عنوان: **تبدیل تعدادی تصویر به یک فایل PDF**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۴/۰۸ ۲۳:۵۳:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

صورت مساله:
تعدادی تصویر داریم، می‌خواهیم این‌ها را تبدیل به فایل PDF کنیم به این شرط که هر تصویر در یک صفحه مجزا قرار داده شود.
در ادامه برای این منظور از کتابخانه‌ی iTextSharp استفاده خواهیم کرد.

iTextSharp چیست؟
iTextSharp کتابخانه‌ی سورس باز و معروفی جهت تولید فایل‌های PDF ، توسط برنامه‌های مبتنی بر دات نت است. آن را از آدرس زیر می‌توان دریافت کرد:

/http://sourceforge.net/projects/itextsharp

کتابخانه iTextSharp نیز جزو کتابخانه‌هایی است که از جاوا به دات نت تبدیل شده‌اند. نام کتابخانه اصلی iText است و اگر کمی جستجو کنید می‌توانید کتاب iText in Action 617 صفحه‌ای از انتشارات MANNING را در این مورد نیز بیابید. هر چند این کتاب برای برنامه نویس‌های جاوا نوشته شده اما نام کلاس‌ها و متدها در iTextSharp تفاوتی با iText اصلی ندارند و مطالب آن برای برنامه نویس‌های دات نت هم قابل استفاده است.

مجوز استفاده از iTextSharp کدام است؟
مجوز این کتابخانه GNU Affero General Public License است. به این معنا که شما موظفید، تغییری در قسمت تهیه کننده خواص فایل PDF تولیدی که به صورت خودکار به نام کتابخانه تنظیم می‌شود، ندهید. اگر می‌خواهید این قسمت را تغییر دهید باید هزینه کنید. همچنین با توجه به اینکه این مجوز، GPL است یعنی زمانیکه از آن استفاده کردید باید کار خود را به صورت سورس باز ارائه دهید؛ درست خوندید! بله! مثل مجوز استفاده از نگارش عمومی و رایگان MySQL و اگر نمی‌خواهید اینکار را انجام دهید، در اینجا تاکید شده که باید کتابخانه را خریداری کنید.

نحوه استفاده از کتابخانه iTextSharp
در ابتدا کد تبدیل تصاویر به فایل PDF را در ذیل مشاهده خواهید کرد. فرض بر این است که ارجاعی را به اسمبلی itextsharp.dll اضافه کرده‌اید:

```
using System.Collections.Generic;
using System.Drawing.Imaging;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpTests
{
   public class ImageToPdf
   {
       public iTextSharp.text.Rectangle PdfPageSize { set; get; }
       public ImageFormat ImageCompressionFormat { set; get; }
       public bool FitImagesToPage { set; get; }

       public void ExportToPdf(IList<string> imageFilesPath, string outPdfPath)
       {
           using (var pdfDoc = new Document(PdfPageSize))
           {
               PdfWriter.GetInstance(pdfDoc, new FileStream(outPdfPath, FileMode.Create));
               pdfDoc.Open();

               foreach (var file in imageFilesPath)
               {
                   var pngImg = iTextSharp.text.Image.GetInstance(file);
```

```
                    if (FitImagesToPage)
                    {
                        pngImg.ScaleAbsolute(pdfDoc.PageSize.Width, pdfDoc.PageSize.Height);
                    }
                    pngImg.SetAbsolutePosition(0, 0);

                    //add to page
                    pdfDoc.Add(pngImg);
                    //start a new page
                    pdfDoc.NewPage();
                }
            }
        }
    }
}
```

توضیحات:

استفاده از کتابخانه‌ی iTextSharp همیشه شامل 5 مرحله است. ابتدا شیء Document ایجاد می‌شود. سپس وهله‌ای از PdfWriter ساخته شده و Document جهت نوشتن در آن گشوده خواهد شد. در طی یک سری مرحله محتویات مورد نظر به Document اضافه شده و نهایتا این شیء بسته خواهد شد. البته در اینجا چون کلاس Document اینترفیس IDisposable را پیاده سازی کرده، بهترین روش استفاده از آن بکارگیری واژه کلیدی using جهت مدیریت منابع آن است. به این ترتیب کامپایلر به صورت خودکار قطعه try/finally مرتبط را جهت پاکسازی منابع، تشکیل خواهد داد.

اندازه صفحات توسط سازنده‌ی شیء Document مشخص خواهند شد. این شیء از نوع iTextSharp.text.Rectangle است؛ اما مقدار دهی آن توسط کلاس iTextSharp.text.PageSize صورت می‌گیرد که انواع و اقسام اندازه صفحات استاندارد در آن تعریف شده‌اند.

متد iTextSharp.text.Image.GetInstance که در این مثال جهت دریافت اطلاعات تصاویر مورد استفاده قرار گرفت، 15 overload دارد که از آدرس مستقیم یک فایل تا استریم مربوطه تا Uri یک آدرس وب را نیز می‌پذیرد و از این لحاظ بسیار غنی است.

مثالی در مورد نحوه استفاده از کلاس فوق:

```
using System.Collections.Generic;
using System.Drawing.Imaging;

namespace iTextSharpTests
{
    class Program
    {
        static void Main(string[] args)
        {
            new ImageToPdf
            {
                FitImagesToPage = true,
                ImageCompressionFormat = ImageFormat.Jpeg,
                PdfPageSize = iTextSharp.text.PageSize.A4
            }.ExportToPdf(
                    imageFilesPath: new List<string>
                            {
                                @"D:\3.jpg",
                                @"D:\4.jpg"
                            },
                    outPdfPath: @"D:\tst.pdf"
                    );
        }
    }
}
```

نظرات خوانندگان

نویسنده: Nima
تاریخ: ۱۳۹۰/۰۶/۱۲ ۲۲:۲۱:۲۵

سلام آقای نصیری
ممنون از آموزش مفیدتون. من این سمپل رو برای خودم نوشتم و فقط FitImagesToPage = false قرار دادم اما عکسها به گوشه پایین سمت چپ میچسبه. میخواستم بدونم چکار باید بکنم که عکس در وسط صفحه نمایش داده بشه؟
باتشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۱۲ ۲۲:۳۳:۵۹

باید همان متد SetAbsolutePosition را مقدار دهی کنید. مثلا به این صورت:

```
pngImg.SetAbsolutePosition((pdfDoc.PageSize.Width - pngImg.Width) / 2, (pdfDoc.PageSize.Height - pngImg.Height) / (2
```

نویسنده: Nima
تاریخ: ۱۳۹۰/۰۶/۱۲ ۲۲:۵۸:۵۷

بسیار عالی بود با تشکر

نویسنده: ali
تاریخ: ۱۳۹۱/۰۶/۲۵ ۱۶:۴۸

با سلام من از این روش استفاده کردم فقط مشکلی که وجود داشت عکسها را از عرض میکشه علت از چی میتونه باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۵ ۱۷:۱۱

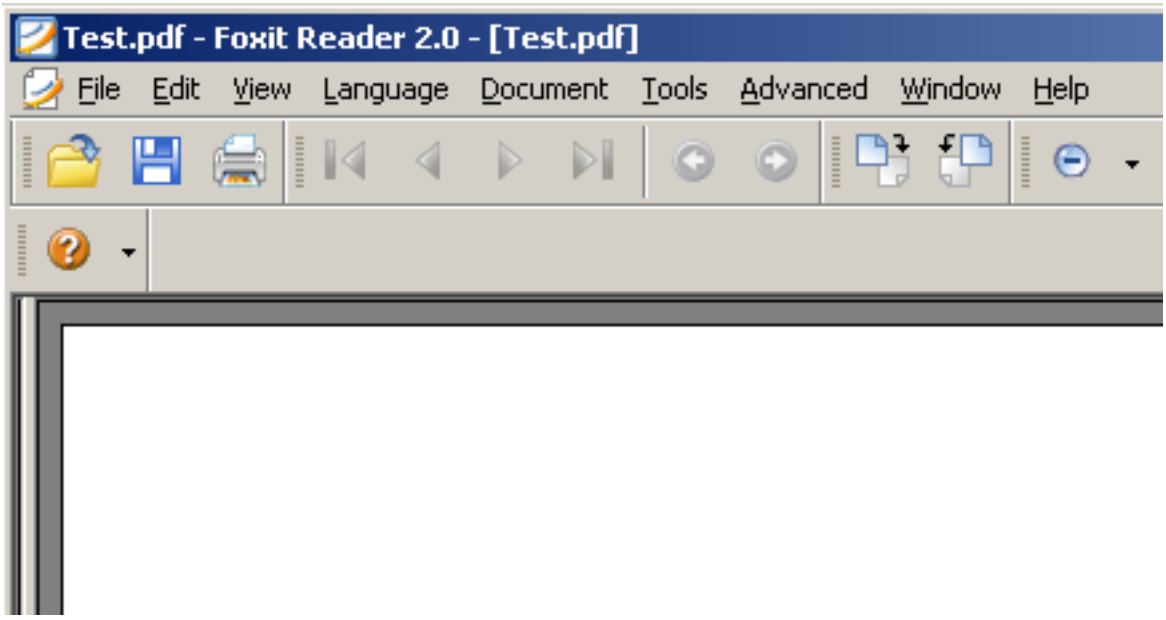
مربوط به گزینه FitImagesToPage است.

عنوان: **فارسی نویسی و iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۴/۱۰ ۱۰:۴۴:۰۰
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp

شرح یک سری سعی و خطا!
سعی اول:

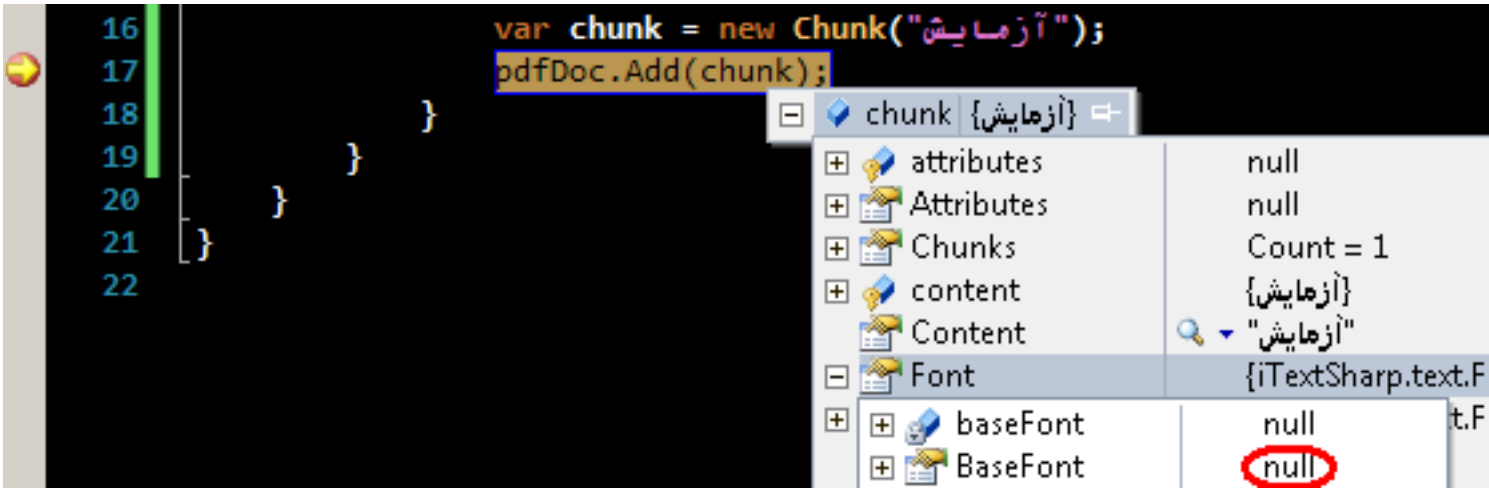
```
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpTests
{
   class Program
   {
       static void Main(string[] args)
       {
           using (var pdfDoc = new Document(PageSize.A4))
           {
               PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf", FileMode.Create));
               pdfDoc.Open();

               var chunk = new Chunk("آزمایش");
               pdfDoc.Add(chunk);
           }
       }
   }
}
```

نتیجه:



بله! هیچی!

مشکل از کجاست؟
در iTextSharp بر اساس نوع فونت انتخابی و encoding مرتبط، نحوه‌ی رندر سازی حروف مشخص می‌شود:



همانطور که ملاحظه می‌کنید، فونت پایه متنی که قرار است اضافه شود، null است.

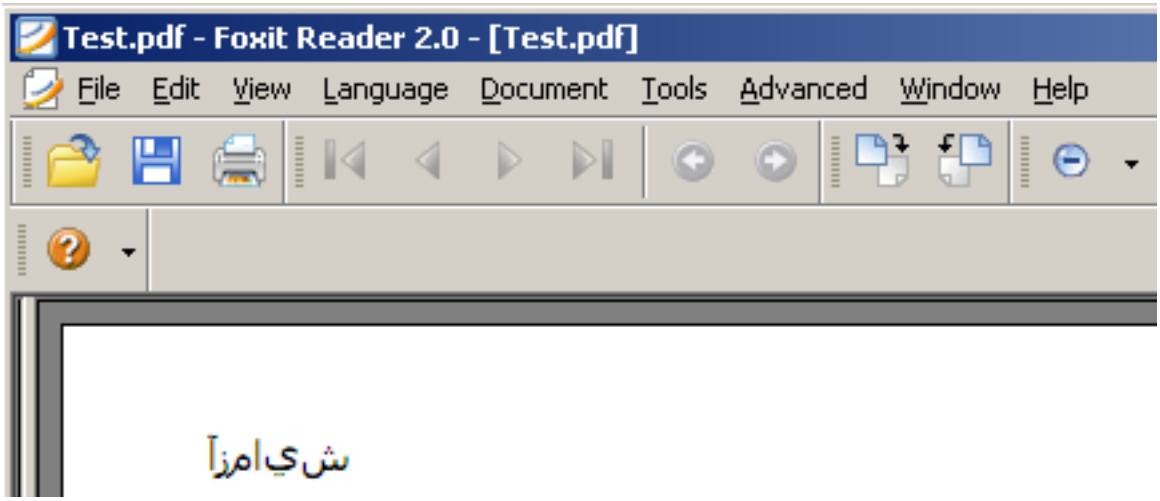
سعی دوم:
اینبار فونت را تنظیم می‌کنیم:

```
using System;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpTests
{
   class Program
   {
       static void Main(string[] args)
       {
           using (var pdfDoc = new Document(PageSize.A4))
           {
               PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf", FileMode.Create));
               pdfDoc.Open();

               var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
               var baseFont = BaseFont.CreateFont(fontPath, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
               var tahomaFont = new Font(baseFont, 10, Font.NORMAL, BaseColor.BLACK);

               var chunk = new Chunk("آزمایش",tahomaFont);
               pdfDoc.Add(chunk);
           }
       }
   }
}
```

توضیحات:
متد BaseFont.CreateFont می‌تواند مسیری از فونت مورد نظر را دریافت کند. این حالت خصوصا برای برنامه‌های وب که ممکن است فونت مورد نظر آن‌ها در سرور نصب نشده باشد، بسیار مفید است و لزومی ندارد که الزاما فونت مورد استفاده در پوشه fonts ویندوز نصب شده باشد.

نکات مهم دیگر بکار گرفته شده در این متد، استفاده از BaseFont.IDENTITY_H و BaseFont.EMBEDDED است. به این صورت encoding متن، جهت نوشتن متون غیر Ansi تنظیم می‌شود و در این حالت حتما باید فونت را در فایل، مدفون (embed) نمود. از این لحاظ که عموما این نوع فونت‌ها در سیستم‌های کاربران نصب نیستند.

نتیجه:



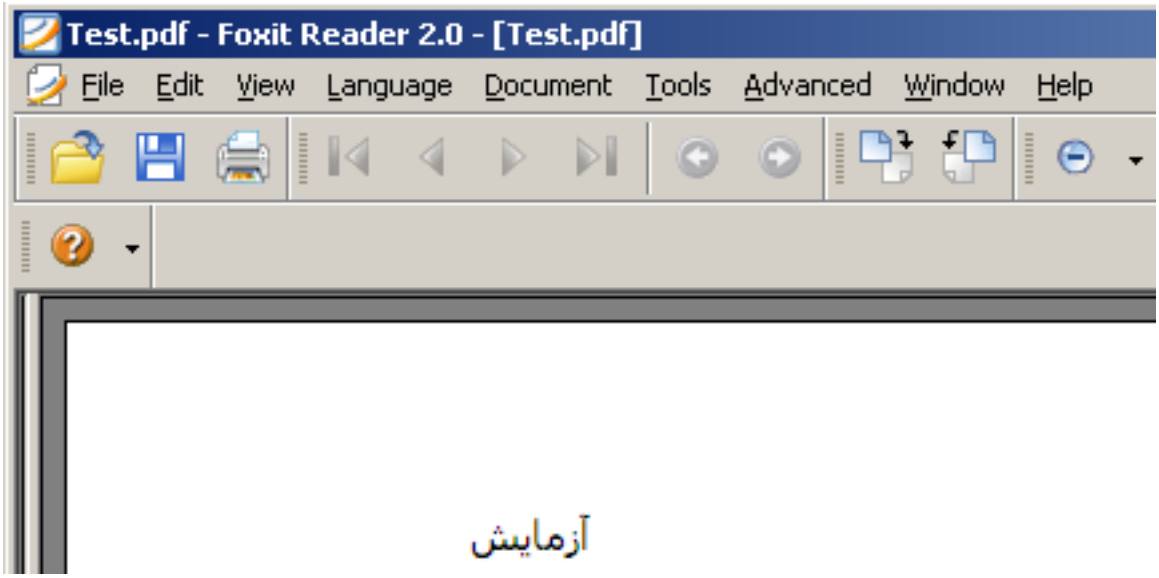
بد نیست! حداقل حروف نمایش داده شدند؛ اما نیاز است تا چرخانده یا معکوس شوند. برای انجام خودکار آن حداقل دو کار را می‌توان انجام داد.

الف) استفاده از ColumnText و اعمال تنظیمات راست به چپ آن

```
using System;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpTests
{
   class Program
   {
       static void Main(string[] args)
       {
           using (var pdfDoc = new Document(PageSize.A4))
           {
               var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
               pdfDoc.Open();

               var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
               var baseFont = BaseFont.CreateFont(fontPath, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
               var tahomaFont = new Font(baseFont, 10, Font.NORMAL, BaseColor.BLACK);

               ColumnText ct = new ColumnText(pdfWriter.DirectContent);
               ct.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
               ct.SetSimpleColumn(100, 100, 500, 800, 24, Element.ALIGN_RIGHT);

               var chunk = new Chunk("آزمایش", tahomaFont);

               ct.AddElement(chunk);
               ct.Go();
           }
       }
   }
}
```

توضیحات:
در اینجا یک ColumnTex جدید تعریف و سپس خصوصیات این ستون تنظیم شده، به همراه RunDirection آن که اصل قضیه است. سپس chunk تعریف شده را به این ستون اضافه کرده‌ایم.

نتیجه:



بله! کار کرد!

ب) استفاده از PdfTable و اعمال تنظیمات راست به چپ آن

```
using System;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpTests
{
   class Program
   {
       static void Main(string[] args)
       {
           using (var pdfDoc = new Document(PageSize.A4))
           {
               var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
               pdfDoc.Open();

               var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
               var baseFont = BaseFont.CreateFont(fontPath, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
               var tahomaFont = new Font(baseFont, 10, Font.NORMAL, BaseColor.BLACK);

               PdfPTable table = new PdfPTable(numColumns: 1);
               table.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
               table.ExtendLastRow = true;

               PdfPCell pdfCell = new PdfPCell(new Phrase("آزمایش", tahomaFont));
               pdfCell.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
```

```
                table.AddCell(pdfCell);
                pdfDoc.Add(table);
            }
        }
    }
}
```

در حین استفاده از PdfTable هم لازم است تا RunDirection مربوط به خود جدول و همچنین هر سلول اضافه شده به آن به RTL تنظیم شوند.

این نکات در هر جایی که با این کتابخانه سر و کار داریم باید اعمال شوند. برای مثال:

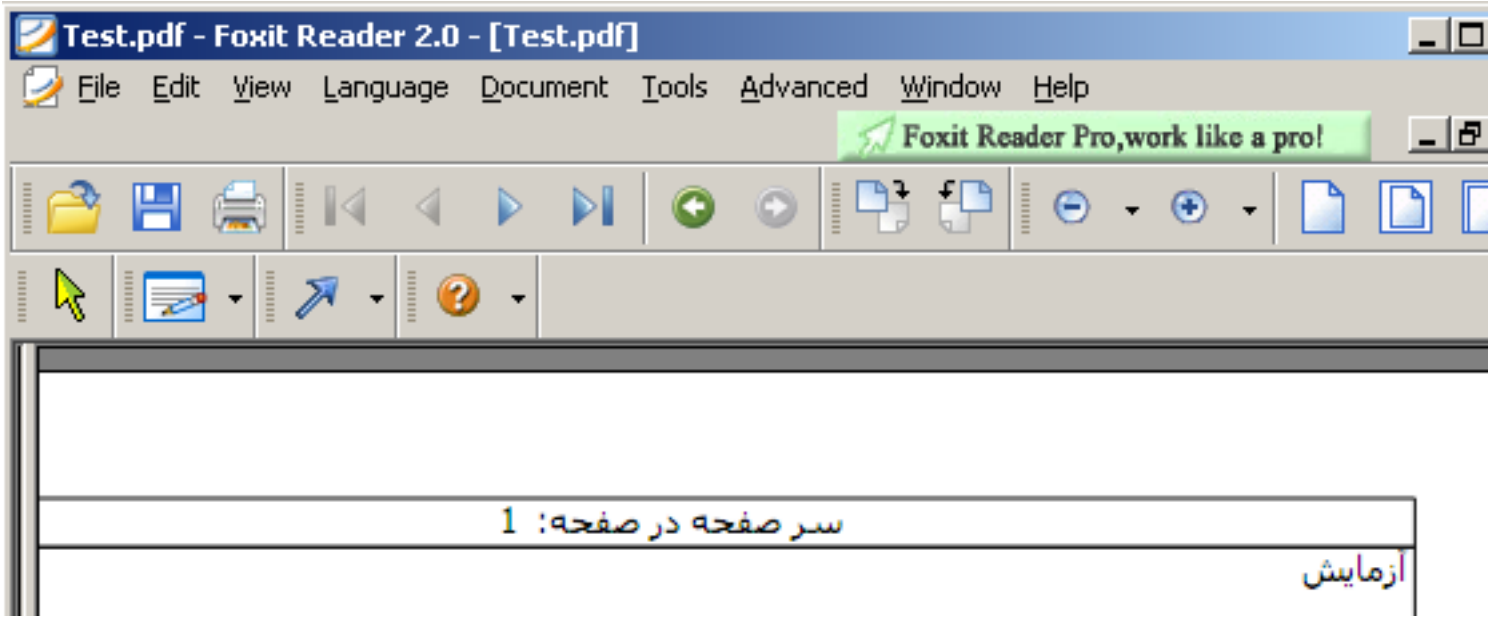
افزودن Header به صفحات Pdf :

افزودن header در نگارش‌های جدید iTextSharp شامل نکته استفاده از کلاس PdfPageEventHelper به شرح زیر است (و مثال‌هایی را که در وب پیدا خواهید کرد، هیچکدام با آخرین نگارش موجود iTextSharp کار نمی‌کنند):

```
using System;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpTests
{
   public class PageEvents : PdfPageEventHelper
   {
       Font _font;
       public PageEvents()
       {
           var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
           var baseFont = BaseFont.CreateFont(fontPath, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
           _font = new Font(baseFont, 10, Font.NORMAL, BaseColor.BLACK);
       }

       public override void OnStartPage(PdfWriter writer, Document document)
       {
           base.OnStartPage(writer, document);

           PdfPTable table = new PdfPTable(numColumns: 1);
           table.RunDirection = PdfWriter.RUN_DIRECTION_RTL;

           PdfPCell pdfCell = new PdfPCell(new Phrase("سر صفحه در صفحه:  " + writer.PageNumber,
_font));
           pdfCell.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
           pdfCell.HorizontalAlignment = Element.ALIGN_CENTER;

           table.AddCell(pdfCell);
           document.Add(table);
       }
   }

   class Program
   {
       static void Main(string[] args)
       {
           using (var pdfDoc = new Document(PageSize.A4))
           {
               var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
               pdfWriter.PageEvent = new PageEvents();
               pdfDoc.Open();

               var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
               var baseFont = BaseFont.CreateFont(fontPath, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
               var tahomaFont = new Font(baseFont, 10, Font.NORMAL, BaseColor.BLACK);

               PdfPTable table = new PdfPTable(numColumns: 1);
               table.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
               table.ExtendLastRow = true;

               PdfPCell pdfCell = new PdfPCell(new Phrase("آزمایش", tahomaFont));
               pdfCell.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
```

```
                table.AddCell(pdfCell);
                pdfDoc.Add(table);

                pdfDoc.NewPage();
            }
        }
    }
}
```

نتیجه:



تنها نکته‌ای که اینجا اضافه شده، تعریف کلاس PageEvents است که از کلاس PdfPageEventHelper مشتق شده است. در این کلاس می‌توان یک سری متد کلاس پایه را تحریف کرد و header و footer و غیره را اضافه نمود. سپس جهت اضافه کردن آن، pdfWriter.PageEvent باید مقدار دهی شود.

در اینجا هم اگر نوع فونت، encoding و PdfTable به همراه RunDirection آن اضافه نمی‌شد، یا چیزی در header صفحه قابل مشاهده نبود یا متن مورد نظر معکوس نمایش داده می‌شد.

نظرات خوانندگان

نویسنده: Alex Kh58
تاریخ: ۱۳۹۰/۰۴/۱۱ ۱۳:۵۶:۱۴

ممنون از مطلب خوبتون مثل همیشه عالی.

نویسنده: Hosein Mohtasham
تاریخ: ۱۳۹۰/۰۵/۰۹ ۱۵:۱۸:۲۲

خدا خیرت بده مشکل جماعتی رو حل کردی

نویسنده: Hossein Hariri
تاریخ: ۱۳۹۰/۰۶/۲۱ ۱۳:۴۱:۴۸

ممنونم، فقط این کار در زمانی که از html استفاده میکنم جواب نمیده.
یعنی با html هم کار میکنه؟ مشکل کار من کجاست؟ باید کار اضافه ای کنم که نمیکنم؟؟؟
لطفا راهنمایی بفرمایید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۲۱ ۱۶:۱۱:۳۲

در مورد html to pdf فارسی، یک سری نکات خاص خودش وجود دارد که مفصل توضیح دادم: (+)

نویسنده: سید مجتبی
تاریخ: ۱۳۹۱/۰۴/۱۵ ۳:۱۰

ممنون آقا وحید، عالی

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۶/۱۲ ۱:۹

با سلام ، من کد زیر رو نوشتم

```
var m = new ITModel.ITModelContainer();
        var list = (from pp in m.PERSONNELs
                    select pp).ToList();

        string pdfpath = Server.MapPath("PDFs");
        using (var pdfDoc = new Document(PageSize.A4))
        {
            var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream(pdfpath + "/Personnel2.pdf",
FileMode.Create));
            pdfDoc.Open();
            var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
            var baseFont = BaseFont.CreateFont(fontPath, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
            var tahomaFont = new Font(baseFont, 10, Font.NORMAL, BaseColor.BLACK);
            float[] widths = new float[] { 1f, 2f };

            PdfPTable table = new PdfPTable(2)
                {
                    TotalWidth = 216f,
                    LockedWidth = true,
                    HorizontalAlignment = 0,
                    SpacingBefore = 20f,
                    SpacingAfter = 30f
                };

            table.SetWidths(widths);
            PdfPCell cell = new PdfPCell(new Phrase("لیست پرسنل", tahomaFont)) { RunDirection =
PdfWriter.RUN_DIRECTION_RTL, Colspan = 2, Border = 0, HorizontalAlignment = 1 };
```

```
            table.AddCell(cell);
            foreach (var item in list)
            {
                PdfPCell cell2 = new PdfPCell(new Phrase(item.PERSON_ID.ToString(), tahomaFont)) {
RunDirection = PdfWriter.RUN_DIRECTION_RTL };
                PdfPCell cell3 = new PdfPCell(new Phrase(item.FIRST_NAME + " " + item.LAST_NAME,
tahomaFont)) { RunDirection = PdfWriter.RUN_DIRECTION_RTL };
                table.AddCell(cell2);
                table.AddCell(cell3);
            }
            pdfDoc.Add(table);
        }
```

آیا امکانش هست که ما به جای اینکه بیایم در هر Cell کد زیر را بنویسیم یک بار برای کد جدول اینو تعریف کنیم؟

```
{ RunDirection = PdfWriter.RUN_DIRECTION_RTL };
```

چون اگه بخواهیم کدهای مربوط به تنظیم فوت رو برای هر Cell بنویسیم یه خورده کدها زیاد میشه.
ممنون میشم راهنمایی کنید
مرسی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۲ ۸:۵۰

نه. لازم است به ازای هر سلول اینکار انجام شود.
ضمنا یک نکته کلی در مورد PDF وجود دارد و آن هم این است که ساختار PDF یک canvas است (یک تابلو نقاشی برداری). یعنی مفاهیمی مانند جدول، سلول، پاراگراف و غیره در پشت صحنه آن وجود خارجی ندارند و فقط کتابخانه‌های تولید PDF است که این نوع امکانات را جهت سهولت کار اختراع کرده‌اند. بنابراین به ازای هر شیءایی که اضافه می‌شود باید اطلاعات دقیق آن نیز درج شود.

نویسنده: بهاره قدمی
تاریخ: ۱۳۹۲/۰۴/۰۱ ۱۵:۵

با عرض خسته نباشید
اگر در داده‌ها ترکیبی از حروف فارسی و انگلیسی باشه با این روش کار نمیکنه.مثلا با فونت B Lotus حروف انگلیسی نمایش داده نمیشه.در بهترین حالت من فونت Tahoma رو استفاده میکنم که بازم اعدادم انگلیسی هست.
آیا اشکال از فونته؟ نمیتونم دوتا فونت براش بفرستم ؟

یا اینکه باید فونتی بسازم که همه جور حروفی رو داشته باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۰۱ ۱۶:۲۹

« iTextSharp و استفاده از قلم‌های محدود فارسی »

نویسنده: samin
تاریخ: ۱۳۹۲/۰۹/۱۱ ۱۰:۲۸

باسلام
ممنون از راهنمایی تون عالی بود
اما برای راست چین کردن متون باید چیکار کرد؟
من این دو خط رو اضافه کردم

```
cell.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
cell.RunDirection = PdfWriter.DirectionR2L;
```

اما باز هم کار نمیکنه !

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۱ ۱۰:۴۰

- به تصویر آخر و کدهای آن دقت کنید. کلمه آزمایش از سمت راست شروع شده.
- DirectionR2L کاربردی ندارد در اینجا. PdfWriter.RUN_DIRECTION_RTL باید باشد.

نویسنده: رسول
تاریخ: ۱۳۹۲/۱۱/۱۵ ۱۷:۱۳

سلام؛ آیا امکانش هست که برای کلمه آزمایش بطور مثال مختصات تعریف کرد تا در اون مختصات توی صفحه نمایش داده بشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۵ ۲۰:۱۷

بله. باید از متد ColumnText.ShowTextAligned استفاده کنید:

```
ColumnText.ShowTextAligned(
                        canvas: pdfWriter.DirectContent,
                        alignment: Element.ALIGN_RIGHT,
                        phrase: new Phrase("لیست پرسنل", tahomaFont),
                        x: 40,
                        y: 30,
                        rotation: 0,
                        runDirection: PdfWriter.RUN_DIRECTION_RTL,
                        arabicOptions: 0);
```

عنوان: **مشکلات طراحی API مرتبط با iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۴/۱۳ ۱۷:۴۳:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

کتابخانه‌ی iTextSharp، یا همان برگردان iText جاوا، انصافا اینقدر حرف برای گفتن دارد که یک کتاب 600 صفحه‌ای برای آن چاپ شده است، اما ... در حین استفاده از آن مشکل زیر (که به شکل وسیعی در قسمت‌های مختلف آن وجود دارد) قابل هضم نیست:
یکی از مواردی را که در حین طراحی یک API خوب باید در نظر گرفت، کمک به استفاده کننده در عدم بکارگیری Magic numbers است. حالا این Magic numbers یعنی چی؟
برای مثال قطعه کد زیر را در نظر بگیرید:

```
new Font(baseFont, 10, 0, BaseColor.BLACK)
```

مقدار پارامتر 3 در اینجا بی‌معنا است، مگر اینکه به مستندات مربوطه مراجعه کنیم.
نگارش بهبود یافته کد فوق به شرح زیر است:

```
new Font(baseFont, 10, Font.NORMAL, BaseColor.BLACK)
```

کمی بهتر شد، اینبار ثوابت به یک کلاس منتقل شده‌اند و به این ترتیب معنای مقدار بکارگرفته شده بدون نیاز به مستندات متد فوق قابل درک است. اما این روش هم (یعنی همان روش متداول iTextSharp) دو مشکل مهم دارد:
استفاده کننده محدودیتی در بکارگیری مقادیر ندارد، چون آرگومان‌ها از نوع int معرفی شده‌اند. ممکن است اشتباهی رخ دهد.
باز هم نیاز است به مستندات کتابخانه مراجعه کرد، زیرا نوع int هیچ نوع منوی intellisense خاصی را ظاهر نمی‌کند.
راه درست، استفاده از enum است بجای یک کلاس ساده که فقط یک سری از ثوابت در آن تعریف شده‌اند. نوع int را باید با enum زیر جایگزین کرد (یا ... بهتر است اینگونه بشود در کتابخانه‌ی اصلی ... روزی!)

```
public enum PdfFontStyle
{
	Normal = 0,
	Bold = 1,
	Italic = 2,
	Underline = 4,
	Strikethru = 8,
	BoldItalic = Bold | Italic
}
```

اگر در طراحی آن از ابتدا این روش پی‌گرفته می‌شد، منوی intellisense تبدیل به بهترین مستند این کتابخانه می‌شد.

عنوان: **نمایش تعداد کل صفحات در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۴/۱۷ ۱۳:۵۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

در مورد نحوه‌ی نمایش شماره صفحه جاری در مثلا header یک گزارش PDF تهیه شده به کمک writer.PageNumber و ارث بری از کلاس PdfPageEventHelper، در پایان مطلب فارسی نویسی در iTextSharp توضیح داده شد. این مورد جزو ضروریات یک گزارش خوب است، اما عموما نیاز است تا تعداد کل صفحات هم نمایش داده شود. مثلا صفحه n از 100 جایی در تمام صفحات درج شود و ... هیچ خاصیتی به نام TotalNumberOfPages را در این کتابخانه نمی‌توان یافت. علت هم این است که تعداد واقعی کل صفحات فقط در حین بسته شدن شیء Document مشخص می‌شود و نه در هنگام تهیه صفحات. بنابراین نکته تهیه و نمایش تعداد صفحات، در iTextSharp به صورت خلاصه به شرح زیر است:

الف) باید در همان کلاسی که از PdfPageEventHelper به ارث رسیده است، متد OnCloseDocument را تحریف (override) کرد.
در اینجا به خاصیت writer.PageNumber دسترسی داریم و writer.PageNumber - 1 مساوی است با تعداد کل صفحات.

ب) در مرحله بعد نیاز است تا این عدد را به نحوی به تمام صفحات تولید شده اضافه کنیم. این کار هم ساده است و مبتنی است بر بکارگیری یک PdfTemplate :

در متد تحریف شده‌ی OnOpenDocument ، یک قالب PDF ساده را تولید می‌کنیم (مثلا یک مستطیل کوچک خالی).
سپس در متد OnEndPage ، این قالب را به انتهای تمام صفحات در حال تولید اضافه خواهیم کرد.
زمانیکه متد OnCloseDocument فراخوانده شد، عدد تعداد کل صفحات را در این قالب که به تمام صفحات اضافه شده، درج خواهیم کرد. به این ترتیب این عدد به صورت خودکار در تمام صفحات نمایش داده خواهد شد.

