عنوان: **اضافه کردن Watermark به تصاویر یک برنامه ASP.NET MVC در صورت لینک شدن در سایتی دیگر**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۹ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, Watermark

درگیر شدن با سایت‌های دیگر که چرا مطالب ما را کپی کرده‌اید نهایتا بجز فرسایش عصبی حاصل دیگری را به همراه ندارد. اساسا زمانیکه مطلبی را به صورت باز در اینترنت انتشار می‌دهید، قید کپی شدن یا نشدن آن‌را باید زد. اما ... می‌توان همین سایت‌ها را تبدیل به تبلیغ کننده‌های رایگان کار خود نمود که در ادامه نحوه انجام آن‌را در یک برنامه ASP.NET MVC بررسی خواهیم کرد:

الف) نیاز است ارائه تصاویر تحت کنترل برنامه باشند.

```
using System.IO;
using System.Net.Mime;
using System.Web.Mvc;

namespace MvcWatermark.Controllers
{
    public class HomeController : Controller
    {
        const int ADay = 86400;

        public ActionResult Index()
        {
            return View();
        }

        [OutputCache(VaryByParam = "fileName", Duration = ADay)]
        public ActionResult Image(string fileName)
        {
            fileName = Path.GetFileName(fileName); // تمیز سازی امنیتی است
            var rootPath = Server.MapPath("~/App_Data/Images");
            var path = Path.Combine(rootPath, fileName);
            if (!System.IO.File.Exists(path))
            {
                var notFoundImage = "notFound.png";
                path = Path.Combine(rootPath, notFoundImage);
                return File(path, MediaTypeNames.Image.Gif, notFoundImage);
            }
            return File(path, MediaTypeNames.Image.Gif, fileName);
        }
    }
}
```

در اینجا یک کنترلر را مشاهده می‌کنید که در اکشن متد Image آن، نام یک فایل دریافت شده و سپس این نام در پوشه App_Data/Images جستجو گردیده و نهایتا در مرورگر کاربر Flush می‌شود. از آنجائیکه الزامی ندارد fileName، واقعا یک fileName صحیح باشد، نیاز است توسط متد استاندارد Path.GetFileName این نام دریافتی اندکی تمیز شده و سپس مورد استفاده قرار گیرد. همچنین جهت کاهش بار سرور، از یک OutputCache به مدت یک روز نیز استفاده گردیده است.
نحوه استفاده از این اکشن متد نیز به نحو زیر است:

```
<img src="@Url.Action(actionName: "Image", controllerName: "Home", routeValues: new { fileName =
"EF_Stra_08.gif" })" />
```

ب) آیا فراخوان تصویر ما را مستقیما در سایت خودش قرار داده است؟

```
private bool isEmbeddedIntoAnotherDomain
        {
            get
```

```
            {
                return this.HttpContext.Request.UrlReferrer != null &&
!this.HttpContext.Request.Url.Host.Equals(this.HttpContext.Request.UrlReferrer.Host,
StringComparison.InvariantCultureIgnoreCase);
            }
        }
```

در ادامه توسط خاصیت سفارشی isEmbeddedIntoAnotherDomain درخواهیم یافت که درخواست رسیده، از دومین جاری صادر شده است یا خیر. اینکار توسط بررسی UrlReferrer ارسال شده توسط مرورگر صورت می‌گیرد. اگر Host این UrlReferrer با Host درخواست جاری یکی بود، یعنی تصویر از سایت خودمان فراخوانی شده‌است.

