عنوان: **کار با اسکنر در برنامه های تحت وب (قسمت اول)**
نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۳/۱۱/۱۷ ۰:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, WCF, TWAIN, Cross-Origin Request

در اکثر برنامه‌های سازمانی، مثل برنامه‌های مدیریت آرشیو اسناد، همواره این نیاز جزو خواسته‌های کاربران بوده که بتوانند به صورت مستقیم و از طریق تنها یک کلیک، تصویر مورد نظر را اسکن کرده و به صورت خودکار در برنامه وارد کنند؛ یعنی بدون اینکه نیاز باشد با استفاده از یک برنامه دیگر ابتدا تصویر را اسکن کرده و سپس در فرم وب، فایل اسکن شده را Browse کنند.
این نیاز اساسا به معنی دسترسی به سخت افزار کاربر از طریق مرورگر می‌باشد که به دلایل متعددی امکان پذیر نیست! مشکلات امنیتی ایجاد شده در این راه بسیار جدی است. اما خوشبختانه راههایی برای رسیدن به این هدف وجود دارند:

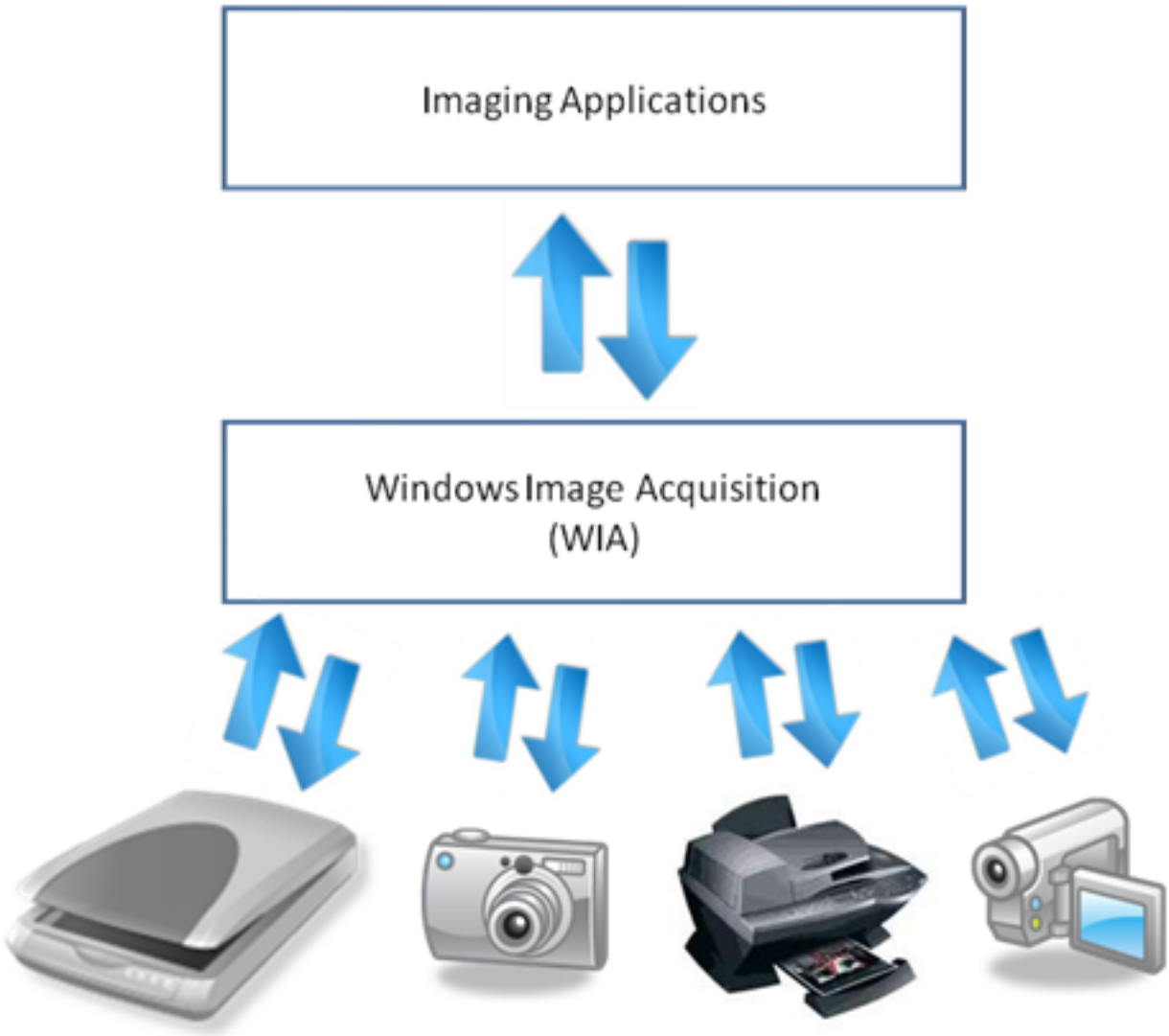
1- **ActiveX** : که به صورت native فقط در IE پشتیبانی می‌شود (در نسخه‌های جدیدتر IE نیاز به بروز رسانی پلاگین ActiveX controls می‌باشد) و برای استفاده‌ی از آن در مرورگرهای Firefox و Chrome هم باید از پلاگین‌های جانبی روی هر مرورگر استفاده کرد که مثلا برای اجرای بر روی Firefox، باید افزونه‌ی IE Tab را نصب کرد که تنها کارش این است که سایت را از طریق موتور IE در پنجره‌ی فایرفاکس اجرا کند! که البته این مورد مثل این می‌ماند که سایت در IE باز شده باشد که ممکن است زیاد خوشایند نباشد؛ در غیر اینصورت باید پروژه را از اول بر مبنای اجرای بر روی IE طراحی و پیاده سازی کرد. در ضمن از مشکلات اجرای پلاگین نوشته شده توسط برنامه نویس در IE و نسخه‌های مختلف آن هم چشم پوشی می‌کنیم. یکی از مزیت‌های این روش نسبت به بقیه این هست که دانلود و نصب پلاگین مورد نیاز به صورت خودکار و توسط مرورگر انجام می‌شود.