پیاده سازی این توضیحات را در ادامه ملاحظه خواهید کرد:

```
using System;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpTests
{
   public class PdfWriterPageEvents : PdfPageEventHelper
   {
       PdfContentByte _pdfContentByte;
       // عدد نهایی تعداد کل صفحات را در این قالب قرار خواهیم داد //
       PdfTemplate _template;
       BaseFont _baseFont;

       public override void OnOpenDocument(PdfWriter writer, Document document)
       {
           _baseFont = BaseFont.CreateFont(BaseFont.HELVETICA, BaseFont.CP1252, BaseFont.NOT_EMBEDDED);
           _pdfContentByte = writer.DirectContent;
           _template = _pdfContentByte.CreateTemplate(50, 50);
       }

       public override void OnEndPage(PdfWriter writer, Document document)
       {
           base.OnEndPage(writer, document);
           String text = writer.PageNumber + "/";
           float len = _baseFont.GetWidthPoint(text, 8);
           Rectangle pageSize = document.PageSize;
           _pdfContentByte.SetRGBColorFill(100, 100, 100);
           _pdfContentByte.BeginText();
           _pdfContentByte.SetFontAndSize(_baseFont, 8);
           _pdfContentByte.SetTextMatrix(pageSize.GetLeft(40), pageSize.GetBottom(30));
           _pdfContentByte.ShowText(text);
           _pdfContentByte.EndText();
           //در پایان هر صفحه یک جای خالی را مخصوص تعداد کل صفحات رزرو خواهیم کرد
           _pdfContentByte.AddTemplate(_template, pageSize.GetLeft(40) + len, pageSize.GetBottom(30));
       }
       public override void OnCloseDocument(PdfWriter writer, Document document)
       {
           base.OnCloseDocument(writer, document);
           _template.BeginText();
```

```
            _template.SetFontAndSize(_baseFont, 8);
            _template.SetTextMatrix(0, 0);
            //درج تعداد کل صفحات در تمام قالب‌های اضافه شده
            _template.ShowText((writer.PageNumber - 1).ToString());
            _template.EndText();
        }
    }

    public class AddTotalNoPages
    {
        public static void CreateTestPdf()
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("tpn.pdf",
FileMode.Create));
                pdfWriter.PageEvent = new PdfWriterPageEvents();
                pdfDoc.Open();


                pdfDoc.Add(new Phrase("Page1"));
                pdfDoc.NewPage();
                pdfDoc.Add(new Phrase("Page2"));
                pdfDoc.NewPage();
                pdfDoc.Add(new Phrase("Page3"));
            }
        }
    }
}
```

نظرات خوانندگان

نویسنده: سید عباس مجاهد
تاریخ: ۱۳۹۱/۰۶/۲۰ ۰:۳۹

با تشکر مطلب جالبی اما یک سوال داشتم و ان اینکه چطور میتوانم در سرصفحه یا پاصفحه به صورت صفحه 1 از 20 بنویسیم من سعی کردم از جدول استفاده کنم اما جواب نداد میشود راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۰ ۱:۲۲

به این صورت قابل انجام است:

```
using System;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpTests
{
    public class PdfWriterPageEvents : PdfPageEventHelper
    {
        PdfContentByte _pdfContentByte;
        // عدد نهایی تعداد کل صفحات را در این قالب قرار خواهیم داد
        PdfTemplate _template;
        Font _font;
        public override void OnOpenDocument(PdfWriter writer, Document document)
        {
            FontFactory.Register(Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\tahoma.ttf");
            _font = FontFactory.GetFont("Tahoma", BaseFont.IDENTITY_H, embedded: true, size: 9);
            _pdfContentByte = writer.DirectContent;
            _template = _pdfContentByte.CreateTemplate(50, 50);
        }

        public override void OnEndPage(PdfWriter writer, Document document)
        {
            base.OnEndPage(writer, document);

            var pageSize = document.PageSize;
            var text = "صفحه " + writer.PageNumber + " از ";
            var textLen = _font.BaseFont.GetWidthPoint(text, _font.Size);
            var center = (pageSize.Left + pageSize.Right) / 2;

            ColumnText.ShowTextAligned(
                _pdfContentByte,
                Element.ALIGN_RIGHT,
                new Phrase(text, _font),
                center,
                pageSize.GetBottom(25),
                0,
                PdfWriter.RUN_DIRECTION_RTL,
                0);

            //در پایان هر صفحه یک جای خالی را مخصوص تعداد کل صفحات رزرو خواهیم کرد
            _pdfContentByte.AddTemplate(_template, center - textLen, pageSize.GetBottom(25));
        }
        public override void OnCloseDocument(PdfWriter writer, Document document)
        {
            base.OnCloseDocument(writer, document);
            _template.BeginText();
            _template.SetFontAndSize(_font.BaseFont, _font.Size);
            _template.SetTextMatrix(0, 0);
            //درج تعداد کل صفحات در تمام قالب‌های اضافه شده
            _template.ShowText((writer.PageNumber - 1).ToString());
            _template.EndText();
        }
    }

    public class AddTotalNoPages
    {
        public static void CreateTestPdf()
        {
```

```
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("tpn.pdf",
FileMode.Create));
                pdfWriter.PageEvent = new PdfWriterPageEvents();
                pdfDoc.Open();


                pdfDoc.Add(new Phrase("Page1"));
                pdfDoc.NewPage();
                pdfDoc.Add(new Phrase("Page2"));
                pdfDoc.NewPage();
                pdfDoc.Add(new Phrase("Page3"));
            }

            System.Diagnostics.Process.Start("tpn.pdf");
        }
    }
}
```

عنوان: **منابع مطالعاتی بیشتر در مورد iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۴/۲۲ ۰۹:۳۶:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

آشنایی با صفحه بندی در iTextSharp : [+]
تعریف هدر و فوتر: [+]
افزودن متن ساده در iTextSharp: [+]
کار با فونت‌های مختلف در iTextSharp: [+]
نحوه‌ی افزودن جدول در iTextSharp: [+]
ترسیم اشکال گرافیکی با iTextSharp: [+]
کار با تصاویر در iTextSharp: [+] و [+]
امکان تبدیل HTML به PDF در iTextSharp: [+]، [+]، [+] و [+]
نحوه‌ی تعریف لینک در iTextSharp: [+]
نحوه‌ی تعریف لیست در iTextSharp: [+]
افزودن نمودار به کمک کنترل‌های چارت مایکروسافت در iTextSharp: [+]
امکان تعریف بارکد در iTextSharp: [+]
یک سری مثال: [+]
یکی کردن چند فایل پی دی اف موجود با هم توسط iTextSharp: [+]

نظرات خوانندگان

نویسنده: حمید کریمی
تاریخ: ۱۳۹۰/۰۴/۲۲ ۱۵:۲۶:۰۲

سلام
جناب آقای نصیری من سالهاست از مقالات و آموزش های شما استفاده می کنم می خواستم یک تشکری کرده باشم و خدمتتان خسته نباشید بگم
واقعاً به گردن من حق استادی دارید من که نمی توانم جبران کنم، فقط آرزو می کنم همیشه سلامت و موفق باشید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۴/۲۲ ۱۶:۱۳:۵۳

سلام، ممنون؛ شما لطف دارید.

نویسنده: Mehdi Payervand
تاریخ: ۱۳۹۰/۰۴/۲۲ ۲۱:۵۴:۱۴

تشکر فراوان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۴/۲۸ ۲۳:۱۸:۰۳

مثال‌های کتاب iText البته به زبان سی شارپ (+)

عنوان: **چک لیست تهیه یک گزارش خوب**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۴/۲۳ ۱۲:۱۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

مشخصات یک گزارش خوب عموما به شرح زیر است:

-1 باید هر سطر گزارش شماره ردیف داشته باشد. (باید امکان ارجاع به هر سطر در صورت بروز مشکل میسر باشد)
-2 باید در هر صفحه، شماره صفحه و تعداد کل صفحات ذکر شود. (اگر چاپ شد بر این اساس بتوان ارتباط بین صفحات را یافت)
-3 در هر صفحه باید تاریخ و ساعت روز تهیه گزارش حتما ذکر شود. (بعدا جهت رفع اختلافات لازم می‌شود. مثلا می‌گویند این عدد اشتباه است. اما واقعا این عدد در زمان تهیه گزارش درست بوده، اما الان بر اساس اطلاعات جدید ... بله ... چیزی دیگری است، یا به قول آن‌ها اشتباه است)
-4 در پایان هر صفحه، یک سری از ستون‌های عددی باید جمع کل داشته باشد.
-5 در ابتدای هر صفحه باید "نقل از صفحه قبل" یا همان سطر جمع کل صفحه قبل ذکر شود.
-6 هدر گزارش باید در تمام صفحات تکرار شود. (باید مشخص باشد این صفحه گزارش که الان به دست من رسیده متعلق به کجاست، عنوانش چیست حداقل؟)
-7 سر ستون‌ها هم باید در هر صفحه تکرار شوند. (مثلا الان صفحه 20 یک گزارش پیش روی شما است. باید بدانید معنای این ستون سوم ظاهر شده در گزارش چیست)
-8 تمام اعداد موجود در گزارش باید جداکننده سه رقمی داشته باشند. (خواندن 4446327531 ساده‌تر است یا خواندن 4,446,327,531 ؟)
-9 تمام اعداد گزارش باید فارسی نمایش داده شوند. (این مورد را می‌شود با فونت‌های دستکاری شده که احتمالا شما هم یک دوجین از آن‌ها را دارید، حل کرد. فونت‌هایی که با یک فونت ادیتور مثل برنامه معروف FontCreator ویرایش شده و بجای اعداد انگلیسی آن‌ها، همان اعداد فارسی قرار گرفته‌اند)

نظرات خوانندگان

نویسنده: Rab Raby
تاریخ: ۱۳۹۰/۰۵/۱۵ ۰۰:۵۶:۲۱

فوق العاده بود . خیلی کاربردیه وحید خان

عنوان: **تکرار خودکار سرستون‌های یک جدول در صفحات مختلف، توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۰۲ ۱۹:۳۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

یکی از نیازهای تهیه یک گزارش خوب، تکرار سرستون‌ها در صفحات مختلف است. شاید در ابتدا این ایده مطرح شود که مثلا می‌خواهیم 25 ردیف را در هر صفحه نمایش دهیم. بر همین اساس می‌توان هر 25 ردیف یکبار، یک سطر footer و در ادامه در صفحه بعد یک سطر header را اضافه کرد و همینطور الی آخر. مهمترین ایراد این روش آن است که الزامی ندارد که واقعا 25 ردیف در یک صفحه جا شوند. عموما بر اساس اندازه‌ی محتوای نمایش داده شده، ممکن است یک صفحه 20 ردیف شود، صفحه‌ای دیگر 10 ردیف. این مورد تمام محاسبات را به هم خواهد ریخت. به همین جهت دو خاصیت مهم به نام‌های HeaderRows و FooterRows در شیء PdfPTable قابل تنظیم است. این دو خاصیت نیاز به اندکی توضیح دارند که در ادامه ذکر خواهد شد:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace HeadersAndFooters
{
  class Program
  {
      static void Main(string[] args)
      {
          using (var pdfDoc = new Document(PageSize.A4))
          {
              var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
              pdfDoc.Open();

              var table1 = new PdfPTable(1);
              table1.HeaderRows = 2;
              table1.FooterRows = 1;

              //header row
              table1.AddCell(new Phrase("header"));

              //footer row
              table1.AddCell(new Phrase("footer"));

              //adding some rows
              for (int i = 0; i < 70; i++)
              {
                  table1.AddCell(new Phrase("Row " + i));
              }

              pdfDoc.Add(table1);
          }

          //open the final file with adobe reader for instance.
          Process.Start("Test.pdf");
      }
  }
}
```

در این مثال، یک جدول ساده با یک ستون تعریف شده و سپس HeaderRows آن به 2 و FooterRows آن به 1 مقدار دهی شده‌اند. HeaderRows = 2 به این معنا است که 2 سطری را که بلافاصله در ادامه اضافه می‌کنید، در محاسبات نمایش خودکار header یا footer قرار می‌گیرند. FooterRows = 1 به این معنا است که از این تعداد، آخرین سطر، معنای footer را می‌دهد. بنابراین اولین table1.AddCell ، همان header خودکار نمایش داده شده در بالای تمام صفحات خواهد بود و table1.AddCell بعدی جهت نمایش footer خودکار بکار می‌رود. این دو سطر کاربرد دیگری ندارند.
مثالی دیگر جهت مشخص شدن این مفاهیم:

```
table1.HeaderRows = 3;
table1.FooterRows = 1;
```

در اینجا HeaderRows = 3 یعنی پس از تعریف وهله‌ای از شیء PdfPTable، سه سطر اولی که اضافه می‌شوند جزو حیطه‌ی header و footer خودکار قرار دارند. در این بین چون FooterRows = 1 تعریف شده، 2 سطر اول header تکرار شونده صفحات مختلف را تشکیل می‌دهند و سومین سطر همان footer خواهد بود.

از این طراحی لذت می‌برید؟!

نظرات خوانندگان

نویسنده: بهار قدمی
تاریخ: ۱۳۹۲/۰۳/۳۰ ۱۱:۳

با سلام و ممنون از مطالب بسیار خوبتون در مورد itextsharp
در این مثال اگر جدول مثلا چهار ستونه باشه و هدر ، فوتر هر کدام سه ستون.به چه صورت خواهد بود؟
ممنونم میشم اگه راهنماییم کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۳۰ ۱۱:۴۰

قسمت « *مثالی دیگر جهت مشخص شدن این مفاهیم:* » را یکبار دیگر مطالعه کنید.

عنوان: **نمایش تصویر پس زمینه در تمام صفحات توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۲۳ ۱۹:۵۰:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

فرض کنید می‌خواهیم تصویری را در پس زمینه‌ی تمام صفحات pdf تولیدی توسط iTextSharp قرار دهیم. برای این منظور شبیه به مطلب « نمایش تعداد کل صفحات در iTextSharp » می‌توان از رخدادهای صفحات استفاده کرد. در متد رویداد گردان OnOpenDocument، یک قالب را به اندازه‌ی یک صفحه‌ی متنی تهیه می‌کنیم. سپس در متد OnStartPage، این قالب را به تمام صفحات اضافه خواهیم کرد. در حقیقت فضایی را به این شکل رزرو می‌کنیم و در نهایت در متد OnCloseDocument ، تصویر مورد نظر را دریافت کرده، Alignment آن‌را طوری تنظیم خواهیم کرد که زیر متون صفحات قرار گیرد و به کمک متد AddImage ، آن‌را به قالب تعریف شده اضافه می‌کنیم. به این ترتیب، تصویر اضافه شده به صورت خودکار به تمام صفحات اضافه می‌شود:

```
public class PageEvents : PdfPageEventHelper
{
        PdfTemplate _backgroundImageTemplate;

        public override void OnStartPage(PdfWriter writer, Document document)
        {
            base.OnStartPage(writer, document);
            writer.DirectContent.AddTemplate(_backgroundImageTemplate, 0, 0);
        }

        public override void OnOpenDocument(PdfWriter writer, Document document)
        {
            _backgroundImageTemplate = writer.DirectContent.CreateTemplate(document.PageSize.Width,
document.PageSize.Height);
        }

        public override void OnCloseDocument(PdfWriter writer, Document document)
        {
            base.OnCloseDocument(writer, document);

            iTextSharp.text.Image img = iTextSharp.text.Image.GetInstance(
@"C:\My Pictures\bg.png");
            img.Alignment = iTextSharp.text.Image.UNDERLYING;
            img.SetAbsolutePosition((document.PageSize.Width - img.Width) / 2,
(document.PageSize.Height - img.Height) / 2);
            _backgroundImageTemplate.AddImage(img);
        }
}
```

عنوان: **فرمت مناسب تصاویر جهت استفاده در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۲۵ ۰۰:۰۲:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

عموما هنگام تهیه یک مستند یا گزارش، هرچقدر حجم نهایی کمتر باشد، توزیع آن ساده‌تر خواهد بود. در اینجا اینطور به نظر می‌رسد که اگر مثلا از تصاویری با فرمت jpg یا png استفاده کنیم، کمترین حجم نهایی را می‌توان بدست آورد. اما حین استفاده از iTextSharp شما با استفاده از تصاویری با فرمت BMP بهترین نتیجه را خواهید گرفت: کمترین حجم و بهترین کیفیت! البته یک نکته‌ی ریز دارد که باید رعایت شود:

```
using (var pdfDoc = new Document(PageSize.A4))
{
    var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("tpn.pdf", FileMode.Create));
    pdfWriter.SetPdfVersion(new PdfName("1.5"));
    pdfWriter.CompressionLevel = PdfStream.BEST_COMPRESSION;
    //...

}
```

در اینجا pdf version و همچنین compression level باید تنظیم شوند. پس از آن فشرده سازی تصاویر BMP به صورت خودکار حین تهیه فایل نهایی انجام خواهد شد.

عنوان: **چند ستونه کردن در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۲۶ ۱۱:۰۶:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

فرض کنید جدولی دارید با چند ستون محدود که نتیجه‌ی نهایی گزارش آن مثلا 100 صفحه است. جهت صرفه جویی در کاغذ مصرفی شاید بهتر باشد که این جدول را به صورت چند ستونی مثلا 5 ستون در یک صفحه نمایش داد؛ چیزی شبیه به شکل زیر:

header	header	header	header	header
192	144	96	48	0
193	145	97	49	1
194	146	98	50	2
195	147	99	51	3
196	148	100	52	4

روش انجام اینکار به کمک iTextSharp به صورت زیر است:

```
using System;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.Diagnostics;

class Program
{
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

                var table1 = new PdfPTable(1);
                table1.WidthPercentage = 100f;
                table1.HeaderRows = 2;
                table1.FooterRows = 1;

                //header row
```

```
var headerCell = new PdfPCell(new Phrase("header"));
table1.AddCell(headerCell);

//footer row
var footerCell = new PdfPCell(new Phrase(" "));
table1.AddCell(footerCell);

//adding some rows
for (int i = 0; i < 400; i++)
{
    var rowCell = new PdfPCell(new Phrase(i.ToString()));
    table1.AddCell(rowCell);
}

// wrapping table1 in multiple columns
ColumnText ct = new ColumnText(pdfWriter.DirectContent);
ct.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
ct.AddElement(table1);

int status = 0;
int count = 0;
int l = 0;
int columnsWidth = 100;
int columnsMargin = 7;
int columnsPerPage = 4;
int r = columnsWidth;
bool isRtl = true;

// render the column as long as it has content
while (ColumnText.HasMoreText(status))
{
    if (isRtl)
    {
        ct.SetSimpleColumn(
            pdfDoc.Right - l, pdfDoc.Bottom,
            pdfDoc.Right - r, pdfDoc.Top
        );
    }
    else
    {
        ct.SetSimpleColumn(
```

```
                        pdfDoc.Left + l, pdfDoc.Bottom,
                        pdfDoc.Left + r, pdfDoc.Top
                    );
                }

                var delta = columnsWidth + columnsMargin;
                l += delta;
                r += delta;

                // render as much content as possible
                status = ct.Go();

                // go to a new page if you've reached the last column
                if (++count > columnsPerPage)
                {
                    count = 0;
                    l = 0;
                    r = columnsWidth;
                    pdfDoc.NewPage();
                }
            }
        }

        //open the final file with adobe reader for instance.
        Process.Start("Test.pdf");
    }
}
```

توضیحات:

تا قسمت تعریف جدول و اضافه کردن سطرها و ستون‌های مورد نظر، همان بحث « تکرار خودکار سرستون‌های یک جدول در صفحات مختلف، توسط iTextSharp » می‌باشد.

اصل مطلب از قسمت ColumnText شروع می‌شود. با استفاده از شیء ColumnText می‌توان محتوای خاصی را در طی چند ستون در صفحه نمایش داد. عرض این ستون‌ها هم توسط متد SetSimpleColumn مشخص می‌شود و همچنین محل دقیق قرارگیری آن‌ها در صفحه. در اینجا دو حالت راست به چپ و چپ به راست در نظر گرفته شده است.

اگر حالت راست به چپ را در نظر بگیریم، محل قرارگیری اولین ستون از سمت راست صفحه (pdfDoc.Right) باید تعیین شود. سپس هربار به اندازه‌ی عرضی که مد نظر است باید محل شروع ستون را مشخص کرد (pdfDoc.Right - l). هر زمانیکه ct.Go فراخوانی می‌شود، تاجایی که میسر باشد، اطلاعات جدول 1 در یک ستون درج می‌شود. سپس بررسی می‌شود که تا این لحظه چند ستون در صفحه نمایش داده شده است. اگر تعداد مورد نظر ما (columnsPerPage) تامین شده باشد، کار را در صفحه‌ی بعد ادامه خواهیم داد (pdfDoc.NewPage)، در غیراینصورت مجددا مکان یک ستون دیگر در همان صفحه تعیین شده و کار افزودن اطلاعات به آن آغاز خواهد شد و این حلقه تا جایی که تمام محتوای جدول 1 را درج کند، ادامه خواهد یافت.

نظرات خوانندگان

نویسنده: Hamidrezabina
تاریخ: ۱۳۹۰/۰۵/۲۷ ۰۸:۲۳:۰۵

با سلام ...
به نظر شما بهتر نیست برای کارهامون از ابزارهای گزارش ساز استفاده کنیم؟
به نظر جای خالی مطلبی درباره ابزارهای گزارش ساز و مقایسه اونها در سایتتون احساس میشه.

ممنون از سایت خوبتون...

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۲۷ ۰۸:۳۳:۴۸

من با استفاده از iTextSharp یک گزارش ساز برای خودم درست کردم. این مطالب هم قسمتی از خلاصه نکاتی است که در ساخت آن استفاده شده...

نویسنده: Ahmad_Akbari2008
تاریخ: ۱۳۹۰/۰۵/۳۰ ۱۸:۰۱:۳۶

سلام مهندس
فکر می کنم در خط 76 ، columnsPerPage باید به (columnsPerPage - 1) تبدیل بشه.(تعداد ستونها در صفحه یکی بیشتره)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۳۰ ۱۹:۰۹:۴۴

بله. درسته.

عنوان: **روش صحیح تعریف قلم در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۳۱ ۱۱:۵۱:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

روش متداول تعریف فونت در iTextSharp به صورت زیر است:

```
public static iTextSharp.text.Font Tahoma()
{
     var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
     var baseFont = BaseFont.CreateFont(fontPath, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
     return new Font(baseFont);
}
```

از آنجائیکه خصوصا برای متون فارسی نیاز است تا به ازای هر المان کوچکی این فونت تنظیم شود و در غیر اینصورت متنی نمایش داده نخواهد شد، با سربار بالایی مواجه خواهیم شد. بنابراین به نظر می‌رسد که بهتر باشد این تولید اشیاء فونت را کش کنیم. خوشبختانه iTextSharp سیستم کش کردن تعریف قلم‌های متفاوت را هم به صورت توکار دارا است:

```
public static iTextSharp.text.Font GetTahoma()
{
    var fontName = "Tahoma";
    if (!FontFactory.IsRegistered(fontName))
    {
         var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
         FontFactory.Register(fontPath);
    }
    return FontFactory.GetFont(fontName, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
}
```

کلاس FontFactory کار ثبت و بازیابی قلم‌های متفاوت را به عهده دارد. تنها کافی است یکبار قلمی در آن ثبت شود (FontFactory.Register)، بار دیگر اطلاعات قلم به سادگی از کش FontFactory خوانده خواهد شد (FontFactory.GetFont).

عنوان: **تعریف رنگ در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۰۲ ۱۹:۳۱:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

در کتابخانه‌ی iTextSharp به جهت سازگاری با کتابخانه‌ی اصلی، رنگ‌ها را بر اساس کلاسی به نام BaseColor تعریف کرده‌اند؛ که ای‌کاش به جای این‌کار، همه را با کلاس Color فضای نام استاندارد System.Drawing جایگزین می‌کردند. همین مشکل با فونت هم هست. یک کلاس فونت در فضای نام iTextSharp.text وجود دارد به علاوه کلاس فونت تعریف شده در فضای نام استاندارد System.Drawing دات نت؛ که خیلی سریع می‌تواند به خطای کامپایل زیر ختم شود:

```
'Font' is an ambiguous reference between 'iTextSharp.text.Font' and 'System.Drawing.Font'
```

و در نهایت مجبور خواهیم شد که به صورت صریح علام کنیم، iTextSharp.text.Font منظور ما است و نه آن یکی.
در کل اگر با کلاس Color فضای نام استاندارد System.Drawing بیشتر راحت هستید به صورت زیر هم می‌توان رنگ‌های متداول را مورد استفاده قرار داد:

تعریف رنگ‌ها بر اساس نام آن‌ها:

```
var color = new BaseColor(Color.LightGray);
```

تعریف رنگ‌ها بر اساس مقادیر Hex متداول در المان‌های HTML :

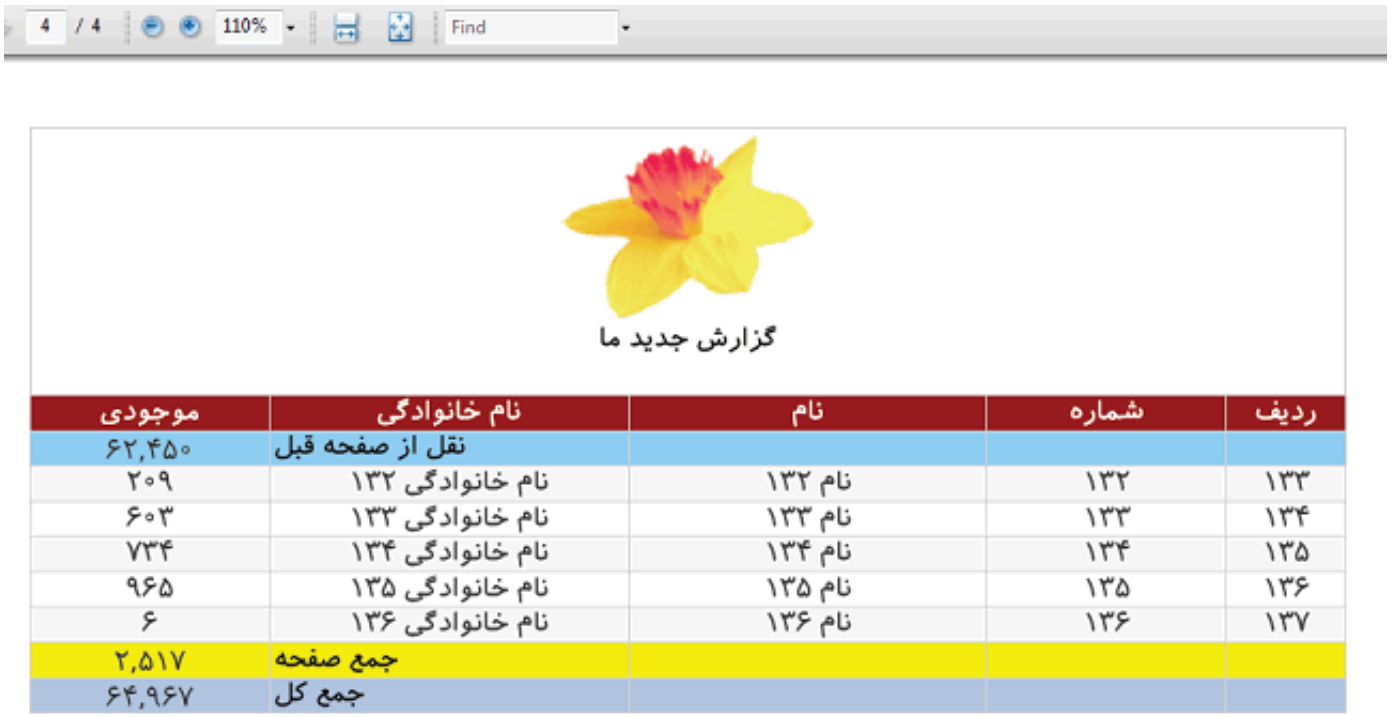
```
var color = new BaseColor(ColorTranslator.FromHtml("#1C5E55"));
```

این نکته‌ای است که شاید خیلی‌ها از وجود آن بی‌اطلاع باشند. به صورت پیش فرض در کلاس استاندارد ColorTranslator، امکان دریافت رنگ‌های بکاررفته در المان‌های HTML به کمک متد ColorTranslator.FromHtml مهیا است.
البته اگر زمانی خواستید خودتان این متد را پیاده سازی کنید، نکته‌ی آن به صورت زیر است:

```
string htmlColor = "#1C5E55";
int x = Convert.ToInt32(htmlColor.Replace("#", "0x"), 16);
byte red   = (byte)((x & 0xff0000) >> 16);
byte green = (byte)((x & 0xff00) >> 8);
byte blue  = (byte)(x & 0xff);
```

سپس کلاس‌های Color و همچنین BaseColor امکان پذیرش این اجزای حاصل را دارند (به کمک متد Color.FromRgb و یا سازنده‌ی BaseColor).
علت ذکر ColorTranslator.FromHtml به این بر می‌گردد که ترکیبات رنگ جالبی را می‌توان از جداول HTML ایی موجود در

سایت‌های مختلف ایده گرفت و یا حتی از قالب‌های پیش فرض GridView در ASP.NET مثلا.



ردیف	شماره	نام	نام خانوادگی	موجودی
			نقل از صفحه قبل	۶۲,۴۵۰
۱۳۳	۱۳۲	نام ۱۳۲	نام خانوادگی ۱۳۲	۲۰۹
۱۳۴	۱۳۳	نام ۱۳۳	نام خانوادگی ۱۳۳	۶۰۳
۱۳۵	۱۳۴	نام ۱۳۴	نام خانوادگی ۱۳۴	۷۳۴
۱۳۶	۱۳۵	نام ۱۳۵	نام خانوادگی ۱۳۵	۹۶۵
۱۳۷	۱۳۶	نام ۱۳۶	نام خانوادگی ۱۳۶	۶
			جمع صفحه	۲,۵۱۷
			جمع کل	۶۴,۹۶۷

اکنون برای ساخت جدولی مانند شکل فوق، به ازای هر سلولی که مشاهده می‌کنید باید یکبار BorderColor و BackgroundColor تنظیم شوند. رنگ متن هم از رنگ فونت دریافت می‌شود:

```
var pdfCell = new PdfPCell(new Phrase(Text, Font))
            {
                RunDirection = ...,
                BorderColor = ...,
                BackgroundColor = ...
            };
```

نظرات خوانندگان

نویسنده: Mohsen Najafzadeh
تاریخ: ۱۳۹۰/۰۶/۰۶ ۱۰:۲۰:۴۴

سلام استاد
اگه راه داره این مثال رو رو سایت قرار بدین ممنون می شم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۰۶ ۱۰:۵۲:۴۹

یک مثال خوب در این زمینه: تهیه خروجی PDF از GridView

نویسنده: Mohsen Najafzadeh
تاریخ: ۱۳۹۰/۰۶/۰۶ ۱۱:۰۱:۳۴

ممنون
یا علی

نویسنده: Nima
تاریخ: ۱۳۹۰/۰۶/۱۳ ۰۰:۵۴:۲۶

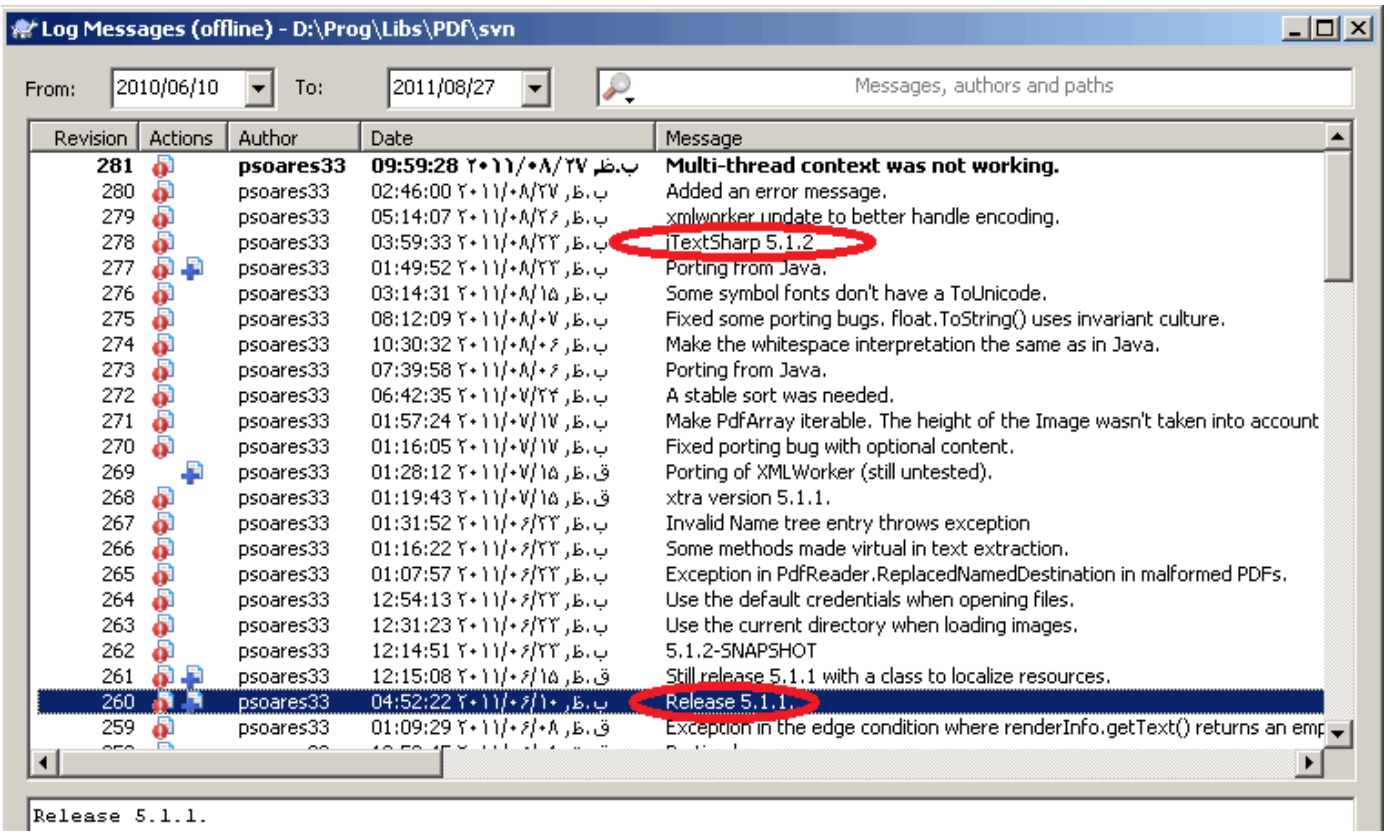
ممنون از مثال مفیدتون
میخواستم بپرسم چطور میتونم عکس رو در کنار متن همینطوری که شما در عکس بالا دارین داشته باشم؟ من به خاصیت Image یک سلول وقتی مقدار میدم کل سلول رو میگیره و وقتی که تو تا سلول تعریف میکنم Border ها دردسر ساز میشن و من میخوام دقیقا مثل عکسی که در بالا هست بدون Border باشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۱۳ ۰۱:۱۰:۳۹

هدر مثال بالا یک جدول است (یک ستون دارد و سه ردیف). عکس هم در یک سلول آن (PdfPCell) قرار گرفته (اولین سلول که می‌شود ردیف یک). برای عدم نمایش حاشیه یا border ، مقدار آن را مساوی صفر قرار دهید (cell.Border = 0). برای اینکه کل سلول را پوشش ندهد خاصیت fit image را مانند مثالی که قبلا در این سایت زده شده، false کنید. برای اینکه در وسط قرار گیرد، cell.HorizontalAlignment = 1 را تنظیم کنید.

عنوان: **تهیه فید از تغییرات SVN**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۰۸ ۱۱:۵۳:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

کتابخانه‌ی iTextSharp 5.1.2 هفته‌ی قبل منتشر شده و ... من هر چقدر سایتی، بلاگی جایی را جستجو کردم که خلاصه‌ای از تغییرات انجام شده آن‌را گزارش دهد، چیزی نیافتم. ولی خوب، مطابق روال متداول کتابخانه‌های سورس باز، حداقل می‌توان به change log مرتبط با سورس کنترل آن‌ها مراجعه کرد. مثلا:



البته این هم خوب است ولی ای‌کاش می‌شد مثلا یک فید هم از این تغییرات تهیه کرد. یک سری از سایت‌های هاستینگ مثل CodePlex و GitHub یک چنین فیدهایی را دارند. اما به نظر SourceForge از این لحاظ اندکی ضعیف است.
سایت روسی زیر می‌تواند با گرفتن آدرس یک مخزن کد SVN (برای مثال: https://itextsharp.svn.sourceforge.net/svnroot/itextsharp/trunk/) یک فید RSS از آن تهیه کند:

http://svn2rss.ru/

در همین راستا برنامه‌ی CommitMonitor هم موجود است.

نظرات خوانندگان

نویسنده: A.Karimi
تاریخ: ۱۳۹۰/۰۶/۰۹ ۲۰:۲۱:۱۱

iText (نسخه جاوا) ظاهراً دو Lisence جداگانه، یکی برای کارهای تجاری و دیگری AGPL است. که AGPL باعث محدودیت هایی برای کارهای تجاری خواهد بود.

در مورد iTextSharp چطور؟ مسأله‌ای از این نظر ندارد؟

نویسنده: A.Karimi
تاریخ: ۱۳۹۰/۰۶/۰۹ ۲۰:۲۲:۵۸

License*

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۰۹ ۲۰:۳۷:۲۱

بله. قبلا توضیح دادم: [+]

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۱۶ ۲۰:۵۰:۲۶

یک مطلب مرتبط:
وبلاگ نویسنده iText : ((+))

عنوان: **تبدیل HTML به PDF با استفاده از کتابخانه‌ی iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۱۲ ۰۹:۵۲:۰۰
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp

روش متداول کار با کتابخانه‌ی iTextSharp ، ایجاد شیء Document ، سپس ایجاد PdfWriter برای نوشتن در آن، گشودن سند و افزودن اشیایی مانند PdfPCell ، PdfPTable ، Paragraph و غیره به آن است و در نهایت بستن سند. راه میانبری هم برای کار با این کتابخانه وجود دارد و آن هم استفاده از امکانات فضای نام iTextSharp.text.html.simpleparser آن می‌باشد. به این ترتیب می‌توان به صورت خودکار، یک محتوای HTML را تبدیل به فایل PDF کرد. ...

مثال : نمایش یک متن HTML ساده انگلیسی

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.html.simpleparser;
using iTextSharp.text.pdf;

namespace HeadersAndFooters
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf", FileMode.Create));
                pdfDoc.Open();

                var html = @"<span style='color:blue'><b>Testing</b></span>
                             <i>iTextSharp's</i> <u>HTML to PDF capabilities</u>";
                var parsedHtmlElements = HTMLWorker.ParseToList(new StringReader(html), null);

                foreach (var htmlElement in parsedHtmlElements)
                {
                    pdfDoc.Add(htmlElement);
                }
            }

            //open the final file with adobe reader for instance.
            Process.Start("Test.pdf");
```

```
        }
    }
}
```

نکته‌ی جدید کد فوق، استفاده از متد HTMLWorker.ParseToList است. به این ترتیب parser کتابخانه‌ی iTextSharp وارد عمل شده و html تعریف شده را به معادل المان‌های بومی خودش تبدیل می‌کند؛ مثلا تبدیل به chunk یا pdfptable و امثال آن. در نهایت در طی یک حلقه، این عناصر به صفحه اضافه می‌شوند.
البته باید دقت داشت که HTMLWorker امکان تبدیل عناصر پیچیده، تودرتو و چندلایه HTML را ندارد؛ اما بهتر از هیچی است!