ج) افزودن خودکار Watermark در صورت کپی شدن در سایتی دیگر

```
private byte[] addWaterMark(string filePath, string text)
        {
            var image = new WebImage(filePath);
            image.AddTextWatermark(text);
            return image.GetBytes();
        }
```

کلاسی در فضای نام System.Web.Helpers وجود دارد به نام WebImage که کار افزودن Watermark را بسیار ساده کرده است.
نمونه‌ای از نحوه استفاده از آن‌را در متد فوق ملاحظه می‌کنید.
اما ... پس از امتحان تصاویر مختلف ممکن است گاها با خطای زیر مواجه شویم:

```
A Graphics object cannot be created from an image that has an indexed pixel format.
```

مشکل از اینجا است که تصاویر با فرمت ذیل برای انجام کار Watermark پشتیبانی نمی‌شوند:

```
PixelFormatUndefined
PixelFormatDontCare
PixelFormat1bppIndexed
PixelFormat4bppIndexed
PixelFormat8bppIndexed
PixelFormat16bppGrayScale
PixelFormat16bppARGB1555
```

اما می‌توان تصویر دریافتی را ابتدا تبدیل به BMP کرد و سپس Watermark دار نمود:

```
private byte[] addWaterMark(string filePath, string text)
        {
            using (var img = System.Drawing.Image.FromFile(filePath))
            {
                using (var memStream = new MemoryStream())
                {
                    using (var bitmap = new Bitmap(img))//avoid gdi+ errors
                    {
                        bitmap.Save(memStream, ImageFormat.Png);
                        var webImage = new WebImage(memStream);
                        webImage.AddTextWatermark(text, verticalAlign: "Top", horizontalAlign: "Left",
fontColor: "Brown");
                        return webImage.GetBytes();
                    }
                }
            }
        }
```

در اینجا نمونه اصلاح شده متد addWaterMark فوق را بر اساس کار با تصاویر bmp و سپس تبدیل آن‌ها به png، ملاحظه می‌کنید. به این ترتیب دیگر به خطای یاد شده بر نخواهیم خورد.

در ادامه، قسمت آخر کار، اعمال این مراحل به اکشن متد Image است:

```
if (isEmbeddedIntoAnotherDomain)
            {
                var text = Url.Action(actionName: "Index", controllerName: "Home", routeValues: null,
protocol: "http");
                var content = addWaterMark(path, text);
                return File(content, MediaTypeNames.Image.Gif, fileName);
            }
            return File(path, MediaTypeNames.Image.Gif, fileName);
```

در اینجا اگر تشخیص داده شود که تصویر، در دومین دیگری لینک شده است، آدرس سایت ما به صورت خودکار در بالای تصویر درج خواهد شد.

کدهای نهایی این کنترلر را از اینجا می‌توانید دریافت کنید:

HomeController.cs

به همراه نمونه تصویری که استثنای یاد شده را تولید می‌کند؛ جهت آزمایش بیشتر:

EFStra08.gif

نظرات خوانندگان

نویسنده: محسن جمشیدی
تاریخ: ۱۳۹۲/۰۲/۱۹ ۸:۵۷

اتفاقا دیروز دیدم در سایتی مطالب این سایت استفاده شده بود که برایم خوشایند نبود. توجهم به تصاویر جلب شد که نام dotnettips درج شده بود فکر کردم که جدیدا این امکان اضافه شده و بعد از بررسی متوجه شدم که این اتفاق در سایتهای دیگر می‌افتد برایم جالب بود

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۴ ۸:۲۷

جهت تکمیل بحث، با استفاده از IIS Url Rewrite نیز می‌توان شبیه چنین کاری را انجام داد.