2- **استفاده از یک کامپوننت جانبی** : مثل کامپوننت‌های LEADTOOLS که ابزارهای متنوع و SDKهای بسیار قدرتمندی را برای اینکار و کارهای دیگر، مانند کار با اسکن تصاویر مغزی، پردازش تصویر، بارکد خوان‌ها، مدیریت اسناد و ... فراهم کرده است. قیمت این ابزار بسیار زیاد است و در برخی موارد امکانات فراهم آورده بسیار بیشتر است از خواسته‌ی ما. این ابزار، هم در HTML & Javascript و هم در DotNet قابل استفاده است و مستندات نسبتا خوبی هم در این زمینه ارائه کرده است. همچنین کامپوننت Dynamsoft که باز هم غیر رایگان هست و قیمت بالایی نیز دارد.

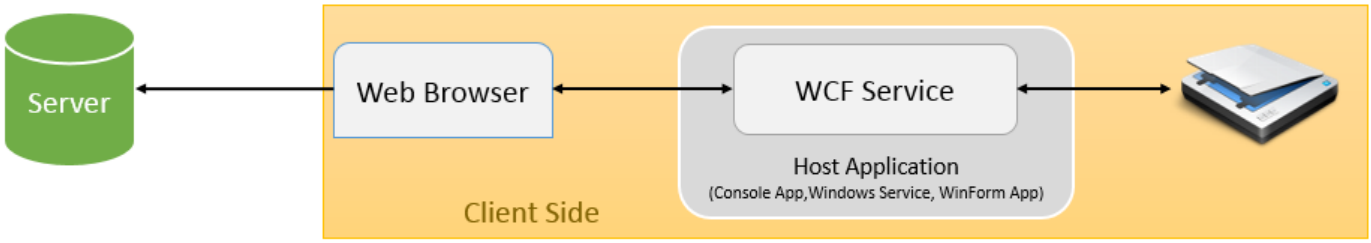
اگر روال کار کامپوننت‌های بالا را مورد بررسی قرار دهید (از طریق اجرای Demo ها، اینجا و اینجا) متوجه خواهید شد که هر دو نیاز به نصب یک سرویس یا App سمت کلاینت جهت اجرای دستورات خود دارند. پس می‌شود اینطور نتیجه گرفت که انجام اینکار بدون اینکه چیزی سمت کاربر نصب شود، ممکن نیست و در هر دو، لینک نصب فایل exe سرویس برای دانلود قرار داده شده است. بر این اساس به راه حل سومی خواهیم رسید که خودمان این سرویس را جهت تعامل با اسکنر سمت کاربر طراحی و پیاده سازی نماییم.

اما چطور ممکن است که با اجرای یک فایل exe سمت کاربر (با این فرض که کاربر در یک دامین مطمئن قرار دارد و می‌شود درخواست نصب سرویس را نمود) این امکان را برای کاربر فراهم نمود که با یک کلیک در مرورگر، اسکنر به صورت خودکار اسکن را آغاز کرده و سپس تصویر حاصل را به یکی از کنترلرهای ما در سمت سرور ارسال نماید؟

برای اینکار ما با دو صورت مساله مواجه هستیم؛ اول اینکه چطور تصویر را سمت کاربر اسکن کنیم و دوم اینکه چطور این تصویر را به سرور ارسال نماییم!
برای مساله‌ی اول از کتابخانه Windows Image Acquisition (WIA) استفاده خواهیم نمود که این کتابخانه به ما این امکان را میدهد تا با سخت افزارهایی که از TWAIN پشتیبانی می‌کنند، بتوانیم ارتباط برقرار نماییم.



برای مساله‌ی دوم هم نیاز به پیاده سازی یک WCF Service و اجرای آن (هاست کردن) در سمت کلاینت داریم. در واقع با صدا زدن متدهای این سرویس، از کتابخانه‌ی بالا استفاده کرده و اسکن را انجام می‌دهیم.



ادامه دارد...

عنوان: **کار با اسکنر در برنامه های تحت وب (قسمت دوم و آخر)**
نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۳/۱۱/۲۰ ۲۳:۳۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, WCF, TWAIN, Cross-Origin Request, Scanning

در قسمت قبل « کار با اسکنر در برنامه‌های تحت وب (قسمت اول) » دیدی از کاری که قرار است انجام دهیم، رسیدیم. حالا سراغ یک پروژه‌ی عملی و پیاده سازی مطالب مطرح شده می‌رویم.

