



@AlexALX

Программист

9,0

карма

19,2

рейтинг



Профиль

1

Публикации

12

Комментарии

0

Избранное

3

Подписчики

24 ноября в 08:35

Разработка → Пишем простую программу захвата скриншотов

из песочницы

Программирование*, C#*, .NET*

Существует множества различных программ для захвата изображений с экрана, их редактирования «прямо на экране» и загрузки на различные сервисы. Это всё хорошо, но большинство программ привязаны к определённым сервисам и не дают возможности загружать куда-либо ещё. В голове давно уже крутилась мысль создать свой простенький сервис загрузки картинок под свои нужды. И я хочу поделиться историей разработки данной программы.

Не долго думая и имея под рукой Visual Studio 2015 конечно же создал новый C# проект т.к. это очень удобно и я уже делал ранее небольшие C# программы.

Задача первая

Глобальный перехват нажатия кнопок PrintScreen и Alt+PrintScreen. Чтобы не изобретать велосипед, пару минут гугления и почти сразу [нашлось решение](#). Суть заключается в использовании callback-функции LowLevelKeyboardProc и функции SetWindowsHookEx с WH_KEYBOARD_LL из user32.dll. С небольшой модификацией под перехват двух комбинаций код заработал и успешно ловит нажатия клавиш.

▼ Код захвата нажатия клавиш

```
namespace ScreenShot_Grab
{
    static class Program
    {
        private static MainForm WinForm;

        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>

        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            _hookID = SetHook(_proc);
            Application.Run(new MainForm());
            UnhookWindowsHookEx(_hookID);
        }

        private const int WH_KEYBOARD_LL = 13;
        //private const int WH_KEYBOARD_LL = 13;
        private const int VK_F1 = 0x70;
        private static LowLevelKeyboardProc _proc = HookCallback;
        private static IntPtr _hookID = IntPtr.Zero;

        private static IntPtr SetHook(LowLevelKeyboardProc proc) {
            using (Process curProcess = Process.GetCurrentProcess())
            using (ProcessModule curModule = curProcess.MainModule) {
                return SetWindowsHookEx(WH_KEYBOARD_LL, proc, GetModuleHandle(curModule.ModuleName));
            }
        }

        private delegate IntPtr LowLevelKeyboardProc(int nCode, IntPtr wParam, IntPtr lParam);

        private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam) {
```

```

        if (nCode >= 0) {
            Keys number = (Keys)Marshal.ReadInt32(lParam);
            //MessageBox.Show(number.ToString());

            if (number == Keys.PrintScreen) {
                if (wParam == (IntPtr)261 && Keys.Alt == Control.ModifierKeys && number == Keys.PrintScreen) {
                    // Alt+PrintScreen
                } else if (wParam == (IntPtr)257 && number == Keys.PrintScreen) {
                    // PrintScreen
                }
            }
        }

        return CallNextHookEx(IntPtr.Zero, nCode, wParam, lParam);
    }

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
private static extern IntPtr SetWindowsHookEx(int idHook, LowLevelKeyboardProc lpfn, IntPtr hMod, uint dwThreadId);

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
[return: MarshalAs(UnmanagedType.Bool)]
private static extern bool UnhookWindowsHookEx(IntPtr hhk);

[DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
private static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam, IntPtr lParam);

[DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]
private static extern IntPtr GetModuleHandle(string lpModuleName);
}
}

```

Задача вторая

Собственно захват скриншота при нажатии клавиш. Вновь гугление и [решение найдено](#). В этом случае используются функции GetForegroundWindow и GetWindowRect всё из того же user32.dll, а также штатная функция .NET Graphics.CopyFromScreen. Пару проверок и код работает, но с одной проблемой — захватывает также границы окна. К решению этого вопроса вернусь чуть позже.

▼ [Код захвата скриншотов](#)

```

class ScreenCapturer
{
    public enum CaptureMode
    {
        Screen,
        Window
    }

    [DllImport("user32.dll")]
    private static extern IntPtr GetForegroundWindow();

    [DllImport("user32.dll")]
    private static extern IntPtr GetWindowRect(IntPtr hWnd, ref Rect rect);

    [StructLayout(LayoutKind.Sequential)]
    public struct Rect
    {
        public int Left;
        public int Top;
        public int Right;
        public int Bottom;
    }
}

```

```

    }

    public Bitmap Capture(CaptureMode screenCaptureMode = CaptureMode.Window)
    {
        Rectangle bounds;

        if (screenCaptureMode == CaptureMode.Screen)
        {
            bounds = Screen.GetBounds(Point.Empty);
            CursorPosition = Cursor.Position;
        }
        else
        {
            var handle = GetForegroundWindow();
            var rect = new Rect();
            GetWindowRect(handle, ref rect);

            bounds = new Rectangle(rect.Left, rect.Top, rect.Right, rect.Bottom);
            //CursorPosition = new Point(Cursor.Position.X - rect.Left, Cursor.Position.Y - rect.Top);
        }

        var result = new Bitmap(bounds.Width, bounds.Height);

        using (var g = Graphics.FromImage(result))
        {
            g.CopyFromScreen(new Point(bounds.Left, bounds.Top), Point.Empty, bounds.Size);
        }

        return result;
    }

    public Point CursorPosition
    {
        get;
        protected set;
    }
}

```

Задача третья

Сохранения скриншота на компьютер, тут всё очень просто достаточно было использовать функцию Bitmap.Save.

```

private void save_Click(object sender, EventArgs e)
{
    if (lastres == null) { return; }

    // генерируем имя с помощью base36
    Int32 unixTimestamp = (Int32)(DateTime.UtcNow.Subtract(new DateTime(1970, 1, 1))).TotalSeconds;
    var FileName = base_convert(unixTimestamp.ToString(), 10, 36);
    lastres.Save(spath + FileName);
}