نویسنده: حمید حسین وند
تاریخ: ۱۳۹۳/۰۲/۱۰ ۰:۲۳

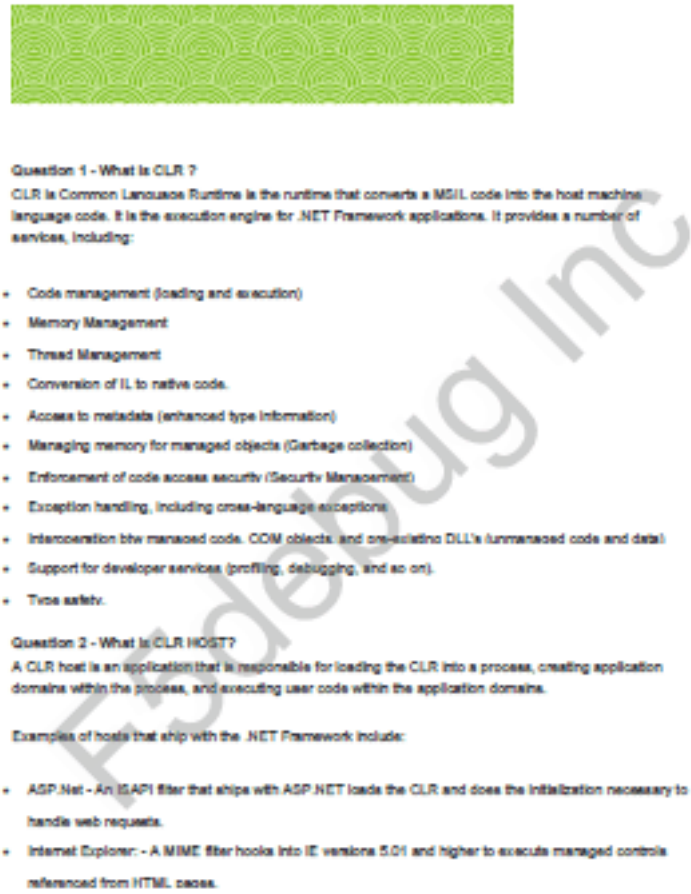
سلام
آیا میشه همچین کاری رو توی asp.net web form کرد؟
یا اصلا یه کاری میشه کرد وقتی تصویر رو از طریق همین redactor (فایل ashx) آپلود میکنیم همون لحظه لوگویی یا متنی رو بهش اضافه کنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۰ ۰:۳۲

کدهای فوق قابل تبدیل به یک Generic handler وب فرم‌ها (فایل‌های ashx) هستند. در اینجا بجای Url.Action فوق برای مقدار دهی img src، خواهید داشت pic.ashx?filename=tst.png.
ضمنا ASP.NET MVC سورس باز است و کدهای متد AddTextWatermark آن‌را در اینجا می‌توانید مطالعه کنید.

عنوان: **نامرئی کردن Watermarkهای صفحات فایل‌های PDF توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۳۰ ۸:۰
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp, Watermark

احتمالا بارها با PDFهایی که یک Watermark بزرگ را در میانه صفحات خود دارند، برخورد داشته‌اید و متاسفانه در اغلب اوقات استفاده ناصحیحی از این قابلیت صورت می‌گیرد. هدف از Watermark دار کردن صفحات PDF، ذکر جملاتی مانند «آزمایشی بودن» یا «محرمانه بودن» است که در هر دو حالت نباید به صورت عمومی منتشر شوند. اما اگر قرار است مطلبی را به صورت عمومی منتشر کنیم، این روش، بدترین حالت تبلیغی برای یک شخص یا شرکت خواهد بود؛ چون مانع خواندن روان متن شده و اعصاب مصرف کننده را به هم خواهد ریخت. بنابراین هیچگونه جنبه تبلیغی مثبتی را در نهایت برای تهیه کننده به همراه نخواهد داشت.
برای نمونه فایل نمونه سؤالات مصاحبات دات را از اینجا دریافت کنید . یک چنین شکلی دارد:



خوب! چطور می‌توان این Watermark را حذف یا حداقل نامرئی کرد؟
برای پاسخ به این سؤال نیاز است ابتدا با نحوه Watermark دار کردن صفحات یک فایل PDF آشناشویم.

الف) ایجاد یک فایل PDF ساده

```
private static void createPdfFile(string pdfFile)
        {
            using (var fs = new FileStream(pdfFile, FileMode.Create, FileAccess.Write, FileShare.None))
            {
                using (var doc = new Document(PageSize.A4))
                {
                    using (var witier = PdfWriter.GetInstance(doc, fs))
                    {
```

```
                        doc.Open();
                        for (int i = 1; i <= 5; i++)
                        {
                            doc.NewPage();
                            doc.Add(new Paragraph(String.Format("This is page {0}", i)));
                        }
                        doc.Close();
                    }
                }
            }
        }
```

در اینجا یک فایل PDF با 5 صفحه ایجاد می‌شود.