ابتدا پروژه‌ی WCF را شروع می‌کنیم. ویژوال استودیو را باز کرده و از قسمت New Project > Visual C# > WCF یک پروژه‌ی WCF Service Application جدید را مثلا با نام "WcfServiceScanner" ایجاد نمایید. پس از ایجاد، دو فایل IService1.cs و Service1.scv موجود را به IScannerService و ScannerService تغییر نام دهید. سپس ابتدا محتویات کلاس اینترفیس IScannerService را به صورت زیر تعریف نمایید :

```
[ServiceContract]
    public interface IScannerService
    {
        [OperationContract]
        [WebInvoke(Method = "GET",
           BodyStyle = WebMessageBodyStyle.Wrapped,
           RequestFormat = WebMessageFormat.Json,
           ResponseFormat = WebMessageFormat.Json,
           UriTemplate = "GetScan")]
        string GetScan();
    }
```

در اینجا ما فقط اعلان متدهای مورد نیاز خود را ایجاد کرده‌ایم. علت استفاده از Attribute ایی با نام WebInvoke ، مشخص نمودن نوع خروجی به صورت Json است و همچنین عنوان آدرس مناسبی برای صدا زدن متد. پس از آن کلاس ScannerService را مطابق کدهای زیر تغییر دهید:

```
public class ScannerService : IScannerService
    {
        public string GetScan()
        {
            // TODO Add code here
        }
    }
```

تا اینجا فقط یک WCF Service معمولی ساخته‌ایم .در ادامه به سراغ کلاس WIA برای ارتباط با اسکنر می‌رویم.
بر روی پروژه‌ی خود راست کلیک کرده و Add Reference را انتخاب نموده و سپس در قسمت COM، گزینه‌ی Microsoft Windows Image Acquisition Library v2.0 را به پروژه‌ی خود اضافه نمایید.
با اضافه شدن این ارجاع به پروژه، دسترسی به فضای نام WIA برای ما امکان پذیر می‌شود که ارجاعی از آن را در کلاس ScannerService قرار می‌دهیم.

```
using WIA;
```

اکنون متد GetScan را مطابق زیر اصلاح می‌نماییم:

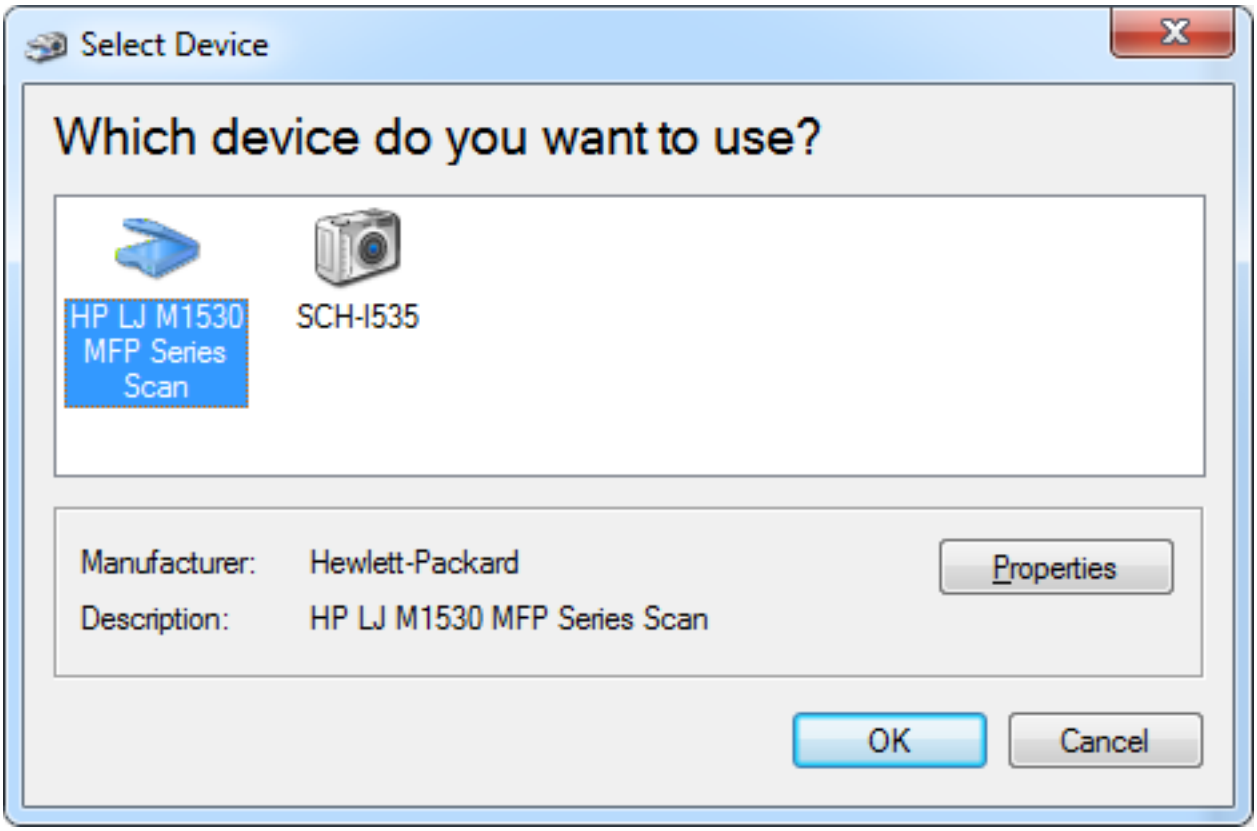
```
public string GetScan()
        {
            var imgResult = String.Empty;
            var dialog = new CommonDialogClass();
            try
            {
                // نمایش فرم پیشفرض اسکنر
                var image = dialog.ShowAcquireImage(WiaDeviceType.ScannerDeviceType);

                // ذخیره تصویر در یک فایل موقت
                var filename = Path.GetTempFileName();
```

```
                image.SaveFile(filename);
                var img = Image.FromFile(filename);

                // img تبدیل تصویر به Base64 جهت ارسال سمت کاربر و نمایش در تگ
                imgResult = ImageHelper.ImageToBase64(img, ImageFormat.Jpeg);
            }
            catch
            {
                // از آنجاییه که امکان نمایش خطا وجود ندارد در صورت بروز خطا رشته خالی
                // بازگردانده می‌شود که به معنای نبود تصویر می‌باشد
            }

            return imgResult;
        }
```