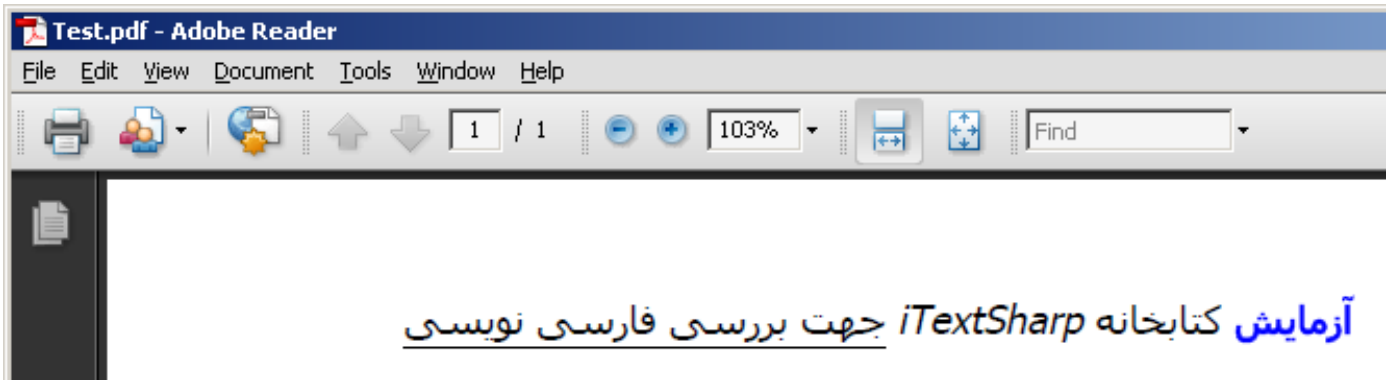
همه‌ی این‌ها خوب! اما به درد ما فارسی زبان‌ها نمی‌خورد. همین متغیر html فوق را با یک متن فارسی جایگزین کنید، چیزی نمایش داده نخواهد شد. البته این هم نکته دارد که در ادامه ذکر خواهد شد.
جهت نمایش متون فارسی نیاز است تا نکات ذکر شده در مطلب « فارسی نویسی و iTextSharp » رعایت شوند که شامل:
- تعیین صریح قلم
- تعیین encoding
- استفاده از عناصر دربرگیرنده‌ای است که خاصیت RunDirection را پشتیبانی می‌کنند؛ مانند PdfPCell و غیره

به این ترتیب خواهیم داشت:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.html.simpleparser;
using iTextSharp.text.pdf;
using iTextSharp.text.html;

namespace HeadersAndFooters
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf", FileMode.Create));
                pdfDoc.Open();

                //روش صحیح تعریف فونت
                FontFactory.Register("c:\\windows\\fonts\\tahoma.ttf");

                StyleSheet styles = new StyleSheet();
                styles.LoadTagStyle(HtmlTags.BODY, HtmlTags.FONTFAMILY, "tahoma");
```

```
                styles.LoadTagStyle(HtmlTags.BODY, HtmlTags.ENCODING, "Identity-H");

                var html = @"<span style='color:blue'><b>آزمایش</b></span>
                                کتابخانه <i>iTextSharp</i> <u>جهت بررسی فارسی نویسی</u>";
                var parsedHtmlElements = HTMLWorker.ParseToList(new StringReader(html), styles);

                PdfPCell pdfCell = new PdfPCell { Border = 0 };
                pdfCell.RunDirection = PdfWriter.RUN_DIRECTION_RTL;

                foreach (var htmlElement in parsedHtmlElements)
                {
                    pdfCell.AddElement(htmlElement);
                }

                var table1 = new PdfPTable(1);
                table1.AddCell(pdfCell);
                pdfDoc.Add(table1);
            }

            //open the final file with adobe reader for instance.
            Process.Start("Test.pdf");
        }
    }
}
```

همانطور که ملاحظه می‌کنید ابتدا قلمی در cache قلم‌های این کتابخانه ثبت می‌شود (FontFactory.Register). سپس نوع قلم و encoding آن توسط یک StyleSheet تعریف شده و به HTMLWorker.ParseToList ارسال می‌گردد و در نهایت به کمک یک المان دارای RunDirection، در صفحه نمایش داده می‌شود.



نکته:
ممکن است که به متغیر html ، یک table ساده html را نسبت دهید. در این حالت پس از تنظیم style یاد شده، در هر سلول این html table ، متون فارسی به صورت معکوس نمایش داده خواهند شد که این هم یک نکته‌ی کوچک دیگر دارد:

```
foreach (var htmlElement in parsedHtmlElements)
                {
                    if (htmlElement is PdfPTable)
                    {
                        var table = (PdfPTable)htmlElement;
                        table.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
                        foreach (var row in table.Rows)
                        {
                            foreach (var cell in row.GetCells())
                            {
                                cell.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
                            }
                        }
                    }

                    pdfCell.AddElement(htmlElement);
                }
```

در قسمتی که قرار است المان‌های معادل به pdfCell اضافه شوند، آن‌ها را بررسی کرده و RunDirection آن‌ها را RTL خواهیم کرد.

کاربردها:
بدیهی است این حالت برای تهیه گزارشات پیشرفته‌تر برای مثال تهیه قالب‌هایی که در حین تهیه PDF ، قسمت‌هایی از آن‌ها توسط برنامه نویس Replace می‌شوند، بسیار مناسب است.
همچنین مطلب « بارگذاری یک یوزرکنترل با استفاده از جی‌کوئری » و متد RenderUserControl مطرح شده در آن که در نهایت یک قطعه کد HTML را به صورت رشته به ما تحویل می‌دهد، می‌تواند جهت تهیه گزارش‌های پویایی که برای مثال قسمتی از آن یک GridView بایند شده حاصل از یک یوزر کنترل است، مورد استفاده قرار گیرد.

نظرات خوانندگان

نویسنده: A.Karimi
تاریخ: ۱۳۹۰/۰۶/۱۳ ۱۶:۴۳:۲۶

بسیار عالی!

نویسنده: Hossein Hariri
تاریخ: ۱۳۹۰/۰۶/۲۱ ۱۲:۱۲:۵۸

با سلام و تشکر
من از روش شما برای تولید pdf استفاده کردم. ولی متاسفانه align متنهای ایجاد شده Left است و هر کاری میکنم درست نمیشود. لطفا راهنمایی بفرمایید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۲۱ ۱۵:۲۵:۲۱

برای اینکار نیاز است تا style تعریف شده را کمی تغییر داد. به صورت زیر:

```
(styles.LoadTagStyle(HtmlTags.BODY, HtmlTags.ALIGN, HtmlTags.ALIGN_LEFT
```

نویسنده: Alisfard
تاریخ: ۱۳۹۰/۰۶/۲۲ ۲۳:۴۰:۳۸

لطفا میشه نمونه کد را برای دانلود قرار دهید چون من هر چی سعی کردم نتونستم اجرا کنم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۲۲ ۲۳:۵۴:۰۷

از اینجا قابل دریافت است: [^]

نویسنده: hamidnch2007
تاریخ: ۱۳۹۰/۰۷/۲۹ ۰۰:۳۵:۰۲

من همین روش رو برای گریدویو تودرتو که داخل یه div هستند و آن div بصورت runat=server میباشد و آن را با استفاده از متد RenderControl داخل یه String Writer ریختم و اونو بعنوان پارامتر StreamReader پاس دادم.نتیجه این شد که فیلدهای گرید بیرونی درست نمایش داده شدند ولی گریدویو داخلی متنهای آن بصورت برعکس و نچسبیده نمایان شدندممنون یشم راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۹ ۰۸:۴۰:۳۴

parsedHtmlElements تولیدی را دیباگ و سپس نکته آخری رو که در متن بالا عنوان شد باید اعمال کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۹ ۱۳:۳۶:۰۶

چند نکته عمومی:
- اینجا انجمن نیست. مشکلات عمومی خودتون رو در انجمن‌ها پیگیری کنید.
- اگر خواستید جایی کدی طولانی را ارسال کنید حداقل از سایت http://pastebin.com استفاده کنید.
- در مورد این مثال جاری، تا دسترسی به html نهایی تولیدی شما نباشد، دیباگ کردن آن بی‌معنا است.

نویسنده: hamidnch2007
تاریخ: ۱۳۹۰/۰۷/۲۹ ۱۵:۰۳:۵۰

بابت اشتباهم عذرخواهی میکنم.
من فقط میخواستم اگر برایتان امکان پذیر است یه مثال کوچک از یه html پارس شده که شامل گریدویو تودرتو هست برایم بزنید و روال را بگویید.جواب شما خیلی کلی بود و بنده متوجه نشدم.باز هم ببخشید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۹ ۲۱:۲۰:۴۶

با گرید تو در تو (جداول تو در تو) هم کار می‌کنه؛ فقط اینبار باید اون حلقه‌ای رو که به عنوان نکته گفتم، تبدیل به یک تابع بازگشتی کنید. به این صورت: http://pastebin.com/zpMsPmMa

نویسنده: hamidnch2007
تاریخ: ۱۳۹۰/۰۷/۲۹ ۲۱:۳۳:۳۳

خدا یک در دنیا و هزار در آخرت بهت بده.الهی خیر ببینی.کارت درسته.آقای نصیری.
یه سوال دیگه بکنم.اگه من تو گریدویو چک باکس داشته باشم اون رو هم میشه به pdf ارسال کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۹ ۲۲:۰۴:۵۰

تعداد تگ‌هایی که iTextSharp ساپورت می‌کنه کم هست. لیست این‌ها رو می‌تونید در کلاس HtmlTags فضای نام iTextSharp.text.html مشاهده کنید.

یک توصیه کلی:
اگر به دنبال یک راه حل حرفه‌ای برای کارهای پیچیده‌تر HTML to PDF هستید، باید سراغ این نوع کتابخانه‌ها بروید:
(wkhtmltopdf ,Convert html to pdf using webkit (qtwebkit
برای مثال این مورد از WebKit یا همان موتور گوگل کروم استفاده می‌کند. بنابراین HTML parser آن مانند iTextSharp محدود نیست و فوق العاده حرفه‌ای است.

نویسنده: hamidnch2007
تاریخ: ۱۳۹۰/۰۷/۲۹ ۲۳:۱۲:۳۷

از خدا برای شما بهترین ها را آرزو میکنم.امشب مشکل من را حل کردید.امیدوارم که ثمره آن را در زندگی تان ببینی.اگر روزی بدانم میتوانم برایتان کاری انجام بدم و در توانم باشد دریغ نخواهم کرد.همیشه پیروز باشید.

<> Disqus 2011/10/21

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۹ ۲۳:۳۹:۰۶

ممنون. سلامت باشید.

نویسنده: mkpro
تاریخ: ۱۳۹۱/۰۷/۲۵ ۱۳:۲۸

سلام خسته نباشید
من جدیداً سایتتون رو کشف کردم سایت خیلی مفید و پر باری دارید.
من یه مشکل توی تبدیل html به pdf داشتم وقتی تعداد صفحاتم بیشتر از 2 بشه بهم اخطاره Object reference not set to an

instance of an object. و این اخطاره مربوط میشه به pdfdoc.add(Table)
ممنون میشم یه راه حلی ارائه بدید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۵ ۱۳:۴۰

HTML worker مطرح شده در این مطلب، مدتی است که از طرف نویسندگان آن منسوخ شده اعلام گردیده و دیگری پشتیبانی یا نگهداری نخواهد شد. بنابراین اگر باگی وجود دارد یا هر مطلبی، کسی دیگر به آن رسیدگی نخواهد کرد.
راه حل جایگزین، استفاده از XML Worker است که بجای HTML worker در حال کار و توسعه می‌باشد .

نویسنده: جواد جعفری
تاریخ: ۱۳۹۱/۰۸/۰۴ ۱۴:۸

بابا دمت گرم خیلی خیلی ممنونم واقعا به دردم خورد

نویسنده: مرتضی موسوی
تاریخ: ۱۳۹۲/۰۴/۱۰ ۱۱:۵۸

آقای مهندس , میتونید راهنمایی کنید من چطور میتونم قابلیت Save a Copy as رو غیر فعال کنم . بعبارتی نباید کاربر بتونه از فایل کپی بگیره .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۱۰ ۱۲:۱۵

مراجعه کنید به مطلب « رمزنگاری فایل‌های PDF با استفاده از کلید عمومی توسط iTextSharp ». در اینجا توسط مقادیری مانند PdfWriter.ALLOW_COPY و غیره می‌شود روی فایل تولیدی محدودیت ایجاد کرد. ضمنا راه برای برطرف کردن این محدودیت‌ها هم هست .

نویسنده: راد
تاریخ: ۱۳۹۲/۰۸/۰۴ ۱۰:۵۷

باسلام
امکان داره نمونه کد این مثال گذاشته شود
باتشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۴ ۱۳:۲۲

- نمونه کد همان مثالی هست که در متن آورده شده. برای اجرا تنها نیاز به کتابخانه iTextSharp دارد. (یک برنامه کنسول ساده را ایجاد کنید. کدهای مثال مطلب فوق را در آن paste کنید و بعد ارجاعی را به اسمبلی iTextSharp به آن اضافه نمائید)
- ضمنا افزونه HTMLWorker این کتابخانه منسوخ شده (مطلب جاری) و به XMLWorker ارتقاء یافته .

نویسنده: فیضی
تاریخ: ۱۳۹۲/۱۱/۱۵ ۹:۴۲

سلام.
من کدهای بالا رو استفاده کردم تا محتویات یک فیلد از جدول که محتویاتش از ادیتور Tinymce پر میشه رو به pdf تبدیل کنم.
در ادیتور عکس هم ممکنه درج بشه.

مشکلی که وجود داره اینه که عکسهایی که در ادیتور در سمت راست قرار داده شدن در pdf در سمت چپ قرار می‌گیرن. این مشکل رو چطور میشه رفع کرد؟
نمونه رو می‌تونید از لینک زیر دانلود کنید.

pdf
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۵ ۹:۵۴

- پردازش CSS کتابخانه HTMLWorker خیلی ضعیف و ابتدایی است. به همین جهت آن‌را کنار گذاشته‌اند و به XMLWorker کوچ کرده‌اند (HTMLWorker هیچ پشتیبانی رسمی دیگر ندارد ؛ به قسمت Deprecated. please switch to XML Worker instead آن دقت کنید). ضمنا HTMLWorker مشکلات دیگری هم دارد. مثلا یک تگ hr در صفحه باشد، کرش می‌کند. پردازش ویژگی‌های مختلف CSS و HTML تقریبا در آن پیاده سازی نشده و ...
- برای کار با ADO.NET بهتر است این روزها از Micro ORMs استفاده کنید.

نویسنده: فیضی
تاریخ: ۱۳۹۲/۱۱/۱۸ ۰:۳

ممنون از پاسخ. بله، من xml worker ای که شما توضیح داده بودید رو هم تست کرده بودم. html ای که شما پاس داده بودید رو به خوبی نشون می‌داد ولی من با همون داده‌های پست قبلی مثال شما رو تست کردم. حروف انگلیسی درست نشون داده میشن ولی فارسی‌ها به شکل نقطه دیده میشن عکسها هم سمت چپ مشاهده می‌شوند
اینم لینک نمونه تستی
Sample
یا شاید من اشتباه فهمیدم که xmlworker اون قدر قوی هست که عناصر html رو می‌تونه با همون استایلی که دارن نشون بده؟
ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۸ ۰:۳۱

- در مورد فارسی نویسی در iTextSharp یک دیباگ مرحله به مرحله قبلا در سایت مطرح شده. اگر خروجی یونیکد نگرفتید یعنی قلم صحیحی در حال استفاده نیست. کدهایی که قبلا ارسال کرده بودم به این نحو است:

```
// روش صحیح تعریف فونت
var systemRoot = Environment.GetEnvironmentVariable("SystemRoot");
FontFactory.Register(Path.Combine(systemRoot, "fonts\\tahoma.ttf"));
```

در کدهای شما به این نحو:

```
var systemRoot = Environment.GetEnvironmentVariable("SystemRoot");
FontFactory.Register(Path.Combine(systemRoot, "c:\\windows\\fonts\\tahoma.ttf"));
```

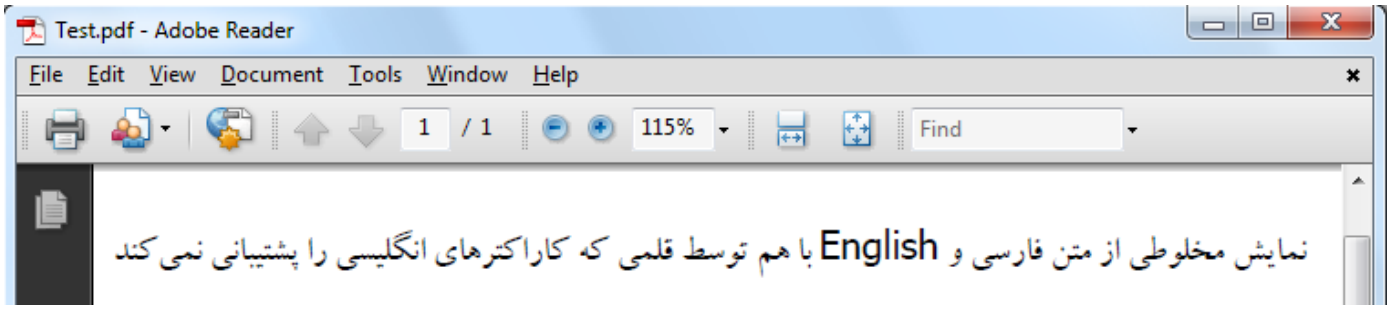
با توجه به استفاده از Path.Combine، مسیری را که معرفی کرده‌اید می‌شود چیزی مانند c:\\windows\\c:\\windows\\fonts\\tahoma.ttf . به همین جهت این فونت یافت نشده و ثبت نمی‌شود (چون دوبار system root در آن وجود دارد).
- بله؛ قدرت پردازش CSS در XML Worker آن خیلی بهتر است از HTML Worker.
- در مورد میزان چرخش جدول، RunDirection = PdfWriter.RUN_DIRECTION_RTL را با حالت LTR هم تست کنید (PdfWriter.RUN_DIRECTION_LTR).

عنوان: **iTextSharp و استفاده از قلم‌های محدود فارسی**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۲۲ ۰۹:۲۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

عموما قلم‌های فارسی، خصوصا مواردی که با B شروع می‌شوند مانند B Zar و امثال آن، فاقد تعاریف حروف مرتبط با glyphs الفبای انگلیسی است. نتیجه این خواهد شد که اگر متن شما مخلوطی از کلمات و حروف فارسی و انگلیسی باشد، فقط قسمت فارسی نمایش داده می‌شود و از قسمت انگلیسی صرفنظر خواهد شد. مرورگرها در این حالت هوشمندانه عمل می‌کنند و به یک قلم پیش فرض مانند Times و همانند آن جهت نمایش اینگونه متون مراجعه خواهند کرد؛ اما اینجا چنین اتفاقی نخواهد افتاد.
برای حل این مشکل، کلاسی به نام FontSelector در کتابخانه‌ی iTextSharp وجود دارد. مثالی در این رابطه:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace HeadersAndFooters
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf", FileMode.Create));
                pdfDoc.Open();

                FontFactory.Register("c:\\windows\\fonts\\bzar.ttf");
                Font bZar = FontFactory.GetFont("b zar", BaseFont.IDENTITY_H);

                FontFactory.Register("c:\\windows\\fonts\\tahoma.ttf");
                Font tahoma = FontFactory.GetFont("tahoma", BaseFont.IDENTITY_H);

                FontSelector fontSelector = new FontSelector();

                //قلم اصلی
                if (bZar.Familyname != "unknown")
                {
                    fontSelector.AddFont(bZar);
                }

                //قلم پیش فرض در صورت نبود تعاریف مناسب در قلم اصلی
                if (tahoma.Familyname != "unknown")
                {
                    fontSelector.AddFont(tahoma);
                }

                var table1 = new PdfPTable(1);
                table1.WidthPercentage = 100;
                table1.RunDirection = PdfWriter.RUN_DIRECTION_RTL;

                var pdfCell = new PdfPCell { RunDirection = PdfWriter.RUN_DIRECTION_RTL, Border = 0 };
                pdfCell.Phrase = fontSelector.Process("با هم توسط English نمایش مخلوطی از متن فارسی و
قلمی که کاراکترهای انگلیسی را پشتیبانی نمی‌کند");

                table1.AddCell(pdfCell);
                pdfDoc.Add(table1);

            }

            //open the final file with adobe reader for instance.
            Process.Start("Test.pdf");
        }
    }
}
```

در این مثال از قلم B Zar استفاده شده است. اولین قلمی که به یک FontSelector اضافه می‌شود، قلم اصلی خواهد بود. قلم بعدی اضافه شده، قلم پیش فرض نام خواهد گرفت؛ به این معنا که در مثال فوق اگر قلم B Zar توانایی نمایش حرف جاری را داشت که خیلی هم خوب، در غیراینصورت به قلم بعدی مراجعه خواهد کرد و همینطور الی آخر. بنابراین این ترتیب اضافه کردن قلم‌ها به FontSelector مهم است. نحوه استفاده نهایی از FontSelector تعریف شده هم در قسمت pdfCell.Phrase = fontSelector.Process مشخص است.



نظرات خوانندگان

نویسنده: Ramin
تاریخ: ۱۳۹۰/۰۶/۲۲ ۲۳:۲۸:۱۶

سلام آقای مهندس
شما برای نمایش PDF در سایت چه روشی رو پیشنهاد میدید؟ نمایش در مرورگر ، دانلود فایل و...

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۲۲ ۲۳:۴۱:۱۱

روش متداول، نصب Adobe reader در سمت کاربر است. اکتیوایکس آن سال‌ها است که با اکثر مرورگرها کار می‌کند و امکان مشاهده فایل pdf را درون خود مرورگر به صورت یکپارچه میسر کرده.
http://www.dotnettips.info/2011/07/pdf-winforms-wpf.html

نویسنده: امیر بختیاری
تاریخ: ۱۳۹۲/۰۳/۱۱ ۱۵:۸

با سلام
می‌خواستم برای حل این مشکل در RDLC چه راهی وجود داره
چون من تمام گزارشات سیستم رو با این ساختم و کلی دردسر برای ساختش کشیدم
البته یه راه حل گیر آوردم و این بود که اومدم با یه نرم افزار فونت هایی که می‌خواستم را ویرایش کردم و مثلا قسمت‌های تعاریف حروف مرتبط با glyphs الفبای انگلیسی را خودم از یه فونت دیگه مثل تایم اضافه کردم عملی هم بود ولی ساخت هر فونت با مشتقاتش 5-6 ساعت وقت میگیره
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۱ ۱۶:۳۴

با توجه به غیرسورس باز بودن PDF سازی که یاد کردید، بجز ویرایش فونت و افزودن دستی glyphs مفقود در آن‌ها، راه دیگری وجود ندارد. در iTextSharp برای اینکار FontSelector طراحی شده. طراحان گزارش ساز مدنظر شما هم باید چنین کاری رو انجام بدن و اضافه کنند. ضمن اینکه در iTextSharp هم اگر کسی این نکته رو ندونه، به صورت پیش فرض از FontSelector استفاده نمیشه و مدتی سردرگم خواهد بود.
در PdfReport این مسایل به صورت توکار در همه جا اعمال شده و استفاده کننده با خیلی از جزئیات و نکات ریز درگیر نخواهد شد.

نویسنده: محسن نجف زاده
تاریخ: ۱۳۹۲/۰۳/۲۱ ۸:۱۶

با سلام
چون من از HTMLWorker استفاده می‌کنم و به کمک کد زیر فونت BNazanin را بکار گرفتم

```
styles.LoadTagStyle(HtmlTags.BODY, HtmlTags.FONTFAMILY, "BNazanin");
```

لازم بود تا کلمات انگلیسی هم نمایش داده شوند در نتیجه از فونت کامل tahoma بایستی استفاده می‌کردم.اما این فونت مورد پسند نیست. در نتیجه از روش زیر (که شاید هم نا متعارف باشد) استفاده کردم

1.ابتدا استایل زیر را اضافه نمودم

```
styles.LoadStyle("english", HtmlTags.FONTFAMILY, "tahoma");
```

2.و سپس تمامی کلمات انگلیسی را به کمک کد زیر یافته و استایل english را به آن نسبت دادم

```
var cleanTagsContent = Regex.Replace(content, @"<[^>]*>", String.Empty);
            var regex = new Regex("[a-zA-Z0-9]*");
            foreach (var word in cleanTagsContent.Split(' '))
                if (regex.Match(word).Value == word && word.Length > 0)
                {
                    content = content.Replace("<" + word, "#!#");
                    content = content.Replace(word + ">", "#^#");

                    content = content.Replace(word, string.Format("<span class='english'>{0}</span>",
word));

                    content = content.Replace("#!#", "<" + word);
                    content = content.Replace("#^#", word + ">");
                }
```

نکته : کدهای به صورت زیر را برای زمانی گذاشتیم که کلمه انگلیسی شامل td,table,div,... باشد

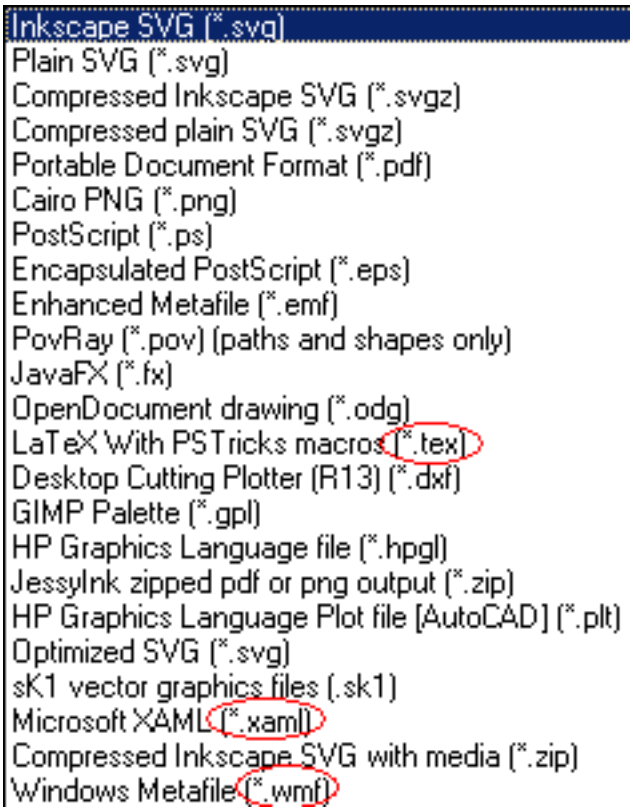
```
content = content.Replace("<" + word, "#!#");
```

باز هم مرا به خاطر این کار نامتعارف ببخشید :)

عنوان: **استفاده از گرافیک برداری در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۳ ۱۹:۴۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

در مورد «ترسیم اشکال گرافیکی با iTextSharp» مطلب مفصلی را در اینجا می‌توانید مطالعه کنید؛ که قصد تکرار مجدد آن‌را ندارم. فقط این روش‌ها یک مشکل مهم دارند : «کار من ترسیم این نوع اشکال گرافیکی نیست!». مثلا من الان نیاز دارم در گزارشی، بجای ستون Boolean آن در مواردی که مقدار ردیف true هست، مثلا یک «چک مارک» را بجای true/false یا بله/خیر نمایش دهم. می‌شود اینکار را با یک تصویر معمولی هم انجام داد. فقط حجم فایل حاصل، بیش از اندازه بالا می‌رود و همچنین نتیجه استفاده از یک bitmap، به زیبایی بکارگیری گرافیک برداری با قابلیت تغییر ابعاد بدون نگرانی در مورد از دست دادن کیفیت آن، نیست.

خوشبختانه هستند سایت‌هایی که این نوع تصاویر برداری را به رایگان ارائه دهند؛ برای مثال: سایت Openclipart ، تعداد قابل توجهی فایل با فرمت SVG دارد. فایل‌های SVG را مستقیما نمی‌توان توسط iTextSharp استفاده کرد؛ اما یک سری برنامه‌ی کمکی برای تبدیل فرمت SVG به مثلا XAML (قابل توجه برنامه نویس‌های WPF و Silverlight) یا WMF و غیره وجود دارد. برای نمونه iTextSharp امکان خواندن فایل‌های WMF را داشته (توسط همان متد معروف Image.GetInstance آن) و اینبار این Image حاصل، یک تصویر برداری است و نه یک Bitmap.
در بین این برنامه‌های تبدیل کننده فرمت‌های برداری، برنامه‌ی معروف و سورس باز Inkscape ، در صدر محبوبیت قرار دارد. تنها کافی است فایل SVG خود را در آن گشوده و سپس به انواع و اقسام فرمت‌های دیگر تبدیل (Save As) کنید:



یکی از فرمت‌های جالب خروجی آن، Tex است (مربوط به یک برنامه ادیتور، به نام LaTeX است). فرض کنید یکی از این «چک مارک»های سایت Openclipart را در برنامه Inkscape باز کرده و سپس با فرمت Tex ذخیره کرده‌ایم. خروجی فایل متنی آن مثلا به شکل زیر خواهد بود:

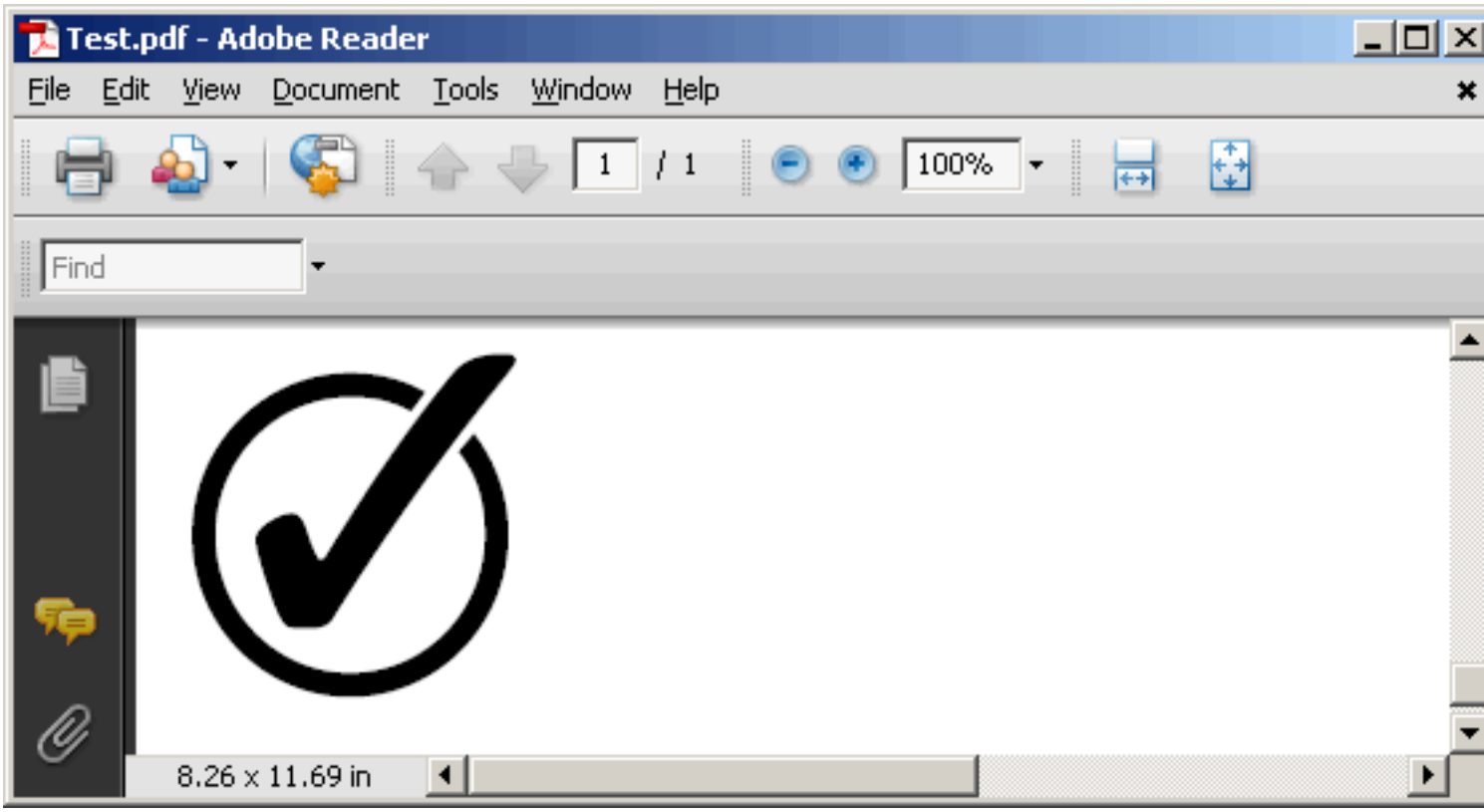
```
%LaTeX with PSTricks extensions
%%Creator: 0.48.0
%%Please note this file requires PSTricks extensions
\psset{xunit=.5pt,yunit=.5pt,runit=.5pt}
\begin{pspicture}(190,190)
{
\newrgbcolor{curcolor}{0 0 0}
\pscustom[linestyle=none,fillstyle=solid,fillcolor=curcolor]
{
\newpath
\moveto(52.73079005,101.89500456)
\curveto(31.29686559,101.89500456)(13.84575258,84.04652127)(13.8457479,62.12456369)
\curveto(13.8457479,40.20259605)(31.29686559,22.35412714)(52.73079005,22.35412235)
\curveto(74.16470983,22.35412235)(91.6158322,40.20259605)(91.61582751,62.12456369)
\curveto(91.61582751,71.60188248)(88.48023622,80.07729424)(83.15553076,87.02034164)
\lineto(79.49425309,82.58209245)
\curveto(84.13622847,76.73639073)(85.95313131,70.24630402)(85.95313131,62.12456369)
\curveto(85.95313131,43.33817595)(71.09893654,28.1547277)(52.73079005,28.1547277)
\curveto(34.36263419,28.15473249)(19.50844879,43.33817595)(19.50844879,62.12456369)
\curveto(19.50844879,80.91094185)(34.36264355,96.10336589)(52.73079005,96.10336589)
\curveto(58.55122776,96.10336589)(62.90459266,95.2476225)(67.65721002,92.5630926)
\lineto(71.13570481,97.23509821)
\curveto(65.57113223,100.3782653)(59.52269945,101.89500456)(52.73079005,101.89500456)
\closepath
}
}
{
\newrgbcolor{curcolor}{0 0 0}
\pscustom[linestyle=none,fillstyle=solid,fillcolor=curcolor]
{
\newpath
\moveto(38.33889376,67.35513328)
\curveto(39.90689547,67.35509017)(41.09296342,66.03921993)(41.89711165,63.40748424)
\curveto(43.50531445,58.47289182)(44.65118131,56.00562195)(45.33470755,56.0056459)
\curveto(45.85735449,56.00562195)(46.40013944,56.41682961)(46.96305772,57.23928802)
\curveto(58.2608517,75.74384316)(68.7143666,90.71198997)(78.32362116,102.14379168)
\curveto(80.81631349,105.10443984)(84.77658911,106.58480942)(90.20445269,106.58489085)
\curveto(91.49097185,106.58480942)(92.35539361,106.46145048)(92.79773204,106.21480444)
\curveto(93.23991593,105.96799555)(93.4610547,105.65958382)(93.46113432,105.28956447)
\curveto(93.4610547,104.71379041)(92.7976618,103.58294901)(91.47094155,101.89705463)
\curveto(75.95141033,82.81670149)(61.55772504,62.66726353)(48.28984822,41.44869669)
\curveto(47.36506862,39.96831273)(45.47540199,39.22812555)(42.62081088,39.22813992)
\curveto(39.72597184,39.22812555)(38.0172148,39.35149407)(37.49457722,39.5982407)
\curveto(36.12755286,40.2150402)(34.51931728,43.36081778)(32.66987047,49.03557823)
\curveto(30.57914689,55.32711903)(29.53378743,59.27475848)(29.53381085,60.87852533)
\curveto(29.53378743,62.60558406)(30.94099884,64.27099685)(33.75542165,65.87476369)
\curveto(35.48425582,66.86164481)(37.01207517,67.35509017)(38.33889376,67.35513328)
}
}

\end{pspicture}
```