ب) افزودن Watermark به فایل PDF تهیه شده

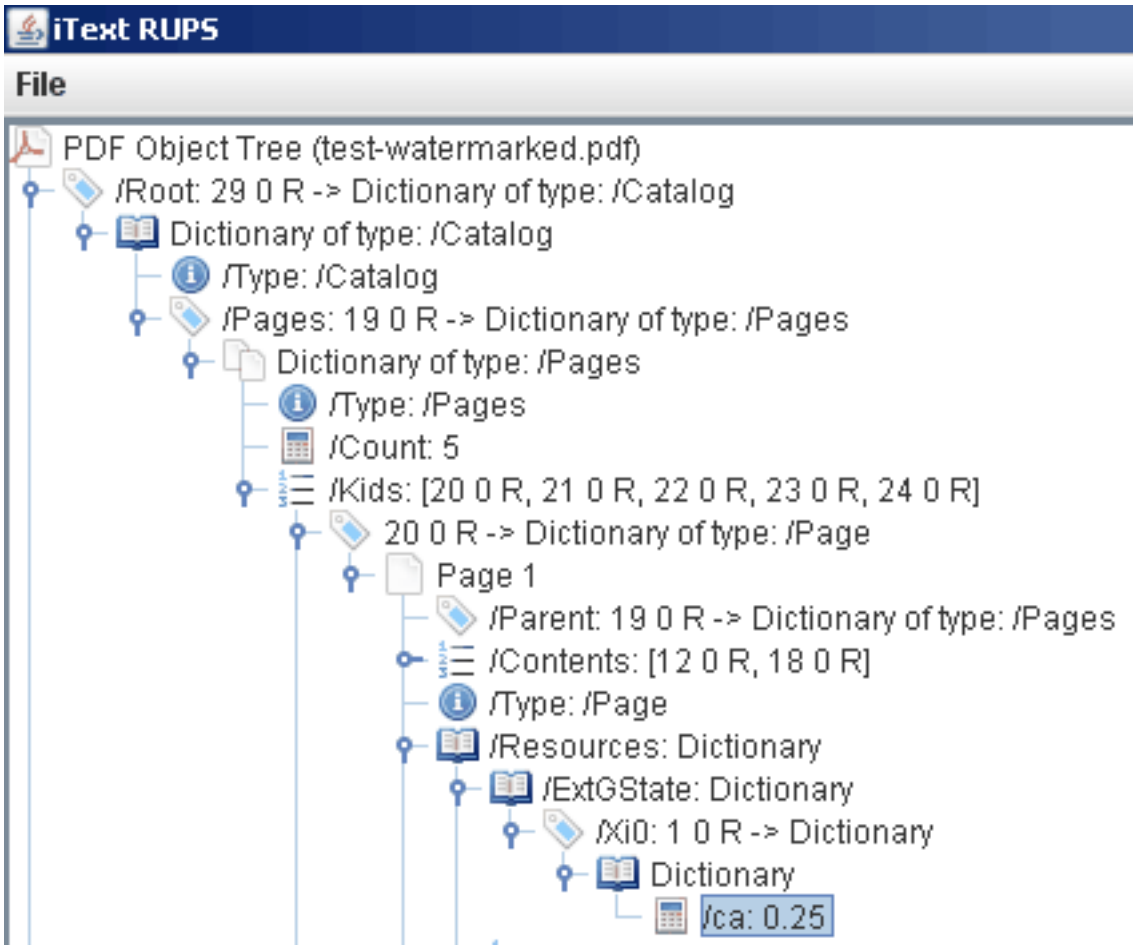
```
private static void addWatermark(string pdfFile, string watermarkedFile, string watermarkText)
        {
            FontFactory.Register("c:\\windows\\fonts\\tahoma.ttf");
            var tahoma = FontFactory.GetFont("Tahoma", BaseFont.IDENTITY_H, 40, Font.NORMAL,
BaseColor.BLACK);

            var reader = new PdfReader(pdfFile);
            using (var fileStream = new FileStream(watermarkedFile, FileMode.Create, FileAccess.Write,
FileShare.None))
            {
                using (var stamper = new PdfStamper(reader, fileStream))
                {
                    int pageCount = reader.NumberOfPages;
                    for (int i = 1; i <= pageCount; i++)
                    {
                        var rect = reader.GetPageSize(i);
                        var cb = stamper.GetUnderContent(i);
                        var gState = new PdfGState();
                        gState.FillOpacity = 0.25f;
                        cb.SetGState(gState);

                        ColumnText.ShowTextAligned(
                            canvas: cb,
                            alignment: Element.ALIGN_CENTER,
                            phrase: new Phrase(watermarkText, tahoma),
                            x: rect.Width / 2,
                            y: rect.Height / 2,
                            rotation: 45f,
                            runDirection: PdfWriter.RUN_DIRECTION_LTR,
                            arabicOptions: 0);
                    }
                }
            }
        }
```

