**6 ЛАБОРАТОРНАЯ РАБОТА №6**

**ОНОВЫ ИНТЕРФЕЙСА WMI.**

**Цель работы:** изучить назначение и основные функциональные возможности интерфейса WMI, получить практические навыки использования WMI в программах на языке C#.

**6.1 Текст программы заданной на самостоятельную работу**

class WMIEventProcessor

    {

        private ManagementEventWatcher watcher;

        private RichTextBox rtb;

        public WMIEventProcessor(RichTextBox rtb, int flag)

        {

            this.rtb = rtb;

            if (flag == 1)

            {

                StringBuilder query = new StringBuilder("SELECT \* FROM \_\_InstanceCreationEvent");

                query.Append(" WITHIN 1 WHERE TargetInstance");

                query.Append(" isa \"Win32\_Process\"");

                watcher = new ManagementEventWatcher(query.ToString());

                watcher.EventArrived += new EventArrivedEventHandler(this.OnProcessCreationEvent);

            }

            else

            {

                StringBuilder query = new StringBuilder("SELECT \* FROM \_\_InstanceCreationEvent");

                query.Append(" WITHIN 1 WHERE TargetInstance");

                query.Append(" isa \"Win32\_PnPEntity\"");

                watcher = new ManagementEventWatcher(query.ToString());

                watcher.EventArrived += new EventArrivedEventHandler(this.OnPnPEvent);

            }

            watcher.Start();

        }

        private void OnProcessCreationEvent(object sender, EventArrivedEventArgs e)

        {

            ManagementBaseObject evnt = (ManagementBaseObject)e.NewEvent;

            ManagementBaseObject row = (ManagementBaseObject)evnt.GetPropertyValue("TargetInstance");

            StringBuilder result = new StringBuilder(row.GetPropertyValue("Name").ToString());

            result.Append(" ").Append(row.GetPropertyValue("ProcessId")).Append("\n");

            this.rtb.Invoke((MethodInvoker)delegate

            {

                rtb.AppendText(result.ToString());

            });

        }

        private void OnPnPEvent(object sender, EventArrivedEventArgs e)

        {

            ManagementBaseObject evnt = (ManagementBaseObject)e.NewEvent;

            ManagementBaseObject row = (ManagementBaseObject)evnt.GetPropertyValue("TargetInstance");

            StringBuilder result = new StringBuilder(row.GetPropertyValue("Name").ToString());

            result.Append(" ").Append(row.GetPropertyValue("Manufacturer")).Append(" ")

                .Append(row.GetPropertyValue("Status")).Append("\n");

            this.rtb.Invoke((MethodInvoker)delegate

            {

                rtb.AppendText(result.ToString());

            });

        }

        public void destroy()

        {

            this.watcher.Stop();

        }

    }

class WMIDisks

    {

        public static string GetLogicalDisksInfo()

        {

            string root = "root\\CIMV2";

            string query\_text = "SELECT \* FROM Win32\_LogicalDisk";

            ManagementObjectSearcher ms = new ManagementObjectSearcher(root, query\_text);

            ManagementObjectCollection list = ms.Get();

            var iter = list.GetEnumerator();

            StringBuilder result = new StringBuilder("");

            ManagementObject item = null;

            while (iter.MoveNext())

            {

                item = (ManagementObject)iter.Current;

                result.Append("Caption:");

                result.Append(item.GetPropertyValue("Caption").ToString());

                result.Append("\n");

                result.Append("Free Space:");

                result.Append(item.GetPropertyValue("FreeSpace"));

                result.Append("\n");

                result.Append("Volume Serial Number:");

                result.Append(item.GetPropertyValue("VolumeSerialNumber"));

                result.Append("\n");

                result.Append("Drive Device ID:");

                result.Append(item.GetPropertyValue("DeviceID").ToString());

                result.Append("\n\n");

            }

            return result.ToString();

        }

    }

    public partial class Form1 : Form

    {

        static int i = 0;

        static int j = 0;

        private WMIEventProcessor process;

        private WMIEventProcessor pnp;

        public Form1()

        {

            InitializeComponent();

        }

        private void button1\_Click(object sender, EventArgs e)

        {

            richTextBox1.AppendText(WMIDisks.GetLogicalDisksInfo());

        }

        private void button2\_Click(object sender, EventArgs e)

        {

            if (i == 0)

            {

                process = new WMIEventProcessor(richTextBox1,1);

                ++i;

            }

        }

        private void Form1\_Close(object sender, EventArgs e)

        {

            if (process != null) process.destroy();

            if (pnp != null) pnp.destroy();

        }

        private void button3\_Click(object sender, EventArgs e)

        {

            if (j == 0)

            {

                pnp = new WMIEventProcessor(richTextBox1, 0);

                ++j;

            }

        }

    }

**Выводы**

В ходе лабораторной работы были изучены основы интерфейса WMI. Было создано приложение с графическим интерфейсом которое перехватывает событие появление нового процесса в системе, а также появление нового PnP-устройства.