استفاده از این خروجی در iTextSharp بسیار ساده است. برای مثال:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace HtmlToPdf
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

                var cb = pdfWriter.DirectContent;

                cb.MoveTo(52.73079005f, 101.89500456f);
                cb.CurveTo(31.29686559f, 101.89500456f, 13.84575258f, 84.04652127f, 13.8457479f,
62.12456369f);
                cb.CurveTo(13.8457479f, 40.20259605f, 31.29686559f, 22.35412714f, 52.73079005f,
22.35412235f);
                cb.CurveTo(74.16470983f, 22.35412235f, 91.6158322f, 40.20259605f, 91.61582751f,
62.12456369f);
                cb.CurveTo(91.61582751f, 71.60188248f, 88.48023622f, 80.07729424f, 83.15553076f,
87.02034164f);
                cb.LineTo(79.49425309f, 82.58209245f);
                cb.CurveTo(84.13622847f, 76.73639073f, 85.95313131f, 70.24630402f, 85.95313131f,
62.12456369f);
                cb.CurveTo(85.95313131f, 43.33817595f, 71.09893654f, 28.1547277f, 52.73079005f,
28.1547277f);
                cb.CurveTo(34.36263419f, 28.15473249f, 19.50844879f, 43.33817595f, 19.50844879f,
62.12456369f);
                cb.CurveTo(19.50844879f, 80.91094185f, 34.36264355f, 96.10336589f, 52.73079005f,
96.10336589f);
                cb.CurveTo(58.55122776f, 96.10336589f, 62.90459266f, 95.2476225f, 67.65721002f,
92.5630926f);
                cb.LineTo(71.13570481f, 97.23509821f);
                cb.CurveTo(65.57113223f, 100.3782653f, 59.52269945f, 101.89500456f, 52.73079005f,
101.89500456f);

                cb.MoveTo(38.33889376f, 67.35513328f);
                cb.CurveTo(39.90689547f, 67.35509017f, 41.09296342f, 66.03921993f, 41.89711165f,
63.40748424f);
                cb.CurveTo(43.50531445f, 58.47289182f, 44.65118131f, 56.00562195f, 45.33470755f,
56.0056459f);
                cb.CurveTo(45.85735449f, 56.00562195f, 46.40013944f, 56.41682961f, 46.96305772f,
57.23928802f);
                cb.CurveTo(58.2608517f, 75.74384316f, 68.7143666f, 90.71198997f, 78.32362116f,
102.14379168f);
                cb.CurveTo(80.81631349f, 105.10443984f, 84.77658911f, 106.58480942f, 90.20445269f,
106.58489085f);
                cb.CurveTo(91.49097185f, 106.58480942f, 92.35539361f, 106.46145048f, 92.79773204f,
106.21480444f);
                cb.CurveTo(93.23991593f, 105.96799555f, 93.4610547f, 105.65958382f, 93.46113432f,
105.28956447f);
                cb.CurveTo(93.4610547f, 104.71379041f, 92.7976618f, 103.58294901f, 91.47094155f,
101.89705463f);
                cb.CurveTo(75.95141033f, 82.81670149f, 61.55772504f, 62.66726353f, 48.28984822f,
41.44869669f);
                cb.CurveTo(47.36506862f, 39.96831273f, 45.47540199f, 39.22812555f, 42.62081088f,
39.22813992f);
                cb.CurveTo(39.72597184f, 39.22812555f, 38.0172148f, 39.35149407f, 37.49457722f,
39.5982407f);
                cb.CurveTo(36.12755286f, 40.2150402f, 34.51931728f, 43.36081778f, 32.66987047f,
49.03557823f);
                cb.CurveTo(30.57914689f, 55.32711903f, 29.53378743f, 59.27475848f, 29.53381085f,
60.87852533f);
                cb.CurveTo(29.53378743f, 62.60558406f, 30.94099884f, 64.27099685f, 33.75542165f,
65.87476369f);
                cb.CurveTo(35.48425582f, 66.86164481f, 37.01207517f, 67.35509017f, 38.33889376f,
67.35513328f);

                cb.SetRGBColorFill(0, 0, 0);
                cb.Fill();
            }

            Process.Start("Test.pdf");
        }
    }
}
```

در اینجا، pdfWriter.DirectContent یک Canvas را جهت ترسیمات گرافیکی در اختیار ما قرار می‌دهد. سپس مابقی هم آن مشخص است و یک تناظر یک به یک را می‌شود بین خروجی Tex و متدهای فراخوانی شده، مشاهده کرد. PDF خروجی هم به شکل زیر است:



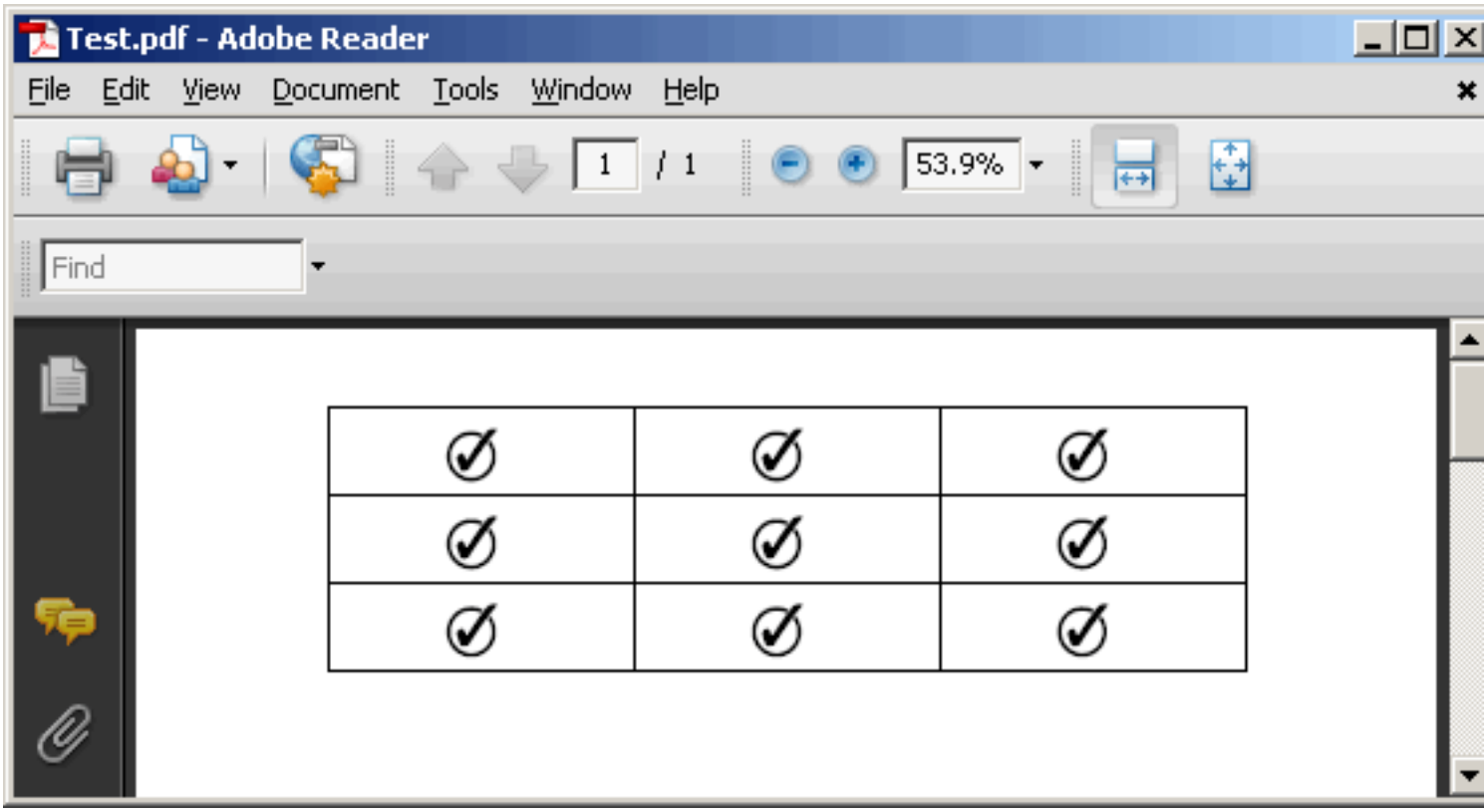
تا اینجا یک مرحله پیشرفت است. مشکل از اینجا شروع می‌شود که خوب! من که یک «چک مارک» این اندازه‌ای لازم ندارم! آن هم قرار گرفته در پایین صفحه. یک راه حل این مشکل استفاده از متد Transform شیء cb فوق است. این متد یک System.Drawing.Drawing2D.Matrix را دریافت می‌کند و سپس می‌شود توسط آن، اعمال تغییر اندازه (Scale)، تغییر مکان (Translate) و غیره را اعمال کرد. راه دیگر تعریف یک Template از دستورات فوق است. سپس متد Image.GetInstance کتابخانه iTextSharp ورودی از نوع Template را هم قبول می‌کند. خروجی حاصل یک تصویر برداری خواهد بود که اکنون با اکثر اشیاء iTextSharp سازگار است. برای مثال متد سازنده PdfPCell، آرگومان از نوع Image را هم قبول می‌کند. به علاوه شیء Image در اینجا متدهای تغییر اندازه و امثال آن‌را نیز به همراه دارد:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace HtmlToPdf
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

                var cb = pdfWriter.DirectContent;
                var template = createCheckMark(cb);

                var image = Image.GetInstance(template);
                image.ScaleAbsolute(40, 40);
```

```
                var table = new PdfPTable(3);
                var cell = new PdfPCell(image)
                {
                    HorizontalAlignment = Element.ALIGN_CENTER
                };

                for (int i = 0; i < 9; i++)
                    table.AddCell(cell);

                pdfDoc.Add(table);
            }

            Process.Start("Test.pdf");
        }

        private static PdfTemplate createCheckMark(PdfContentByte cb)
        {
            var template = cb.CreateTemplate(140, 140);

            template.MoveTo(52.73079005f, 101.89500456f);
            template.CurveTo(31.29686559f, 101.89500456f, 13.84575258f, 84.04652127f, 13.8457479f,
62.12456369f);
            template.CurveTo(13.8457479f, 40.20259605f, 31.29686559f, 22.35412714f, 52.73079005f,
22.35412235f);
            template.CurveTo(74.16470983f, 22.35412235f, 91.6158322f, 40.20259605f, 91.61582751f,
62.12456369f);
            template.CurveTo(91.61582751f, 71.60188248f, 88.48023622f, 80.07729424f, 83.15553076f,
87.02034164f);
            template.LineTo(79.49425309f, 82.58209245f);
            template.CurveTo(84.13622847f, 76.73639073f, 85.95313131f, 70.24630402f, 85.95313131f,
62.12456369f);
            template.CurveTo(85.95313131f, 43.33817595f, 71.09893654f, 28.1547277f, 52.73079005f,
28.1547277f);
            template.CurveTo(34.36263419f, 28.15473249f, 19.50844879f, 43.33817595f, 19.50844879f,
62.12456369f);
            template.CurveTo(19.50844879f, 80.91094185f, 34.36264355f, 96.10336589f, 52.73079005f,
96.10336589f);
            template.CurveTo(58.55122776f, 96.10336589f, 62.90459266f, 95.2476225f, 67.65721002f,
92.5630926f);
            template.LineTo(71.13570481f, 97.23509821f);
            template.CurveTo(65.57113223f, 100.3782653f, 59.52269945f, 101.89500456f, 52.73079005f,
101.89500456f);

            template.MoveTo(38.33889376f, 67.35513328f);
            template.CurveTo(39.90689547f, 67.35509017f, 41.09296342f, 66.03921993f, 41.89711165f,
63.40748424f);
            template.CurveTo(43.50531445f, 58.47289182f, 44.65118131f, 56.00562195f, 45.33470755f,
56.0056459f);
            template.CurveTo(45.85735449f, 56.00562195f, 46.40013944f, 56.41682961f, 46.96305772f,
57.23928802f);
            template.CurveTo(58.2608517f, 75.74384316f, 68.7143666f, 90.71198997f, 78.32362116f,
102.14379168f);
            template.CurveTo(80.81631349f, 105.10443984f, 84.77658911f, 106.58480942f, 90.20445269f,
106.58489085f);
            template.CurveTo(91.49097185f, 106.58480942f, 92.35539361f, 106.46145048f, 92.79773204f,
106.21480444f);
            template.CurveTo(93.23991593f, 105.96799555f, 93.4610547f, 105.65958382f, 93.46113432f,
105.28956447f);
            template.CurveTo(93.4610547f, 104.71379041f, 92.7976618f, 103.58294901f, 91.47094155f,
101.89705463f);
            template.CurveTo(75.95141033f, 82.81670149f, 61.55772504f, 62.66726353f, 48.28984822f,
41.44869669f);
            template.CurveTo(47.36506862f, 39.96831273f, 45.47540199f, 39.22812555f, 42.62081088f,
39.22813992f);
            template.CurveTo(39.72597184f, 39.22812555f, 38.0172148f, 39.35149407f, 37.49457722f,
39.5982407f);
            template.CurveTo(36.12755286f, 40.2150402f, 34.51931728f, 43.36081778f, 32.66987047f,
49.03557823f);
            template.CurveTo(30.57914689f, 55.32711903f, 29.53378743f, 59.27475848f, 29.53381085f,
60.87852533f);
            template.CurveTo(29.53378743f, 62.60558406f, 30.94099884f, 64.27099685f, 33.75542165f,
65.87476369f);
            template.CurveTo(35.48425582f, 66.86164481f, 37.01207517f, 67.35509017f, 38.33889376f,
67.35513328f);

            template.SetRGBColorFill(0, 0, 0);
            template.Fill();

            return template;
        }
```

```
    }
}
```

در این مثال، با کمک متد CreateTemplate مرتبط با Canvas دریافتی، یک قالب جدید ایجاد و سپس روی آن نقاشی خواهیم کرد. اکنون می‌توان از این قالب تهیه شده، یک Image دریافت کرده و سپس مثلا در سلول‌های یک جدول نمایش داد. اینبار خروجی نهایی ما به شکل زیر خواهد بود:



عنوان: **نکته‌ای تکمیلی در مورد مجوز استفاده از iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۱۲ ۱۲:۱۶:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

یکی از سؤ برداشت‌های متداول از کارهای سورس باز موجود این است:
«من مجازم از این کتابخانه‌ی سورس باز هرجایی و هر طوری که دوست دارم استفاده کنم.»

در کل این یک «توهم» بزرگ است. بسته به مجوز پروژه (^)، جمله‌ی فوق می‌تواند صحیح یا کاملا نادرست باشد.
برای نمونه من خیلی‌ها رو می‌بینم که می‌گن: «از MySQL استفاده کن که رایگانه». نه دوست عزیز؛ اشتباه می‌کنید! فقط برای کارهای سورس باز رایگان است. مجوز نگارش Community و رایگان آن در رده‌ی مجوزهای GPL است (^). به این معنا که اگر روزی مطابق قوانین کپی رایت قرار شد رفتار شود، به سراغ کار سورس بسته شما که دارد از MySQL رایگان استفاده می‌کند، خواهند آمد. جهت اطلاع!
به همین جهت کسانی که کار تجاری سورس بسته انجام می‌دهند از طرف کتابخانه‌های دارای مجوز GPL حتی رد هم نمی‌شوند؛ چه برسد به اینکه بخواهند آزادانه از آن استفاده کنند.

در مورد مجوز کتابخانه‌ی iTextSharp پیشتر مطلبی را در این سایت خوانده‌اید:
مجوز این کتابخانه، GNU Affero General Public License است. به این معنا که شما موظفید، تغییری در قسمت تهیه کننده خواص فایل PDF تولیدی که به صورت خودکار به نام کتابخانه تنظیم می‌شود، ندهید. اگر می‌خواهید این قسمت را تغییر دهید باید هزینه کنید. همچنین با توجه به اینکه این مجوز، GPL است یعنی زمانیکه از آن استفاده کردید باید کار خود را به صورت سورس باز ارائه دهید (^).

و ... نکته تکمیلی مهم اینکه:
این کتابخانه تا نگارش 4.1.7 تحت مجوز MPL/LGPL ارائه شده و «بدون مشکل» در کارهای تجاری سورس بسته قابل استفاده است. از نگارش 5 به بعد، AGPL شده و برای کارهای تجاری سورس بسته «رایگان نیست» (^).
برای نمونه سورس نسخه 4.1.7 از این آدرس قابل دریافت است.
این سورس را از پروژه " FDFToolkit .NET " اینجا نقل کردم چون تهیه کننده این پروژه دقیقا به این مطلب اشاره کرده و کار خود را به نگارش 4.1.7 کتابخانه iTextSharp عمدا محدود کرده است.

نظرات خوانندگان

نویسنده: mSafdel
تاریخ: ۱۳۹۰/۰۸/۱۲ ۱۴:۴۷:۴۷

باور کنید زبونم مو درآورد از بس این نکته رو توی فروم ها و اجتماعات مختلف گوشزد کردم. ممنون بابت تکرار مجدد این نکته مهم. صرف علاقمند بودن به Open Source کافی نیست باید در مورد اون اطلاعات هم داشت.

عنوان: **رمزنگاری فایل‌های PDF با استفاده از کلید عمومی توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۲۱ ۱۳:۴۴:۰۰
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp

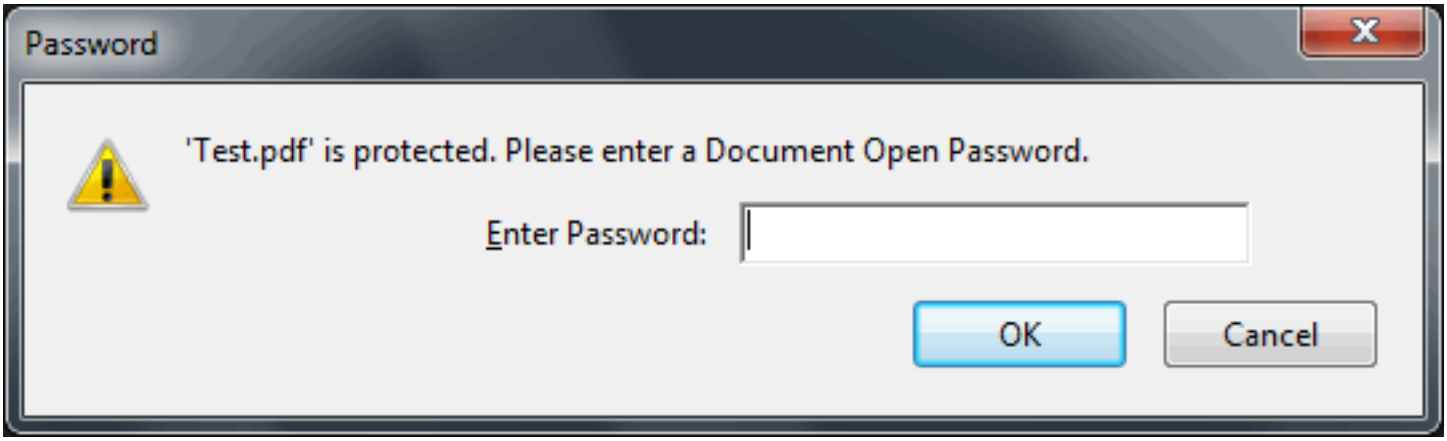
دو نوع رمزنگاری را می‌توان توسط iTextSharp به PDF تولیدی و یا موجود، اعمال کرد:
الف) رمزنگاری با استفاده از کلمه عبور
ب) رمزنگاری توسط کلید عمومی

الف) رمزنگاری با استفاده از کلمه عبور

در اینجا امکان تنظیم read password و edit password به کمک متد SetEncryption شیء pdfWrite وجود دارد. همچنین می‌توان مشخص کرد که مثلا آیا کاربر می‌تواند فایل PDF را چاپ کند یا خیر (PdfWriter.ALLOW_PRINTING).
ذکر read password اختیاری است؛ اما جهت اعمال permissions حتما نیاز است تا edit password ذکر گردد:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.Text;

namespace EncryptPublicKey
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));

                var readPassword = Encoding.UTF8.GetBytes("123");//it can be null.
                var editPassword = Encoding.UTF8.GetBytes("456");
                int permissions = PdfWriter.ALLOW_PRINTING | PdfWriter.ALLOW_COPY;
                pdfWriter.SetEncryption(readPassword, editPassword, permissions,
PdfWriter.STRENGTH128BITS);

                pdfDoc.Open();

                pdfDoc.Add(new Phrase("tst 0"));
                pdfDoc.NewPage();
                pdfDoc.Add(new Phrase("tst 1"));
            }

            Process.Start("TestEnc.pdf");
        }
    }
}
```

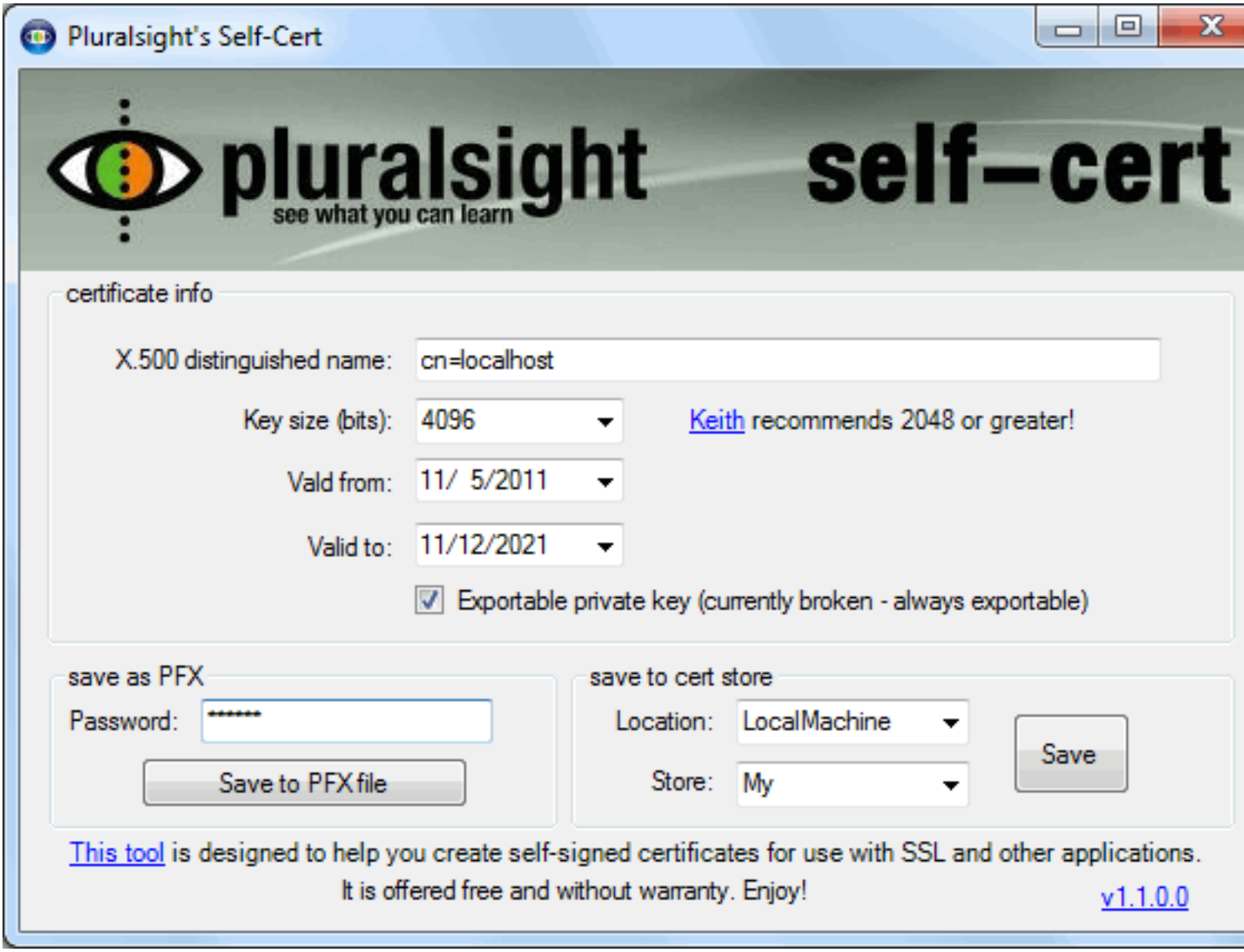
اگر read password ذکر شود، کاربران برای مشاهده محتویات فایل نیاز خواهند داشت تا کلمه‌ی عبور مرتبط را وارد نمایند:



این روش آنچنان امنیتی ندارد. هستند برنامه‌هایی که این نوع فایل‌ها را «آنی» به نمونه‌ی غیر رمزنگاری شده تبدیل می‌کنند (حتی نیازی هم ندارند که از شما کلمه‌ی عبوری را سؤال کنند). بنابراین اگر کاربران شما آنچنان حرفه‌ای نیستند، این روش خوب است؛ در غیراینصورت از آن صرفنظر کنید.

ب) رمزنگاری توسط کلید عمومی
این روش نسبت به حالت الف بسیار پیشرفته‌تر بوده و امنیت قابل توجهی هم دارد و «نیستند» برنامه‌هایی که بتوانند این فایل‌ها را بدون داشتن اطلاعات کافی، به سادگی رمزگشایی کنند.

برای شروع به کار با public key encryption نیاز است یک فایل PFX یا Personal Information Exchange داشته باشیم. یا می‌توان این نوع فایل‌ها را از CA's یا Certificate Authorities خرید، که بسیار هم نیکو یا اینکه می‌توان فعلا برای آزمایش، نمونه‌ی self signed این‌ها را هم تهیه کرد. مثلا با استفاده از این برنامه .



در ادامه نیاز خواهیم داشت تا اطلاعات این فایل PFX را جهت استفاده توسط iTextSharp استخراج کنیم. کلاس‌های زیر اینکار را انجام می‌دهند و نهایتا کلیدهای عمومی و خصوصی ذخیره شده در فایل PFX را بازگشت خواهند داد:

```
using Org.BouncyCastle.Crypto;
using Org.BouncyCastle.X509;

namespace EncryptPublicKey
{
    /// <summary>
    /// A Personal Information Exchange File Info
    /// </summary>
    public class PfxData
    {
        /// <summary>
        /// Represents an X509 certificate
        /// </summary>
        public X509Certificate[] X509PrivateKeys { set; get; }

        /// <summary>
        /// Certificate's public key
        /// </summary>
        public ICipherParameters PublicKey { set; get; }
    }
}
```

```
using System;
using System.IO;
using Org.BouncyCastle.Crypto;
using Org.BouncyCastle.Pkcs;
using Org.BouncyCastle.X509;

namespace EncryptPublicKey
{
    /// <summary>
    /// A Personal Information Exchange File Reader
    /// </summary>
    public class PfxReader
    {
        X509Certificate[] _chain;
        AsymmetricKeyParameter _asymmetricKeyParameter;

        /// <summary>
        /// Reads A Personal Information Exchange File.
        /// </summary>
        /// <param name="pfxPath">Certificate file's path</param>
        /// <param name="pfxPassword">Certificate file's password</param>
        public PfxData ReadCertificate(string pfxPath, string pfxPassword)
        {
            using (var stream = new FileStream(pfxPath, FileMode.Open, FileAccess.Read))
            {
                var pkcs12Store = new Pkcs12Store(stream, pfxPassword.ToCharArray());
                var alias = findThePublicKey(pkcs12Store);
                _asymmetricKeyParameter = pkcs12Store.GetKey(alias).Key;
                constructChain(pkcs12Store, alias);
                return new PfxData { X509PrivateKeys = _chain, PublicKey = _asymmetricKeyParameter };
            }
        }

        private void constructChain(Pkcs12Store pkcs12Store, string alias)
        {
            var certificateChains = pkcs12Store.GetCertificateChain(alias);
            _chain = new X509Certificate[certificateChains.Length];

            for (int k = 0; k < certificateChains.Length; ++k)
                _chain[k] = certificateChains[k].Certificate;
        }

        private static string findThePublicKey(Pkcs12Store pkcs12Store)
        {
            string alias = string.Empty;
            foreach (string entry in pkcs12Store.Aliases)
            {
                if (pkcs12Store.IsKeyEntry(entry) && pkcs12Store.GetKey(entry).Key.IsPrivate)
                {
                    alias = entry;
                    break;
                }
            }

            if (string.IsNullOrEmpty(alias))
                throw new NullReferenceException("Provided certificate is invalid.");

            return alias;
        }
    }
}
```

اکنون رمزنگاری فایل PDF تولیدی توسط کلید عمومی، به سادگی چند سطر کد زیر خواهد بود:

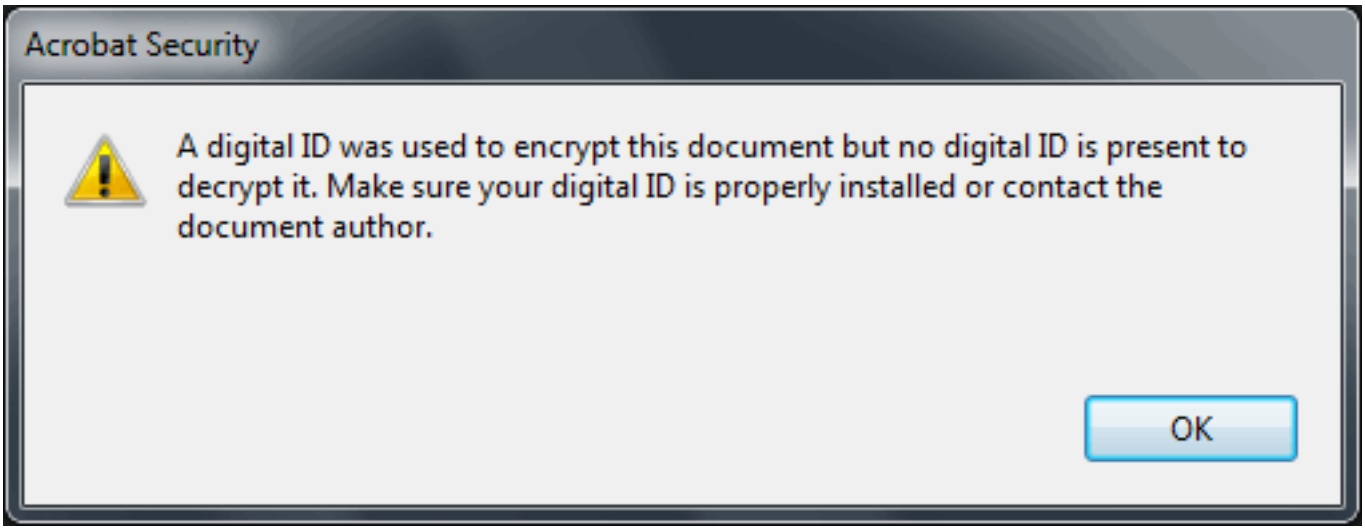
```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace EncryptPublicKey
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));

                var certs = new PfxReader().ReadCertificate(@"D:\path\cert.pfx", "123");
                pdfWriter.SetEncryption(
                          certs: certs.X509PrivateKeys,
                          permissions: new int[] { PdfWriter.ALLOW_PRINTING, PdfWriter.ALLOW_COPY },
                          encryptionType: PdfWriter.ENCRYPTION_AES_128);

                pdfDoc.Open();

                pdfDoc.Add(new Phrase("tst 0"));
                pdfDoc.NewPage();
                pdfDoc.Add(new Phrase("tst 1"));
            }

            Process.Start("Test.pdf");
        }
    }
}
```

پیش از فراخوانی متد Open باید تنظیمات رمزنگاری مشخص شوند. در اینجا ابتدا فایل PFX خوانده شده و کلیدهای عمومی و خصوصی آن استخراج می‌شوند. سپس به متد SetEncryption جهت استفاده نهایی ارسال خواهند شد.

نحوه استفاده از این نوع فایل‌های رمزنگاری شده:

اگر سعی در گشودن این فایل رمزنگاری شده نمائیم با خطای زیر مواجه خواهیم شد:



کاربران برای اینکه بتوانند این فایل‌های PDF را بار کنند نیاز است تا فایل PFX شما را در سیستم خود نصب کنند. ویندوز فایل‌های PFX را می‌شناسد و نصب آن‌ها با دوبار کلیک بر روی فایل و چندبار کلیک بر روی دکمه‌ی Next و وارد کردن کلمه عبور آن، به پایان می‌رسد.

سؤال: آیا می‌توان فایل‌های PDF موجود را هم به همین روش رمزنگاری کرد؟

بله. iTextSharp علاوه بر PdfWriter دارای PdfReader نیز می‌باشد:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace EncryptPublicKey
{
    class Program
    {
        static void Main(string[] args)
        {
            PdfReader reader = new PdfReader("TestDec.pdf");
            using (var stamper = new PdfStamper(reader, new FileStream("TestEnc.pdf",
FileMode.Create)))
            {
                var certs = new PfxReader().ReadCertificate(@"D:\path\cert.pfx", "123");
                stamper.SetEncryption(
                         certs: certs.X509PrivateKeys,
                         permissions: new int[] { PdfWriter.ALLOW_PRINTING, PdfWriter.ALLOW_COPY },
                         encryptionType: PdfWriter.ENCRYPTION_AES_128);
                stamper.Close();
            }

            Process.Start("TestEnc.pdf");
        }
    }
}
```

سؤال: آیا می‌توان نصب کلید عمومی را خودکار کرد؟

سورس برنامه SelfCert که معرفی شد، در دسترس است. این برنامه قابلیت انجام نصب خودکار مجوزها را دارد.

نظرات خوانندگان

نویسنده: مجتبی
تاریخ: ۱۳۹۰/۰۸/۲۱ ۱۵:۳۵:۲۹

خیلی عالی بود . من همیشه برام سؤال بود که این کدها چگونه ساخته میشدند . در ضمن من این کدهای pfx رو در فایلهای کتابهای " اداره کل چاپ و کتب درسی " دیده بودم . فایلهای pdf اونها برای خواندن احتیاج به نصب pfx داشتند.

نویسنده: Mojtaba Shagi
تاریخ: ۱۳۹۰/۰۸/۲۲ ۰۸:۳۴:۵۶

خیلی ممنون از مطالب خوب و مفید شما

نویسنده: K Liberal
تاریخ: ۱۳۹۰/۱۰/۰۸ ۱۴:۲۹:۰۵

سلام.دستت بابت این اطلاعات درد نکنه.
راستش من یه سری کتاب درسی دانلود کردم میخوام ادیتش کنم.چون از برنامه نویسی چیز زیادی سرم نمیشه از توضیحات بالا زیاد چیزی نفهمیدم حتی نمیدونم تو چه محیطی این دستوراتو باید نوشت.ولی واقعا به اون فایل های پی دی اف نیاز دارم اگه میشه 1 توضیحی بدین که باید چیکار کنم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۰/۰۸ ۱۷:۰۵:۳۸

ادیت کردن نیاز به رمزگشایی دارد؛ مطلب ارسالی من در اینجا «رمزنگاری» است.

برای رمزگشایی برنامه پر است در اینترنت مثلا: (^)

نویسنده: سینا
تاریخ: ۱۳۹۲/۱۱/۰۲ ۸:۳۷

با تشکر - ایا فایلی که با استفاده از PFX رمز شده در تبلت و مک بوک غیر ویندوز باز میشه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۲ ۹:۵

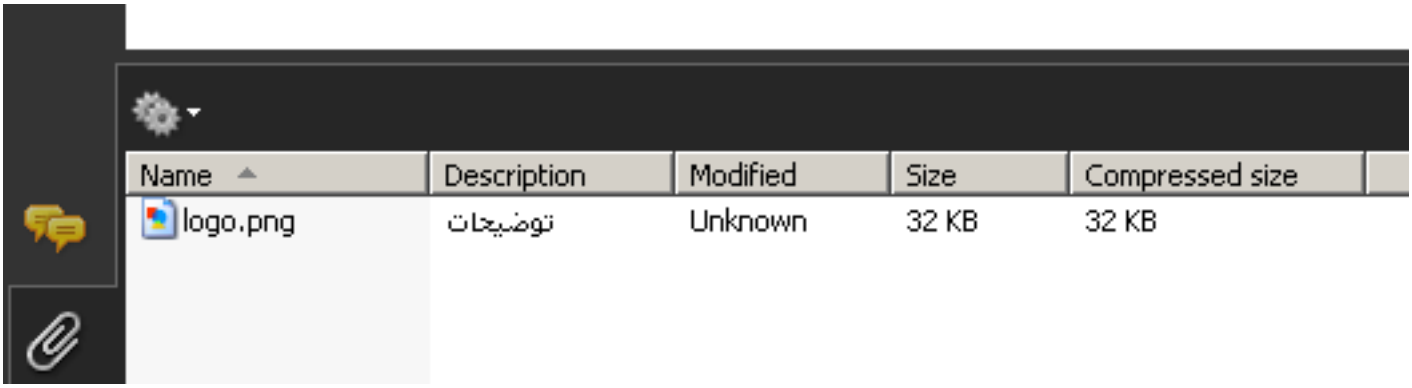
PFX منحصر به ویندوز نیست. اگر برنامه‌ها قادر به پردازش رمزنگاری‌های از نوع کلیدهای عمومی باشند، قادر به خواندن آن نیز خواهند بود. عموما برنامه‌های ساده تا این حد انواع و اقسام استانداردها را رعایت نمی‌کنند. ولی در کل امکان « حذف محدودیت‌های فایل‌های PDF توسط iTextSharp » وجود دارد تا فایل رمزنگاری شده‌ی توسط کلیدهای عمومی را بتوان تبدیل به فایل‌های غیر رمزنگاری شده کرد. پس از رفع محدودیت، برنامه‌های ضعیف PDF خوان هم قادر به گشودن آن‌ها خواهند بود.