در متد فوق pdfFile نام و مسیر فایل PDF ایی است که قرار است به صفحات آن Watermark اضافه شود. نام و مسیر فایل خروجی توسط watermarkedFile مشخص می‌شود و watermarkText متنی است که در میانه صفحه نمایش داده خواهد شد. در اینجا توسط PdfReader، فایل موجود گشوده می‌شود. به این ترتیب می‌توان به تک تک صفحات این فایل دسترسی یافت. از PdfStamper برای نوشتن در این فایل باز شده استفاده می‌کنیم. متد stamper.GetUnderContent، لایه زیرین متن صفحات را در اختیار ما قرار می‌دهد. اگر علاقمند به نوشتن بر روی لایه رویی متون هستید از متد stamper.GetOverContent استفاده کنید. در اینجا از PdfGState برای مشخص سازی میزان شفافیت متن درحال نمایش، با مقدار دهی FillOpacity آن استفاده شده است. نهایتا از متد ColumnText.ShowTextAligned برای نمایش متن مورد نظر در مکان و زاویه‌ای مشخص استفاده می‌کنیم. این متد با زبان فارسی سازگاری دارد و run direction آن قابل تنظیم است.

ج) آشنایی با ساختار سطح پایین Watermark اضافه شده به صفحات

تقریبا اکثر Watermarkهایی که بر روی صفحات PDF درج می‌شوند، نیمه شفاف هستند تا بتوان متن زیر آن‌ها را مطالعه کرد. این شفافیت همانطور که ذکر شد توسط مقدار دهی شیء PdfGState حاصل می‌شود. اگر فایل PDF تولیدی در قسمت ب را توسط

برنامه iText Rups باز کنیم، به شکل زیر خواهیم رسید:



هدف ما این است که به شیء PdfGState موجود در هر صفحه، دسترسی یافته و مقدار FillOpacity آن‌را صفر کنیم. به این ترتیب این Watermark، کاملا شفاف یا نامرئی خواهد شد. در PDF نهایی، چیزی به نام شیء PdfGState وجود ندارد، بلکه با یک سری Dictionary و Array سر و کار داریم.

همانطور که در شکل فوق ملاحظه می‌کنید، برای رسیدن به Gstateها باید مراحل زیر طی شوند:

1- فایل PDF گشوده شده و سپس به هر صفحه دسترسی یافت.

2- نیاز است RESOURCES صفحه جاری استخراج شوند.

3- در این منابع، باید EXTGSTATE را که همان PdfGStateها هستند، بیابیم.