```

Задача четвёртая

Загрузка скриншота на сервер, тут вроде кажется, что всё просто, но это не совсем так. После небольшого размышления пришла в голову довольно простая идея — загружать скриншот при помощи WebClient в бинарном формате используя заголовок «application/octet-stream» и функцию WebClient.UploadData, а на стороне сервера брать данные с помощью file_get_contents(«php://input»). Собственно так и поступил, написал очень простой php скрипт в пару строк и привязал всё это дело к программе. Итог — скриншоты сохраняет и загружает. Вместе с этим надо было найти простой алгоритм генерации коротких ссылок, итогом нагуглил очень [простой и элегантный способ](#) заключающийся в использовании Base36, взяв за int unix время в секундах (linux epoch).

```
// переводим bitmap в byte[]
private Byte[] BitmapToArray(Bitmap bitmap)
{
    if (bitmap == null) return null;
    using (MemoryStream stream = new MemoryStream()) {
        bitmap.Save(stream, ImageFormat[Properties.Settings.Default.format]);
        return stream.ToArray();
    }
}

private void upload_Click(object sender, EventArgs e)
{
    using (var client = new WebClient()) {
        client.Headers.Add("Content-Type", "application/octet-stream");
        try {
            var response = client.UploadData(surl, BitmapToArray(lastres));
            var result = Encoding.UTF8.GetString(response);
            if (result.StartsWith("http")) {
                System.Diagnostics.Process.Start(result);
            }
        } catch { }
    }
}
```

Принимающий PHP-скрипт

```
<?php
$file = file_get_contents("php://input");
$id = base_convert(time(), 10, 36);
file_put_contents("img/" . $id . ".png", $file);
echo "http://" . $_SERVER['SERVER_NAME'] . "/img/" . $id . ".png";
?>
```

Редактирование скриншотов

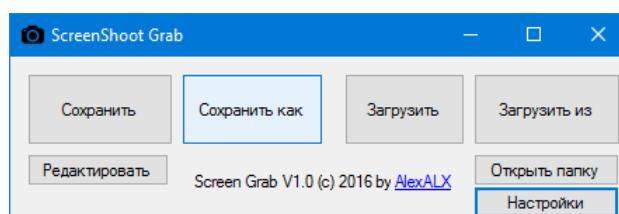
Далее захотелось также как-то быстро редактировать скриншоты и загружать их на сервер. Вместо изобретения очередного редактора изображений родилась очень простая идея — сделать кнопку «редактировать» которая открывала paint с захваченным скриншотом (последним что сохранил на диск), а после редактирования можно было спокойно загрузить этот файл на сервер.

```
private void edit_Click(object sender, EventArgs e)
{
    if (lastres == null) return;
    if (lastfile == "") save_Click(sender, e);
    Process.Start("mspaint.exe", "\"" + lastfile + "\"");
}
```

Настройки

Также надо было где-то указывать url сайта и папку по умолчанию куда сохранять скриншоты, в итоге создал простенькую форму настроек где это можно было указать. Ну и вдобавок сделал кнопку «открыть папку» чтобы всё было ещё проще и быстрее с помощью функции System.Diagnostics.Process.Start. Кроме этого быстро научил программу сворачиваться в трей.

Итак после всего этого был готов **первый рабочий прототип**, и выглядел он так:



Предосмотр

Всё вроде бы хорошо, но стало понятно чего не хватает. А не хватало кнопки предпросмотра! Было несколько не удобно открывать папку или нажимать редактировать чтобы только посмотреть что же захватилось с экрана перед отправкой. В итоге быстро набросал форму предпросмотра, была небольшая проблема с отображением полноэкранного скриншота в форме (она ведь с рамками), рамки удалять не хотелось (даже не знаю почему), в итоге сделал скрол в форме и меня такое полностью устроило.

```
private void PreviewForm_Load(object sender, EventArgs e)
{
    if (form1.lastfile != "") {
        img.Image = Image.FromFile(form1.lastfile);
    } else {
        img.Image = form1.lastres;
    }
    ClientSize = new Size(img.Image.Width + 10, img.Image.Height + 10);
    img.Width = img.Image.Width + 10;
    img.Height = img.Image.Height + 10;
    if (img.Image.Width >= Screen.PrimaryScreen.Bounds.Width || img.Image.Height >= Screen.PrimaryScreen.Bounds.Height) {
        WindowState = FormWindowState.Maximized;
    }
    CenterToScreen();
}
```

Формат изображений

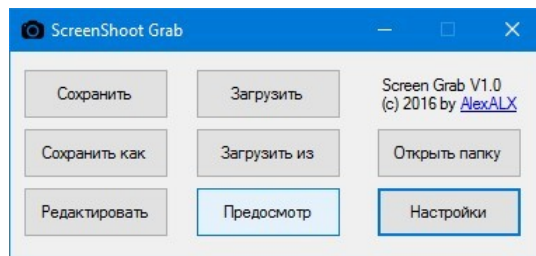
Кроме этого появилась также необходимость сохранения скриншотов в разных форматах (а не только PNG как по умолчанию), благо всё это легко решается с помощью всё той же функции `Bitmap.Save`, правда вот качество jpg изображений меня не устроило. Возможность указать качество у jpg было не так очевидно, быстрое гугление и [есть решение](#). Реализуется с помощью доп параметра `EncoderParameter` к `Bitmap.Save`.

```
// получаем энкодер по формату
private ImageCodecInfo GetEncoder(ImageFormat format)
{
    ImageCodecInfo[] codecs = ImageCodecInfo.GetImageDecoders();
    foreach (ImageCodecInfo codec in codecs) {
        if (codec.FormatID == format.Guid) {
            return codec;
        }
    }
    return null;
}

internal void SaveFile(string FilePath, ImageFormat format)
{
    var curimg = lastres;
    if (format == ImageFormat.Jpeg) {
        System.Drawing.Imaging.Encoder myEncoder = System.Drawing.Imaging.Encoder.Quality;
        ImageCodecInfo Encoder = GetEncoder(format);
        EncoderParameters myEncoderParameters = new EncoderParameters(1);
        myEncoderParameters.Param[0] = new EncoderParameter(myEncoder, Properties.Settings.Default.quality);
        curimg.Save(stream, Encoder, myEncoderParameters);
    } else {
        curimg.Save(FilePath, format);
    }
}
```

Также родилась идея автоматического открытия папки после сохранения скриншота, а также авто открытию ссылки после загрузки. Быстро это реализовал и добавил галочки в настройки. Ещё добавил функцию копирования ссылки в буфер обмена.