عنوان: **اضافه کردن پیوست به فایل‌های PDF با استفاده از iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۲۹ ۲۱:۵۳:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

فایل PDF موجود عجیب و غریبی است. می‌شود به آن فایل پیوست اضافه کرد. مثلا اگر یک راهنمای آموزشی را با فرمت PDF تهیه می‌کنید، لازم نیست تا فایل‌های مرتبط با آن‌را جداگانه ارائه دهید. می‌شود تمام این‌ها را داخل همان فایل PDF مدفون کرد. روش انجام اینکار به کمک iTextSharp ساده است اما چند نکته را نیز به همراه دارد:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace PDFAttachment
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

                pdfDoc.Add(new Phrase("Test"));

                var fs = PdfFileSpecification.FileEmbedded(pdfWriter, @"C:\path\logo.png", "logo.png",
null);
                pdfWriter.AddFileAttachment("توضیحات",fs);
            }

            Process.Start("Test.pdf");
        }
    }
}
```

در ساده‌ترین حالت ممکن، با استفاده از متد AddFileAttachmen شیء PdfWriter می‌توان پیوستی را به یک فایل PDF در حال تولید اضافه کرد. اگر به فایل نهایی مراجعه کنیم و همچنین قسمت attachments را هم دستی در Adobe reader انتخاب نمائیم، شکل زیر حاصل خواهد شد:



روش متداول بکارگرفته شده دو مشکل را به همراه دارد:
قسمت modified مقدار دهی نشده است.
پنل مربوط به پیوست‌ها باید دستی باز شود.

نحوه مقدار دهی ستون modified پس از تعریف یک PdfDictionary و قرار دادن PdfName.MODDATE در آن، به صورت زیر است:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace PDFAttachment
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

                pdfDoc.Add(new Phrase("Test"));

                var filePath = @"C:\path\logo.png";
                var fileInfo = new FileInfo(filePath);
                var pdfDictionary = new PdfDictionary();
                pdfDictionary.Put(PdfName.MODDATE, new PdfDate(fileInfo.LastWriteTime));
                var fs = PdfFileSpecification.FileEmbedded(pdfWriter, filePath, fileInfo.Name, null,
true, null, pdfDictionary);
                pdfWriter.AddFileAttachment("توضیحات", fs);
            }

            Process.Start("Test.pdf");
        }
    }
}
```

که اینبار خروجی زیر را به همراه دارد:

و برای نمایش خودکار پنل پیوست‌ها در Adobe reader به طوری که کاربر نهایی متوجه وجود این فایل‌های پیوست شده گردد، می‌توان ViewerPreferences شیء pdfWriter را مقدار دهی نمود:

```
pdfWriter.ViewerPreferences = PdfWriter.PageModeUseAttachments;
```

در مورد فایل‌های موجود چطور؟ آیا می‌توان به یک فایل PDF از پیش تهیه شده هم فایل پیوست کرد؟
پاسخ: بله. باید از امکانات شیء PdfReader استفاده کرد:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace PDFAttachment
{
    class Program
    {
        static void Main(string[] args)
        {
            var reader = new PdfReader("Test.pdf");
            using (var stamper = new PdfStamper(reader, new FileStream("newTest.pdf",
FileMode.Create)))
            {
                var filePath = @"C:\path\logo.png";
                addAttachment(stamper, filePath, "توضیحات");
                stamper.Close();
            }

            Process.Start("newTest.pdf");
        }

        private static void addAttachment(PdfStamper stamper, string filePath, string description)
        {
            var fileInfo = new FileInfo(filePath);
            var pdfDictionary = new PdfDictionary();
            pdfDictionary.Put(PdfName.MODDATE, new PdfDate(fileInfo.LastWriteTime));
            var pdfWriter = stamper.Writer;
            var fs = PdfFileSpecification.FileEmbedded(pdfWriter, filePath, fileInfo.Name, null, true,
null, pdfDictionary);
            stamper.AddFileAttachment(description, fs);
        }
    }
}
```

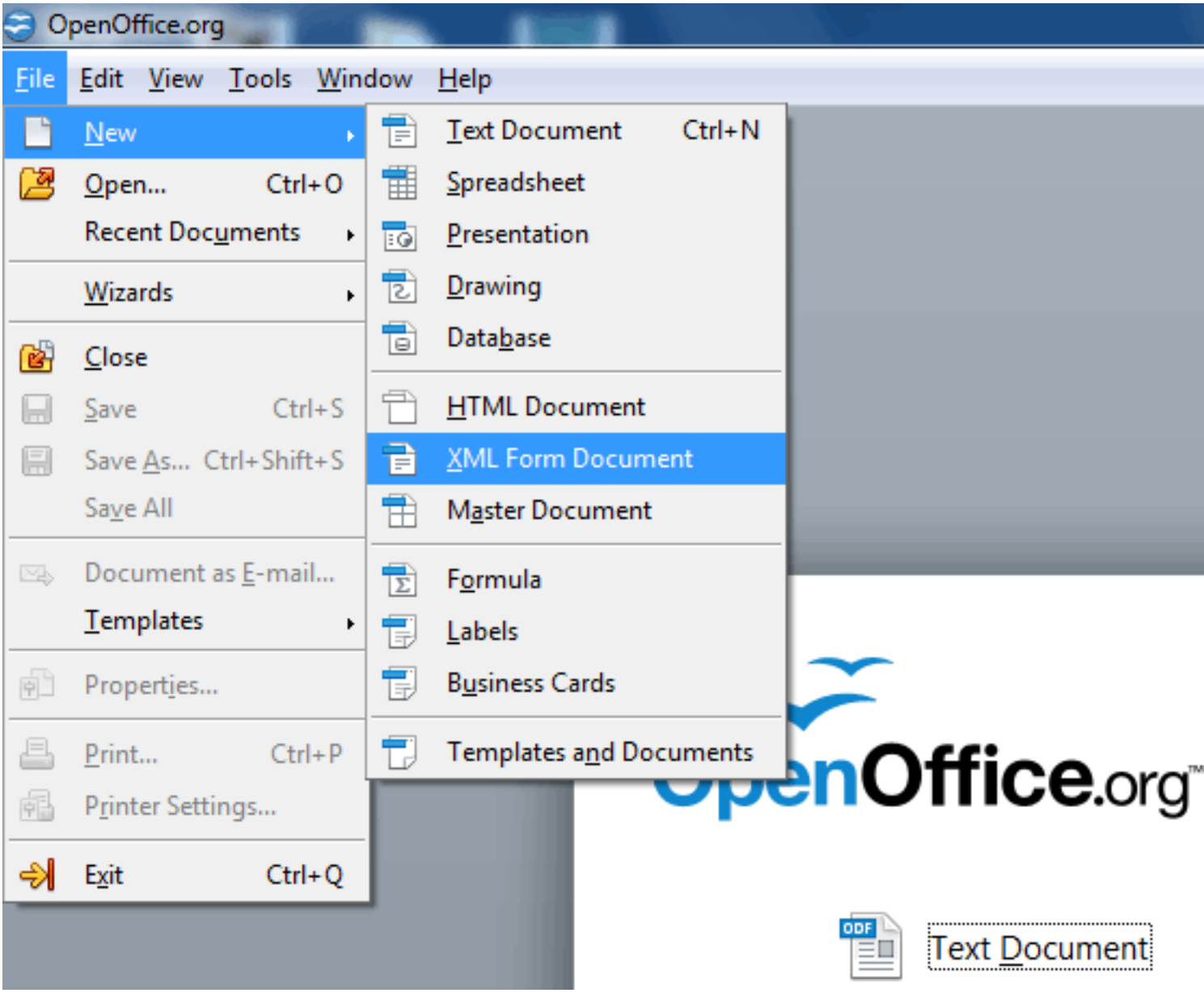
در اینجا به کمک کلاس PdfReader، یک فایل موجود خوانده شده و سپس با استفاده از امکانات کلاس PdfStamper که خاصیت Writer آن همان pdfWriter است می‌توان فایل مورد نظر را به فایل موجود افزود.

عنوان: **ساخت یک گزارش ساز به کمک iTextSharp و Open Office**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۰۱ ۰۰:۱۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

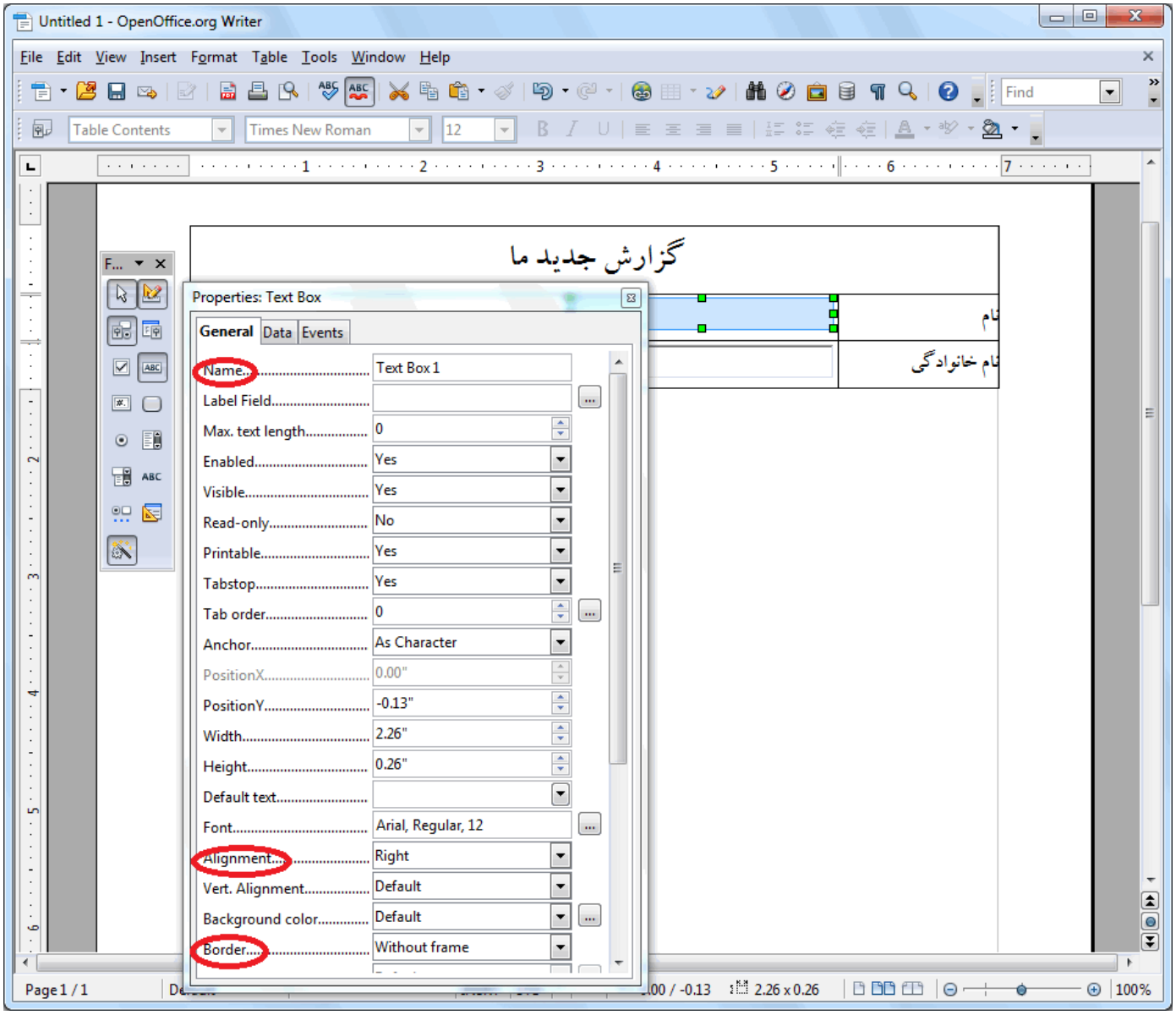
iTextSharp پایه کار با فایل‌های PDF را ارائه می‌دهد اما ابزاری را جهت ساده‌تر سازی تولید فایل‌های PDF به همراه ندارد؛ هر چند مثلا امکان تبدیل HTML به PDF را دارا است اما باید گفت: «تا حدودی البته». اگر نیاز باشد جدولی را ایجاد کنیم باید کد نویسی کرد، اگر نیاز باشد تصویری اضافه شود به همین ترتیب و الی آخر. البته این را هم باید در نظر داشت که کد نویسی انعطاف قابل توجهی را در اختیار برنامه نویس قرار می‌دهد؛ شاید به همین دلیل این روزها مباحث «Code first» بیشتر مورد توجه برنامه نویس‌ها است، تا مباحث «Wizard first» یک دهه قبل!
اما باز هم داشتن یک طراح بد نیست و می‌تواند در کاهش مدت زمان تولید نهایی یک فایل PDF مؤثر باشد. برای این منظور می‌توان از برنامه‌ی رایگان و معروف Open office استفاده کرد. توسط آن می‌توان یک فرم PDF را طراحی و سپس فیلدهای آن‌را (این قالب تهیه شده را) با iTextSharp پر کرد. این مورد می‌تواند برای تهیه گزارش‌هایی که تهیه آن‌ها با ابزارهای متداول گزارش سازی عموما میسر نیست، بسیار مناسب باشد.

طراحی یک فرم PDF با استفاده از برنامه Open Office

آخرین نگارش برنامه Open office را از اینجا می‌توانید دریافت کنید و آنچنان حجمی هم ندارد؛ حدودا 154 مگابایت است.
پس از نصب و اجرای برنامه، حداقل به دو طریق می‌توان یک فرم جدید را شروع کرد:
الف) آغاز یک XML Form document جدید در Open office سبب خواهد شد که نوارهای ابزار طراحی فرم، مانند قرار دادن CheckBox ، TextBox و غیره به صورت خودکار ظاهر شوند.
ب) و یا آغاز یک سند معمولی و سپس مراجعه به منوی View->Toolbars->Form Controls هم همان حالت را به همراه خواهد داشت.

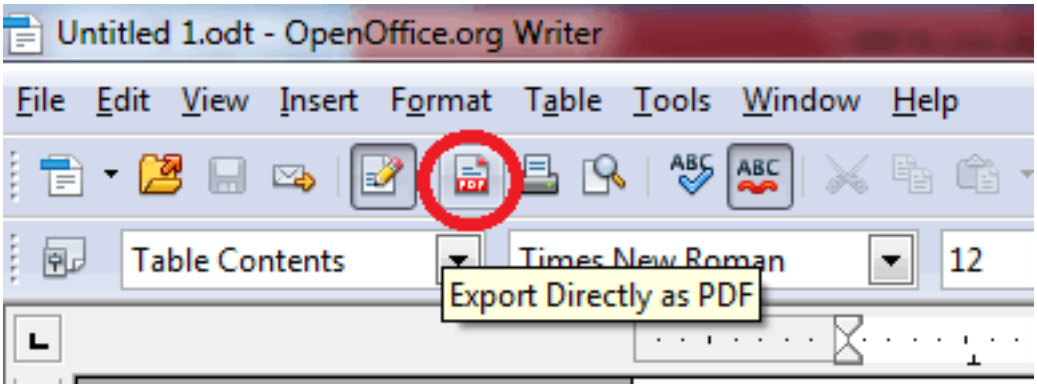


در اینجا برای طراحی یک گزارش یا فرم جدید تنها کافی است همانند روش‌های متداول تهیه یک سند معمولی رفتار کنیم و مواردی را که قرار است توسط iTextSharp مقدار دهی کنیم، با کنترل‌های نوار ابزار Form آن بر روی صفحه قرار دهیم که نمونه‌ی ساده آن‌را در شکل زیر ملاحظه می‌کنید:



برای گزارش‌های فارسی بهتر است Alignment یک کنترل به Right تنظیم شود و Border به حالت Without frame مقدار دهی گردد. نام این کنترل را هم بخاطر بسپارید و یا تغییر دهید. از این نام‌ها در iTextSharp استفاده خواهیم کرد. (صفحه خواص فوق با دوبار کلیک بر روی یک کنترل قرار گرفته بر روی فرم ظاهر می‌شود)

مرحله بعد، تبدیل این فرم به فایل PDF است. کلیک بر روی دکمه تهیه خروجی به صورت PDF در نوار ابزار اصلی آن برای اینکار کفایت می‌کند. این گزینه در منوی File نیز موجود است.



فرم‌های PDF تهیه شده در اینجا، فقط خواندنی هستند. مثلا یک کاربر می‌تواند آن‌ها را پر کرده و چاپ کند. اما ما از آن‌ها در ادامه به عنوان قالب گزارشات استفاده خواهیم کرد. بنابراین جهت ویرایش فرم‌های تهیه شده بهتر است فایل‌های اصلی Open Office مرتبط را نیز درجایی نگهداری کرد و هر بار پس از ویرایش، نیاز است تا خروجی جدید PDF آن‌ها تهیه شود.

استفاده از iTextSharp جهت مقدار دهی فیلدهای یک فرم PDF

در ادامه می‌خواهیم این قالب گزارشی را که تهیه کردیم با کمک iTextSharp پر کرده و یک فایل PDF جدید تهیه کنیم. سورس کامل اینکار را در ذیل مشاهده می‌کنید:

```
using System;
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace PdfForm
{
    class Program
    {
        //روش صحیح ثبت و معرفی فونت در این کتابخانه
        public static iTextSharp.text.Font GetTahoma()
        {
            var fontName = "Tahoma";
            if (!FontFactory.IsRegistered(fontName))
            {
                var fontPath = Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\tahoma.ttf";
                FontFactory.Register(fontPath);
            }
            return FontFactory.GetFont(fontName, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
        }

        static void Main(string[] args)
        {
            string fileNameExisting = @"form.pdf";
            string fileNameNew = @"newform.pdf";

            using (var existingFileStream = new FileStream(fileNameExisting, FileMode.Open))
            using (var newFileStream = new FileStream(fileNameNew, FileMode.Create))
            {
                var pdfReader = new PdfReader(existingFileStream);
                using (var stamper = new PdfStamper(pdfReader, newFileStream))
                {
                    //نکته مهم جهت کار با اطلاعات فارسی
                    //در غیراینصورت شاهد ثبت اطلاعات نخواهید بود
                    stamper.AcroFields.AddSubstitutionFont(GetTahoma().BaseFont);

                    //form.Fields.Keys = تمام فیلدهای موجود در فرم
                    var form = stamper.AcroFields;

                    //مقدار دهی فیلدهای فرم
                    form.SetField("TextBox1", "مقدار1");
```

```
                    form.SetField("TextBox2", "مقدار2");

                    // "Yes" and "Off" are valid values here
                    form.SetField("Check Box 1", "Yes");

                    // "" and "Off" are valid values here
                    form.SetField("Option Button 1", "");

                    // نحوه مقدار دهی لیست
                    form.SetListOption("ListBox1", new[] { "مقدار یک", "مقدار دو11" }, null);
                    form.SetField("ListBox1", null);

                    // به این ترتیب فرم دیگر توسط کاربر قابل ویرایش نخواهد بود
                    //stamper.PartialFormFlattening --> جهت غیرقابل ویرایش نمودن فیلدی مشخص
                    stamper.FormFlattening = true;

                    stamper.Close();
                    pdfReader.Close();
                }
            }

            Process.Start("newform.pdf");
        }
    }
}
```

توضیحات:

چون در اینجا فایل PDF، از پیش تهیه شده است، پس باید از اشیاء PdfReader و PdfStamper جهت خواندن و نوشتن اطلاعات در آن‌ها استفاده کرد. سپس توسط شیء stamper.AcroFields می‌توان به این فیلدها یا همان کنترل‌هایی که در برنامه‌ی Open office بر روی فرم قرار دادیم، دسترسی پیدا کنیم.

در ابتدا نیاز است فونت این فیلدها توسط متد AddSubstitutionFont مقدار دهی شود. این مورد برای گزارش‌های فارسی الزامی است؛ در غیراینصورت متنی را در خروجی مشاهده نخواهید کرد.

ادامه کار هم مشخص است. توسط متد form.SetField مقداری را به کنترل‌های قرار گرفته بر روی فرم نسبت می‌دهیم. آرگومان اول آن نام کنترل است و آرگومان دوم، مقدار مورد نظر می‌باشد. اگر کنترل CheckBox را بر روی صفحه قرار دادید، تنها مقدارهای Yes و Off را می‌پذیرد (آن هم با توجه به اینکه به کوچکی و بزرگی حروف حساس است). اگر یک Radio button یا در اینجا Option button را بر روی فرم قرار دادید، تنها مقدارهای خالی و Off را قبول خواهد کرد. نحوه‌ی مقدار دهی یک لیست هم در اینجا ذکر شده است.

در پایان چون نمی‌خواهیم کاربر نهایی قادر به ویرایش اطلاعات باشد، FormFlattening را true خواهیم کرد و به این ترتیب، کنترل‌ها فقط خواندنی خواهند شد. البته اگر همانطور که ذکر شد، border کنترل‌ها را در حین طراحی حذف کنید، PDF نهایی تولیدی یکپارچه و یک دست به نظر می‌رسد و اصلا مشخص نخواهد بود که این فایل پیشتر یک فرم قابل پر کردن بوده است.

نظرات خوانندگان

نویسنده: باربد
تاریخ: ۱۳۹۱/۰۷/۱۰ ۱۰:۸

سلام
آیا میشه از طریق همین ادیتور ، پارامتر تصویری هم به فایل اضافه کرد؟
(مثلا عکس شخص ...)
- یک مشکل : وقتی Open office را دانلود میکنم ، موقع اجرا پیغام میده که مشکل داره و کامل دانلود نشده ، در صورتیکه اینجوری نیست . (حجم فایلش هم 130 MB هست)
سپاسگزارم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۰ ۱۰:۵۶

- در مورد اجرا نشدن برنامه نصاب نظری ندارم. عموما این فایل‌ها دارای یک امضای دیجیتال md5 یا sha1 منتشر شده در سایت اصلی هم هستند. مقایسه کنید آیا کامل دریافت شده یا نه.
- در مورد تصویر، می‌تونید از روش‌های متداول iTextSharp استفاده کنید. PDF در اصل یک قالب برداری است. شما یک Canvas دارید که می‌تونید در هر جایی از آن هر شیءایی را قرار دهید. برای نمونه در مثال فوق:

```
PdfContentByte content = stamper.GetOverContent(pdfReader.NumberOfPages);
Image image = Image.GetInstance(imagePath);
image.SetAbsolutePosition(450,650);
image.ScaleAbsolute(200,200);
content.AddImage(image);
```

شما به کمک stamper دسترسی به این Canvas پیدا می‌کنید. سپس در هر مختصات دلخواهی مطابق کدهای فوق، تصویر مورد نظر را قرار دهید.

نویسنده: باربد
تاریخ: ۱۳۹۱/۰۷/۱۰ ۱۱:۳۷

سپاس آقای نصیری

نویسنده: M.B
تاریخ: ۱۳۹۱/۰۷/۲۸ ۱۰:۵۱

با سلام، من زمانی که کدهای مربوطه را می‌نویسم و فرم رو اجرا می‌کنم بلافاصله یک فایل PDF برای من باز میشه که در اون کلمه آزمایش نوشته شده است کدهای من به صورت زیر هستند

```
public static Font GetTahoma()
    {
        var fontName = "Tahoma";
        if (!FontFactory.IsRegistered(fontName))
        {
            var fontPath = Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.ttf";
            FontFactory.Register(fontPath);
        }
        return FontFactory.GetFont(fontName, BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
    }
```

```
string fileNameExisting = @"Test.pdf";
        string fileNameNew = @"newform.pdf";
         using (var existingFileStream = new FileStream(fileNameExisting, FileMode.Open))
            using (var newFileStream = new FileStream(fileNameNew, FileMode.Create))
```

```
{
    var pdfReader = new PdfReader(existingFileStream);
    using (var stamper = new PdfStamper(pdfReader, newFileStream))
    {
        //نکته مهم جهت کار با اطلاعات فارسی
        //در غیراینصورت شاهد ثبت اطلاعات نخواهید بود
        stamper.AcroFields.AddSubstitutionFont(GetTahoma().BaseFont);

        //form.Fields.Keys = تمام فیلدهای موجود در فرم
        var form = stamper.AcroFields;

        //مقدار دهی فیلدهای فرم
        form.SetField("Text1", "مقدار1");
         form.SetField("Text2", "مقدار2");
         form.SetField("Text3", "مقدار3");
         form.SetField("Text4", "مقدار4");
         form.SetField("Text5", "مقدار5");
         form.SetField("Text6", "مقدار6");


        // به این ترتیب فرم دیگر توسط کاربر قابل ویرایش نخواهد بود
        //stamper.PartialFormFlattening --> جهت غیرقابل ویرایش نمودن فیلدی مشخص
        stamper.FormFlattening = true;

        stamper.Close();
        pdfReader.Close();
    }
}

Process.Start("newform.pdf");
```

و محتوای فایل Test.PDF

شماره پرسنلی		شماره شناسنامه	
نام و نام خانوادگی		کد ملی	
نام پدر		مقطع	

و محتوای فایل جدید که برای من ایجاد می‌کند

آزمایش

ممنون .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۸ ۱۱:۲۱

در سیستم شما تداخل وجود دارد. کلمه «آزمایش» متعلق به فایل قبلی دیگری است که دارید.
فایل‌های آزمایشی خروجی PDF موجود را کلا حذف کنید و بعد کدهای فوق را اجرا کنید.
Process.Start را هم حذف کرده و خروجی را دستی و خارج از VS.NET بررسی کنید.

نویسنده: پویا امینی
تاریخ: ۱۳۹۱/۰۷/۲۸ ۱۱:۴۰

ببخشید آقای نصیری من وقتی Process.Start رو حذف کنم کجا می‌تونمم محتوای فایل NewForm.PDF رو ببینم؟ چون درون Sloution چنین فایلی برای من ایجاد نمی‌شود.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۸ ۱۱:۵۳

معمولا درون پوشه bin\debug یا bin\release تشکیل می‌شود.
اما جهت اطمینان می‌تونید مسیر دهی کامل کنید:

```
string fileNameExisting = @"c:\Test.pdf";
string fileNameNew = @"c:\newform.pdf";
```

نویسنده: پویا امینی
تاریخ: ۱۳۹۱/۰۷/۲۹ ۰:۳۲

خیلی کارت درسته جناب نصیری، من اومدم مسیر دهی خودم رو به صورت زیر انجام دادم

```
string fileNameExisting =HttpRuntime.AppDomainAppPath+ @"TestOpenOffice.pdf";
 string fileNameNew =HttpRuntime.AppDomainAppPath+ @"newform.pdf";
```

و Process.Start

```
Process.Start(HttpRuntime.AppDomainAppPath+"newform.pdf");
```

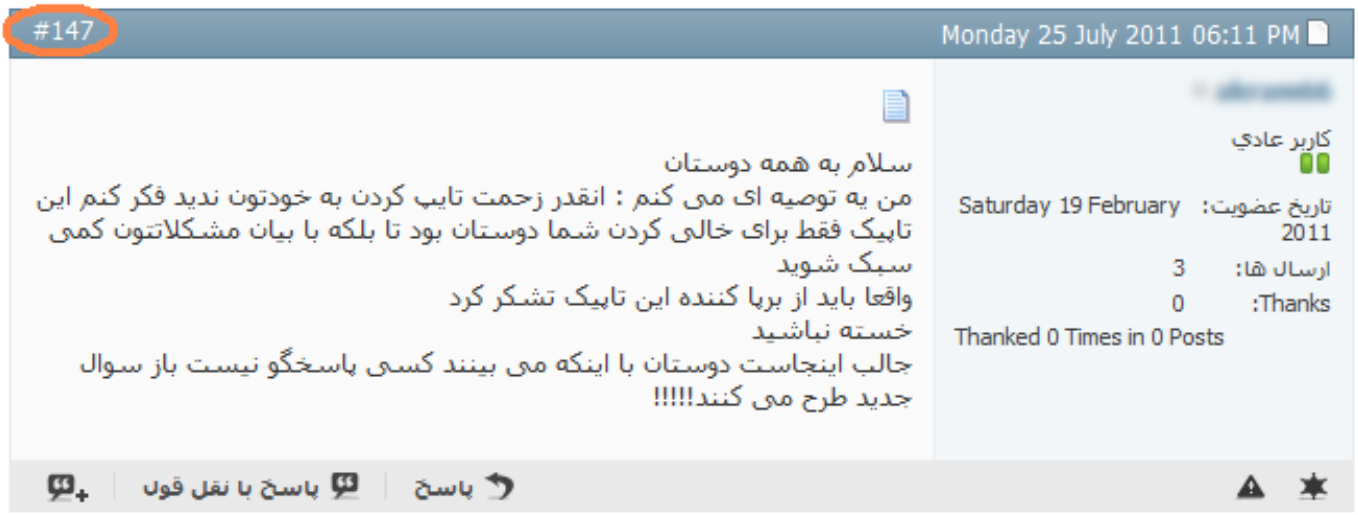
یه دنیا ممنون جناب نصیری

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۹ ۰:۴۵

اگر برنامه ویندوزی است بهتره از Application.StartupPath استفاده کنید (تعریف شده در اسمبلی System.Windows.Forms).
اگر برنامه وب است، از Server.MapPath استفاده کنید.

عنوان: **شرح حال ابزارهای گزارشگیری موجود**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۱۲ ۲۳:۱۸:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

مدتی هست که در حال تهیه یک کتابخانه گزارشگیری بر پایه iTextSharp هستم. برای تهیه backlog هم چه جایی بهتر از بررسی سؤالات موجود در انجمن‌ها؛ چیزی مثل این:



بله، تاپیکی با 13 صفحه که حتی یک مورد از درخواست‌های آن هم دارای پاسخ نبود؛ اما باز هم کاربران با علاقه هرچه تمام‌تر یا می‌دونید، از روی عجز درخواستشون رو مطرح می‌کردند و کسی نبود که جواب بده. حقیقتش این است که مشکل از افراد نیست یا اینکه «کسی نبود» یا «کسی نخواست» که جواب بده. مشکل این است که اکثر برنامه‌های گزارشگیری یا گزارش سازی موجود در حد یک Demo ware هستند. «نمی‌تونند» با مشکلات واقعی کاری موجود (در طی 13 صفحه که ذکر شد) راحت کنار بیان و راه حل بدرد بخوری رو ارائه بدن.

نظرات خوانندگان

نویسنده: Salar Khalilzadeh
تاریخ: ۱۳۹۰/۰۹/۱۳ ۰۸:۵۲:۳۹

در مورد این iTextSharp نمی دونم اما با ابزارهای گزارش سازی زیادی کار کردم، منجمله CrystalReport, FastReport, Telerik Reporting, QuickReport و چند تا دیگه
اما هیچ کدوم به پای DevExpress XtraReports نمی رسند! این رو بدون اغراق می گم. تنها مشکلش عدم پشتیبانی از راست به چپ هست که چون مشتریش کمه برا راست به چپ پیاده سازی نمی کنند.

واقعا اگر امکانش بود قیمت یک میلیونیش رو هم می دادیم.
با استفاده از این کامپوننت یک برنامه گزارش سازی پویا تهیه کردم، امکاناتی که اون می داد فراتر از اونی هستند که تا حالا دیده بودم.

اما ابزارهای گزارش گیری اوپن سورس شاید زمان زیادی لازم باشه تا به قدرت ابزارهای تجاری برسند. البته این نظر منه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۱۳ ۱۴:۰۴:۴۹

لبته این iTextSharp فقط یک Pdf Writer خام هست. برای گزارشگیری و گزارش سازی ابزاری رو نداره ولی ... میشه برفراز آن خیلی کارها رو میسر کرد.
من منهای طراح گرافیکی DevExpress XtraReports که ذکر کردید، مابقی امکاناتش رو تا الان با iTextSharp پیاده سازی کردم. به نظرم نیازی هم به طراح ندارد. روش Code first هست. البته فقط خروجی PDF داره. با پشتیبانی کامل فارسی و راست به چپ.
اصلا برای راست به چپ درستش کردم!
این یک نمونه خروجی Dynamic crosstab ایی است که چند وقت قبل در اینجا (^) در موردش توضیح دادم. فکر نمی‌کنم هیچکدوم از ابزارهای موجود بتونند از یک کوئری LINQ و اون هم Dynamic یک خروجی به این شکل رو تولید کنند : (^)

نویسنده: Salar Khalilzadeh
تاریخ: ۱۳۹۰/۰۹/۱۳ ۱۵:۵۸:۱۰

مثال PivotGrid که البته با asp.net است
http://demos.devexpress.com/xtrareportsdemos/ReportControls/XRPivotGrid.aspx
هیچ کد نویسی لازم ندارید! فقط یک کنترل PivotGrid رو تو فرم قرار دهید و گرید رو طراحی کنید، نتیجه مانند دمو میشه!
لیست امکاناتش.
http://documentation.devexpress.com/#XtraReports/CustomDocument2161

یک امکان بسیار جالب که این گزارش گیری داره امکان قرار دادن کنترل های معمولی رو گزارش هست، و همچنین امکان دسترسی به تمامی کنترل های از دورن کد برنامه.
تصور می کنم که اگه یک کنترل PivotGrid معمولی (pivotGrid گزارش امکان تولید خودکار ستون ها رو نداره) رو در گزارش قرار بدیم و datasourceرو برابر خروجی اون تابع در پست بگذاریم، نتیجه مورد نظر که گفتید بدست میاد.

مطالب شما هم همیشه مورد استفاده و بسیار کاربردی بوده، بابت زحماتتون ازتون تشکر می کنم :)

نویسنده: shahin kiassat
تاریخ: ۱۳۹۰/۱۰/۰۶ ۱۱:۱۶:۱۳

سلام.
آقای نصیری نظرتون درباره ی این پروژه چیه ؟

/http://doddlereport.codeplex.com

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۰/۰۶ ۱۲:۳۰:۴۱

سلام؛ برای شروع و ایده گرفتن خوبه؛ ولی جای کار زیاد داره. حداقل خروجی PDF اون درکی از راست به چپ نداره و نیاز به اصلاحات اساسی داره.

عنوان: **نمایش گرادیان در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۱۵ ۱۳:۵۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

کتابخانه iTextSharp نمایش گرادیانی از رنگ‌ها را هم پشتیبانی می‌کند و بدیهی است این نمایش برداری است. روش استفاده از آن هم بسیار ساده است؛ مثلا:

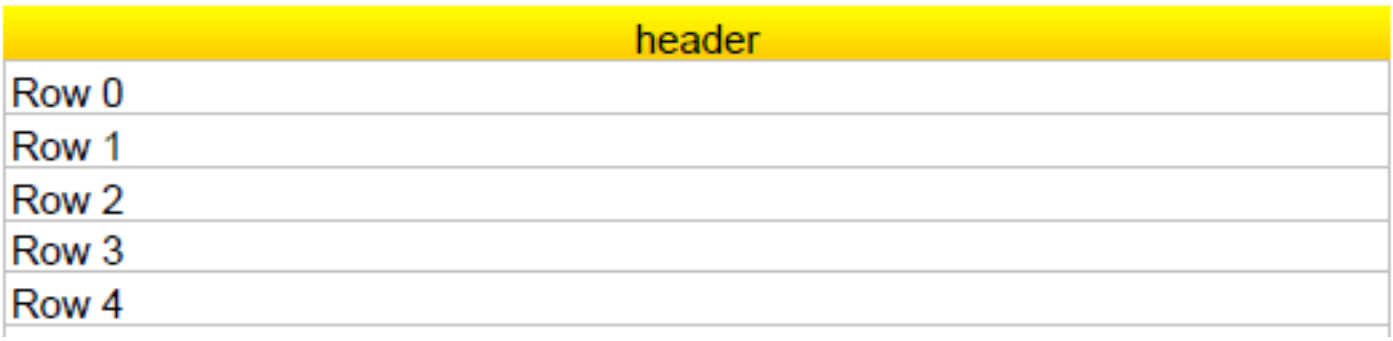
```
PdfShading shading = PdfShading.SimpleAxial(pdfWriter, x0, y0, x1, y1, BaseColor.YELLOW,
BaseColor.RED);
PdfShadingPattern pattern = new PdfShadingPattern(shading);
ShadingColor color = new ShadingColor(pattern);
```

متد PdfShading.SimpleAxial بر اساس شیء PdfWriter که توسط آن به Canvas صفحه دسترسی پیدا می‌کند، در مختصاتی مشخص، یک طیف رنگی را ایجاد می‌کند. بر این اساس می‌توان به یک ShadingColor هم رسید که از آن مثلا به عنوان BackgroundColor یک PdfPCell قابل استفاده است.
تا اینجا ساده به نظر می‌رسد اما واقعیت این است که مختصات ذکر شده، مهم است و آن‌را در مورد مثلا سلول‌های یک جدول تنها در زمان تهیه نهایی یک جدول می‌توان به دست آورد. البته اگر شیء ساده‌ای را روی صفحه رسم کنیم، این مورد بر اساس مختصات ابتدایی شیء واضح به نظر می‌رسد.
برای حل این مشکل در مورد جداول و سلول‌های آن خاصیتی به نام CellEvent وجود دارد که از نوع IPdfPCellEvent است. به عبارتی با ارسال یک وهله از کلاسی که اینترفیس IPdfPCellEvent را پیاده سازی می‌کند، می‌توان در زمان نمایش نهایی یک سلول به مختصات دقیق آن دسترسی پیدا کرد.
یک مثال کامل را در مورد پیاده سازی IPdfPCellEvent و استفاده از آن جهت نمایش گرادیان در Header و footer یک جدول، در ادامه مشاهده خواهید نمود:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace iTextSharpGradientTest
{
    public class GradientCellEvent : IPdfPCellEvent
    {
        public void CellLayout(PdfPCell cell, Rectangle position, PdfContentByte[] canvases)
        {
            var cb = canvases[PdfPTable.BACKGROUNDCANVAS];
            cb.SaveState();

            var shading = PdfShading.SimpleAxial(
                                cb.PdfWriter,
                                position.Left, position.Top, position.Left, position.Bottom,
                                BaseColor.YELLOW, BaseColor.ORANGE);
            var shadingPattern = new PdfShadingPattern(shading);

            cb.SetShadingFill(shadingPattern);
            cb.Rectangle(position.Left, position.Bottom, position.Width, position.Height);
            cb.Fill();

            cb.RestoreState();
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
```

```
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

                var table1 = new PdfPTable(1);
                table1.HeaderRows = 2;
                table1.FooterRows = 1;

                //header row
                var headerCell = new PdfPCell(new Phrase("header"))
                {
                    HorizontalAlignment = Element.ALIGN_CENTER,
                    Border = 0
                };
                headerCell.CellEvent = new GradientCellEvent();
                table1.AddCell(headerCell);

                //footer row
                var footerCell = new PdfPCell(new Phrase("footer"))
                {
                    HorizontalAlignment = Element.ALIGN_CENTER,
                    Border = 0
                };
                footerCell.CellEvent = new GradientCellEvent();
                table1.AddCell(footerCell);

                //adding some rows
                for (int i = 0; i < 70; i++)
                {
                    var rowCell = new PdfPCell(new Phrase("Row " + i)) { BorderColor =
BaseColor.LIGHT_GRAY };
                    table1.AddCell(rowCell);
                }

                pdfDoc.Add(table1);
            }

            //open the final file with adobe reader for instance.
            Process.Start("Test.pdf");
        }
    }
}
```

با خروجی:



نکته مهم این مثال نحوه مقدار دهی CellEvent است. به این ترتیب در زمان نمایش نهایی یک سلول می‌توان در متد CellLayout، به خواص فقط خواندنی آن سلول دسترسی یافت. مثلا position، مختصات نهایی مستطیل مرتبط با سلول جاری را بر می‌گرداند؛ یا از طریق canvases می‌توان برای آخرین بار فرصت یافت تا در یک سلول نقاشی کرد.