4- سپس مقدار ca این EXTGSTATE یافت شده را به صفر مقدار دهی کنیم.

```
private static void removeWatermark(string watermarkedFile, string unwatermarkedFile)
        {
            PdfReader.unethicalreading = true;
            PdfReader reader = new PdfReader(watermarkedFile);
            reader.RemoveUnusedObjects();
            int pageCount = reader.NumberOfPages;
            for (int i = 1; i <= pageCount; i++)
            {
                var page = reader.GetPageN(i);
                PdfDictionary resources = page.GetAsDict(PdfName.RESOURCES);
                PdfDictionary extGStates = resources.GetAsDict(PdfName.EXTGSTATE);
                if (extGStates == null)
                    continue;

                foreach (PdfName name in extGStates.Keys)
```

```
                {
                    var obj = extGStates.Get(name);
                    PdfDictionary extGStateObject = (PdfDictionary)PdfReader.GetPdfObject(obj);
                    var stateNumber = extGStateObject.Get(PdfName.ca_);
                    if (stateNumber == null)
                        continue;

                    var caNumber = (PdfNumber)PdfReader.GetPdfObject(stateNumber);
                    if (caNumber.FloatValue != 1f)
                    {
                        extGStateObject.Remove(PdfName.ca_);
                        //با تنظیم مقدار به صفر، نامرئی خواهد شد
                        extGStateObject.Put(PdfName.ca_, new PdfNumber(0f));
                    }
                }
            }

            using (FileStream fs = new FileStream(unwatermarkedFile, FileMode.Create, FileAccess.Write,
FileShare.None))
            {
                using (PdfStamper stamper = new PdfStamper(reader, fs))
                {
                    stamper.SetFullCompression();
                    stamper.Close();
                }
            }
        }
```

نحوه پیاده سازی مراحل یاد شده را در کدهای فوق ملاحظه می‌کنید. ابتدا توسط PdfReader فایل موجود باز شده و سپس تک تک صفحات آن‌را بررسی می‌کنیم. در این صفحات اگر EXTGSTATE ایی یافت شد، مقدار ca آن‌را به صفر تنظیم خواهیم کرد. از مقدار 1 صرفنظر شده، چون این مقدار عموما برای Watermark دار کردن صفحات استفاده نمی‌شود.
در این متد، watermarkedFile فایلی است که باید watermark آن نامرئی شود و unwatermarkedFile فایل تولیدی حاصل است.

Question 1 - What is CLR ?

CLR is Common Language Runtime is the runtime that converts a MSIL code into the host machine language code. It is the execution engine for .NET Framework applications. It provides a number of services, including:

- Code management (loading and execution)
- Memory Management
- Thread Management
- Conversion of IL to native code.
- Access to metadata (enhanced type information)
- Managing memory for managed objects (Garbage collection)
- Enforcement of code access security (Security Management)
- Exception handling, including cross-language exceptions
- Interoperation btw managed code, COM objects, and pre-existing DLL's (unmanaged code and data)
- Support for developer services (profiling, debugging, and so on).
- Type safety.

Question 2 - What is CLR HOST?

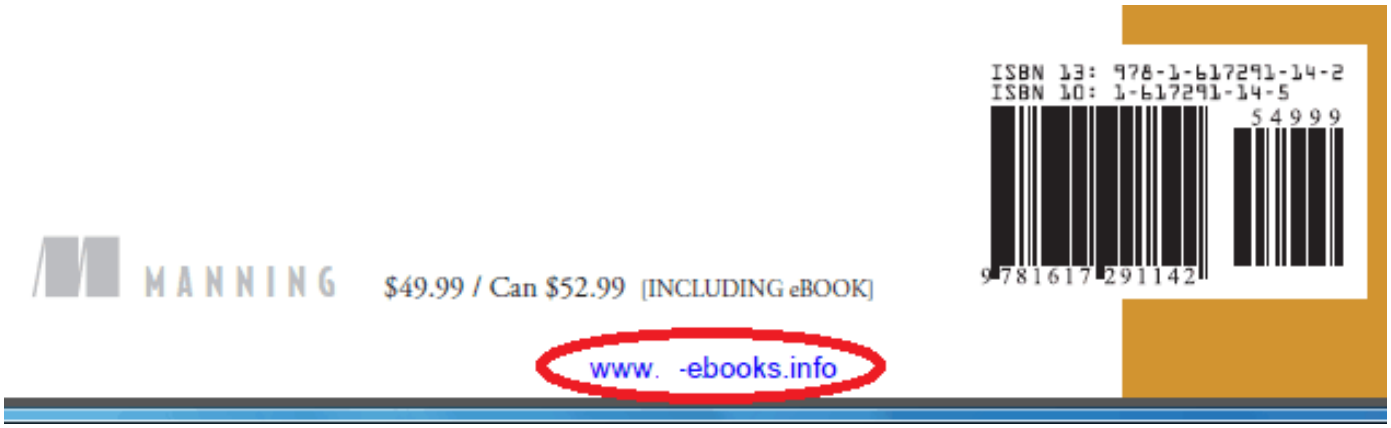
A CLR host is an application that is responsible for loading the CLR into a process, creating application domains within the process, and executing user code within the application domains.

