عنوان: **Base64 و کاربرد جالب آن**
نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۹:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: HTML, Embeded Images, Base64

Base64 یک مبنای عددی است که یکی از کاربردهای آن در نرم افزارها انتقال اطلاعات فایل باینری است. به عنوان مثال با تبدیل محتوای باینری یک تصویر به مبنای 64 میتونید اون تصویر رو در دل فایل هایی نظیر HTML و CSS و SVG قرار بدید؛ یا به اصطلاح اون تصویر رو Embeded (توکار) کنید.

نکته: در Base64 برای هر 6 بیت یک کاراکتر معادل در مبنای 64 در نظر گرفته میشه، به همین دلیل در هنگام تبدیل برای جلوگیری از Lost (گم) شدن داده‌ها عملیات مورد نظر رو بر روی سه بایت داده انجام میدیم. که با این کار برای هر 3 بایت (24 بیت) 4 کاراکتر معادل خواهیم داشت؛ به این ترتیب حجم نهایی فایل تبدیل شده در واحد بایت، برابر است با:

(حجم نهایی) = (3 / حجم کل فایل) + (حجم کل فایل)

و حالا نحوه تبدیل فایل تصویر به Base64:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Image2Base64Converter
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnOpen_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Multiselect = false;
            ofd.Filter = "Pictures |*.bmp;*.jpg;*.jpeg;*.png";

            if (ofd.ShowDialog(this) == DialogResult.OK)
            {
                txtImagePath.Text = ofd.FileName;
            }
        }

        private void btnConvert_Click(object sender, EventArgs e)
        {
            if (!File.Exists(txtImagePath.Text))
            {
                MessageBox.Show("File not found!");
                return;
            }

            btnCopy.Enabled = false;
            btnConvert.Enabled = false;
            txtOutput.Enabled = false;

            BackgroundWorker bworker = new BackgroundWorker();
            bworker.DoWork += (o, a) =>
            {
                string header = "";
                header = string.Format(header, txtImagePath.Text.GetExtension());
```

```
                StringBuilder data = new StringBuilder();

                byte[] buffer;
                BinaryReader head = new BinaryReader(
                    new FileStream(txtImagePath.Text, FileMode.Open));

                buffer = head.ReadBytes(6561);
                while (buffer.Length > 0)
                {
                    data.Append(Convert.ToBase64String(buffer));
                    buffer = head.ReadBytes(6561);
                }

                head.Close();
                head.Dispose();

                this.Invoke(new Action(() =>
                {
                    txtOutput.ResetText();
                    txtOutput.AppendText(header);
                    txtOutput.AppendText(data.ToString());

                    btnCopy.Enabled = true;
                    btnConvert.Enabled = true;
                    txtOutput.Enabled = true;
                }));
            };

            bworker.RunWorkerAsync();
        }

        private void btnCopy_Click(object sender, EventArgs e)
        {
            Clipboard.SetText(txtOutput.Text);
        }
    }

    public static class ExtensionMethods
    {
        public static string GetExtension(this string str)
        {
            string ext = string.Empty;
            for (int i = str.Length - 1; i >= 0; i--)
            {
                if (str[i] == '.')
                {
                    break;
                }