نظرات خوانندگان

نویسنده: alireza
تاریخ: ۱۳۹۰/۰۹/۱۹ ۰۹:۲۸:۱۸

با سلام و خسته نباشید ، از این کلاس من در سایت خودم برای تبدیل مقالات به فرمت pdf استفاده کردم متون فارسی را تبدیل به حروف ناخوانا می کند در این آدرس این مشکل را می توانید مشاهده نمایید :
http://article.kiansoftware.com/article.aspx?id=117&idauthore=1
راه حلش رو اگر توضیح بفرمایید ممنون می شم . با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۹/۱۹ ۰۹:۴۹:۴۲

لطفا مراجعه کنید به برچسب iTextSharp که در کنار صفحه هست. چند مقاله در این مورد منتشر کردم:
فارسی نویسی و iTextSharp
روش صحیح تعریف قلم در iTextSharp
تبدیل HTML به PDF با استفاده از کتابخانه‌ی iTextSharp
iTextSharp و استفاده از قلم‌های محدود فارسی

عنوان: **تعیین تعداد ردیف در صفحه جداول خودکار iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۰/۱۴ ۱۴:۲۰:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

پیشنیاز : « تکرار خودکار سرستون‌های یک جدول در صفحات مختلف، توسط iTextSharp »
همانطور که در مطلب پیشنیاز عنوان شده ذکر گردید، iTextSharp امکان درج خودکار header و footer به علاوه محاسبه خودکار تعداد ردیف‌های یک جدول در یک صفحه را بر اساس طول و اندازه محتوای هر ردیف، دارد. برای مثال یک صفحه ممکن است 2 ردیف شود و یک صفحه 20 ردیف. تمام این‌ها را به صورت خودکار محاسبه می‌کند و بسیار عالی است. (این امکان مهمی است که خیلی از ابزارهای گزارشگیری موجود هنوز با آن مشکل دارند)
اما اگر فرض را بر این بگذاریم که اندازه سلول‌ها و در نتیجه طول هر ردیف ثابت است و مثلا تمام صفحات نهایتا از یک تعداد ردیف مشخص تشکیل خواهند شد، خاصیتی را به نام number of rows یا rows count و امثال آن‌را ندارد که مثلا به آن گفت، من در هر صفحه فقط 5 ردیف را می‌خواهم نمایش دهم و نه 20 ردیف را.
روش حل این مساله را در ادامه ملاحظه خواهید کرد و یک نکته‌ی خیلی ساده و مستند نشده دارد!

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace RowsCountSample
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

                var table1 = new PdfPTable(3);
                table1.HeaderRows = 2;
                table1.FooterRows = 1;

                //header row
                var headerCell = new PdfPCell(new Phrase("header"));
                headerCell.Colspan = 3;
                headerCell.HorizontalAlignment = Element.ALIGN_CENTER;
                table1.AddCell(headerCell);

                //footer row
                var footerCell = new PdfPCell(new Phrase("footer"));
                footerCell.Colspan = 3;
                footerCell.HorizontalAlignment = Element.ALIGN_CENTER;
                table1.AddCell(footerCell);

                //adding some rows
                for (int i = 0; i < 70; i++)
                {
                    //adds a new row
                    table1.AddCell(new Phrase("Cell[0], Row[" + i + "]"));
                    table1.AddCell(new Phrase("Cell[1], Row[" + i + "]"));
                    table1.AddCell(new Phrase("Cell[2], Row[" + i + "]"));

                    //sets the number of rows per page
                    if (i > 0 && table1.Rows.Count % 7 == 0)
                    {
                        pdfDoc.Add(table1);
                        table1.DeleteBodyRows();
                        pdfDoc.NewPage();
                    }
                }

                pdfDoc.Add(table1);
```

```
            }

            //open the final file with adobe reader for instance.
            Process.Start("Test.pdf");
        }
    }
}
```

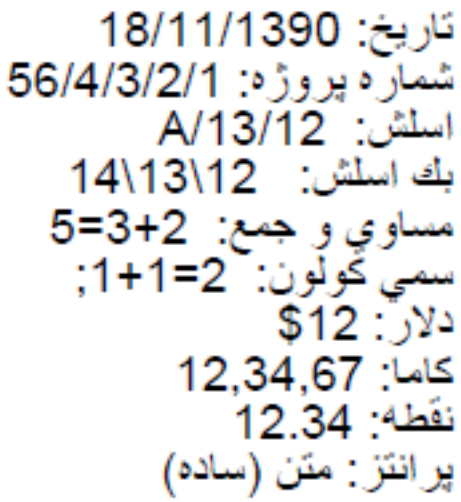
نکته جدید این مثال، قسمت زیر است:

```
if (i > 0 && table1.Rows.Count % 7 == 0)
{
      pdfDoc.Add(table1);
      table1.DeleteBodyRows();
      pdfDoc.NewPage();
}
```

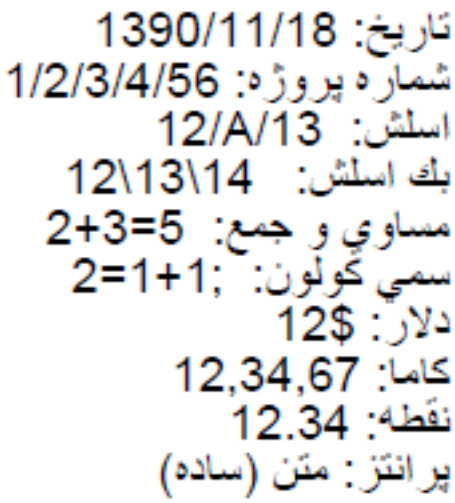
هر زمان که table1 به صفحه اضافه شود، header و footer هم اضافه خواهند شد، اما اگر BodyRows آن حذف نشود، دفعه‌ی دومی که این table به صفحه اضافه می‌شود، شامل ردیف‌های مثلا یک تا 10 خواهد بود بجای 6 تا 10 .

عنوان: **iTextSharp و نمایش صحیح تاریخ در متنی راست به چپ**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۱/۱۹ ۱۹:۴۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

خروجی PDF زیر را در نظر بگیرید:



مشکلی را در آن مشاهده می‌کنید؟ اصل آن یا صحیح آن باید به شکل زیر باشد:



و این وارونه نمایش دادن‌ها، دقیقا مشکلی است که حین کار با iTextSharp برای نمایش متنی مثلا به همراه یک تاریخ شمسی وجود دارد. البته این مشکل هم اساسا به خود استاندارد یونیکد برمی‌گرد که یک سری کاراکتر را « کاراکتر ضعیف » معرفی کرده؛ برای مثال کاراکتر اسلش بکار رفته در یک تاریخ هم از این دست است. بنابراین PDF تولیدی توسط iTextSharp از دید استاندارد

یونیکد مشکلی ندارد، زیرا یک «نویسه ضعیف» مثل اسلش نمی‌تواند جهت را تغییر دهد؛ مگر اینکه از یک « نویسه قوی » برای دستکاری آن استفاده شود. برای مثال این نویسه‌ها قوی هستند:

```
U+202A:   LEFT-TO-RIGHT EMBEDDING (LRE)
U+202B:   RIGHT-TO-LEFT EMBEDDING (RLE)
U+202D:   LEFT-TO-RIGHT OVERRIDE (LRO)
U+202E:   RIGHT-TO-LEFT OVERRIDE (RLO)
U+202C:   POP DIRECTIONAL FORMATTING (PDF)
```

برای رسیدن به تصویر صحیح نمایش داده شده در بالا، متد FixWeakCharacters زیر را تهیه کرده‌ام که حداقل با iTextSharp جواب می‌ده:

```
using System;
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace RleTests
{
    class Program
    {
        const char RightToLeftEmbedding = (char)0x202B;
        const char PopDirectionalFormatting = (char)0x202C;

        static string FixWeakCharacters(string data)
        {
            if (string.IsNullOrWhiteSpace(data)) return string.Empty;
            var weakCharacters = new[] { @"\", "/", "+", "-", "=", ";", "$" };
            foreach (var weakCharacter in weakCharacters)
            {
                data = data.Replace(weakCharacter, RightToLeftEmbedding + weakCharacter +
PopDirectionalFormatting);
            }
            return data;
        }

        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf", FileMode.Create));
                pdfDoc.Open();

                FontFactory.Register("c:\\windows\\fonts\\Arial.ttf");
                Font tahoma = FontFactory.GetFont("Arial", BaseFont.IDENTITY_H);

                var table1 = new PdfPTable(1);
                table1.WidthPercentage = 100;

                var pdfCell = new PdfPCell
                {
                    RunDirection = PdfWriter.RUN_DIRECTION_RTL,
                    Border = 0,
                    Phrase = new Phrase(FixWeakCharacters(
                        "18/11/1390" + " :تاریخ" + Environment.NewLine +
                        "56/4/3/2/1" + " :شماره پروژه" + Environment.NewLine +
                        "12 " + " :اسلش/A/13 " + Environment.NewLine +
                        "14\\13\\12  " + " بک اسلش: " + Environment.NewLine +
                        "5=3+2 " + " مساوی و جمع: " + Environment.NewLine +
                        "1+1=2 " + " ;سمی کولون: " + Environment.NewLine +
                        "12" + " :دلار$" + Environment.NewLine +
                        "12,34,67" + " :کاما" + Environment.NewLine +
                        "12.34" + " :نقطه" + Environment.NewLine +
                        "متن (ساده)" + " :پرانتز)"
                        ),
                        tahoma)
                };

                table1.AddCell(pdfCell);
                pdfDoc.Add(table1);
```

```
            }
            Process.Start("Test.pdf");
        }
    }
}
```

از این نوع مشکلات حین کار با HTML هم هست؛ وارونه نمایش داده شدن تاریخ فارسی در بین یک متن راست به چپ. البته در آنجا راه حل زیر هم توصیه شده (بدون نیاز به دستکاری نویسه‌ها):

```
<span dir="ltr" style="display:inline">1390/11/19</span>
```

نظرات خوانندگان

نویسنده: فرهاد یزدان پناه
تاریخ: ۱۳۹۰/۱۱/۱۹ ۲۱:۱۶:۴۲

ممنون. جناب نصیری.
آقای حاجلو هم مطلبه بسیار مفیدی در همین موضوع دارند.
/http://hajloo.wordpress.com/2009/03/02/persian-text-problem-in-ltr-forms

نویسنده: linux
تاریخ: ۱۳۹۰/۱۱/۲۲ ۲۳:۱۹:۴۵

برای جدا سازی اجزای تاریخ شمسی، ماه ، روز و سال نباید از / استفاده کرد در unicode برای این کار از Unicode Character ('ARABIC DATE SEPARATOR' (U+060D استفاده کنید برای دیدن جزییات بیشتر
http://www.fileformat.info/info/unicode/char/60d/index.htm

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۱/۲۳ ۰۰:۰۰:۴۲

شاید در زبان عربی اینطور باشه. حتما می‌دونید که نحوه نمایش و نویسه‌های اعداد 4 و 6 عربی و فارسی یکی نیست. ک و ی عربی و فارسی هم یکی نیست. حتی ممیز فارسی هم شیوه خاص خودش را دارد و کلا بحث من اینجا در مورد نحوه متداول ورود اطلاعات در زبان فارسی است؛ در مورد هزاران هزار سطر موجود. ضمن اینکه اگر به مثال دقت کرده باشید یک شماره پروژه‌ای هم این وسط هست که الگویی شبیه به تاریخ ندارد؛ به علاوه یک سری نویسه ضعیف دیگر مثل مساوی و جمع و منها و غیره. به علاوه بحث من در مورد کتابخانه تولید PDF ذکر شده است و راه حلی که با آن جواب بدهد.
راه حل بالایی که من مطرح کردم در نمایش هیچ تغییری ایجاد نمی‌کنه. این حرف بکارگرفته شده، نامرئی هستند. PDF هم یک لایه Presentation است. بنابراین زمانیکه اطلاعاتی را درست نمایش می‌دهد، یعنی هدف اصلی خودش را برآورده کرده.

نویسنده: Nasser Mansouri
تاریخ: ۱۳۹۰/۱۲/۰۸ ۱۵:۲۹:۴۷

سلام، می خواستم اگر ممکن باشه من رو راهنمایی کنید، من می خوام با استفاده از itextsharp محتوای یک فایل پی دی اف رو به صورت txt ذخیره کنم، با زبانهای چپ به راست خیلی آسون هست ولی در زبان راست به چپ، کلمات از اخر به اول نوشته می شن مثلا کلمه ارزیابی به صورت "یبایزرا" خوانده می شه، ممنون می شم من رو راهنمایی بکنید.

```
()private static string ParsPDFToString
}
;("PdfReader reader = new PdfReader("c:/2v.pdf
;(PdfReaderContentParser parser = new PdfReaderContentParser(reader
;()StringBuilder sb = new StringBuilder
;(())Console.WriteLine("reader.NumberOfPages : " + reader.NumberOfPages.ToString
;()Console.ReadKey
(++for (int i = 1; i <= reader.NumberOfPages; i
}
)ITextExtractionStrategy strategy = parser.ProcessContent
()i , new SimpleTextExtractionStrategy
;(
;(())sb.Append(strategy.GetResultantText
{
;()return sb.ToString
```

{

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۰۸ ۲۰:۰۶:۵۱

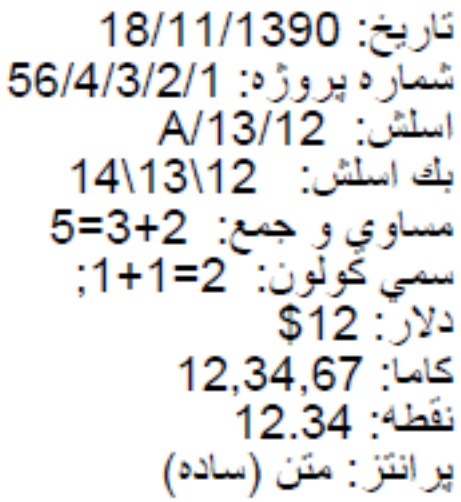
بستگی داره نرم افزار تولید PDF از چه روشی استفاده کرده باشه. بعضی‌ها از چرخاندن حروف استفاده می‌کنند و این روش بسیار متداولی هست. یعنی مشکل از iTextSharp نیست. در اصل به همین ترتیب حروف ذخیره شدن. الگوریتم اولیه به همین صورت بوده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۰۹ ۱۲:۲۹:۱۸

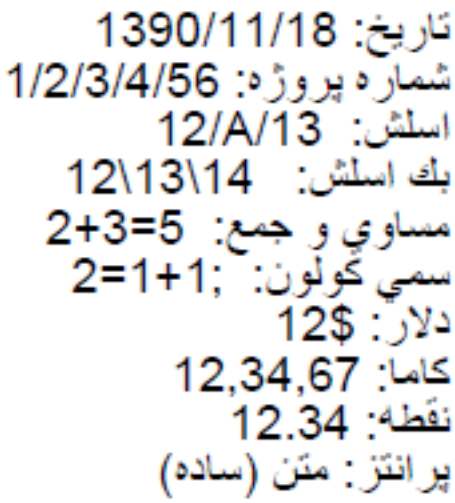
دو مورد تکمیلی:
- کار این چرخاندن‌ها توسط دو کلاس ArabicLigaturizer و BidiLine در iTextSharp انجام می‌شود. سورس کتابخانه را دریافت و این دو کلاس را مطالعه کنید (ضمن اینکه PDF های فارسی هم وجود دارند که اصلا با این الگوریتم‌ها تهیه نشده‌اند و خلاصه راه سختی را پیش رو دارید).iTextSharp انجمنی نداره ولی یک mailing list فعال داره:
https://lists.sourceforge.net/lists/listinfo/itext-questions

عنوان: **iTextSharp و نمایش صحیح تاریخ در متنی راست به چپ**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۱/۱۹ ۱۹:۴۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

خروجی PDF زیر را در نظر بگیرید:



مشکلی را در آن مشاهده می‌کنید؟ اصل آن یا صحیح آن باید به شکل زیر باشد:



و این وارونه نمایش دادن‌ها، دقیقا مشکلی است که حین کار با iTextSharp برای نمایش متنی مثلا به همراه یک تاریخ شمسی وجود دارد. البته این مشکل هم اساسا به خود استاندارد یونیکد برمی‌گردد که یک سری کاراکتر را « کاراکتر ضعیف » معرفی کرده؛ برای مثال کاراکتر اسلش بکار رفته در یک تاریخ هم از این دست است. بنابراین PDF تولیدی توسط iTextSharp از دید

استاندارد یونیکد مشکلی ندارد، زیرا یک «نویسه ضعیف» مثل اسلش نمی‌تواند جهت را تغییر دهد؛ مگر اینکه از یک « نویسه قوی » برای دستکاری آن استفاده شود. برای مثال این نویسه‌ها قوی هستند:

```
U+202A:   LEFT-TO-RIGHT EMBEDDING (LRE)
U+202B:   RIGHT-TO-LEFT EMBEDDING (RLE)
U+202D:   LEFT-TO-RIGHT OVERRIDE (LRO)
U+202E:   RIGHT-TO-LEFT OVERRIDE (RLO)
U+202C:   POP DIRECTIONAL FORMATTING (PDF)
```

برای رسیدن به تصویر صحیح نمایش داده شده در بالا، متد FixWeakCharacters زیر را تهیه کرده‌ام که حداقل با iTextSharp جواب می‌ده:

```
using System;
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace RleTests
{
    class Program
    {
        const char RightToLeftEmbedding = (char)0x202B;
        const char PopDirectionalFormatting = (char)0x202C;

        static string FixWeakCharacters(string data)
        {
            if (string.IsNullOrWhiteSpace(data)) return string.Empty;
            var weakCharacters = new[] { @"\", "/", "+", "-", "=", ";", "$" };
            foreach (var weakCharacter in weakCharacters)
            {
                data = data.Replace(weakCharacter, RightToLeftEmbedding + weakCharacter +
PopDirectionalFormatting);
            }
            return data;
        }

        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf", FileMode.Create));
                pdfDoc.Open();

                FontFactory.Register("c:\\windows\\fonts\\Arial.ttf");
                Font tahoma = FontFactory.GetFont("Arial", BaseFont.IDENTITY_H);

                var table1 = new PdfPTable(1);
                table1.WidthPercentage = 100;

                var pdfCell = new PdfPCell
                {
                    RunDirection = PdfWriter.RUN_DIRECTION_RTL,
                    Border = 0,
                    Phrase = new Phrase(FixWeakCharacters(
                        "18/11/1390" + " :تاریخ" + Environment.NewLine +
                        "56/4/3/2/1" + " :شماره پروژه" + Environment.NewLine +
                        "12 " + " :اسلش/A/13 " + Environment.NewLine +
                        "14\\13\\12  " + " :بک اسلش " + Environment.NewLine +
                        "5=3+2 " + " :مساوی و جمع " + Environment.NewLine +
                        "1+1=2 " + " :سمی کولون; " + Environment.NewLine +
                        "12" + " :دلار$" + Environment.NewLine +
                        "12,34,67" + " :کاما" + Environment.NewLine +
                        "12.34" + " :نقطه" + Environment.NewLine +
                        "متن (ساده)" + " :پرانتز)"
                        ),
                        tahoma)
                };

                table1.AddCell(pdfCell);
                pdfDoc.Add(table1);
```

```
            }
            Process.Start("Test.pdf");
        }
    }
}
```

از این نوع مشکلات حین کار با HTML هم هست؛ وارونه نمایش داده شدن تاریخ فارسی در بین یک متن راست به چپ. البته در آنجا راه حل زیر هم توصیه شده (بدون نیاز به دستکاری نویسه‌ها):

```
<span dir="ltr" style="display:inline">1390/11/19</span>
```

نظرات خوانندگان

نویسنده: فرهاد یزدان پناه
تاریخ: ۱۳۹۰/۱۱/۱۹ ۲۱:۱۶:۴۲

ممنون. جناب نصیری.
آقای حاجلو هم مطلبه بسیار مفیدی در همین موضوع دارند.
/http://hajloo.wordpress.com/2009/03/02/persian-text-problem-in-ltr-forms

نویسنده: linux
تاریخ: ۱۳۹۰/۱۱/۲۲ ۲۳:۱۹:۴۵

برای جدا سازی اجزای تاریخ شمسی، ماه ، روز و سال نباید از / استفاده کرد در unicode برای این کار از Unicode Character ('ARABIC DATE SEPARATOR' (U+060D استفاده کنید برای دیدن جزییات بیشتر
http://www.fileformat.info/info/unicode/char/60d/index.htm

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۱/۲۳ ۰۰:۰۰:۴۲

شاید در زبان عربی اینطور باشه. حتما می‌دونید که نحوه نمایش و نویسه‌های اعداد 4 و 6 عربی و فارسی یکی نیست. ک و ی عربی و فارسی هم یکی نیست. حتی ممیز فارسی هم شیوه خاص خودش را دارد و کلا بحث من اینجا در مورد نحوه متداول ورود اطلاعات در زبان فارسی است؛ در مورد هزاران هزار سطر موجود. ضمن اینکه اگر به مثال دقت کرده باشید یک شماره پروژه‌ای هم این وسط هست که الگویی شبیه به تاریخ ندارد؛ به علاوه یک سری نویسه ضعیف دیگر مثل مساوی و جمع و منها و غیره. به علاوه بحث من در مورد کتابخانه تولید PDF ذکر شده است و راه حلی که با آن جواب بدهد.
راه حل بالایی که من مطرح کردم در نمایش هیچ تغییری ایجاد نمی‌کنه. این حرف بکارگرفته شده، نامرئی هستند. PDF هم یک لایه Presentation است. بنابراین زمانیکه اطلاعاتی را درست نمایش می‌دهد، یعنی هدف اصلی خودش را برآورده کرده.

نویسنده: Nasser Mansouri
تاریخ: ۱۳۹۰/۱۲/۰۸ ۱۵:۲۹:۴۷

سلام، می خواستم اگر ممکن باشه من رو راهنمایی کنید، من می خوام با استفاده از itextsharp محتوای یک فایل پی دی اف رو به صورت txt ذخیره کنم، با زبانهای چپ به راست خیلی آسون هست ولی در زبان راست به چپ، کلمات از اخر به اول نوشته می شن مثلا کلمه ارزیابی به صورت "یبایزرا" خوانده می شه، ممنون می شم من رو راهنمایی بکنید.

```
()private static string ParsPDFToString
}
;("PdfReader reader = new PdfReader("c:/2v.pdf
;(PdfReaderContentParser parser = new PdfReaderContentParser(reader
;()StringBuilder sb = new StringBuilder
;(((Console.WriteLine("reader.NumberOfPages : " + reader.NumberOfPages.ToString
;()Console.ReadKey
(++for (int i = 1; i <= reader.NumberOfPages; i
}
)ITextExtractionStrategy strategy = parser.ProcessContent
()i , new SimpleTextExtractionStrategy
;(
;(((sb.Append(strategy.GetResultantText
{
;()return sb.ToString
```

{

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۰۸ ۲۰:۰۶:۵۱

بستگی داره نرم افزار تولید PDF از چه روشی استفاده کرده باشه. بعضی‌ها از چرخاندن حروف استفاده می‌کنند و این روش بسیار متداولی هست. یعنی مشکل از iTextSharp نیست. در اصل به همین ترتیب حروف ذخیره شدن. الگوریتم اولیه به همین صورت بوده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۰۹ ۱۲:۲۹:۱۸

دو مورد تکمیلی:
- کار این چرخاندن‌ها توسط دو کلاس ArabicLigaturizer و BidiLine در iTextSharp انجام می‌شود. سورس کتابخانه را دریافت و این دو کلاس را مطالعه کنید (ضمن اینکه PDF های فارسی هم وجود دارند که اصلا با این الگویتم‌ها تهیه نشده‌اند و خلاصه راه سختی را پیش رو دارید).iTextSharp انجمنی نداره ولی یک mailing list فعال داره:
https://lists.sourceforge.net/lists/listinfo/itext-questions

عنوان: **تبدیل html به pdf با کیفیت بالا**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۰ ۹:۵۴
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp, wkhtmltopdf, WkHtmlToXSharp

کتابخانه iTextSharp دارای کلاسی است به نام HTMLWorker که کار تبدیل عناصر HTML را به عناصر متناظر خودش، انجام می‌دهد. این کلاس در حال حاضر منسوخ شده درنظر گرفته می‌شود (اینطور توسط نویسندگان آن اعلام شده) و دیگر توسعه نخواهد یافت. بنابراین اگر از HTMLWorker استفاده می‌کنید با یک کلاس قدیمی که دارای HTML Parser ایی بسیار بدوی است طرف هستید و در کل برای تبدیل محتوای HTML ایی با ساختار بسیار ساده بد نیست؛ اما انتظار زیادی از آن نداشته باشید.
جایگزین کلاس HTMLWorker در این کتابخانه در حال حاضر کتابخانه itextsharp.xmlworker است، که به صورت یک افزونه در کنار کتابخانه اصلی در حال توسعه می‌باشد. مشکل اصلی این کتابخانه، عدم پشتیبانی از UTF8 و راست به چپ است. بنابراین حداقل به درد کار ما نمی‌خورد.

راه حل بسیار بهتری برای موضوع اصلی بحث ما وجود دارد و آن هم استفاده از موتور WebKit (همان موتوری که برای مثال در Apple Safari استفاده می‌شود) برای HTML parsing و سپس تبدیل نتیجه نهایی به PDF است. پروژه‌ای که این مقصود را میسر کرده، wkhtmltopdf نام دارد.
توسط آن به کمک موتور WebKit، کار HTML Parsing انجام شده و سپس برای تبدیل عناصر نهایی به PDF از امکانات کتابخانه‌ای به نام QT استفاده می‌شود. کیفیت نهایی آن کپی مطابق اصل HTML قابل مشاهده در یک مرورگر است و با یونیکد و زبان فارسی هم مشکلی ندارد.

برای استفاده از این کتابخانه‌ی native در دات نت، شخصی پروژه‌ای را ایجاد کرده است به نام WkHtmlToXSharp که محصور کننده‌ی wkhtmltopdf می‌باشد. در ادامه به نحوه استفاده از آن خواهیم پرداخت:

الف) دریافت پروژه WkHtmlToXSharp
پروژه WkHtmlToXSharp را از آدرس زیر می‌توانید دریافت کنید.
https://github.com/pruiz/WkHtmlToXSharp

این پروژه به همراه فایل‌های کامپایل شده نهایی wkhtmltopdf نیز می‌باشد و حجمی حدود 40 مگ دارد. به علاوه فعلا نسخه 32 بیتی آن در دسترس است. بنابراین باید دقت داشت که نباید تنظیمات پروژه دات نت خود را بر روی Any CPU قرار دهیم، زیرا در این حالت برنامه شما در یک سیستم 64 بیتی بلافاصله کرش خواهد کرد. تنظیمات target platform پروژه دات نتی ما حتما باید بر روی X86 تنظیم شود.

ب) پس از دریافت این پروژه و افزودن ارجاعی به اسمبلی WkHtmlToXSharp.dll، استفاده از آن به نحو زیر می‌باشد:

```
using System.IO;
using WkHtmlToXSharp;
using System;

namespace Test2
{
    public class WkHtmlToXSharpTest
    {
        public static void ConvertHtmlStringToPdfTest()
        {
            using (var wk = new MultiplexingConverter())
            {
                wk.Begin += (s, e) => Console.WriteLine("Conversion begin, phase count: {0}", e.Value);
                wk.Error += (s, e) => Console.WriteLine(e.Value);
                wk.Warning += (s, e) => Console.WriteLine(e.Value);
                wk.PhaseChanged += (s, e) => Console.WriteLine("PhaseChanged: {0} - {1}", e.Value,
e.Value2);
                wk.ProgressChanged += (s, e) => Console.WriteLine("ProgressChanged: {0} - {1}",
e.Value, e.Value2);
```

```
                wk.Finished += (s, e) => Console.WriteLine("Finished: {0}", e.Value ? "success" :
"failed!");

                wk.GlobalSettings.Margin.Top = "0cm";
                wk.GlobalSettings.Margin.Bottom = "0cm";
                wk.GlobalSettings.Margin.Left = "0cm";
                wk.GlobalSettings.Margin.Right = "0cm";

                wk.ObjectSettings.Web.EnablePlugins = false;
                wk.ObjectSettings.Web.EnableJavascript = false;
                wk.ObjectSettings.Load.Proxy = "none";

                var htmlString = File.ReadAllText(@"c:\page.xhtml");
                var tmp = wk.Convert(htmlString);

                File.WriteAllBytes(@"tst.pdf", tmp);
            }
        }
    }
}
```

کار با وهله سازی از کلاس MultiplexingConverter شروع می‌شود. اگر علاقمند باشید که درصد پیشرفت کار به همراه خطاهای احتمالی پردازشی را ملاحظه کنید می‌توان از رخدادگردان‌هایی مانند ProgressChanged و Error استفاده نمائید که نمونه‌ای از آن در کد فوق بکارگرفته شده است.

تبدیل HTML به PDF آنچنان تنظیمات خاصی ندارد زیرا فرض بر این است که قرار است از همان تنظیمات اصلی HTML مورد نظر استفاده گردد. اما اگر نیاز به تنظیمات بیشتری وجود داشت، برای مثال به کمک GlobalSettings آن می‌توان حاشیه‌های صفحات فایل نهایی تولیدی را تنظیم کرد.

موتور WebKit با توجه به اینکه موتور یک مرورگر است، امکان پردازش جاوا اسکریپت را هم دارد. بنابراین اگر قصد استفاده از آن‌را نداشتید می‌توان خاصیت ObjectSettings.Web.EnableJavascript را به false مقدار دهی کرد.

کار اصلی، در متد Convert انجام می‌شود. در اینجا می‌توان یک رشته را که حاوی فایل HTML مورد نظر است به آن ارسال کرد و نتیجه نهایی، آرایه‌ای از بایت‌ها، حاوی فایل باینری PDF تولیدی است.

روش دیگر استفاده از این کتابخانه، مقدار دهی wk.ObjectSettings.Page می‌باشد. در اینجا می‌توان Url یک صفحه اینترنتی را مشخص ساخت. در این حالت دیگر نیازی نیست تا به متد Convert پارامتری را ارسال کرد. می‌توان از overload بدون پارامتر آن استفاده نمود.

یک نکته:

اگر می‌خواهید زبان فارسی را توسط این کتابخانه به درستی پردازش کنید، نیاز است حتما یک سطر زیر را به header فایل html خود اضافه نمائید:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

نظرات خوانندگان

نویسنده: حمید
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۳:۴۲

سلام.خسته نباشید.ممنون از لطلاعات پربارتون.
بنده با توجه به توضیحاتتوون عمل کردم و خروجی Pdf هم دریافت کردم
اما مشکلی که وجود داره،بیشتر وقتها ارور میده و ویژوال استودیو بسته می‌شه و میگه فضای کافی برای لود این dll وجود نداره.(WkHtmlToXSharp)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۴:۱۵

در عمل عموم کدهای native نوشته شده با سی پلاس پلاس این مشکلات را دارند:
- ناپایدار
- دارای نشتی‌های حافظه بالا
- نا امن
- نیاز به کامپایل مجزا برای سیستم‌های 64 بیتی و 32 بیتی
فقط از این کلمه لذت می‌برند: «سرعت»! اما در 4 مورد فوق حرفی برای گفتن ندارند.

ولی خوب بازسازی این پروژه‌ها با دات نت وقت زیادی می‌گیرد به همین جهت کسی طرف تبدیل آن‌ها نرفته. نوشتن یک html parser خوب و تمام عیار، یک پروژه چند میلیون دلاری است که موزیلا، مایکروسافت، اپل، گوگل و غیره درگیر آن هستند!

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۵/۲۲ ۱۹:۴۸

با سلام؛
ببخشید شما برای قابلیت نسخه چاپی این سایت از QT استفاده کرده اید یا iTextSharp؟ ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۲ ۲۰:۱۸

ترکیب iTextSharp و Html Agility Pack است. به عبارتی مجبور شدم قابلیت تبدیل html به pdf غیرمطلوب iTextSharp رو در حد نیاز سایت از صفر بازنویسی کنم. html parser سورس باز Html Agility Pack واقعا باکیفیت است.

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۵/۲۲ ۲۰:۵۳

من می‌خواهم در یک وب سایت که یک اپلیکیشن است یک سری گزارش تهیه بکنم سناریو به این صورت است که ابتدا درون وب سایت گزارش را نمایش می‌دهیم و اگر کاربر خواست می‌تواند اون گزارش رو به فرمت PDF دانلود کند. برای این سناریو به نظر شما از چه چیزی استفاده کنم؟

ممنونم

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۵/۲۲ ۲۱:۱۵

باید برنامه نویسی کنید. تبدیل html به pdf در این نوع موارد خاص جواب نمی‌دهد چون یک گزارش نیاز به خیلی از جزئیات دیگر مانند شماره صفحه، header و footer و غیره هم دارد. از این مثال می‌تونید ایده بگیرید.

نویسنده: hasani
تاریخ: ۱۳۹۱/۰۵/۲۳ ۲۳:۲۵

شاید این مثال هم کمک کند http://www.codeproject.com/Articles/260470/PDF-reporting-using-ASP-NET-MVC3

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۴ ۰:۲۷

از این روش استفاده می‌کنه. به علاوه باید درنظر داشت HTMLWorker کتابخانه iTextSharp منسوخ شده در نظر گرفته می‌شود و دیگر توسعه نخواهد یافت و حتی نگهداری هم نمی‌شود. با Xml Worker آن جایگزین شده که فعلا از RTL و یونیکد پشتیبانی نمی‌کند.

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۵/۲۹ ۲۰:۰

ببخشید استاد می‌خواست بپرسم که وقتی که در وب سایت مطالب رو از نسخه چاپی دانلود می‌کنیم نام تمام فایل‌ها dotnettips هست آیا شما یه همچین فایل فیزیکی دارید که محتوای آن را به صورت داینامیک ایجاد می‌کنید یا اینکه درون حافظه این فایل را ایجاد می‌کنید و اصلاً فایل فیزیکی وجود ندارد؟ اگه از حالت فایل فیزیکی استفاده می‌کنید تداخل داده به وجود نمی‌آید؟ مثلاً یکی از کاربران روی مقاله 100 کلیک کند و هم زمان یک کاربر دیگر روی مقاله 101 کلیک کند اون موقعه متن مقاله 101 برای هر 2 نمایش داده می‌شود

ممنونم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۹ ۲۰:۱۳

iTextSharp با Stream کار می‌کند. این استریم می‌تواند فایل استریم یا memory stream باشد؛ که من در اینجا از حالت memory stream استفاده می‌کنم.
ضمن اینکه اگر فایل استریم هم می‌بود چون نام فایل‌ها یکی نیست، تداخلی رخ نمی‌داد.

نویسنده: نوید
تاریخ: ۱۳۹۱/۰۸/۰۶ ۱۴:۴۴

جناب نصیری سلام
از مقالات آموزنده شما بسیار سپاسگذارم
میخاستم بدونم شما در سایتتون از iTextSharp استفاده کردید یا PdfReport ؟
چون من از iTextSharp استفاده کردم و داخل جداولم متون فارسی نوشتم که کلا به هم ریخته نمایش میده ، اگه امکانش هست یک راهنمایی بفرمائید
متشکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۶ ۱۶:۱۱

لطفا برچسب iTextSharp را در سایت جاری دنبال بفرمائید. در این مورد کاملا توضیح داده شده. برای نمونه: (^)

نویسنده: سامان

تاریخ: ۱۳۹۲/۰۲/۲۷ ۲۱:۳۰

سلام
پروژه wkhtmltopdf برای ویندوز فقط معماری i386 رو پشتیبانی می‌کنه. آیا این میتونه برای یه برنامه که قراره رو سیستم مشتری‌های مختلفی اجرا بشه مشکل ایجاد کنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۷ ۲۲:۹

بله. platform target رو باید روی X86 قرار بدید تا همه جا بدون مشکل اجرا بشه.
یا اینکه نسخه 64 بیتی اون رو هم پیدا یا کامپایل کنید و نهایتا مانند خیلی از کارهای native باید دو نسخه 64 بیتی و 32 بیتی از برنامه خودتون رو منتشر کنید.

نویسنده: آتوسا فتوحی
تاریخ: ۱۳۹۳/۰۳/۱۱ ۱۰:۷

اگر از ابزارهای گزارش گیری مانند: Telerik, Stimul, ... استفاده شود, بصورت Built-in قابلیت تبدیل به PDF و یا Excel, ... را به ما می‌دهد.

عنوان: **عبارت using و نحوه استفاده صحیح از آن**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۲ ۱۳:۳۱
آدرس: www.dotnettips.info
برچسب‌ها: C#, Refactoring, iTextSharp

مثال ساده زیر را که در مورد تعریف یک کلاس Disposable و سپس استفاده از آن توسط عبارت using است را به همراه سه استثنایی که در این متدها تعریف شده است، در نظر بگیرید:

```
using System;

namespace TestUsing
{
    public class MyResource : IDisposable
    {
        public void DoWork()
        {
            throw new ArgumentException("A");
        }

        public void Dispose()
        {
            throw new ArgumentException("B");
        }
    }

    public static class TestClass
    {
        public static void Test()
        {
            using (MyResource r = new MyResource())
            {
                throw new ArgumentException("C");
                r.DoWork();
            }
        }
    }
}
```

به نظر شما قطعه کد زیر چه عبارتی را نمایش می‌دهد؟ C یا B یا A؟

```
try
{
     TestClass.Test();
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

پاسخ: برخلاف تصور (که احتمالا C است؛ چون قبل از فراخوانی متد DoWork سبب بروز استثناء شده است)، فقط B را در خروجی مشاهده خواهیم کرد!
و این دقیقا مشکلی است که در حین کار با کتابخانه iTextSharp برای اولین بار با آن مواجه شدم. روش استفاده متداول از iTextSharp به نحو زیر است:

```
using (var pdfDoc = new Document(PageSize.A4))
{
   //todo: ...
}
```

در این بین هر استثنایی رخ دهد، در لاگ‌های خطای سیستم شما تنها خطاهای مرتبط با خود iTextSharp را مشاهده خواهید کرد و نه مشکل اصلی را که در کدهای ما وجود داشته است. البته این یک مشکل عمومی است و اگر « using statement and suppressed exceptions » را در گوگل جستجو کنید به نتایج مشابه زیادی خواهید رسید.
و خلاصه نتایج هم این است:

اگر به ثبت جزئیات خطاهای سیستم اهمیت می‌دهید (یکی از مهم‌ترین مزیت‌های دات نت نسبت به بسیاری از فریم ورک‌های مشابه که حداکثر خطای 0xABC12EF را نمایش می‌دهند)، از using استفاده نکنید! using در پشت صحنه به try/finally ترجمه می‌شود و بهتر است این مورد را دستی نوشت تا اینکه کامپایلر اینکار را به صورت خودکار انجام دهد.
در اینجا باز هم به یک سری کد تکراری try/finally خواهیم رسید و همانطور که در مباحث کاربردهای Action و Func در این سایت ذکر شد، می‌توان آن‌را تبدیل به کدهایی با قابلیت استفاده مجدد کرد. یک نمونه از پیاده سازی آن‌را در این سایت « C# Using Blocks can Swallow Exceptions » می‌توانید مشاهده کنید که خلاصه آن کلاس زیر است:

```
using System;

namespace Guard
{
    public static class SafeUsing
    {
        public static void SafeUsingBlock<TDisposable>(this TDisposable disposable, Action<TDisposable>
action)
            where TDisposable : IDisposable
        {
            disposable.SafeUsingBlock(action, d => d);
        }

        internal static void SafeUsingBlock<TDisposable, T>(this TDisposable disposable, Action<T>
action, Func<TDisposable, T> unwrapper)
            where TDisposable : IDisposable
        {
            try
            {
                action(unwrapper(disposable));
            }
            catch (Exception actionException)
            {
                try
                {
                    disposable.Dispose();
                }
                catch (Exception disposeException)
                {
                    throw new AggregateException(actionException, disposeException);
                }

                throw;
            }

            disposable.Dispose();
        }
    }
}
```

برای استفاده از کلاس فوق مثلا در حالت بکارگیری iTextSharp خواهیم داشت:

```
new Document(PageSize.A4).SafeUsingBlock(pdfDoc =>
{
  //todo: ...
});
```

علاوه بر اینکه SafeUsingBlock یک سری از اعمال تکراری را کپسوله می‌کند، از AggregateException نیز استفاده کرده است (معرفی شده در دات نت 4). به این صورت چندین استثنای رخ داده نیز در سطحی بالاتر قابل دریافت و بررسی خواهند بود و استثنایی در این بین از دست نخواهد رفت.