Examples of hosts that ship with the .NET Framework include:

- ASP.Net - An ISAPI filter that ships with ASP.NET loads the CLR and does the initialization necessary to handle web requests.
- Internet Explorer: - A MIME filter hooks into IE versions 5.01 and higher to execute managed controls referenced from HTML pages.

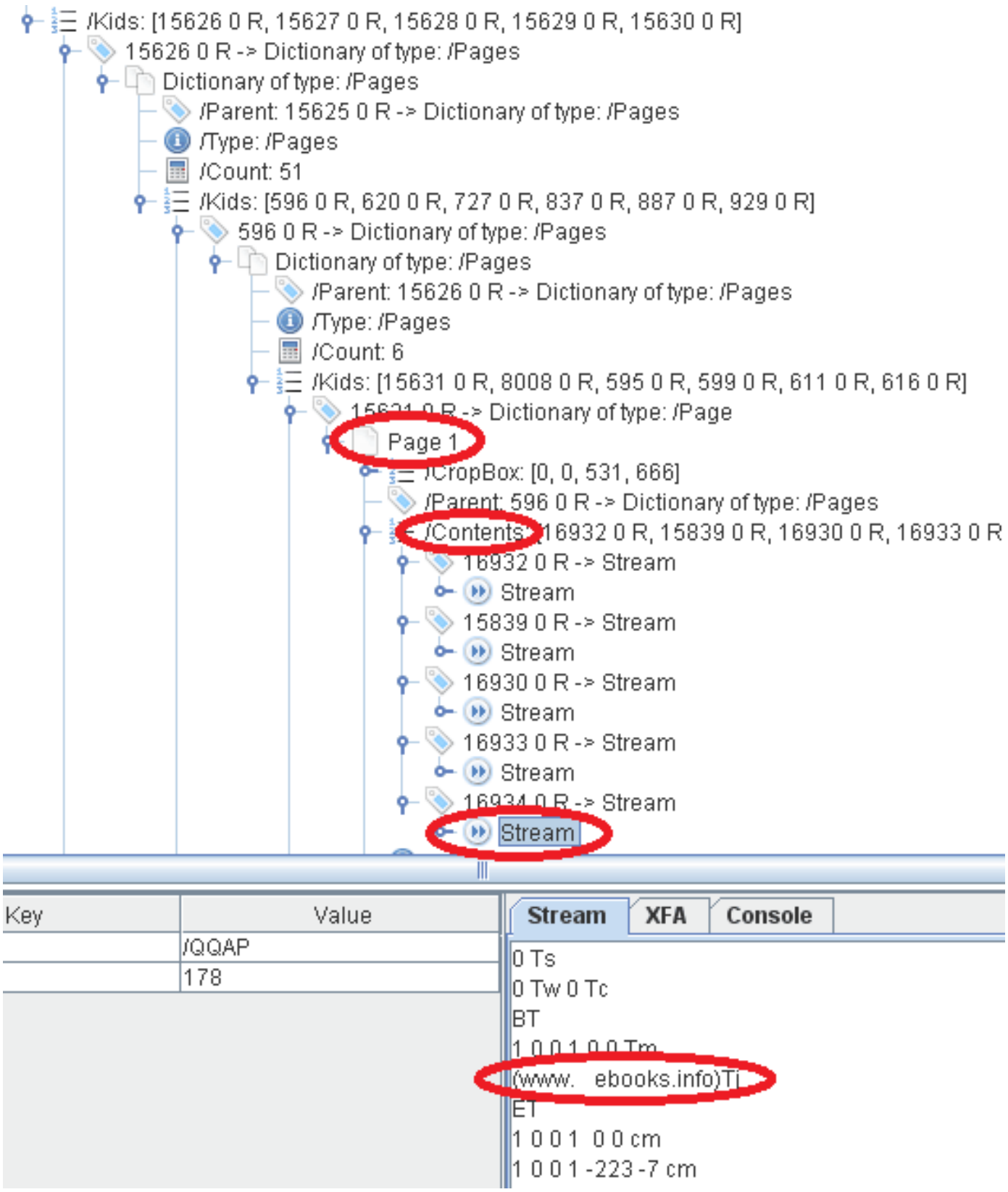
عنوان: **حذف لایه‌های جدید اضافه شده به فایل‌های PDF توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۳۱ ۸:۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp, Watermark