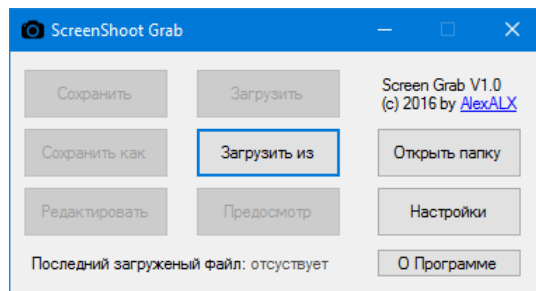
После добавления кнопки предпросмотр, программа как-то стала выглядеть «не так», расположение кнопок было разбросано, подумал немного, и переставлял кнопки, так что вышло следующее:



Мелкие доработки

Немного передохнув и подумав, понял чего ещё не хватает — информации о последней загрузке скриншота. Сделал соответствующее поле, при нажатии на которое можно было перейти по ссылке. Помимо этого сделал кнопки сохранения/редактирования недоступными пока не сделаешь скриншот. Ну и ещё один штрих — добавил кнопку «о программе» с кратким описанием, версией и датой билда (кстати для получения даты опять [нагуглил решение](#), получая дату с заголовка самого приложения).

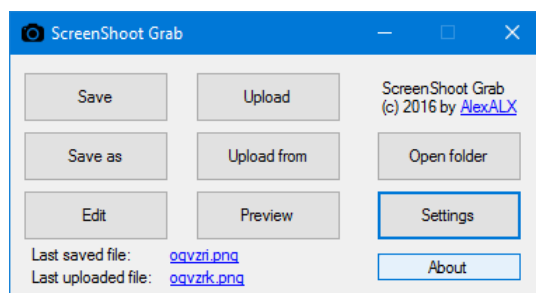
Итого после этих действий вышло следующее:



Чуть позже понял что также не хватает отображения последнего сохранённого файла, что быстро добавил, а ещё сделал эти поля более функциональными — прикрутив контекстное меню (при щелчке правой кнопкой мышью) где можно было скопировать ссылку/путь в буфер обмена при помощи `Clipboard.SetText`.

Готовность программы, локализация

Ну и вроде бы основной функционал был готов, всё работало, и подумал я — может поделиться программой с народом? Если делать это, тогда нужно как минимум сделать возможность локализации и добавить английский язык. Благо студия легко позволяет всё это реализовать штатными средствами, начал я всё это дело переводить. Итого получилось:



Для перевода некоторых сообщений нужно было создать новые файлы ресурсов и потом брать из него строки следующим образом:

```
internal ResourceManager LocM = new ResourceManager("ScreenShot_Grab.Resources.WinFormsStrings", typeof(MainForm).Assembly);

LocM.GetString("key_name");
```

Файл с русским языком у меня WinFormStrings.resx, для английского WinFormStrings.en.resx, которые положил в папку Resources.

Но чтобы сменить язык требовалась перезагрузка приложения, конечно хотелось чтобы можно было обойтись без этого, к счастью есть [решение](#) этого вопроса, которое я быстро применил. Помимо этого также надо было получить список поддерживаемых языков приложением (на будущее, если вдруг будут ещё локализации), итог нагугли [такое решение](#), всё это объединив получилась следующая конструкция:

▼ [Код смены языка в реальном времени](#)

```

private void ChangeLanguage(string lang)
{
    foreach (Form frm in Application.OpenForms) {
        localizeForm(frm);
    }
}

private void localizeForm(Form frm)
{
    var manager = new ComponentResourceManager(frm.GetType());
    manager.ApplyResources(frm, "$this");
    applyResources(manager, frm.Controls);
}

private void applyResources(ComponentResourceManager manager, Control.ControlCollection ctls)
{
    foreach (Control ctl in ctls) {
        manager.ApplyResources(ctl, ctl.Name);
        Debug.WriteLine(ctl.Name);
        applyResources(manager, ctl.Controls);
    }
}

private void language_SelectedIndexChanged(object sender, EventArgs e)
{
    var lang = ((ComboBoxItem) language.SelectedItem).Value;
    if (Properties.Settings.Default.language == lang) return;
    UpdateLang(lang);
}

private void UpdateLang(string lang)
{
    Thread.CurrentThread.CurrentUICulture = new CultureInfo(lang);
    ChangeLanguage(lang);
    Properties.Settings.Default.language = lang;
    Properties.Settings.Default.Save();
    form1.OnLangChange();
}

private void Form2_Load(object sender, EventArgs e)
{
    language.Items.Clear();
    foreach (CultureInfo item in GetSupportedCulture()) {
        var lc = item.TwoLetterISOLanguageName;
        var citem = new ComboBoxItem(item.NativeName, lc);
        //Debug.WriteLine(item.NativeName);
        // Задаём для дефолтного языка свой код и заголовок в списке
        if (item.Name == CultureInfo.InvariantCulture.Name) {
            lc = "ru";
            citem = new ComboBoxItem("Русский", lc);
        }
        language.Items.Add(citem);
        if (Properties.Settings.Default.language == lc) {
            language.SelectedItem = citem;
        }
    }
}

private IList<CultureInfo> GetSupportedCulture()
{
    //Get all culture
    CultureInfo[] culture = CultureInfo.GetCultures(CultureTypes.AllCultures);

    //Find the location where application installed.
    string exeLocation = Path.GetDirectoryName(Uri.UnescapeDataString(new UriBuilder(Assembly.GetExecutingAssembly()).CodeB