                ext = str[i] + ext;
            }

            return ext;
        }
    }
}
```

توضیح کد:

در خط 44 یک BackgroundWorker تعریف شده است تا عملیات تبدیل در یک ترد جداگانه انجام شود، که در عملیات‌های سنگین ترد اصلی برنامه آزاد باشد.

در خطوط 47 تا 64 کدهای مربوط به تبدیل قرار گرفته است:

در خط 47 و 48 ابتدا یک سرآیند برای معرفی داده‌های تبدیل شده ایجاد میکنیم؛ "" که در ان نوع فایل و پسوند آن را مشخص کرده و سپس الگوریتم به کار گرفته شده برای تبدیل آن را نیز ذکر میکنیم: "data:" + نوع فایل (image) + "/" + پسوند فایل + ";" + الگوریتم تبدیل + ",".

در انتهای کار این سرآیند تولید شده در ابتدای داده‌های تبدیل شده فایل، قرار خواهد گرفت.

در خط 50 یک StringBuilder برای نگهداری اطلاعات تبدیل شده ایجاد کرده ایم.

در خط 52 یک بافر برای خواندن اطلاعات باینری فایل ایجاد کرده ایم.

در خط 53 فایل مورد نظر را برای خواندن باز کرده ایم.

و در خطوط 56 تا 61 فایل مورد نظر را خوانده و پس از تبدیل در متغیر data ذخیره کرده ایم.

در نظر داشته باشید که برای عملکرد صحیح لازم است که عملیات تبدیل بر روی 3 بایت داده انجام شود؛ پس در هر بار خواندن

داده‌های فایل، باید مقدار داده ای که خوانده میشود ضریبی از 3 باشد.

در خطوط 63 و 64 نیز منابع اشغال شده سیستم را آزاد کرده ایم.

در انتها داده‌های مورد استفاده برای Embeded کردن تصویر برابر است با:

header + data.ToString()

embeded-images.htm

: فایل نمونه برای نحوه استفاده از تصویر توکار. Image2Base64-Converter.zip

: سورس کامل برنامه ارائه شده.

نظرات خوانندگان

نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۹:۲۰

در کد فوق یک مشکل کوچیک وجود داره که امیدوارم آقای وحید نصیری و یا دیگر دوستان نظر خودشون رو در این رابطه بیان کنن.
مشکل اینه که با اینکه عملیات تبدیل به سرعت انجام میشه، ولی در خط 70 ریختن نتیجه عملیات درون RichTextBox به کندی انجام میشه. برای رفع این مشکل چه کاری میشه انجام داد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۰۲ ۹:۲۹

- مرسوم نیست این همه اطلاعات کد شده رو در یک tetxbox نمایش بدن. به چه دلیلی و چه مقصودی را برآورده می‌کند؟
- برای RichTextBox یک سری متد BeginUpdate و EndUpdate هست.
- در کد فوق می‌شود بجای GetExtension از متد توکار Path.GetExtension استفاده کرد.
- تصاویر وب عموما حجیم نیستند (خصوصا زمانیکه قرار است تبدیل به base64 شوند). یعنی حجم بالای 10 مگابایت ندارند که بخواهید با استریم‌ها کار کنید. بهتر است یک ضرب از متد File.ReadAllBytes استفاده کنید. همچنین در این حالت مشخص (با توجه به حجم پایین تصاویر مورد استفاده در وب سایت‌ها)، استفاده از تردها را هم حذف کنید.
- همیشه زمانیکه کدنویسی می‌کنید این سؤال رو از خودتون بپرسید:
آیا کاری که دارم انجام می‌دم قابلیت استفاده مجدد داره؟ میشه از قسمتی از نتیجه‌اش در یک پروژه دیگر استفاده کرد؟
مثلا در کد شما بهتر است قسمت تبدیل تصویر به معادل base64 آن تبدیل به یک متد کمکی با قابلیت استفاده مجدد شود تا اینکه منطق پیاده سازی آن در بین کدهای UI دفن شده باشد. مطالعه قسمت Refactoring در سایت در این زمینه مفید است.

نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۹:۵۳

آقای نصیری ممنون از نظرات خوبتون.
حتما تو کدنویسی‌های آینده نکات مفیدی که گفتید رو اعمال میکنم.

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۲/۰۲ ۱۲:۴۶

ضمن تشکر از نویسنده عزیز. لازم است کمی در مورد کاربردهای آن بیشتر بحث شود. به طور مثال علت Embed کردن یک تصویر در فایل html.

نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۲ ۱۳:۳۴

- خب البته من این رو در حین کار با فایل‌های SVG یاد گرفتم؛ برای اینکه بتونم روی فایل‌های SVG تولید شده توی برنامه ام مدیریت بهتری داشته باشم، دنبال راهی بودم که تصاویر رو درون اون Embed کنم. که با این راه آشنا شدم.

- برای فایل‌های HTML و CSS کاربرد خاصی نمیتونم مثال بزنم؛ شایدم اصلا کاربرد خاصی نداشته باشه!. ولی توی کار با فایل‌های SVG میتونه مفید واقع بشه و خیالتون رو در مورد Lost شدن تصاویر راحت کنه. هرچند که کلا این روش برای تصاویر حجیم مناسب نیست.

نویسنده: مهدی
تاریخ: ۱۳۹۲/۰۲/۰۳ ۲:۵۷

شاید یک کاربرد آن در CSS این هست که تصویر استفاده شده در فایل CSS به همراه سند CSS قابل جابجایی می‌باشد و نیاز نیست که تصویر بر روی هاست Upload شود.

نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۳۹۲/۰۲/۰۳ ۸:۳۴

یک کاربرد جالب دیگه که به ذهنم رسید اینه که، مثلا فرض کنید میخواید یه برنامه برای آرشیو کردن صفحات وب روی کامپیوتر بنویسید، خب مسلما گنجوندن تمام اطلاعات صفحه وب مورد نظر توی یک فایل، مدیریت اون رو راحت‌تر میکنه.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۰۳ ۹:۱۲

تصاویر base64 از IE8 به بعد پشتیبانی میشه.