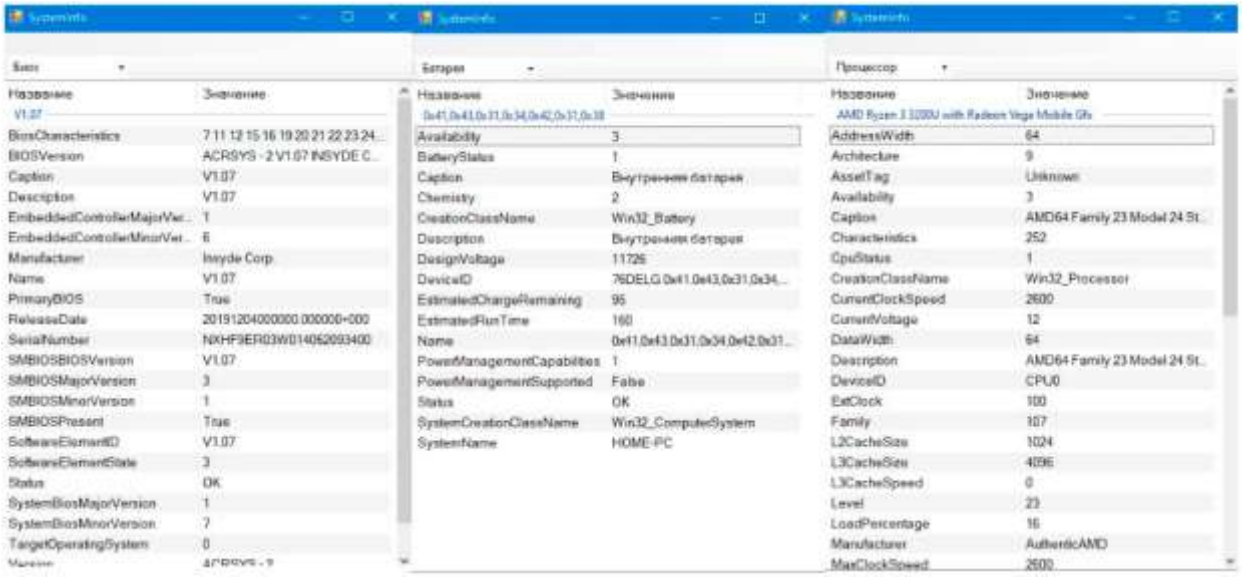

Рисунок 5. Демонстрация работы приложения на ПК2, сведения (слева направо) о BIOS, батареи и процессоре

Таким образом, реализовано настольное приложения для сбора и вывода информации о системе компьютера. Программа была описана, скомпилирована в исполняемый файл и протестирована на двух различных компьютерах.

Библиографический список

1. Опенкин Д.Ю. Применение инструментария WMI запросов при разработке программного обеспечения // Ученые записки УлГУ. Серия: Математика и информационные технологии. 2019. № 2. С. 76-80.
2. Королёнок А.Ю. Использование инструментария управления Windows (WMI) для получения системной информации с применением техник объектно-ориентированного программирования // Постулат. 2020. № 12 (62).
3. Опенкин Д.Ю. Разработка программы для мониторинга аппаратного обеспечения персонального компьютера с помощью инструментария WMI // В сборнике: Системы управления, сложные системы: моделирование, устойчивость, стабилизация, интеллектуальные технологии. материалы VI Международной научно-практической конференции, посвященной 100-летию со дня рождения профессора А. А. Шестакова. Елецкий государственный университет им. И.А. Бунина. Елец, 2020. С. 310-314.
4. Южаков А.А., Верещагина Е.А. Исследование библиотек и фреймворков прямого доступа к периферийным USB устройствам с целью разработки системы универсального токен-ключа // В сборнике: Новейшие исследования в современной науке: опыт, традиции, инновации. Научно-издательский центр «Открытие». 2015. С. 25-28.
5. Белов В.А., Шакуров Р.А. Разработка системы мониторинга работы и состояния компьютера // В сборнике: Прикладные информационные системы. Сборник научных трудов. 2016. С. 618-623.