```

```

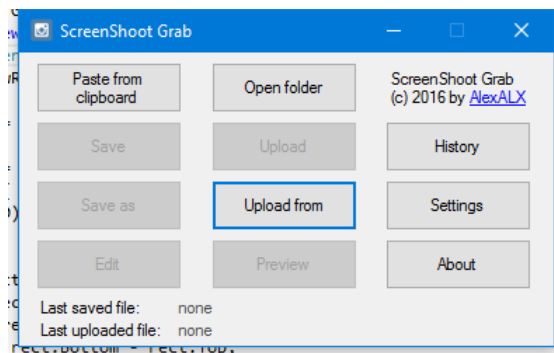
ase).Path));

//Return all culture for which satellite folder found with culture code.
IList<CultureInfo> cultures = new List<CultureInfo>();
foreach(var cultureInfo in culture) {
    if (Directory.Exists(Path.Combine(exeLocation, cultureInfo.Name))) {
        cultures.Add(cultureInfo);
    }
}
return cultures;
}
}

```

Проблема захвата границ у окна

А теперь я вернусь к проблеме захвата границ окна, этот вопрос сначала был решён с помощью функции автоматической обрезки окна (которую я добавил в настройки), указав значения для windows 10, но это был скорее костыль чем решение. Чтобы было понятнее о чём речь вот скриншот того что я имею ввиду:



(скриншот с более новой версии)

Как видно на скриншоте — кроме окна захватывало его границы и то что под ними. Довольно долго гуглил как решить эту проблему, но потом наткнулся на [эту статью](#), где собственно описывалось решение вопроса, суть заключается в том что на windows vista и новее нужно использовать dwmapi для получения корректных границ окна с учётом аеро и тд. С небольшой модификацией своего кода успешно привязал к dwmapi и проблема наконец была полностью решена. Но т.к. функционал обрезки окна уже был написан, решил оставить его, возможно кому-то будет полезен.

```

[DllImport(@"dwmapi.dll")]
private static extern int DwmGetWindowAttribute(IntPtr hwnd, int dwAttribute, out Rect pvAttribute, int cbAttribute);

public Bitmap Capture(CaptureMode screenCaptureMode = CaptureMode.Window, bool cutborder = true)
{
    ...
    var handle = GetForegroundWindow();
    var rect = new Rect();
    // Если Win XP и ранее то используем старый способ
    if (Environment.OSVersion.Version.Major < 6) {
        GetWindowRect(handle, ref rect);
    } else {
        var res = -1;
        try {
            res = DwmGetWindowAttribute(handle, 9, out rect, Marshal.SizeOf(typeof(Rect)));
        } catch { }
        if (res < 0) GetWindowRect(handle, ref rect);
    }
    ...
}

```

Поддержка imgur

Потом ещё подумав, раз я собираюсь публиковать программу для всех, то наверное было бы неплохо кроме загрузки на свой сервер сделать загрузку на какой-то сервис, ведь тогда программа будет более полезной, и не нужно иметь обязательно свой сервер для её использования, т.к. я давно использую [imgur.com](#) и у него есть простой [api](#), то решил сделать привязку к нему.

Посидев поизучав его api сначала реализовал анонимную загрузку, а чуть позже и возможность привязки аккаунта. Кроме этого реализовал возможность удаления последнего загруженного изображения в программе (для их сервиса только).

Полностью описывать код реализации их api я не буду, скажу лишь что для загрузки изображений на imgur использовал HttpClient и MultipartFormDataContent из .NET Framework 4.5 и при этом я переделал код загрузки изображений на свой сервер, вместо бинарной отправки использовал полноценную загрузку с помощью формы чтобы унифицировать код. Попутно для своего скрипта как способ идентификации использовал user-agent и \$_GET[key] ключ, что-то не захотелось возиться с полноценной авторизацией (хотя это по идее не сложно).

```
private void uploadfile(bool bitmap = true)
{
    byte[] data;
    if (bitmap && !imgedit) {
        data = BitmapToArray(lastres);
    } else {
        if (!File.Exists(lastfile)) {
            MessageBox.Show(LocM.GetString("file_nf"), LocM.GetString("error"), MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        data = File.ReadAllBytes(lastfile);
    }
    HttpContent bytesContent = new ByteArrayContent(data);
    using (var client = new HttpClient())
    using (var formData = new MultipartFormDataContent()) {

        ...

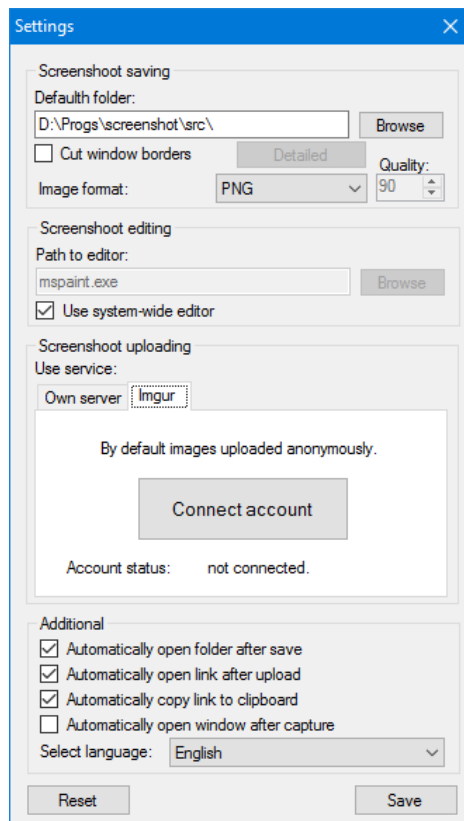
        formData.Add(bytesContent, "image", "image");
        try {
            var response = client.PostAsync(url, formData).Result;

            if (!response.IsSuccessStatusCode) {
                MessageBox.Show(response.ReasonPhrase, LocM.GetString("error"), MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

                lastlabel.Text = LocM.GetString("error");
                lastlabel.Enabled = false;
            } else {
                ...
            }
        }
    }
}
```

Итого получилась вполне работоспособная и функциональная программа, которой уже можно было делать куда больше вещей, чем я планировал делать изначально.