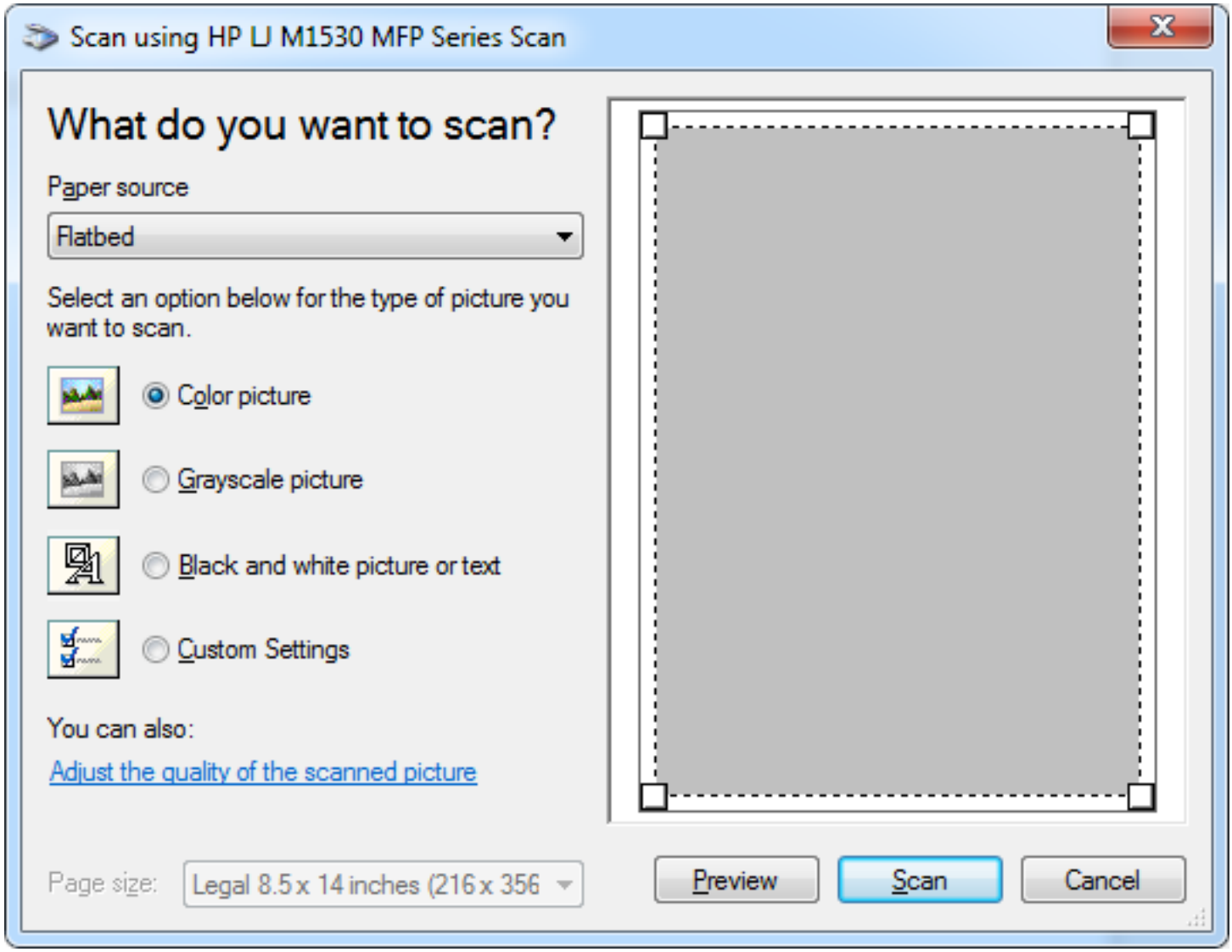
دقت داشته باشید که کدها را در زمان توسعه بین Try..Catch قرار ندهید چون ممکن‌است در این زمان به خطاهایی برخورد کنید که نیاز باشد در مرورگر آنها را دیده و رفع خطا نمایید.