نظرات خوانندگان

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۱/۰۶/۱۳ ۷:۵۹

با سلام خدمت شما جناب نصیری
با توجه به این مطلب که زیاد هم قدیمی نیست به نظر شما استفاده از using پیشنهاد میشود یا استفاده از try catch ؟
چون بنده در برنامه‌های کاربردی تحت وب بیشتر از using استفاده میکنم و از Elmah هم برای ثبت خطاهای سیستم استفاده میکنم .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۳ ۸:۴۲

مطلب جاری بیشتر به شبیه سازی **finally** /try معادل using که توسط کامپایلر به صورت خودکار تولید می‌شود مرتبط است نه try/catch کلی. بحث dispose خودکار اشیاء disposable و اینکه استفاده از using به دلیلی که عنوان شد مناسب نیست. بنابراین بجای using از SafeUsingBlock استفاده کنید (شبیه سازی بهتر کاری است که کامپایلر در پشت صحنه جهت معادل سازی یا پیاده سازی using انجام می‌دهد؛ اما بدون از دست رفتن استثناهای رخ داده). مابقی را هم ELMAH انجام می‌دهد.
اگر از using استفاده کنید و ELMAH، فقط خطاهای مرتبط با مثلا iTextSharp رو در لاگ‌ها خواهید یافت؛ مثلا شیء document آن dispose شده، اما خطا و مشکل اصلی که به کدهای ما مرتبط بوده و نه iTextSharp، این میان گم خواهد شد. اما با استفاده از SafeUsingBlock ، دلیل اصلی نیز لاگ می‌شود.

نویسنده: مرادی
تاریخ: ۱۳۹۱/۰۶/۱۳ ۹:۹

البته از حق هم نمی‌شه گذشت که طراح‌های iText Sharp در این جا هم از Best Practiceها پیروی نکردند
این قاعده کلی کار هستش که در Dispose و متد Finalize خطایی ایجاد نشه
حتی چند بار فراخوانی Dispose هم نباید ایجاد خطا کنه، حتی خطای Object Disposed Exception
متاسفانه تفاوت‌های Java و .NET. تونسته رو این تیم که iText Sharp رو از Java به .NET. برده، رو به یک سری اشتباهات بندازه، مثل این مورد، و عدم استفاده از Enum و چند تا مورد دیگه
اگه فرض کنیم ما Dispose رو فراخونی نکنیم و این فراخونی توسط CLR و در فاینالایزر کلاس رخ بده، این خطایی موجود در Dispose می‌تونه مسائل بدتری رو هم در پی داشته باشه
به دوستان توصیه می‌کنم حتما با Best Practiceهای مدیریت خطا مخصوص .NET. آشنا بشن، که در اینترنت منابع زیادی برای این مهم موجوده

نویسنده: افشین
تاریخ: ۱۳۹۱/۰۶/۱۳ ۱۳:۱۰

واقا خسته نباشید.
هر روز بابات این سایت شما رو دعا میکنیم.
ببخشید این موضوع رو این جا مطرح میکنم چون راه دیگه ای پیدا نکردم

نویسنده: سید امیر سجادی
تاریخ: ۱۳۹۲/۰۲/۰۵ ۹:۴۹

سلام. من این مشکلی که گفتید رو توی VB.NET تست کردم مشکلی نداشت. آیا .NET این مشکل رو داره و یا فقط C#?

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۰۲/۰۵ ۱۰:۱۴

این رفتار در VB.NET هم قابل مشاهده است:

```
Public Class MyResource
    Implements IDisposable
    Public Sub DoWork()
        Throw New ArgumentException("A")
    End Sub

    Public Overloads Sub Dispose() Implements System.IDisposable.Dispose
        Throw New ArgumentException("B")
    End Sub
End Class

Public NotInheritable Class TestClass
    Private Sub New()
    End Sub
    Public Shared Sub Test()
        Using r As New MyResource()
            Throw New ArgumentException("C")
            r.DoWork()
        End Using
    End Sub
End Class
Module Module1

    Sub Main()
        Try
            TestClass.Test()
        Catch ex As Exception
            Console.WriteLine(ex.Message)
        End Try
    End Sub

End Module
```

عبارت نمایش داده شده در اینجا هم B است.

عنوان: **تبدیل صفحات یک فایل PDF به تصویر، با استفاده از Acrobat SDK**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۱۲ ۰:۱۰
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp, PdfReport, PDF

با استفاده از اشیاء Com همراه با Acrobat SDK می‌توان تمام صفحات یک فایل PDF را تبدیل به تصویر کرد. این SDK به همراه نگارش کامل Adobe Acrobat نیز بر روی سیستم نصب می‌شود و یا می‌توان آن‌را به صورت جداگانه از سایت Adobe دریافت کرد.
http://www.adobe.com/devnet/acrobat/downloads.html
پس از آن، برای تبدیل صفحات یک فایل PDF به تصویر، مراحل زیر باید طی شود:
الف) وهله سازی از شیء AcroExch.PDDoc
در صورتیکه SDK یاد شده بر روی سیستم نصب نباشد، این وهله سازی با شکست مواجه خواهد شد و همچنین باید دقت داشت که این SDK به همراه نگارش رایگان Adobe reader ارائه نمی‌شود.
ب) گشودن فایل PDF به کمک شیء Com وهله سازی شده (pdfDoc.Open)
ج) دریافت اطلاعات صفحه مورد نظر (pdfDoc.AcquirePage)
د) کپی این اطلاعات به درون clipboard ویندوز (pdfPage.CopyToClipboard)
به این ترتیب به یک تصویر Bmp قرار گرفته شده در clipboard ویندوز خواهیم رسید
ه) مرحله بعد تغییر ابعاد و ذخیره سازی این تصویر نهایی است.

کدهای زیر، روش انجام این مراحل را بیان می‌کنند:

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.IO;
using System.Runtime.InteropServices;
using System.Threading;
using System.Windows.Forms;
using Acrobat; //Add a Com Object ref. to "Adobe Acrobat 10.0 Type Library" => Program
Files\Adobe\Acrobat 10.0\Acrobat\acrobat.tlb
using Microsoft.Win32;

namespace PdfThumbnail.Lib
{
    public static class PdfToImage
    {
        const string AdobeObjectsErrorMessage = "Failed to create the PDF object.";
        const string BadFileErrorMessage = "Failed to open the PDF file.";
        const string ClipboardError = "Failed to get the image from clipboard.";
        const string SdkError = "This operation needs the Acrobat
SDK(http://www.adobe.com/devnet/acrobat/downloads.html), which is combined with the full version of
Adobe Acrobat.";

        public static byte[] PdfPageToPng(string pdfFilePath, int thumbWidth = 600, int thumbHeight =
750, int pageNumber = 0)
        {
            byte[] imageData = null;
            runJob((pdfDoc, pdfRect) =>
                {
                    imageData = pdfPageToPng(thumbWidth, thumbHeight, pageNumber, pdfDoc, pdfRect);
                }, pdfFilePath);
            return imageData;
        }

        public static void AllPdfPagesToPng(Action<byte[], int, int> dataCallback, string pdfFilePath,
int thumbWidth = 600, int thumbHeight = 750)
        {
            runJob((pdfDoc, pdfRect) =>
                {
                    var numPages = pdfDoc.GetNumPages();
                    for (var pageNumber = 0; pageNumber < numPages; pageNumber++)
                    {
                        var imageData = pdfPageToPng(thumbWidth, thumbHeight, pageNumber, pdfDoc,
pdfRect);

                        dataCallback(imageData, pageNumber + 1, numPages);
                    }
                }, pdfFilePath);
```

```
        }

        static void runJob(Action<CAcroPDDoc, CAcroRect> job, string pdfFilePath)
        {
            if (!File.Exists(pdfFilePath))
                throw new InvalidOperationException(BadFileErrorMessage);

            var acrobatPdfDocType = Type.GetTypeFromProgID("AcroExch.PDDoc");
            if (acrobatPdfDocType == null || !isAdobeSdkInstalled)
                throw new InvalidOperationException(SdkError);

            var pdfDoc = (CAcroPDDoc)Activator.CreateInstance(acrobatPdfDocType);
            if (pdfDoc == null)
                throw new InvalidOperationException(AdobeObjectsErrorMessage);

            var acrobatPdfRectType = Type.GetTypeFromProgID("AcroExch.Rect");
            var pdfRect = (CAcroRect)Activator.CreateInstance(acrobatPdfRectType);

            var result = pdfDoc.Open(pdfFilePath);
            if (!result)
                throw new InvalidOperationException(BadFileErrorMessage);

            job(pdfDoc, pdfRect);

            releaseComObjects(pdfDoc, pdfRect);
        }

        public static byte[] ResizeImage(this Image image, int thumbWidth, int thumbHeight)
        {
            var srcWidth = image.Width;
            var srcHeight = image.Height;
            using (var bmp = new Bitmap(thumbWidth, thumbHeight, PixelFormat.Format32bppArgb))
            {
                using (var gr = Graphics.FromImage(bmp))
                {
                    gr.SmoothingMode = SmoothingMode.HighQuality;
                    gr.PixelOffsetMode = PixelOffsetMode.HighQuality;
                    gr.CompositingQuality = CompositingQuality.HighQuality;
                    gr.InterpolationMode = InterpolationMode.High;

                    var rectDestination = new Rectangle(0, 0, thumbWidth, thumbHeight);
                    gr.DrawImage(image, rectDestination, 0, 0, srcWidth, srcHeight,
GraphicsUnit.Pixel);

                    using (var memStream = new MemoryStream())
                    {
                        bmp.Save(memStream, ImageFormat.Png);
                        return memStream.ToArray();
                    }
                }
            }
        }

        static bool isAdobeSdkInstalled
        {
            get
            {
                return Registry.ClassesRoot.OpenSubKey("AcroExch.PDDoc", writable: false) != null;
            }
        }

        private static Bitmap pdfPageToBitmap(int pageNumber, CAcroPDDoc pdfDoc, CAcroRect pdfRect)
        {
            var pdfPage = (CAcroPDPage)pdfDoc.AcquirePage(pageNumber);
            if (pdfPage == null)
                throw new InvalidOperationException(BadFileErrorMessage);

            var pdfPoint = (CAcroPoint)pdfPage.GetSize();

            pdfRect.Left = 0;
            pdfRect.right = pdfPoint.x;
            pdfRect.Top = 0;
            pdfRect.bottom = pdfPoint.y;

            pdfPage.CopyToClipboard(pdfRect, 0, 0, 100);

            Bitmap pdfBitmap = null;
            var thread = new Thread(() =>
            {
                var data = Clipboard.GetDataObject();
                if (data != null && data.GetDataPresent(DataFormats.Bitmap))
```

```
                    pdfBitmap = (Bitmap)data.GetData(DataFormats.Bitmap);
            });
            thread.SetApartmentState(ApartmentState.STA);
            thread.Start();
            thread.Join();

            Marshal.ReleaseComObject(pdfPage);

            return pdfBitmap;
        }

        private static byte[] pdfPageToPng(int thumbWidth, int thumbHeight, int pageNumber, CAcroPDDoc
pdfDoc, CAcroRect pdfRect)
        {
            var pdfBitmap = pdfPageToBitmap(pageNumber, pdfDoc, pdfRect);
            if (pdfBitmap == null)
                throw new InvalidOperationException(ClipboardError);

            var pdfImage = pdfBitmap.GetThumbnailImage(thumbWidth, thumbHeight, null, IntPtr.Zero);
            // (+ 7 for template border)
            var imageData = pdfImage.ResizeImage(thumbWidth + 7, thumbHeight + 7);
            return imageData;
        }

        private static void releaseComObjects(CAcroPDDoc pdfDoc, CAcroRect pdfRect)
        {
            pdfDoc.Close();
            Marshal.ReleaseComObject(pdfRect);
            Marshal.ReleaseComObject(pdfDoc);
        }
    }
}
```

و برای استفاده از آن خواهیم داشت:

```
using System;
using System.IO;
using System.Windows.Forms;
using PdfThumbnail.Lib;

namespace PdfThumbnail
{
    class Program
    {
        static void Main(string[] args)
        {
            var pdfPath = Application.StartupPath + @"\test.pdf";
            PdfToImage.AllPdfPagesToPng((pageImageData, pageNumber, numPages) =>
                {
                    Console.WriteLine("Page {0}/{1}", pageNumber, numPages);
                    File.WriteAllBytes(string.Format("{0}\\page-{1}.png", Application.StartupPath,
pageNumber), pageImageData);
                }, pdfPath);
        }
    }
}
```

کدهای این قسمت را از اینجا نیز می‌توانید دریافت کنید:

PdfThumbnail.zip

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۱۲ ۰:۲۰

امکان دریافت SDK فوق با IP ایرانی نیست. آن‌را از اینجا هم می‌توانید دریافت کنید.

نویسنده: molana11
تاریخ: ۱۳۹۲/۰۱/۲۲ ۱۴:۴۷

با سلام.
می توان از این مثال در وب نیز استفاده کرد؟
باتشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۲ ۱۵:۳

- اگر سرور دار خودتون هستید و می‌تونید روی سرور وابستگی‌های مربوط به نگارش کامل Adobe Acrobat را نصب کنید و همچنین برنامه در حالت Full trust اجرا می‌شود؛ بله.
- راه حل‌های دیگری هم هستند که در قسمت اشتراک‌های سایت مطرح شدند .

نویسنده: molana11
تاریخ: ۱۳۹۲/۰۱/۲۲ ۱۵:۱۰

مهندس جان.
کیفیت تصویر تولیدی را چگونه می‌توان بالا برد؟
باتشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۲ ۱۶:۲۹

- کیفیت بالایی داره.
- حداکثر از ResizeImage استفاده نکنید (مستقیما از Bitmap تولیدی استفاده کنید) یا آن‌را تغییر دهید. در کل متد ResizeImage هم بر اساس تولید تصویر با کیفیت بالا تنظیم شده.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۶:۳۲

با سلام . روش دیگری که نیاز به نصب کتابخانه جانبی دیگر نداشته باشد و بتوان از آن تحت وب نیز استفاده کرد (برای هر دو ورژن x64 و x86) را چه توصیه میکنید؟ با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۷:۶

کتابخانه‌ی GhostScript هم هست . یک سری ابزار دیگر هم در اینجا لیست شدند.

عنوان: **استفاده از HTML برای تهیه قالب‌های سفارشی ستون‌ها در PdfReport**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۱۸ ۱۵:۵
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp, PdfReport

فرض کنید که لیستی از کاربران را به همراه نام و تصاویر آن‌ها داریم. قصد داریم این اطلاعات را در یک سلول نمایش دهیم و نه اینکه هر کدام را در سلول‌های جداگانه‌ای قرار دهیم. روش متداول انجام اینکار تعریف یک قالب سلول سفارشی با پیاده سازی اینترفیس IColumnItemsTemplate است. راه میانبری نیز برای حل این مساله وجود دارد:

```
columns.AddColumn(column =>
                {
                    column.PropertyName("User");
                    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
                    column.IsVisible(true);
                    column.Order(1);
                    column.Width(3);
                    column.HeaderCell("User");
                    column.CalculatedField(list =>
                        {
                            var user = list.GetSafeStringValueOf("User");
                            var photo = new Uri(list.GetSafeStringValueOf("Photo"));
                            var image = string.Format("<img src='{0}' />", photo);
                            return
                                   @"<table style='width: 100%;'>
                                                <tr>
                                                    <td>" + user + @"</td>
                                                </tr>
                                                <tr>
                                                    <td>" + image + @"</td>
                                                </tr>
                                      </table>
                                     ";
                        });
                    column.ColumnItemsTemplate(template =>
                        {
                            template.Html(); // Using iTextSharp's limited HTML to PDF capabilities
(HTMLWorker class).
                        });
                });
```

می‌توان از قابلیت‌های محدود تبدیل HTML به PDF موجود در کلاس HTMLWorker استفاده کرد. البته نباید انتظار زیادی از این کلاس داشت، اما برای اینگونه مقاصد بسیار مفید است. در اینجا به کمک یک CalculatedField، مقدار جدید سلول را که یک جدول HTMLایی است، به منبع داده مورد استفاده تزریق می‌کنیم. سپس با استفاده از قالب Html، آن‌را پردازش و نمایش خواهیم داد.
کدهای کامل این مثال را در اینجا می‌توانید ملاحظه کنید: (^)

عنوان: **بهینه سازی حجم فایل PDF تولیدی در حین کار با تصاویر در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۵ ۲۱:۵
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp, PdfReport

ابتدا مثال ساده زیر را درنظر بگیرید:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace OptimizeImageSizes
{
    class Program
    {
        static void Main(string[] args)
        {
            test1();
            test2();
        }

        private static void test2()
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test2.pdf",
FileMode.Create));
                pdfDoc.Open();

                var table = new PdfPTable(new float[] { 1, 2 });
                table.AddCell(Image.GetInstance("myImage.png"));
                table.AddCell(Image.GetInstance("myImage.png"));
                pdfDoc.Add(table);
            }

            Process.Start("test2.pdf");
        }

        private static void test1()
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test1.pdf",
FileMode.Create));
                pdfDoc.Open();

                var table = new PdfPTable(new float[] { 1, 2 });
                var image = Image.GetInstance("myImage.png");
                table.AddCell(image);
                table.AddCell(image);
                pdfDoc.Add(table);
            }

            Process.Start("test1.pdf");
        }
    }
}
```

در اینجا یک تصویر به نام myImage.png به دو طریق، به صفحه‌ای اضافه شده است:
الف) در متد test1، یک وهله از آن تهیه و دو بار به صفحه اضافه شده است.
ب) در متد test2، به نحوی متداول، هربار که نیاز به نمایش تصویری بوده، یک وهله جدید از تصویر تهیه و اضافه شده است.

نکته‌ی مهم در اینجا، حجم نهایی دو فایل حاصل است:
حجم فایل test2.pdf دقیقا دوبرابر حجم فایل test1.pdf است. علت هم به این بر می‌گردد که هر وهله جدیدی از شیء Image، صرفنظر از محتوای آن، توسط iTextSharp به صورت جداگانه‌ای در فایل pdf نهایی ثبت خواهد شد.
این مورد خصوصا در تهیه گزارشاتی که تصویری را در پشت صحنه صفحات نمایش می‌دهد یا در هدر صفحه یک تصویر مشخص و ثابتی قرار گرفته است و نیاز است این تصویر در تمام صفحات تکرار شود، بسیار مهم است و در صورت عدم رعایت نکته تهیه یک

وهله از تصاویری تکراری، می‌تواند حجم فایل را بی‌جهت تا چندمگابایت افزایش دهد.

نظرات خوانندگان

نویسنده: محمدرضا
تاریخ: ۱۳۹۱/۰۸/۲۸ ۹:۴۸

سلام
آیا این امکان برای تصویر به کاربرده شده در htmlheader هم فراهم است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۸ ۱۰:۵۲

مطلب فوق در مورد iTextSharp بود.
در PdfReport متد header.CacheHeader وجود دارد که کل هدر را کش می‌کند (حالت پیش فرض است) و اشیاء آن را یکبار محاسبه و به صفحات اضافه خواهد کرد (در آخرین نگارش موجود در SVN). به این ترتیب فرقی نمی‌کند که هدر سفارشی است یا هر نوع پیش فرض دیگری. برای تمام آن‌ها کش توکار اعمال خواهد شد.
اگر خواستید به ازای صفحات مختلف هدرهای مختلفی داشته باشید نیاز است header.CacheHeader را false کنید. در این حالت بهینه سازی صورت نخواهد گرفت.

عنوان: **فشرده سازی حجم فایل‌های PDF توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۰ ۱۷:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp, PdfReport

پیشتر در سایت جاری مطلبی را در مورد « بهینه سازی حجم فایل PDF تولیدی در حین کار با تصاویر در iTextSharp » مطالعه کرده‌اید. خلاصه آن به این نحو است که می‌توان در یک فایل PDF، ده‌ها تصویر را که تنها به یک فایل فیزیکی اشاره می‌کنند قرار داد. به این ترتیب حجم فایل نهایی تا حد بسیار قابل ملاحظه‌ای کاهش می‌یابد. البته آن مطلب در مورد تولید یک فایل PDF جدید صدق می‌کند. اما در مورد فایل‌های PDF موجود و از پیش آماده شده چطور؟

سؤال: آیا در فایل PDF ما تصاویر تکراری وجود دارند؟

نحوه یافتن تصاویر تکراری موجود در یک فایل PDF را به کمک iTextSharp در کدهای ذیل ملاحظه می‌کنید:

```
public static int FindDuplicateImagesCount(string pdfFileName)
        {
            int count = 0;
            var pdf = new PdfReader(pdfFileName);

            var md5 = new MD5CryptoServiceProvider();
            var enc = new UTF8Encoding();
            var imagesHashList = new List<string>();

            int intPageNum = pdf.NumberOfPages;
            for (int i = 1; i <= intPageNum; i++)
            {
                var page = pdf.GetPageN(i);
                var resources = PdfReader.GetPdfObject(page.Get(PdfName.RESOURCES)) as PdfDictionary;
                if (resources == null) continue;

                var xObject = PdfReader.GetPdfObject(resources.Get(PdfName.XOBJECT)) as PdfDictionary;
                if (xObject == null) continue;

                foreach (var name in xObject.Keys)
                {
                    var pdfObject = xObject.Get(name);
                    if (!pdfObject.IsIndirect()) continue;

                    var imgObject = PdfReader.GetPdfObject(pdfObject) as PdfDictionary;
                    if (imgObject == null) continue;

                    var subType = PdfReader.GetPdfObject(imgObject.Get(PdfName.SUBTYPE)) as PdfName;
                    if (subType == null) continue;

                    if (!PdfName.IMAGE.Equals(subType)) continue;

                    byte[] imageBytes = PdfReader.GetStreamBytesRaw((PRStream)imgObject);
                    var md5Hash = enc.GetString(md5.ComputeHash(imageBytes));

                    if (!imagesHashList.Contains(md5Hash))
                    {
                        imagesHashList.Add(md5Hash);
                    }
                    else
                    {
                        Console.WriteLine("Found duplicate image @page: {0}.", i);
                        count++;
                    }
                }
            }

            pdf.Close();
            return count;
        }
```

در این کد، از قابلیت‌های سطح پایین PdfReader استفاده شده است. یک فایل PDF از پیش آماده، توسط این شیء گشوده شده و سپس محتویات تصاویر آن یافت می‌شوند. در ادامه هش MD5 آن‌ها محاسبه و با یکدیگر مقایسه می‌شوند. اگر هش تکراری یافت شد، یعنی تصویر یافت شده تکراری است و این فایل قابلیت بهینه سازی و کاهش حجم (قابل ملاحظه‌ای) را دارا می‌باشد.

سؤال: چگونه اشیاء تکراری یک فایل PDF را حذف کنیم؟

کلاسی در iTextSharp به نام PdfSmartCopy وجود دارد که شبیه به عملیات فوق را انجام داده و یک کپی سبک از هر صفحه را تهیه می‌کند. سپس می‌توان این کپی‌ها را کنار هم قرار داد و فایل اصلی را مجددا بازسازی کرد:

```
public class PdfSmartCopy2 : PdfSmartCopy
    {
        public PdfSmartCopy2(Document document, Stream os)
            : base(document, os)
        { }

        /// <summary>
        /// This is a forgotten feature in iTextSharp 5.3.4.
        /// Actually its PdfSmartCopy is useless without this!
        /// </summary>
        protected override PdfIndirectReference CopyIndirect(PRIndirectReference inp, bool
keepStructure, bool directRootKids)
        {
            return base.CopyIndirect(inp);
        }
    }

        public static void RemoveDuplicateObjects(string inFile, string outFile)
        {
            var document = new Document();
            var copy = new PdfSmartCopy2(document, new FileStream(outFile, FileMode.Create));
            document.Open();

            var reader = new PdfReader(inFile);

            var n = reader.NumberOfPages;
            for (int page = 0; page < n; )
            {
                copy.AddPage(copy.GetImportedPage(reader, ++page));
            }
            copy.FreeReader(reader);

            document.Close();
        }
```

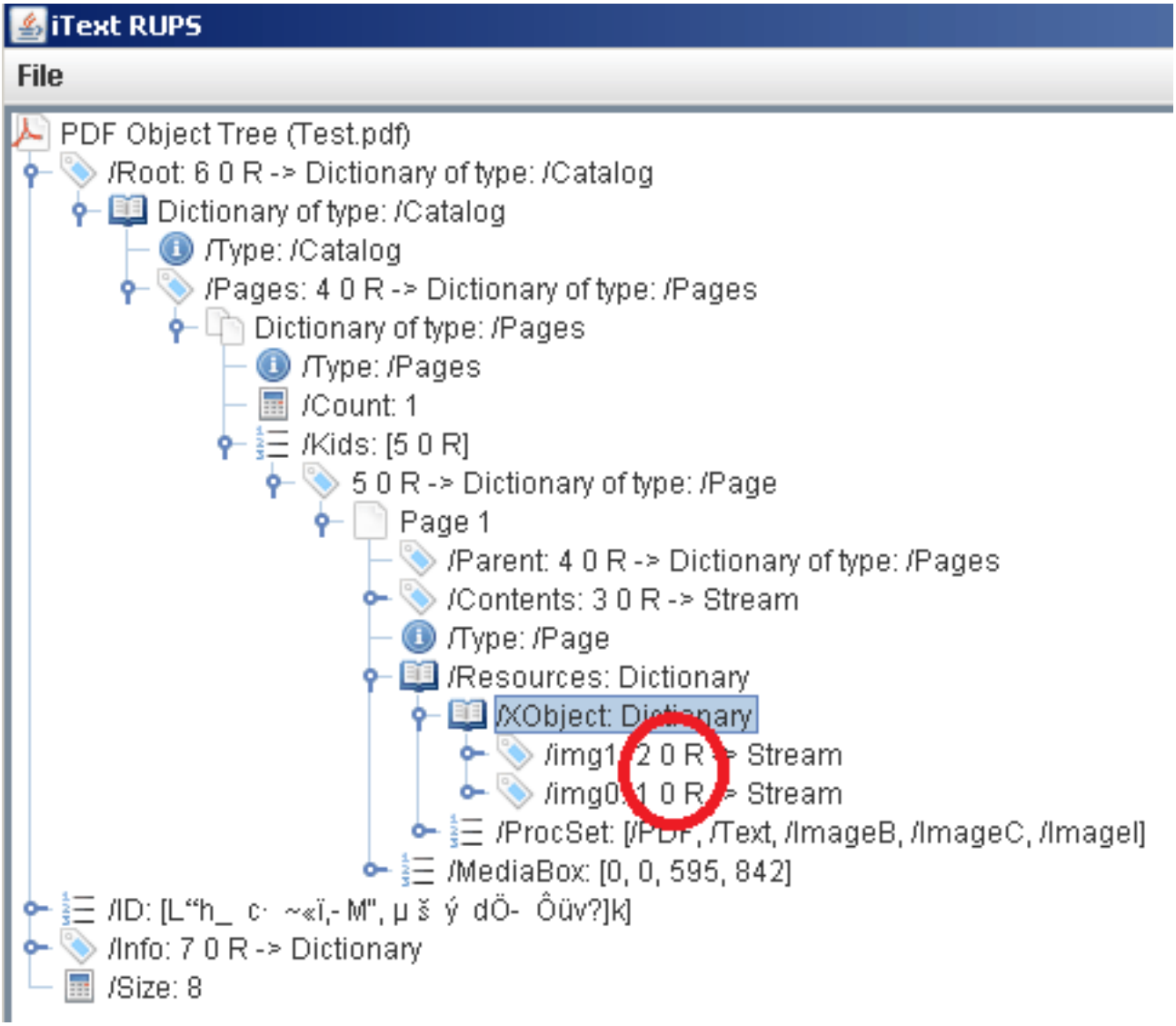
به نظر در نگارش iTextSharp 5.3.4 نویسندگان این کتابخانه اندکی فراموش کرده‌اند که باید تعدادی متد دیگر را نیز override کنند! به همین جهت کلاس PdfSmartCopy2 را مشاهده می‌کنید (اگر از نگارش‌های پایین‌تر استفاده می‌کنید، نیازی به آن نیست). استفاده از آن هم ساده است. در متد RemoveDuplicateObjects، ابتدا هر صفحه موجود توسط متد GetImportedPage دریافت شده و به وهله‌ای از PdfSmartCopy اضافه می‌شود. در پایان کار، فایل نهایی تولیدی، حاوی عناصر تکراری نخواهد بود. احتمالا برنامه‌های PDF compressor تجاری را در گوشه و کنار اینترنت دیده‌اید. متد RemoveDuplicateObjects دقیقا همان کار را انجام می‌دهد.
اگر علاقمند هستید که متد فوق را آزمایش کنید یک فایل جدید PDF را به صورت زیر ایجاد نمائید:

```
private static void CreateTestFile()
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

                var table = new PdfPTable(new float[] { 1, 2 });
                table.AddCell(Image.GetInstance("01.png"));
                table.AddCell(Image.GetInstance("01.png"));
                pdfDoc.Add(table);
            }
        }
```

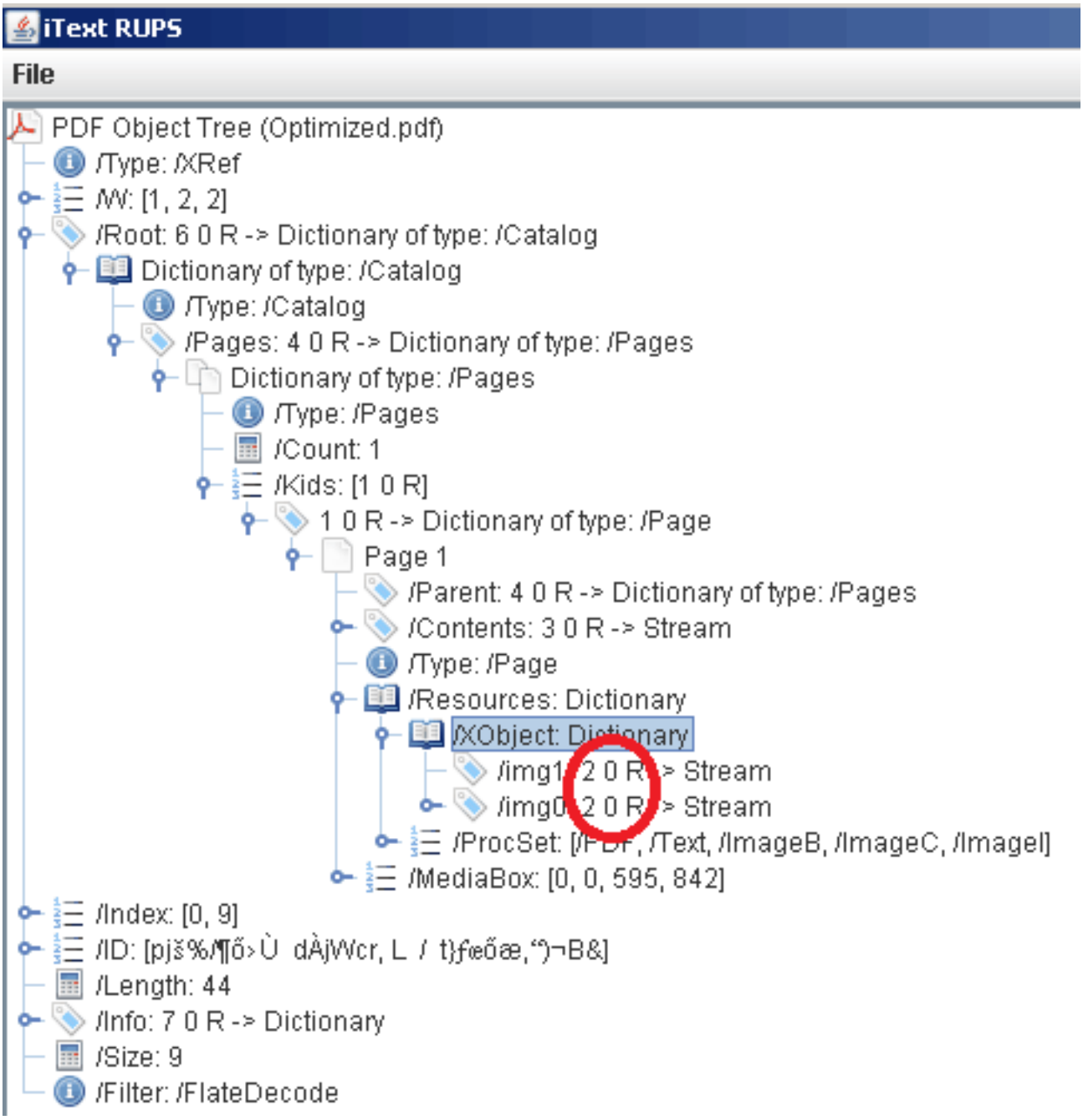
در این فایل دو وهله از تصویر 01.png به صفحه اضافه شده‌اند. بنابراین دقیقا دو تصویر در فایل نهایی تولیدی وجود خواهد داشت.

سپس متد RemoveDuplicateObjects را روی test.pdf تولید شده فراخوانی کنید. حجم فایل حاصل تقریبا نصف خواهد شد. از این جهت که PdfSmartCopy توانسته است بر اساس هش MD5 موجود در فایل PDF نهایی، موارد تکراری را یافته و ارجاعات را تصحیح کند.

در شکل زیر ساختار فایل test.pdf اصلی را ملاحظه می‌کنید. در اینجا img1 و img0 به دو stream متفاوت اشاره می‌کنند:



در شکل زیر همان test.pdf را پس از بکارگیری PDFSmartCopy ملاحظه می‌کنید:



اینبار دو تصویر داریم که هر دو به یک stream اشاره می‌کنند. تصاویر فوق به کمک برنامه iText RUPS تهیه شده‌اند.

عنوان: **حذف محدودیت‌های فایل‌های PDF توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۱ ۲۰:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

پیشنیاز

« رمزنگاری فایل‌های PDF با استفاده از کلید عمومی توسط iTextSharp »

در مطلب فوق در مورد رمزنگاری اطلاعات فایل‌های PDF به کمک iTextSharp بحث شد. در مطلب جاری به نحوه رفع این محدودیت‌ها خواهیم پرداخت.