Список настроек на тот момент выглядел так:



Совместимость с Win XP

После я стал думать о совместимости с Windows XP, в итоге оказалось что она поддерживает лишь .NET Framework 4.0, а MultipartFormDataContent доступен лишь в v4.5, но её можно по прежнему подключить в v4.0 установив пакет System.Net.Http. По началу я так и сделал. И вроде всё хорошо, кроме того что на Windows Vista/7 нужно устанавливать .NET Framework 4.0 для того чтобы программа заработала. Переключил проект на 3.5, переписал загрузку изображений на WebClient, и вместо загрузки файла использовал обычное поле с закодированным изображением в формате base64, благо api у imgur позволяет так загружать изображения, да и переписать свой php скрипт не составило труда под этот вариант. А потом решил также переключить проект на версию 2.0, и в итоге банальной правкой пары строк получил полностью рабочий .NET Framework 2.0 проект.

```
using (var client = new WebClient()) {
    var pdata = new NameValueCollection();

    ...

    pdata.Add("image", Convert.ToBase64String(data));

    try {
        var response = client.UploadValues(url, "POST", pdata);
        var result = Encoding.UTF8.GetString(response);

        ...
    }
}
```

```
$file = base64_decode($_POST["image"]);
```

Это всё позволило запускать программу на старых фреймворках, а на Windows Vista/7 запускать без установки чего либо, т.к. согласно [этой статье](#) Windows Vista содержит v2.0, а Windows 7 содержит v3.5 по умолчанию. Но на этом проблемы не закончились. На Windows 8 и новее начало просить установку .NET Framework v3.5, что конечно плохо, но вопрос был быстро решен благодаря [этой информации](#), подправив опции supportedRuntime в конфиге, позволяя запускать приложение на новой или старой версии без каких либо проблем. Кроме этого сделал возможность использования протокола TLS 1.2 если он доступен (т.е. на системах с .NET Framework 4.5).

app.config

```
<startup>
  <supportedRuntime version="v4.0"/>
  <supportedRuntime version="v2.0.50727"/>
</startup>
```

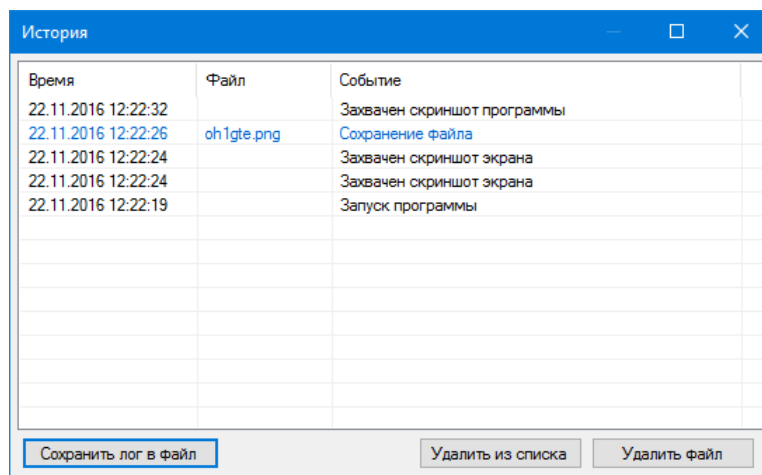
Поддержка TLS 1.2

```
System.Net.ServicePointManager.SecurityProtocol |= SecurityProtocolType.Tls;
try {
    System.Net.ServicePointManager.SecurityProtocol |= (SecurityProtocolType)3072; //SecurityProtocolType.Tls12;
} catch { }
```

История событий

По большому счёту думал что всё, на этом хватит, можно релизить, но всё же по прежнему чего-то не хватало — истории действий с логом. Начал разработку соответствующего окна с некоторыми функциями, как удаления файла с ПК и imgur, открытие файла/ссылки, копирования пути/ссылки с помощью контекстного меню. Также сделал возможность сохранить события в лог файл как из списка, так и автоматически задав в настройках.

Вышло вполне информативное окно:



Проблема в работе HookCallback на Win XP

Но вылезла одна проблема — на Windows XP при захвате скриншотов запись добавлялась дважды. В ходе тестов выяснил, что HookCallback вызывается дважды при отпускании клавиши, причина такого поведения мне была не ясна, но решил вопрос довольно легко — сделал дополнительную проверку нажатия клавиши сохраняя это в переменную, а при отпускании клавиши изменение переменной на false, в итоге нужный мне код стал обрабатываться лишь 1 раз при отпускании клавиши.

```
private static bool pressed = false;

private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam) {
    if (nCode >= 0) {
        Keys number = (Keys)Marshal.ReadInt32(lParam);
        //MessageBox.Show(number.ToString());
        if (number == Keys.PrintScreen) {
            if (pressed && wParam == (IntPtr)261 && Keys.Alt == Control.ModifierKeys && number == Keys.PrintScreen) {
                var res = Scr.Capture(ScreenCatcher.CaptureMode.Window, Properties.Settings.Default.cutborder);
                WinForm.OnGrabScreen(res, false, true);
                pressed = false;
            } else if (pressed && wParam == (IntPtr)257 && number == Keys.PrintScreen) {
                var res = Scr.Capture(ScreenCatcher.CaptureMode.Screen);
                WinForm.OnGrabScreen(res);
                pressed = false;
            } else if (wParam == (IntPtr)256 || wParam == (IntPtr)260) {
                pressed = true; // fix for win xp double press
            }
        }
    }
}
```

```

    }
}

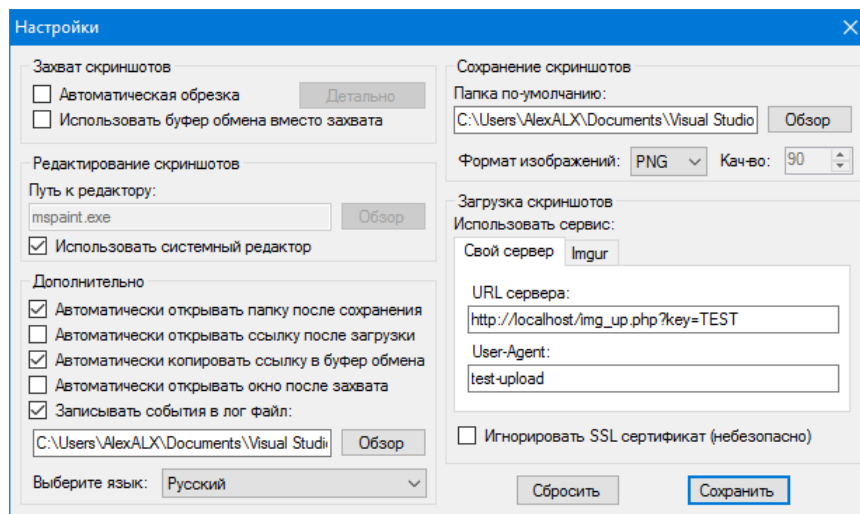
return CallNextHookEx(IntPtr.Zero, nCode, wParam, lParam);
}

```

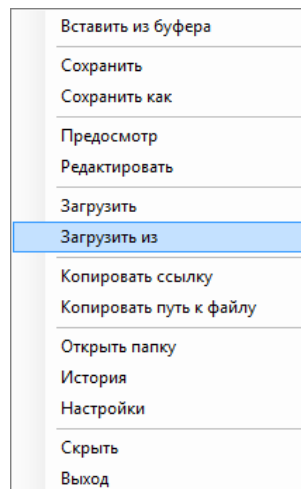
Проблема захвата скриншотов из игр

Чуть позже в ходе тестирования столкнулся с проблемой захвата скриншотов из полноэкранных приложений (например игры), заметил что в windows 10 штатный printscreen захватывает это дело без проблем, в итоге добавил функцию вставки изображения из буфера обмена, а также добавил галочку «использовать буфер обмена вместо захвата» в настройки, тем самым «решил вопрос» для себя, но как оказалось в win 7 и ниже это не работает, начал изучать вопрос, и понял что это довольно сложная задача, с необходимостью использования directx инъекций, в итоге попросту забил на эту проблему, всё-таки основная цель не захват скриншотов из игр, для этого существует множество других программ и инструментов.

Попутно добавив настройки переделал меню настроек, сделал его более компактным чтобы вмещалось на экран с разрешением 640*480 пикселей, и оно стало выглядеть так:



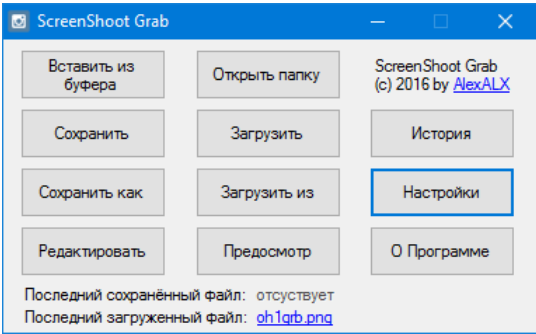
Также сделал более функциональным иконку в трее, добавив туда все важные функции при щелчке правой кнопкой:



Проверка на Win98 и Win2000

Ну и уже чисто ради эксперимента развернул на виртуалке windows 2000 SP4 и 98 SE, поставил там .NET Framework 2.0. Это было сделать не так просто, т.к. требовалась установка некоторых патчей и обновить Windows Installer. Но всё же всё получилось и я попробовал запустить приложение.

Как оказалось на Windows 2000 SP4 приложение оказалось полностью рабочим, а вот на Windows 98 SE захват клавиш не работал, вставка из буфера тоже не работает, однако загрузка скриншота из файла работает без проблем. Собственно эти проблемы решить не получилось, информации крайне мало, всё что смог выяснить — параметр «WH_KEYBOARD_LL» добавили лишь в Windows 2000. А о причине не работающей вставки изображения из буфера вообще не нашёл никакой информации. Итого мин требования — Windows 2000.



Всё что осталось — создать github репозиторий, загрузить исходники, скомпилировать приложение, написать ридми и сделать релиз. На этом история разработки заканчивается. Готовую программу можно скачать и посмотреть исходный код на [GitHub](#). Надеюсь статья была полезной.

🔖 захват изображений, .NET framework, c#

↑ — ↓

👁 7,4k ⭐ 120

@AlexALX

карма рейтинг

9,0 19,2

Программист

Похожие публикации

- +5

Обработка изображений 1C средствами .Net framework при выгрузке на веб-сайт

👁 10,2k ⭐ 30 💬 6
- +10

Мобильный захват изображений: одни лишь разговоры или нечто большее?

👁 1,9k ⭐ 9 💬 3
- +27

.NET Framework: Какую версию вы используете?

👁 23,1k ⭐ 57 💬 50

Реклама помогает поддерживать и развивать наши сервисы

Подробнее

Спецпроект

Самое читаемое				Разработка
Сейчас	Сутки	Неделя	Месяц	
+12	Что означает ноябрьское обновление Steam для инди-разработчиков?			
👁 2,5k	⭐ 8	💬 5		
+44	Консоль разработчика Google Chrome: десять неочевидных полезностей			
👁 10,2k	⭐ 285	💬 26		
+5	10 причин почему именно сейчас стоит попробовать Microsoft SQL Server			
👁 4,7k	⭐ 33	💬 60		

+17

Издательство Питер. Черная пятница 2016

7,5k 19 13

+270 5 способов, которыми игры пытаются вызвать зависимость

108k 766 247

Комментарии (31)



MonkAlex 24 ноября 2016 в 08:57 #

+3 ↑ ↓

Код в формах.