CommonDialogClass کلاس اصلی در اینجا جهت نمایش فرم کار با اسکنر می‌باشد و متدهای مختلفی را جهت ارتباط با اسکنر در اختیار ما قرار می‌دهد که بسته به نیاز خود می‌توانید از آنها استفاده کنید. برای نمونه در مثال ما نیز متد اصلی که مورد استفاده قرار گرفته، ShowAcquireImage می‌باشد که این متد، فرم پیش فرض دریافت اسکنر را به کاربر نمایش می‌دهد و کاربر از طریق آن می‌تواند قبل از شروع اسکن، یکسری تنظیمات را انجام دهد.

این متد ابتدا به صورت خودکار فرم تعیین دستگاه اسکنر ورودی را نمایش داده :



و سپس فرم پیش فرض اسکنرهای TWAIN را جهت تعیین تنظیمات اسکن نمایش می‌دهد که این امکان نیز در این فرم فراهم است تا دستگاه‌های Feeder یا Flated انتخاب گردند.



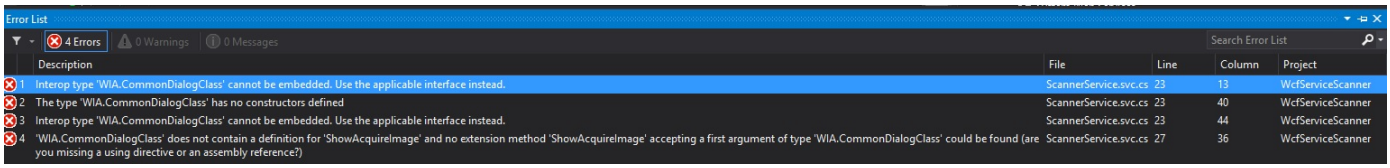
خروجی این متد همان عکس اسکن شده است که از نوع WIA.ImageFile می‌باشد و ما پس از دریافتش، ابتدا آن را در یک فایل موقت ذخیره نموده و سپس با استفاده از یک متد کمکی آن را به فرمت Base64 برای درخواست کننده اسکن ارسال می‌نماییم.

کدهای کلاس کمکی ImageHelper:

```
public static string ImageToBase64(Image image, System.Drawing.Imaging.ImageFormat format)
        {
            if (image != null)
            {
                using (MemoryStream ms = new MemoryStream())
                {
                    // Convert Image to byte[]
                    image.Save(ms, format);
                    byte[] imageBytes = ms.ToArray();

                    // Convert byte[] to Base64 String
                    string base64String = Convert.ToBase64String(imageBytes);
                    return base64String;
                }
            }
            return String.Empty;
        }
```

توجه داشته باشید که خروجی این متد قرار است توسط callBack یک متد جاوا اسکریپتی مورد استفاده قرار گرفته و احیانا عکس مورد نظر در صفحه نمایش داده شود. پس بهتر است که از قالب تصویر به شکل Base64 استفاده گردد. ضمن اینکه پلاگین‌های Jquery مرتبط با ویرایش تصویر هم از این قالب پشتیبانی می‌کنند. (اینجا)
این مثال به ساده‌ترین شکل نوشته شد. کلاس دیگری هم در اینجا وجود دارد و در صورتیکه از اسکنر نوع Feeder استفاده می‌کنید، می‌توانید از کدهای آن استفاده کنید.

کار ما تا اینجا در پروژه‌ی WCF Service تقریبا تمام است. اگر پروژه را یکبار Build نمایید برای اولین بار احتمالا پیغام خطاهای زیر ظاهر خواهند شد:



جهت رفع این خطا، در قسمت Referenceهای پروژه خود، WIA را انتخاب نموده و از Propertiesهای آن خصوصیت Embed Interop Types را به False تغییر دهید؛ مشکل حل می‌شود.

به سراغ پروژه‌ی ویندوز فرم جهت هاست کردن این WCF سرویس می‌رویم. می‌توانید این سرویس را بر روی یک Console App یا Windows Service هم هاست کنید که در اینجا برای سادگی مثال، از WinForm استفاده می‌کنیم.
یک پروژه‌ی WinForm جدید را ایجاد کنید و سپس از قسمت Solution > Add Reference به مسیر پروژه‌ی قبلی رفته و dllهای آن را به پروژه خود اضافه نمایید.
Form1.cs را باز کرده و ابتدا دو متغیر زیر را در آن به صورت عمومی تعریف نمایید:

```
private readonly Uri _baseAddress = new Uri("http://localhost:6019");
        private ServiceHost _host;
```

برای استفاده از کلاس ServiceHost لازم است تا ارجاعی به فضای نام System.ServiceModel داده شود. متغیر _baseAddress نگه دارنده‌ی آدرس ثابت سرویس اسکنر در سمت کلاینت می‌باشد و به این ترتیب ما دقیقا می‌دانیم باید سرویس را با کدام آدرس در کدهای جاوا اسکریپتی خود فراخوانی نماییم.
حال در رویداد Form_Load برنامه، کدهای زیر را جهت هاست کردن سرویس اضافه می‌نماییم:

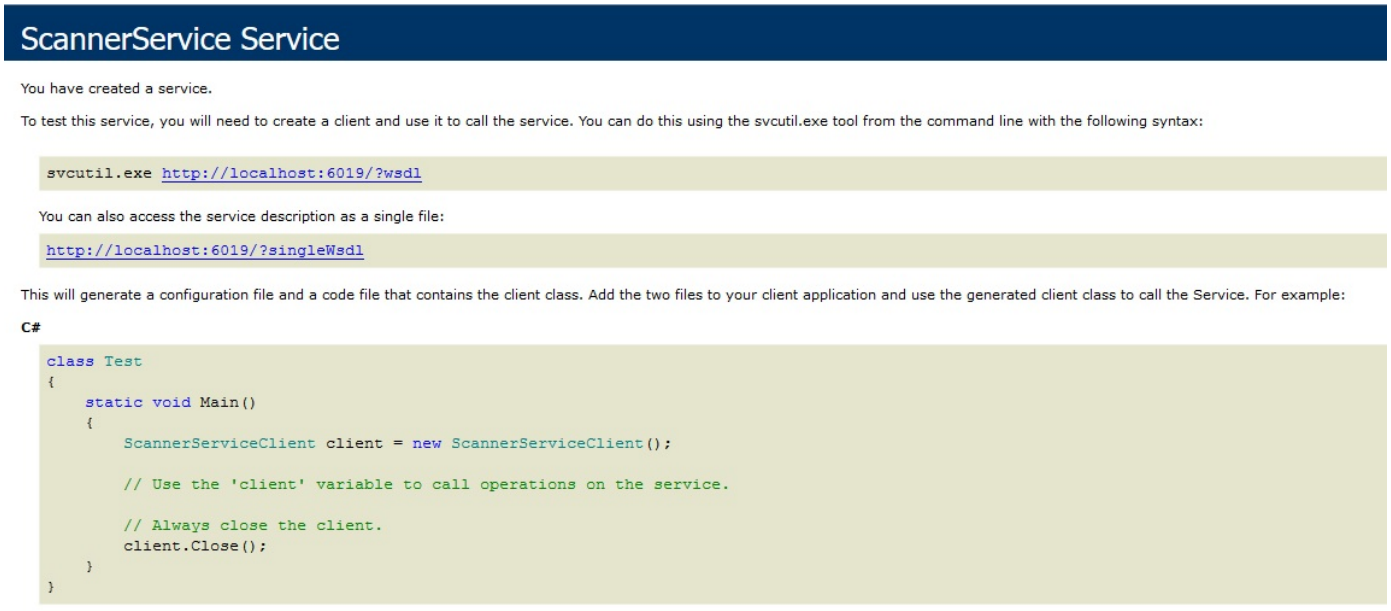
```
private void Form1_Load(object sender, EventArgs e)
        {
            _host = new ServiceHost(typeof(WcfServiceScanner.ScannerService), _baseAddress);
            _host.Open();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            _host.Close();
        }
```

همین چند خط برای هاست کردن سرویس روی آدرس localhost و پورت 8010 کامپیوتر کلاینت کافی است. اما یکسری تنظیمات مربوط به خود سرویس هم وجود دارد که باید در زمان پیاده سازی سرویس، در خود پروژه‌ی سرویس، ایجاد می‌گردید. اما از آنجا که ما قرار است سرویس را در یک پروژه‌ی دیگر هاست کنیم، بنابراین این تنظیمات را باید در همین پروژه‌ی WinForm قرار دهیم.
فایل App.Config پروژه‌ی WinForm را باز کرده و کدهای آنرا مطابق زیر تغییر دهید:

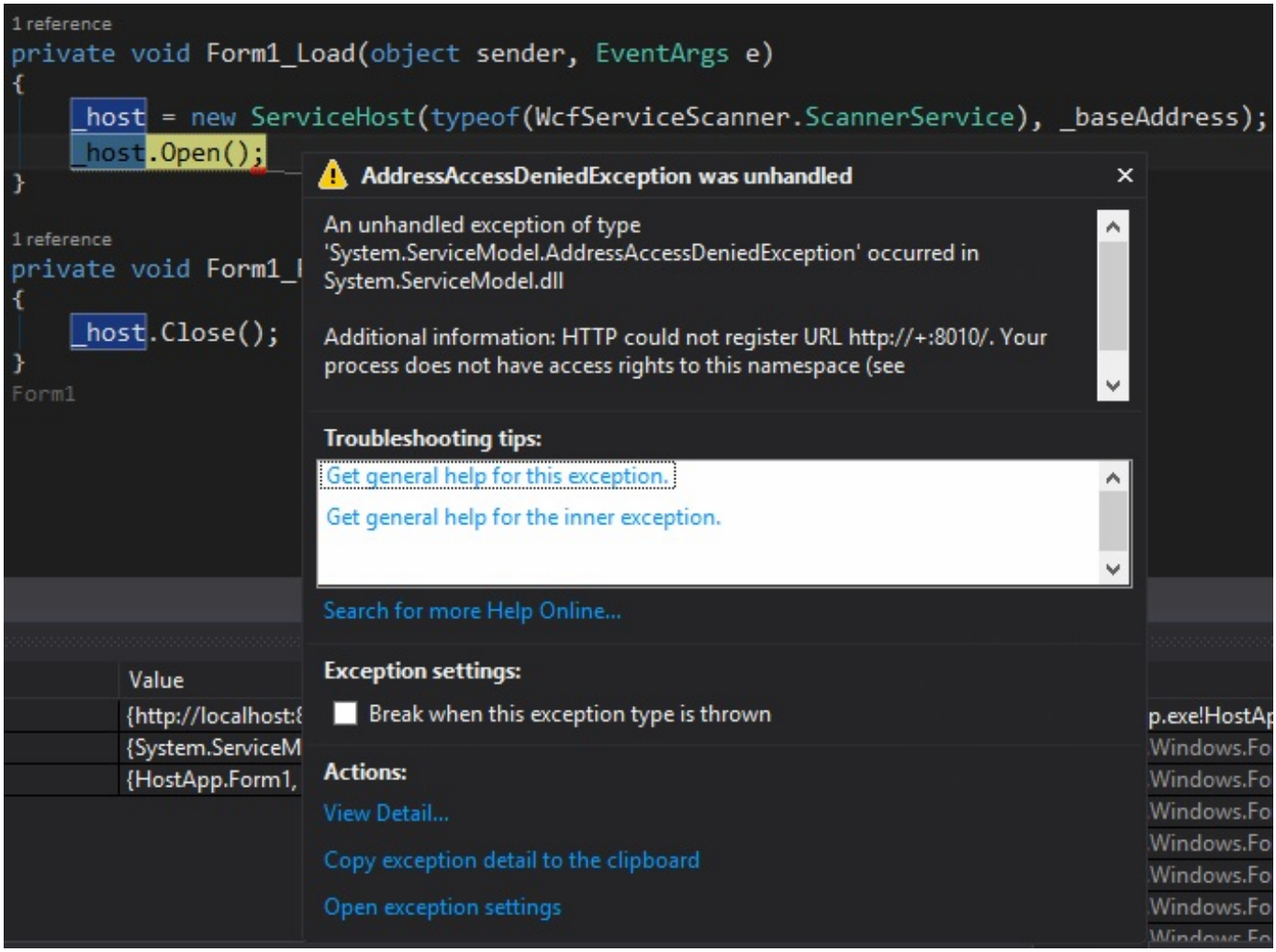
```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>

  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="BehaviourMetaData">
          <serviceMetadata httpGetEnabled="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <services>
      <service name="WcfServiceScanner.ScannerService"
               behaviorConfiguration="BehaviourMetaData">
        <endpoint address=""
                  binding="basicHttpBinding"
                  contract="WcfServiceScanner.IScannerService" />
      </service>
    </services>
  </system.serviceModel>

</configuration>
```