شاید یک سری از Ebookهای PDF ایی را دیده باشید که سایت‌های ثالث، آن‌ها را پس از افزودن لایه‌ای متنی، مثلا در ذیل تمام صفحات به همراه آدرس وب سایت خودشان، باز انتشار می‌دهند. در مطلب جاری قصد داریم، نحوه حذف این لایه‌های اضافی را توسط iTextSharp بررسی کنیم.



یافتن و حذف لایه‌های اضافه شده به صفحات یک فایل PDF

برای آشنایی با ساختار سطح پایین لایه‌های اضافه شده نیاز است به برنامه iText Rups مراجعه کنیم.



همانطور که مشاهده می‌کنید، برای رسیدن به لایه‌ای که حاوی متن اضافه شده به ذیل تمام صفحات است، نیاز است ابتدا صفحات را گشوده و سپس CONTENTS آن‌ها را استخراج کنیم. در این CONTENTS کلیه streamهای موجود را بررسی و هر کدام که حاوی متن مورد نظر ما بودند، یافته و سپس آن استریم را با مقدار دهی طول آن به صفر، حذف کنیم. روش کار را در متد ذیل مشاهده می‌کنید:

```
private static void removeWatermarkLayer(string watermarkedFile, string text,  string
unwatermarkedFile)
        {
            PdfReader.unethicalreading = true;
            PdfReader reader = new PdfReader(watermarkedFile);
            reader.RemoveUnusedObjects();
            int pageCount = reader.NumberOfPages;
            for (int i = 1; i <= pageCount; i++)
            {
                var page = reader.GetPageN(i);
                var contentarray = page.GetAsArray(PdfName.CONTENTS);
                if (contentarray == null)
                    continue;

                for (int j = 0; j < contentarray.Size; j++)
                {
                    var stream = (PRStream)contentarray.GetAsStream(j);
                    //دریافت محتوای خام صفحه
                    var content =
System.Text.Encoding.ASCII.GetString(PdfReader.GetStreamBytes(stream));
                    if (content.Contains(text))
                    {
                        //حذف کامل محتوا از فایل
                        stream.Put(PdfName.LENGTH, new PdfNumber(0));
                        stream.SetData(new byte[0]);
                    }
                }
            }

            using (var fileStream = new FileStream(unwatermarkedFile, FileMode.Create,
FileAccess.Write, FileShare.None))
            {
                using (var stamper = new PdfStamper(reader, fileStream))
                {
                    stamper.SetFullCompression();
                    stamper.Close();
                }
            }
        }
```

در این متد watermarkedFile همان فایل اصلی دارای لایه‌های اضافی است. Text متنی است که در استریم‌های صفحات به دنبال آن خواهیم گشت و unwatermarkedFile نام و مسیر فایل تصحیح شده نهایی است که قرار است تولید شود.