С одной стороны — хорошая программа, неплохая практика и прочая.
С другой — обычная лапша в формах, не то, что радует глаз =)



Sinatr 24 ноября 2016 в 13:15 # h ↑

-5 ↑ ↓

Полностью согласен с «лапшей», ну не солидно это, делится таким кодом (закомментированный отладочный код и стиль уже о многом говорят). Было бы неплохо, если бы первый абзац был вступительный, в стиле «Здравствуйте, меня зовут Алекс и я программист. Мне 10 лет ...». ;)

Статья, кстате, неплохо написана (идея, последовательность, подача материала, совместимость, ...). Единственная претензия — это winforms в 2016. Seriously? Подобное было актуально лет 10+ тому. Локализация эта через ж... сателиты... аж коробит, извините.



AlexALX 24 ноября 2016 в 13:25 (комментарий был изменён) # h ↑

+2 ↑ ↓

Я согласен что имеет место «лапша», писалось на скорою руку. Да и честно скажу — опыта в с# конкретно не очень много, я больше по веб части. По поводу закомментированного отладочного кода — а что здесь не так? Я часто так делаю и в рабочих проектах, более того ещё и подписываю иногда, разве у вас никогда не бывало такого что надо вернуться к отладке? А писать заново код отладки тоже не всегда хочется. Ладно если там одна строчка, а бывают и более сложные конструкции.

это winforms в 2016. Seriously?

А что не так? Писал статью просто чтобы поделиться опытом. К тому же самому вон пришлось собирать инфу по крупицам с гугла. Более того пробую делать что-то новое, написать статью, чего ранее не делал. Разве это плохо?

Локализация эта через ж... сателиты... аж коробит, извините.

Видимо я чего-то не понимаю в этой формулировке, о чём именно речь? Если о качестве перевода программы, то да, мой английский не идеален.

«Здравствуйте, меня зовут Алекс и я программист. Мне 10 лет ...». ;)

Вы почти точно описали вступительную фразу в моём портфолио (правда мне 24), ну уж извиняйте, я пока не профи как вы.



geckyan 24 ноября 2016 в 14:42 # h ↑

0 ↑ ↓

По поводу отладочного кода, в .Net Framework есть неймспейс System.Diagnostics. Там можно найти и Debug, и Trace для логирования соответствующей информации. На их методах навешан атрибут ConditionalAttribute для «включения»/«отключения» методов по дефайну. Собственно, поэтому методы класса Debug «включаются» при дефайне DEBUG, а Trace — при TRACE. И не нужно никаких комментов, голимые директивы, которые можно один раз установить в настройках сборки приложения. Не знаю, хороший ли это способ (сам потихоньку изучаю шарпы:), но, вроде бы, работает, и отпадает проблема каждый раз (раз)комментировать куски кода.



Klotos 24 ноября 2016 в 14:36 # h ↑

+1 ↑ ↓

Без всякой подколки хочу поинтересоваться насчёт локализации. Наверное, я чего-то не знаю, но чем плоха локализация через ресурсные файлы RESX, описанная в статье, и какие есть альтернативы? К тому же, автор не упоминает сборки-сателлиты, а, насколько я знаю, можно скомпилированные RESX внедрять в основную сборку. (Сразу скажу: слабо разбираюсь в локализации десктопных приложений, поэтому и спрашиваю.)



Tujh 24 ноября 2016 в 09:09 #

+1 ↑ ↓

Проверка на Win98 и Win2000

Это имело какой-то практический смысл, или сделано просто из любопытства? Про поддержку XP понимаю, таких ещё относительно много, W2k ещё более-менее понимаю, знаю несколько крупных организаций с серверами на этой ОС, работающими до сих пор, но W9x всё же слишком глубокая археология, на мой взгляд.



AlexALX 24 ноября 2016 в 13:14 # h ↑

+1 ↑ ↓

По большей части всё же любопытство, хотя я тоже знаю людей которые имеют 2000 винду на древних офисных пк. Да и раз работает — почему бы и нет?

Ну а вин98 решил проверить т.к. там была поддержка .NET 2.0 (был несколько удивлён этому на самом деле). Но да, это археология уже))



SlavikF 24 ноября 2016 в 09:52 #

+2 ↑ ↓

Бросается в глаза слово «Screenshoot».

С одной стороны — это как бы ошибка в написании слова «Screenshot», с другой — интересное обыгрывание фразы «Screen Shoot». Возможно...

Однако в Readme.MD у вас там это слово неправильно написано не только в названии (где уместно написать такой «бренд»), но и в тексте... Думаю, стоит исправить.



AlexALX 24 ноября 2016 в 12:56 # h ↑

0 ↑ ↓

Вы абсолютно правы, это ошибка, которую я почему то в упор не замечал аж до сего момента. Самое интересное что я давал ссылки своим зарубежным коллегам чисто для оценки, и никто слова не сказал за то ошибку в слове «screenshot». Но вообще если уж на то пошло, то видимо стоит оставить как есть название программы, и есть в этом некая изюминка) а вот в описании уже поправить слово. Спасибо за наводку.



impwx 24 ноября 2016 в 10:21 #

+2 ↑ ↓

Поставить себе задачу и выполнить ее так, чтобы результат был похож на готовый продукт — это очень похвально. Статья тоже написана доступно и по делу. По поводу качества кода и шероховатостей UI уже сказали выше, но это со временем устаканится :)



KvanTTT 24 ноября 2016 в 12:41 #

+1 ↑ ↓

Приятно, что [инфа](#) из моего старого ответа на [stackoverflow](#) оказалась полезной, и об этом написали :)



Klotos 24 ноября 2016 в 14:32 #

+1 ↑ ↓

Спасибо за статью. Можно поинтересоваться, сколько времени заняла разработка? От появления идеи и начала работы над ней до выкладывания на ГитХаб?