اکنون پروژه‌ی هاست آماده اجرا می‌باشد. اگر آنرا اجرا کنید و در مرورگر خود آدرس ذکر شده را وارد کنید، صفحه‌ی زیر را مشاهده خواهد کرد که به معنی صحت اجرای سرویس اسکنر می‌باشد.



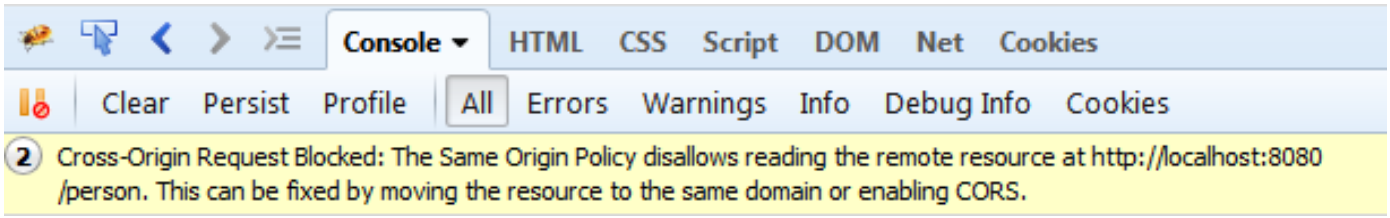
اگر موفق به اجرا نشدید و احیانا با خطای زیر مواجه شدید، اطمینان حاصل کنید که ویژوال استودیو Run as Administrator باشد. مشکل حل خواهد شد.



به سراغ پروژه‌ی بعدی، یعنی وب سایت خود می‌رویم. یک پروژه‌ی MVC جدید ایجاد نمایید و در View مورد نظر خود، کدهای زیر را جهت صدا زدن متد GetScan اضافه می‌کنیم.
(از آنجا که کدها به صورت جاوا اسکریپت می‌باشد، پس مهم نیست که حتما پروژه MVC باشد؛ یک صفحه‌ی HTML ساده هم کافی است).

```
<a href="#" id="get-scan">Get Scan</a>
<img src="" id="img-scanned" />
<script>
    $("#get-scan").click(function () {
        var url = 'http://localhost:6019/';
        $.get(url, function (data) {
            $("#img-scanned").attr("src","  "+ data.GetScanResult);
        });
    });
</script>
```

دقت کنید در هنگام دریافت اطلاعات از سرویس، نتیجه به شکل GetScanResult خواهد بود. الان اگر پروژه را اجرا نمایید و روی لینک کلیک کنید، اسکنر شروع به دریافت اسکن خواهد کرد اما نتیجه‌ای بازگشت داده نخواهد شد و علت هم مشکل امنیتی CORS می‌باشد که به دلیل دریافت اطلاعات از یک دامین دیگر رخ می‌دهد و اگر با Firebug درخواست را بررسی کنید متوجه خطا به شکل زیر خواهید شد.



راه حل‌های زیادی برای این مشکل ارائه شده است، و متاسفانه بسیاری از آنها در شرایط پروژه‌ی ما جوابگو نمی‌باشد (به دلیل هاست روی یک پروژه ویندوزی). تنها راه حل مطمئن (تست شده) استفاده از یک کلاس سفارشی در پروژه‌ی WCF Service می‌باشد که مثال آن در اینجا آورده شده است.

برای رفع مشکل به پروژه WcfServiceScanner بازگشته و کلاس جدیدی را به نام CORSSupport ایجاد کرده و کدهای زیر را به آن اضافه کنید:

```
public class CORSSupport : IDispatchMessageInspector
    {
        Dictionary<string, string> requiredHeaders;
        public CORSSupport(Dictionary<string, string> headers)
        {
            requiredHeaders = headers ?? new Dictionary<string, string>();
        }

        public object AfterReceiveRequest(ref System.ServiceModel.Channels.Message request,
System.ServiceModel.IClientChannel channel, System.ServiceModel.InstanceContext instanceContext)
        {
            var httpRequest = request.Properties["httpRequest"] as HttpRequestMessageProperty;
            if (httpRequest.Method.ToLower() == "options")
                instanceContext.Abort();
            return httpRequest;
        }

        public void BeforeSendReply(ref System.ServiceModel.Channels.Message reply, object
correlationState)
        {
            var httpResponse = reply.Properties["httpResponse"] as HttpResponseMessageProperty;
            var httpRequest = correlationState as HttpRequestMessageProperty;

            foreach (var item in requiredHeaders)
            {
                httpResponse.Headers.Add(item.Key, item.Value);
            }
            var origin = httpRequest.Headers["origin"];
            if (origin != null)
                httpResponse.Headers.Add("Access-Control-Allow-Origin", origin);

            var method = httpRequest.Method;
            if (method.ToLower() == "options")
                httpResponse.StatusCode = System.Net.HttpStatusCode.NoContent;

        }
    }

    // Simply apply this attribute to a DataService-derived class to get
    // CORS support in that service
    [AttributeUsage(AttributeTargets.Class)]
    public class CORSSupportBehaviorAttribute : Attribute, IServiceBehavior
    {
        #region IServiceBehavior Members

        void IServiceBehavior.AddBindingParameters(ServiceDescription serviceDescription,
ServiceHostBase serviceHostBase, System.Collections.ObjectModel.Collection<ServiceEndpoint> endpoints,
BindingParameterCollection bindingParameters)
        {
        }

        void IServiceBehavior.ApplyDispatchBehavior(ServiceDescription serviceDescription,
ServiceHostBase serviceHostBase)
        {
            var requiredHeaders = new Dictionary<string, string>();

            //Chrome doesn't accept wildcards when authorization flag is true
```

```
            //requiredHeaders.Add("Access-Control-Allow-Origin", "*");
            requiredHeaders.Add("Access-Control-Request-Method", "POST,GET,PUT,DELETE,OPTIONS");
            requiredHeaders.Add("Access-Control-Allow-Headers", "Accept, Origin, Authorization, X-
Requested-With,Content-Type");
            requiredHeaders.Add("Access-Control-Allow-Credentials", "true");
            foreach (ChannelDispatcher cd in serviceHostBase.ChannelDispatchers)
            {
                foreach (EndpointDispatcher ed in cd.Endpoints)
                {
                    ed.DispatchRuntime.MessageInspectors.Add(new CORSSupport(requiredHeaders));
                }
            }
        }

        void IServiceBehavior.Validate(ServiceDescription serviceDescription, ServiceHostBase
serviceHostBase)
        {
        }

        #endregion
    }
```

فضاهای نام لازم برای این کلاس به شرح زیر می‌باشد:

```
using System.ServiceModel;
using System.ServiceModel.Channels;
using System.ServiceModel.Description;
using System.ServiceModel.Dispatcher;
```

کلاس ScannerService را باز کرده و آنرا به ویژگی

```
[CORSSupportBehavior]
    public class ScannerService : IScannerService
    {
```

مزین نمایید.

کار تمام است، یکبار دیگر ابتدا پروژه‌ی WcfServiecScanner و سپس پروژه هاست را Build کرده و برنامه‌ی هاست را اجرا کنید. اکنون مشاهده می‌کنید که با زدن دکمه‌ی اسکن، اسکنر فرم تنظیمات اسکن را نمایش می‌دهد که پس از زدن دکمه‌ی Scan، پروسه آغاز شده و پس از اتمام، تصویر اسکن شده در صفحه‌ی وب سایت نمایش داده می‌شود.

نظرات خوانندگان

نویسنده: امیر بختیاری
تاریخ: ۱۳۹۳/۱۱/۲۱ ۱:۴۸

با سلام و تشکر خدمت شما دوست گرامی
من تمام مراحل را به دقت ایجاد کردم ولی بازم همون خطایی که خودتون هم عنوان کردید را میده

```
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at http://localhost:13748/. This can be fixed by moving the resource to the same domain or enabling CORS.
```

فایل پروژه را به طور کامل میزارم ببینید علت چیه WcfServiceScanner.zip

با تشکر مجدد

نویسنده: شروین ایرانی
تاریخ: ۱۳۹۳/۱۱/۲۱ ۹:۵

با سلام؛ آیا امکانش هست که سورس پروژه‌ی مثال رو هم قرار بدید؟ تشکر

نویسنده: امیر بختیاری
تاریخ: ۱۳۹۳/۱۱/۲۱ ۱۰:۳۱

سورس پروژه را از نظر قبلی می‌تونید دریافت کنید