AlexALX 24 ноября 2016 в 15:38 # h ↑

0 ↑ ↓

Смотрите ответ ниже, случайно не там ответить нажал)



AlexALX 24 ноября 2016 в 15:38 #

+1 ↑ ↓

Если говорить о появлении идеи — то очень давно, но всё никак руки не доходили. А со временем частая необходимость быстро загружать скриншоты всё же сподвигла написать программу. Судя по файлам проекта — создал я его 16 ноября (конечно этого года), но активно разрабатывал программу по сути на выходных и уже во вторник (22го) опубликовал (как видно на [github](#)). Итого можно сказать что сделал за 2-3 дня с перерывами.



mayorovp 24 ноября 2016 в 15:45 #

+1 ↑ ↓

Я в своей программе сделал возможность выбрать "рамочкой" участок экрана. Довольно удобно получилось, редактировать скришноты приходится не чаще раза в месяц.



AlexALX 24 ноября 2016 в 15:53 # h ↑

0 ↑ ↓

Я тоже об этом думаю, но честно говоря мне и хватает паинта где можно это выделить и обрезать. Просто не стал пока заморачиваться как это реализуется, позже может доработаю. Но а так, конечно это было бы удобней.



mayorovp 24 ноября 2016 в 16:11 (комментарий был изменён) # h ↑

+1 ↑ ↓

Держите реализацию: <https://github.com/mayorovp/tool-printscreens/blob/master/SelectionFrame.cs>

но честно говоря мне и хватает паинта где можно это выделить и обрезать

В том-то и фишка, что я теперь открываю Paint раз в месяц, хотя скриншоты для JIRA делаю иногда раз 20 в день.



AlexALX 24 ноября 2016 в 23:20 # h ↑

0 ↑ ↓

Да спасибо за ссылку, как-нибудь разберусь на досуге, когда будет больше времени. А пока зарелизил мелкие фиксы и правки.



sYsTem1337 24 ноября 2016 в 17:18 #

0 ↑ ↓

Зачем изобретать велосипед если уже давным давно существует именна та программа что и требуется здесь написать? Image Uploader.

https://ru.wikipedia.org/wiki/Image_Uploader

**AlexALX** 24 ноября 2016 в 17:19 (комментарий был изменён)

↵ ↑

0 ↑ ↓

А где в этой программе загрузка изображения по http на свой сервер? задача была не в загрузке по ftp. К тому же не знал про данную программу, и дописать на C++ было бы куда сложнее для меня чем своё решение сделать, которое знаю от и до. А так конечно от части изобретения велосипеда, согласен, но зато и опыт получил дополнительный)

**rdifb0** 24 ноября 2016 в 19:14 # ↵ ↑

0 ↑ ↓

Каждый программист должен написать свою скриптитилку.

**zigrus** 24 ноября 2016 в 17:52 #

0 ↑ ↓

по работе иногда делать снимок экрана на удаленных машинах в домене пока использую связку psexec и nircmd

**MrMerak** 24 ноября 2016 в 19:43 #

0 ↑ ↓

GreenShot

**mafia8** 24 ноября 2016 в 21:09 #

0 ↑ ↓

Добавить автосохранение после нажатия PrintScreen Alt-PrintScreen как опцию.

С чем связан размер программы полмегабайта?

**AlexALX** 24 ноября 2016 в 21:28 # ↵ ↑

0 ↑ ↓

Да тоже уже об этом подумал, сделаю в новой версии.

По поводу размера — дело в иконке, весит она 361 кб, остальное реальный вес программы.

**mafia8** 24 ноября 2016 в 22:55 # ↵ ↑

+1 ↑ ↓

Greenfish Icon Editor Pro редактирует и сохраняет иконки. Для Висты и выше сохраняет в сжатом виде (PNG), можно настроить количество цветов (32 бита, 24 бита, 256 цветов).

**AlexALX** 25 ноября 2016 в 00:30 # ↵ ↑

0 ↑ ↓

О, спасибо, «ужало» картинку до 48кб при всех нужных размерах. Пожалуй перезалью архив с релизом с новой иконкой.

**NeverIn** 25 ноября 2016 в 14:01 #

0 ↑ ↓

Может ли программа захватывать произвольную область экрана, а не только активное окно или весь экран?
К сожалению, не нашел этой информации в статье и на вики проекта.

**AlexALX** 25 ноября 2016 в 14:08 # ↵ ↑

0 ↑ ↓

На данный момент нет, но мне выше уже прислали пример реализации, обязательно чуть позже реализую т.к. это действительно удобней, чем захватывать весь экран и редактировать обрезаю ненужное (хотя это тоже не сложно).

**HAOSov** 25 ноября 2016 в 21:39 #

0 ↑ ↓

Возможно ли сделать захват через видеодрайвер? Я имею ввиду захват из игр и/или с оверлейных окон? Понимаю что хочу слишком много, но было бы неплохо.

**AlexALX** 25 ноября 2016 в 21:39 (комментарий был изменён) # ↵ ↑



0 ↑ ↓

Мне данный функционал не нужен, потому не хочу тратить лишнее время на изучение сего вопроса. Но может както...

Только зарегистрированные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

Интересные публикации

[«Кузькина мать» по-оверклокерски: как русский энтузиаст выбил три мировых рекорда в один день](#) 8[Что означает ноябрьское обновление Steam для инди-разработчиков?](#) 4[Идем по приборам](#) 1

 Франция, Финляндия, Великобритания и Канада полностью откажутся от сжигания угля  24

 Поднимаем собственный репозиторий пакетов для Ubuntu (Debian)  10