الف) رمزگشایی با استفاده از کلمه عبور

```
using System.IO;
using iTextSharp.text.pdf;

namespace PdfDecryptor.Core
{
    public class PasswordDecryptor
    {
        public string ReadPassword { set; get; }
        public string PdfPath { set; get; }
        public string OutputPdf { set; get; }

        public void DecryptPdf()
        {
            PdfReader.unethicalreading = true;

            PdfReader reader;
            if (string.IsNullOrWhiteSpace(ReadPassword))
                reader = new PdfReader(PdfPath);
            else
                reader = new PdfReader(PdfPath, System.Text.Encoding.UTF8.GetBytes(ReadPassword));

            using (var stamper = new PdfStamper(reader, new FileStream(OutputPdf, FileMode.Create)))
            {
                stamper.Close();
            }
        }
    }
}
```

کلاس فوق دوکاربرد را می‌تواند به همراه داشته باشد:
- اگر PDF ایی صرفا دارای محدودیت چاپ بوده و این قابلیت ویژه آن غیرفعال شده است، فقط کافی است مسیر فایل PDF موجود (PdfPath) و مسیر فایل جدیدی که قرار است تولید شود (OutputPdf) ذکر گردد. خروجی فایلی خواهد بود که هیچگونه محدودیتی ندارد. این مساله هم صرفا توسط PdfReader.unethicalreading میسر شده است. به عبارتی ذکر و تنظیم edit password در فایل‌های PDF فاقد امنیت است. همین اندازه که PdfReader می‌تواند فایلی را بخواند، امکان تهیه یک کپی بدون محدودیت از آن توسط PdfStamper وجود خواهد داشت.
در مورد ReadPassword در پیشنیاز ذکر شده ، توضیحات کافی به همراه تصویر وجود دارد؛ حالت خاصی که کاربران برای مشاهده محتویات فایل نیاز خواهند داشت تا کلمه‌ی عبور مرتبط را وارد نمایند. در اینجا ذکر ReadPassword الزامی است. خروجی نهایی کلاس فوق رفع کامل این محدودیت است.

ب) رمزگشایی توسط کلید عمومی

```
using System.IO;
using iTextSharp.text.pdf;

namespace PdfDecryptor.Core
{
    public class Decryptor
    {
        public string PfxPath { set; get; }
```

```
        public string PfxPassword { set; get; }
        public string InputPdf { set; get; }
        public string OutputPdf { set; get; }

        public void DecryptPdf()
        {
            var certs = new PfxReader().ReadCertificate(PfxPath, PfxPassword);
            var reader = new PdfReader(InputPdf, certs.X509Certificates[0], certs.PrivateKey);
            using (var stamper = new PdfStamper(reader, new FileStream(OutputPdf, FileMode.Create)))
            {
                stamper.Close();
            }
        }
    }
}
```

در اینجا کدهای کامل رمزگشایی فایل PDF ایی که توسط فایل‌های مخصوص PFX رمزنگاری شده است را مشاهده می‌کنید. کلاس PfxReader آن در پیشنیاز بحث موجود است.
در این حالت مسیر فایل PFX به همراه کلمه عبور آن (PfxPassword) باید مشخص شود. خروجی فایلی است بدون محدودیت خاصی.

پ.ن.
این مثال را به صورت یک فایل اجرایی از اینجا می‌توانید دریافت کنید.

عنوان: **تهیه پردازنده‌های سفارشی برای HTMLWorker کتابخانه iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۳ ۱۰:۵
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

پیشنیاز

« تبدیل HTML به PDF با استفاده از کتابخانه‌ی iTextSharp »

هرچند کلاس HTMLWorker دیگر توسعه نخواهد یافت (با کتابخانه XML Worker جایگزین شده‌است)، اما برای تبدیل یک سری از کارهای ابتدایی بسیار مناسب است. در این بین اگر تگ خاصی توسط کلاس HTMLWorker پشتیبانی نشود یا پیاده سازی آن ناقص باشد، امکان جایگزین کردن کامل آن با پیاده سازی اینترفیس IHTMLTagProcessor وجود دارد. در کدهای ذیل نحوه جایگزین کردن پردازش کننده تصاویر آن‌را ملاحظه می‌کنید. در اینجا پشتیبانی از تصاویر base64 مدفون شده در صفحات html به آن اضافه شده است:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.html;
using iTextSharp.text.html.simpleparser;
using iTextSharp.text.pdf;

namespace CustomHtmlWorkerTag
{
    /// <summary>
    /// Our custom HTML Tag to add an IElement.
    /// </summary>
    public class CustomImageHTMLTagProcessor : IHTMLTagProcessor
    {
        /// <summary>
        /// Tells the HTMLWorker what to do when a close tag is encountered.
        /// </summary>
        public void EndElement(HTMLWorker worker, string tag)
        {
        }

        /// <summary>
        /// Tells the HTMLWorker what to do when an open tag is encountered.
        /// </summary>
        public void StartElement(HTMLWorker worker, string tag, IDictionary<string, string> attrs)
        {
            Image image;
            var src = attrs["src"];

            if (src.StartsWith("data:image/"))
            {
                // data:[<MIME-type>][;charset=<encoding>][;base64],<data>
                var base64Data = src.Substring(src.IndexOf(",") + 1);
                var imagedata = Convert.FromBase64String(base64Data);
                image = Image.GetInstance(imagedata);
            }
            else
            {
                image = Image.GetInstance(src);
            }

            worker.UpdateChain(tag, attrs);
            worker.ProcessImage(image, attrs);
            worker.UpdateChain(tag);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf", FileMode.Create));
                pdfDoc.Open();
```

```
                FontFactory.Register("c:\\windows\\fonts\\tahoma.ttf");

                var tags = new HTMLTagProcessors();
                // Replace the built-in image processor
                tags[HtmlTags.IMG] = new CustomImageHTMLTagProcessor();

                var html = "<img alt=''
src='
bWFnZVJlYWR5ccllPAAAAodJREFUeNpsk0tME1EUhv87UwlCREhRFpi4cGMMRrTE4MaoxBhAsDyMssFHfCQu3BlXGuNKNy5NmqALoqE
EMJWCgEUjYojllSpofIUNBNqmIKU6OnQennunUxvgJF86957z/+d27hkGigMlDJfOAmV7AcYsKGqIZljRSvhNE+CMTwEtXmBy2gQb7m
CQJUBKkTIQYtfJYCNMAxO9hzq5CYmFiWFY6ISE9VFLRedc1SONeqwf+uJLuKreNPI9nltbLG0orhpqUCM90DRVoEbJ5MSLho1MMg1O0
bHOuyoD9crCcxL+xa0HqwL+rEQHsb/CW89reO1aAyEuq+yp+zXvg66rgng8LrDXSmwYpUc8dZkmDsJNL+NCeVVXbWK+O32cpJ7E6Ogk
wuEwrl8phaHrVsfYD+x03XTPjN3nzZnD0HGxvPppTSLcLwo0I4lldRFK8jdCoZBlJquAbBnr0BD9GUTRvubahclW5qDukqkpIqlodGQ
1At3UxZXaIUvauqsyjBV+jZJEJ3s83HO5j+UWI7E6C4mp2EQCTixyV2CvbbKzNmN2zNfHtbzPM3p4FOy/M5CXtwsOKZmmsOi2IHMvyy
FhJhgY4BqutQ/aRRstocEngZzswnQnO+x1lqTjy8hIgNdyDc+x5nomxrKJhpcSp2lSrx48WlZhGArynG5hsLLoE7/jQ59f0aR7ZBkdb
f7U6Ge+mKYaBvdx8wwZXjtWvfswfTrp3Over29J8NAXYO1t/v/7csZA5U5/Q35nH+aKt8OMR2POPSUFOyRmorvje3BiCt4b9zBANTmw
GvP/aMoZRluJbURB8APmnPlQliNLzk8flxbeh9Du8eId5bYQ2SnxH36b/wQYABNFRsIaESsTAAAAAElFTkSuQmCC' />";

                var styles = new StyleSheet();
                styles.LoadTagStyle(HtmlTags.BODY, HtmlTags.FONTFAMILY, "tahoma");
                styles.LoadTagStyle(HtmlTags.BODY, HtmlTags.ENCODING, "Identity-H");

                PdfPCell pdfCell = new PdfPCell { Border = 0 };
                pdfCell.RunDirection = PdfWriter.RUN_DIRECTION_LTR;

                using (var reader = new StringReader(html))
                {
                    var parsedHtmlElements = HTMLWorker.ParseToList(reader, styles, tags, null);

                    foreach (var htmlElement in parsedHtmlElements)
                    {
                        pdfCell.AddElement(htmlElement);
                    }
                }

                var table1 = new PdfPTable(1);
                table1.AddCell(pdfCell);
                pdfDoc.Add(table1);
            }

            Process.Start("Test.pdf");
        }
    }
}
```

همانطور که ملاحظه می‌کنید، پس از پیاده سازی اینترفیس IHTMLTagProcessor و تهیه یک پردازش کننده جدید که اینبار می‌تواند تصاویر شروع شده با data:image را مورد استفاده قرار دهد، برای معرفی آن به کتابخانه HTMLWorker فقط کافی است وهله‌ای از HTMLTagProcessors موجود را ایجاد نمائیم و سپس در این Dictionary ، نمونه قدیمی را جایگزین کنیم:

```
var tags = new HTMLTagProcessors();
// Replace the built-in image processor
tags[HtmlTags.IMG] = new CustomImageHTMLTagProcessor();
```

در ادامه فقط کافی است لیست جدید پردازنده‌ها را به متد ParseToList ارسال نمائیم تا مورد استفاده قرار گیرد:

```
HTMLWorker.ParseToList(reader, styles, tags, null)
```

عنوان: **نحوه ذخیره شدن متن در فایل‌های PDF**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۳۰ ۱۳:۳۸
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp, PdfReport, PDF

تبدیل بی عیب و نقص یک فایل PDF (انواع و اقسام آن‌ها) به متن قابل درک بسیار مشکل است. در ادامه بررسی خواهیم کرد که چرا.
برخلاف تصور عموم، ساختار یک صفحه PDF شبیه به یک صفحه فایل Word نیست. این صفحات درحقیقت نوعی Canvas برای نقاشی هستند. در این بوم نقاشی، شکل، تصویر، متن و غیره در مختصات خاصی قرار خواهند گرفت. حتی کلمه «متن» می‌تواند به صورت سه حرف در سه مختصات خاص یک صفحه PDF نقاشی شود. برای درک بهتر این مورد نیاز است سورس یک صفحه PDF را بررسی کرد.

نحوه استخراج سورس یک صفحه PDF

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace TestReaders
{
    class Program
    {
        static void writePdf()
        {
            using (var document = new Document(PageSize.A4))
            {
                var writer = PdfWriter.GetInstance(document, new FileStream("test.pdf",
FileMode.Create));
                document.Open();

                document.Add(new Paragraph("Test"));

                PdfContentByte cb = writer.DirectContent;
                BaseFont bf = BaseFont.CreateFont();
                cb.BeginText();
                cb.SetFontAndSize(bf, 12);
                cb.MoveText(88.66f, 367);
                cb.ShowText("ld");
                cb.MoveText(-22f, 0);
                cb.ShowText("Wor");
                cb.MoveText(-15.33f, 0);
                cb.ShowText("llo");
                cb.MoveText(-15.33f, 0);
                cb.ShowText("He");
                cb.EndText();

                PdfTemplate tmp = cb.CreateTemplate(250, 25);
                tmp.BeginText();
                tmp.SetFontAndSize(bf, 12);
                tmp.MoveText(0, 7);
                tmp.ShowText("Hello People");
                tmp.EndText();
                cb.AddTemplate(tmp, 36, 343);
            }

            Process.Start("test.pdf");
        }

        private static void readPdf()
        {
            var reader = new PdfReader("test.pdf");
            int intPageNum = reader.NumberOfPages;
            for (int i = 1; i <= intPageNum; i++)
            {
                byte[] contentBytes = reader.GetPageContent(i);
                File.WriteAllBytes("page-" + i + ".txt", contentBytes);
            }
            reader.Close();
        }
```

```
        static void Main(string[] args)
        {
            writePdf();
            readPdf();
        }
    }
}
```

فایل PDF تولیدی حاوی سه عبارت کامل و مفهوم می‌باشد:

Test

Hello World
Hello People

اگر علاقمند باشید که سورس واقعی صفحات یک فایل PDF را مشاهده کنید، نحوه انجام آن توسط کتابخانه iTextSharp به صورت فوق است.
هرچند متد GetPageContent آرایه‌ای از بایت‌ها را بر می‌گرداند، اما اگر حاصل نهایی را در یک ادیتور متنی باز کنیم، قابل مطالعه و خواندن است. برای مثال، سورس مثال فوق (محتوای فایل page-1.txt تولید شده) به نحو زیر است:

```
q
BT
36 806 Td
0 -18 Td
/F1 12 Tf
(Test)Tj
0 0 Td
ET
Q
BT
/F1 12 Tf
```

```
88.66 367 Td
(ld)Tj
-22 0 Td
(Wor)Tj
-15.33 0 Td
(llo)Tj
-15.33 0 Td
(He)Tj
ET
q 1 0 0 1 36 343 cm /Xf1 Do Q
```

و تفسیر این عملگرها به این ترتیب است:

```
SaveGraphicsState(); // q
BeginText(); // BT
MoveTextPos(36, 806); // Td
MoveTextPos(0, -18); // Td
SelectFontAndSize("/F1", 12); // Tf
ShowText("(Test)"); // Tj
MoveTextPos(0, 0); // Td
EndTextObject(); // ET
RestoreGraphicsState(); // Q
BeginText(); // BT
SelectFontAndSize("/F1", 12); // Tf
MoveTextPos(88.66, 367); // Td
ShowText("(ld)"); // Tj
MoveTextPos(-22, 0); // Td
ShowText("(Wor)"); // Tj
MoveTextPos(-15.33, 0); // Td
ShowText("(llo)"); // Tj
MoveTextPos(-15.33, 0); // Td
ShowText("(He)"); // Tj
EndTextObject(); // ET
SaveGraphicsState(); // q
TransMatrix(1, 0, 0, 1, 36, 343); // cm
XObject("/Xf1"); // Do
RestoreGraphicsState(); // Q
```

همانطور که ملاحظه می‌کنید کلمه Test به مختصات خاصی انتقال داده شده و سپس به کمک اطلاعات فونت F1، ترسیم می‌شود. تا اینجا استخراج متن از فایل‌های PDF ساده به نظر می‌رسد. باید به دنبال Tj گشت و حروف مرتبط با آن‌را ذخیره کرد. اما در مورد «ترسیم» عبارات hello world و hello people اینطور نیست. عبارت hello world به حروف متفاوتی تقسیم شده و سپس در مختصات مشخصی ترسیم می‌گردد. عبارت hello people به صورت یک شیء ذخیره شده در قسمت منابع فایل PDF، بازیابی و نمایش داده می‌شود و اصلا در سورس صفحه جاری وجود ندارد.

این تازه قسمتی از نحوه عملکرد فایل‌های PDF است. در فایل‌های PDF می‌توان قلم‌ها را مدفون ساخت. همچنین این قلم‌ها نیز تنها زیر مجموعه‌ای از قلم اصلی مورد استفاده هستند. برای مثال اگر عبارت Test قرار است نمایش داده شود، فقط اطلاعات T، e و s در فایل نهایی PDF قرار می‌گیرند. به علاوه امکان تغییر کلی شماره Glyph متناظر با هر حرف نیز توسط PDF writer وجود دارد. به عبارتی الزامی نیست که مشخصات اصلی فونت حتما حفظ شود.

شاید بعضی از PDFهای فارسی را دیده باشید که پس از کپی متن آن‌ها در برنامه Adobe reader و سپس paste آن در جایی دیگر، متن حاصل قابل خواندن نیست. علت این است که نحوه ذخیره سازی قلم مورد استفاده کاملا تغییر کرده است و برای بازیابی متن اینگونه فایل‌ها، استفاده از OCR ساده‌ترین روش است. برای نمونه در این قلم جدید مدفون شده، دیگر شماره کاراکتر 0x41 مساوی A نیست. بنابر سلیقه PDF writer این شماره به Glyph دیگری انتساب داده شده و چون قلم و مشخصات هندسی Glyph مورد استفاده در فایل PDF ذخیره می‌شود، برای نمایش این نوع فایل‌ها هیچگونه مشکلی وجود ندارد. اما متن آن‌ها به سادگی قابل بازیابی نیست.

عنوان: **استخراج متن از فایل‌های PDF توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۰۱ ۱۲:۵۳
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp, PDF

پیشنیاز

نحوه ذخیره شدن متن در فایل‌های PDF

حتما نیاز است پیشنیاز فوق را یکبار مطالعه کنید تا علت خروجی‌های متفاوتی را که در ادامه ملاحظه خواهید نمود، بهتر مشخص شوند. همچنین فایل PDF ایی که مورد بررسی قرار خواهد گرفت، همان فایلی است که توسط متد writePdf ذکر شده در پیشنیاز تهیه شده است.

دو کلاس متفاوت برای استخراج متن از فایل‌های PDF در iTextSharp وجود دارند:

الف) SimpleTextExtractionStrategy

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
using iTextSharp.text.pdf.parser;

namespace TestReaders
{
    class Program
    {
        private static void readPdf1()
        {
            var reader = new PdfReader("test.pdf");
            int intPageNum = reader.NumberOfPages;
            for (int i = 1; i <= intPageNum; i++)
            {
               var text = PdfTextExtractor.GetTextFromPage(reader, i, new
SimpleTextExtractionStrategy());
                File.WriteAllText("page-" + i + "-text.txt", text);
            }
            reader.Close();
        }

        static void Main(string[] args)
        {
            readPdf1();
        }
    }
}
```

مثال فوق، متن موجود در تمام صفحات یک فایل PDF را در فایل‌های txt جداگانه‌ای ثبت می‌کند. برای نمونه اگر از PDF پیشنیاز یاد شده استفاده کنیم، خروجی آن به نحو زیر خواهد بود:

```
Test
ld Wor llo He
Hello People
```

علت آن نیز پیشتر بررسی گردید. متن، در این فایل ویژه در مختصات خاصی ترسیم شده است. حاصل از دیدگاه خواننده نهایی بسیار خوانا است؛ اما خروجی hello world متنی جالبی از آن استخراج نمی‌شود. SimpleTextExtractionStrategy دقیقا بر اساس همان عملگرهای Tj و همچنین منابع صفحه، عبارات را یافته و سر هم می‌کند.

ب) LocationTextExtractionStrategy

همان مثال قبل را درنظر بگیرید، اینبار به شکل زیر:

```
private static void readPdf2()
        {
            var reader = new PdfReader("test.pdf");
            int intPageNum = reader.NumberOfPages;
            for (int i = 1; i <= intPageNum; i++)
            {
                var text = PdfTextExtractor.GetTextFromPage(reader, i, new
LocationTextExtractionStrategy());
                File.WriteAllText("page-" + i + "-text.txt", text);
            }
            reader.Close();
        }
```

کلاس LocationTextExtractionStrategy هوشمندتر عمل کرده و بر اساس عملگرهای هندسی یک فایل PDF، سعی می‌کند جملات و حروف را کنار هم قرار دهد و در نهایت خروجی متنی بهتری را تولید کند. برای نمونه اینبار خروجی متنی حاصل به صورت زیر خواهد بود:

```
Test
Hello World
Hello People
```

این خروجی با آنچه که در صفحه نمایش داده می‌شود تطابق دارد.

استخراج متون فارسی از فایل‌های PDF توسط iTextSharp

روش‌های فوق با PDFهای فارسی هم کار می‌کنند اما خروجی حاصل آن مفهوم نیست و نیاز به پردازش ثانوی دارد. ابتدا مثال زیر را درنظر بگیرید:

```
static void writePdf2()
        {
            using (var document = new Document(PageSize.A4))
            {
                var writer = PdfWriter.GetInstance(document, new FileStream("test.pdf",
FileMode.Create));
                document.Open();

                FontFactory.Register("c:\\windows\\fonts\\tahoma.ttf");
                var tahoma = FontFactory.GetFont("tahoma", BaseFont.IDENTITY_H);

                ColumnText.ShowTextAligned(
                            canvas: writer.DirectContent,
                            alignment: Element.ALIGN_CENTER,
                            phrase: new Phrase("تست می‌شود", tahoma),
                            x: 100,
                            y: 100,
                            rotation: 0,
                            runDirection: PdfWriter.RUN_DIRECTION_RTL,
                            arabicOptions: 0);
            }

            Process.Start("test.pdf");
        }
```

از متد فوق، برای تولید یک فایل PDF که متنی فارسی را نمایش می‌دهد استفاده خواهیم کرد. اگر متد readPdf2 را که به همراه LocationTextExtractionStrategy تعریف شده است، بر روی فایل حاصل فراخوانی کنیم، خروجی آن به صورت زیر خواهد بود:

```
ïº©ï»®ïº·ï»²ï»£ ïº–ïº´ïº–
```

برای تبدیل آن به یونیکد خواهیم داشت:

```
private static void readPdf2()
```

```
        {
            var reader = new PdfReader("test.pdf");
            int intPageNum = reader.NumberOfPages;
            for (int i = 1; i <= intPageNum; i++)
            {
                var text = PdfTextExtractor.GetTextFromPage(reader, i, new
LocationTextExtractionStrategy());
                text = Encoding.UTF8.GetString(Encoding.UTF8.GetBytes(text));
                File.WriteAllText("page-" + i + "-text.txt", text, Encoding.UTF8);
            }
            reader.Close();
        }
```

اکنون خروجی ثبت شده در فایل متنی حاصل به صورت زیر است:

```
دوشيم تست
```

دقیقا به همان نحوی است که iTextSharp و اکثر تولید کننده‌های PDF فارسی از آن استفاده می‌کنند و اصطلاحا چرخاندن حروف یا تولید Glyph mirrors صورت می‌گیرد. روش‌های زیادی برای چرخاندن حروف وجود دارند. در ادامه از روشی استفاده خواهیم کرد که خود ویندوز در کارهای داخلی‌اش از آن استفاده می‌کند:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Security;

namespace TestReaders
{
    [SuppressUnmanagedCodeSecurity]
    class GdiMethods
    {
        [DllImport("GDI32.dll")]
        public static extern bool DeleteObject(IntPtr hgdiobj);

        [DllImport("gdi32.dll", CharSet = CharSet.Auto, SetLastError = true)]
        public static extern uint GetCharacterPlacement(IntPtr hdc, string lpString, int nCount, int
nMaxExtent, [In, Out] ref GcpResults lpResults, uint dwFlags);

        [DllImport("GDI32.dll")]
        public static extern IntPtr SelectObject(IntPtr hdc, IntPtr hgdiobj);
    }

    [StructLayout(LayoutKind.Sequential)]
    struct GcpResults
    {
        public uint lStructSize;
        [MarshalAs(UnmanagedType.LPTStr)]
        public string lpOutString;
        public IntPtr lpOrder;
        public IntPtr lpDx;
        public IntPtr lpCaretPos;
        public IntPtr lpClass;
        public IntPtr lpGlyphs;
        public uint nGlyphs;
        public int nMaxFit;
    }

    public class UnicodeCharacterPlacement
    {
        const int GcpReorder = 0x0002;
        GCHandle _caretPosHandle;
        GCHandle _classHandle;
        GCHandle _dxHandle;
        GCHandle _glyphsHandle;
        GCHandle _orderHandle;

        public Font Font { set; get; }

        public string Apply(string lines)
        {
            if (string.IsNullOrWhiteSpace(lines))
                return string.Empty;
```

```
        return Apply(lines.Split('\n')).Aggregate((s1, s2) => s1 + s2);
    }

    public IEnumerable<string> Apply(IEnumerable<string> lines)
    {
        if (Font == null)
            throw new ArgumentNullException("Font is null.");

        if (!hasUnicodeText(lines))
            return lines;

        var graphics = Graphics.FromHwnd(IntPtr.Zero);
        var hdc = graphics.GetHdc();
        try
        {
            var font = (Font)Font.Clone();
            var hFont = font.ToHfont();
            var fontObject = GdiMethods.SelectObject(hdc, hFont);
            try
            {
                var results = new List<string>();
                foreach (var line in lines)
                    results.Add(modifyCharactersPlacement(line, hdc));
                return results;
            }
            finally
            {
                GdiMethods.DeleteObject(fontObject);
                GdiMethods.DeleteObject(hFont);
                font.Dispose();
            }
        }
        finally
        {
            graphics.ReleaseHdc(hdc);
            graphics.Dispose();
        }
    }

    void freeResources()
    {
        _orderHandle.Free();
        _dxHandle.Free();
        _caretPosHandle.Free();
        _classHandle.Free();
        _glyphsHandle.Free();
    }

    static bool hasUnicodeText(IEnumerable<string> lines)
    {
        return lines.Any(line => line.Any(chr => chr >= '\u00FF'));
    }

    void initializeResources(int textLength)
    {
        _orderHandle = GCHandle.Alloc(new int[textLength], GCHandleType.Pinned);
        _dxHandle = GCHandle.Alloc(new int[textLength], GCHandleType.Pinned);
        _caretPosHandle = GCHandle.Alloc(new int[textLength], GCHandleType.Pinned);
        _classHandle = GCHandle.Alloc(new byte[textLength], GCHandleType.Pinned);
        _glyphsHandle = GCHandle.Alloc(new short[textLength], GCHandleType.Pinned);
    }

    string modifyCharactersPlacement(string text, IntPtr hdc)
    {
        var textLength = text.Length;
        initializeResources(textLength);
        try
        {
            var gcpResult = new GcpResults
            {
                lStructSize = (uint)Marshal.SizeOf(typeof(GcpResults)),
                lpOutString = new String('\0', textLength),
                lpOrder = _orderHandle.AddrOfPinnedObject(),
                lpDx = _dxHandle.AddrOfPinnedObject(),
                lpCaretPos = _caretPosHandle.AddrOfPinnedObject(),
                lpClass = _classHandle.AddrOfPinnedObject(),
                lpGlyphs = _glyphsHandle.AddrOfPinnedObject(),
                nGlyphs = (uint)textLength,
                nMaxFit = 0
            };
```

```
                var result = GdiMethods.GetCharacterPlacement(hdc, text, textLength, 0, ref gcpResult,
GcpReorder);
                return result != 0 ? gcpResult.lpOutString : text;
            }
            finally
            {
                freeResources();
            }
        }
    }
}
```

از کلاس فوق در هر برنامه‌ای که راست به چپ را به نحو صحیحی پشتیبانی نمی‌کند، می‌توان استفاده کرد؛ خصوصا برنامه‌های گرافیکی.

در اینجا برای اصلاح متد readPdf2 خواهیم داشت:

```
private static void readPdf2()
        {
            var reader = new PdfReader("test.pdf");
            int intPageNum = reader.NumberOfPages;
            for (int i = 1; i <= intPageNum; i++)
            {
                var text = PdfTextExtractor.GetTextFromPage(reader, i, new
LocationTextExtractionStrategy());
                text = Encoding.UTF8.GetString(Encoding.UTF8.GetBytes(text));
                text = new UnicodeCharacterPlacement
                {
                    Font = new System.Drawing.Font("Tahoma", 12)
                }.Apply(text);
                File.WriteAllText("page-" + i + "-text.txt", text, Encoding.UTF8);
            }
            reader.Close();
        }
```

اگر خروجی متد اصلاح شده فوق را بررسی کنیم، دقیقا به «تست می‌شود» خواهیم رسید.

سؤال: آیا این روش با تمام PDFهای فارسی کار می‌کند؟

پاسخ: خیر! همانطور که در پیشنیاز مطلب جاری عنوان شد، در یک حالت خاص، PDF writer می‌تواند شماره Glyphها را کاملا عوض کرده و در فایل PDF نهایی ثبت کند. خروجی حاصل در برنامه Adobe reader خوانا است، چون نمایش را بر اساس اطلاعات هندسی Glyphها انجام می‌دهد؛ اما خروجی متنی آن به نوعی obfuscated است چون مثلا حرف A آن به کاراکتر مرسوم دیگری نگاشت شده است.

نظرات خوانندگان

نویسنده: ابراهیم بیاگوی
تاریخ: ۱۳۹۱/۱۰/۰۲ ۱۴:۳۱

بسیار عالی هست، بسیار بسیار عالی. فقط یک سوال، راه حل بدون P/Invoke هم در حال حاضر سراغ دارید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۰۲ ۱۴:۳۸

در مورد پیاده سازی «UnicodeCharacterPlacement»؟
بله. یک نمونه جاوا اسکریپتی هست که به سادگی قابل تبدیل به سی شارپ خالص است (mirror Glyphs را پیاده سازی کرده):
bidi.js

نویسنده: ابراهیم بیاگوی
تاریخ: ۱۳۹۱/۱۰/۰۲ ۱۴:۴۲

عالی. ممنون

نویسنده: هیمن روحانی
تاریخ: ۱۳۹۲/۱۱/۰۱ ۱۵:۳۸

برای استخراج متن و عکس و رندر کردن می‌تونید از این کتابخانه PdfLib.Net با چند خط کد، متن کامل فارسی رو بدون مشکل استخراج کنید. البته حجم این کتابخونه یکم زیاده چون کار اصلیش رندر کردن PDF.

```
PDFLibNet.PDFWrapper wrapper = new PDFLibNet.PDFWrapper();
            wrapper.LoadPDF(pdfPath);
            string page1Text = wrapper.Pages[1].Text;
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۱ ۱۶:۶

از MuPDF هم استفاده می‌کنه (کد خالص دات نتی نیست و استفاده ازش در برنامه‌های وب مشکل ساز هست؛ نیاز به فول تراست دارد و همچنین 32 بیتی و 64 بیتی آن باید بر اساس نوع سرور لحاظ شود).

عنوان: **اصلاح Urlها در فایل‌های PDF با استفاده از iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۸ ۱۸:۵
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp, PDF

نحوه ایجاد لینک در فایل‌های PDF به کمک iTextSharp

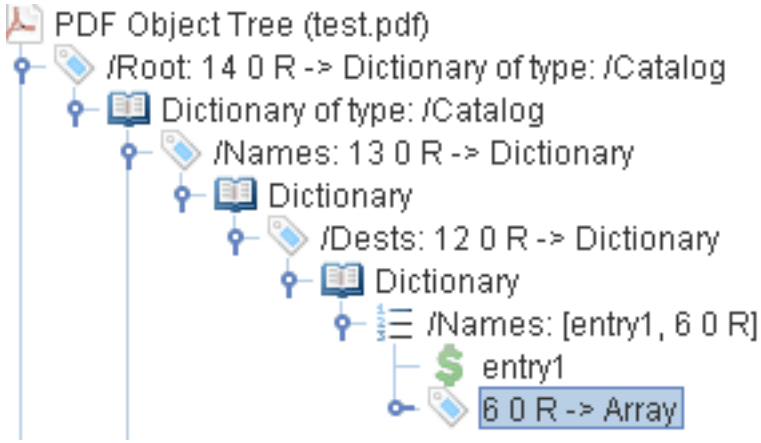
حداقل دو نوع لینک را در فایل‌های PDF می‌توان ایجاد کرد:
الف) لینک به منابع خارجی؛ مانند یک وب سایت
ب) لینک به صفحه‌ای داخل فایل PDF
در ادامه مثالی را مشاهده خواهید نمود که شامل هر دو نوع لینک است:

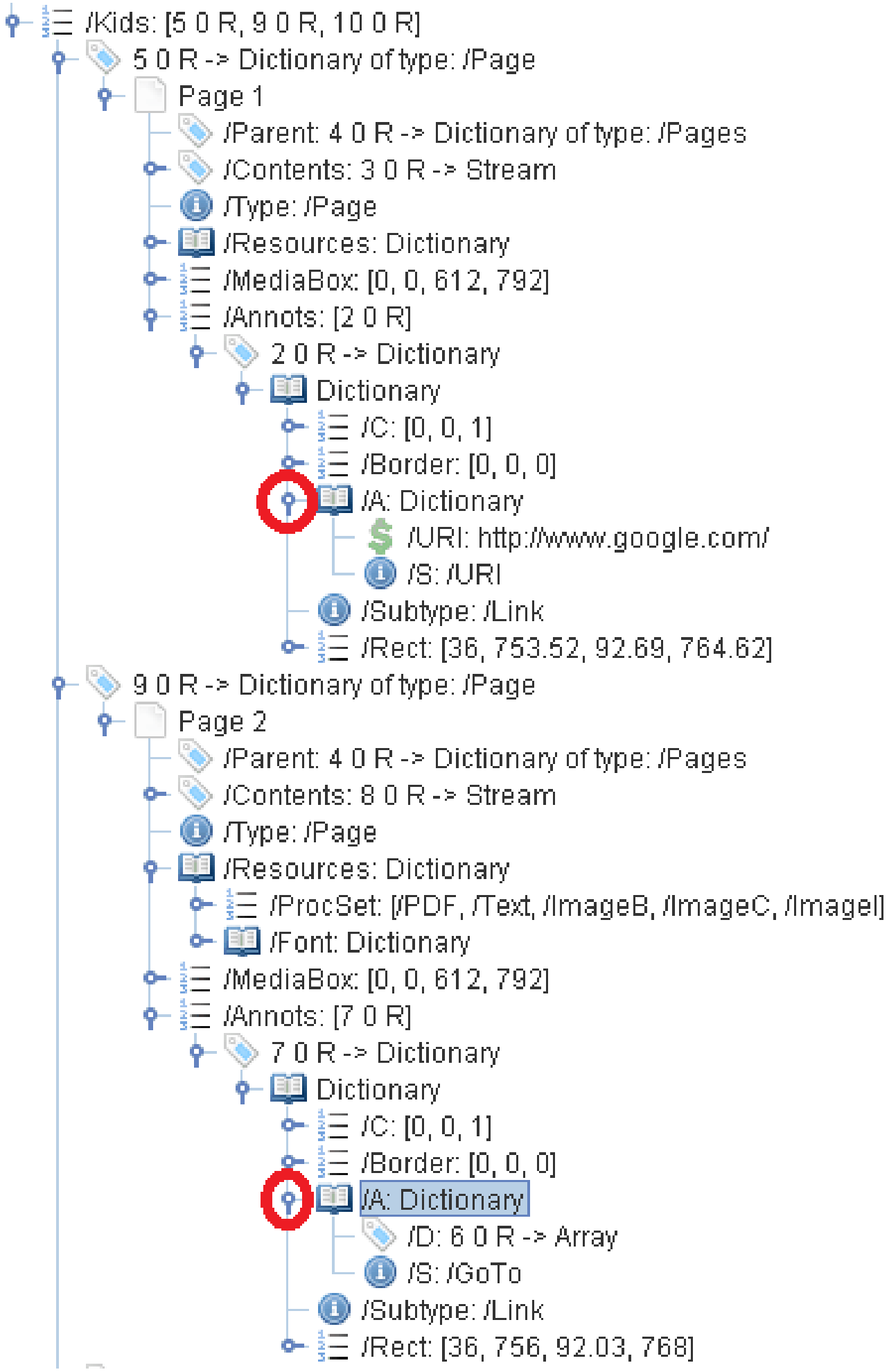
```
void WriteFile()
        {
            using (var doc = new Document(PageSize.LETTER))
            {
                using (var fs = new FileStream("test.pdf", FileMode.Create))
                {
                    using (var writer = PdfWriter.GetInstance(doc, fs))
                    {
                        doc.Open();
                        var blueFont = FontFactory.GetFont("Arial", 12, Font.NORMAL, BaseColor.BLUE);
                        doc.Add(new Chunk("Go to URL", blueFont).SetAction(new
PdfAction("http://www.google.com/", false)));

                        doc.NewPage();
                        doc.Add(new Chunk("Go to Test", blueFont).SetLocalGoto("entry1"));

                        doc.NewPage();
                        doc.Add(new Chunk("Test").SetLocalDestination("entry1"));

                        doc.Close();
                    }
                }
            }
        }
```

حاصل این مثال، یک فایل PDF است با سه صفحه. در صفحه اول لینکی به سایت Google وجود دارد. در صفحه دوم، لینکی به صفحه سوم تهیه شده است.
در صفحه سوم یک Local Destination تعبیه شده است. در صفحه دوم به کمک یک Local Goto، لینکی به این مقصد داخلی ایجاد خواهد شد.

اصلاح لینک‌ها در فایل‌های PDF

همان مثال فوق را درنظر بگیرید. فرض کنید لینک خارجی ذکر شده در ابتدای فایل را می‌خواهیم به مقصدی که در صفحه دوم ایجاد کرده‌ایم، تغییر دهیم. برای مثال خروجی PDF ایی را درنظر بگیرید که لینک‌های اصلی آن به مقالاتی در یک سایت اشاره می‌کنند. اما همین مقالات اکنون در فایل نهایی خروجی نیز قرار دارند. بهتر است این لینک‌های خارجی را به لینک‌های ارجاع دهنده به مقالات موجود در فایل اصلاح کنیم، تا استفاده از نتیجه حاصل، ساده‌تر گردد.
پیش از اینکه کدهای این قسمت را بررسی کنیم، نیاز است کمی با ساختار سطح پایین فایل‌های PDF آشنا شویم . پس از آن قادر خواهیم بود تا نسبت به اصلاح این لینک‌ها اقدام کنیم.


PDF Object Tree (test.pdf)
/Root: 14 0 R -> Dictionary of type: /Catalog
Dictionary of type: /Catalog
/Names: 13 0 R -> Dictionary
Dictionary
/Dests: 12 0 R -> Dictionary
Dictionary
/Names: [entry1, 6 0 R]
entry1
6 0 R -> Array


/Kids: [5 0 R, 9 0 R, 10 0 R]
5 0 R -> Dictionary of type: /Page
Page 1
/Parent: 4 0 R -> Dictionary of type: /Pages
/Contents: 3 0 R -> Stream
/Type: /Page
/Resources: Dictionary
/MediaBox: [0, 0, 612, 792]
/Annots: [2 0 R]
2 0 R -> Dictionary
Dictionary
/C: [0, 0, 1]
/Border: [0, 0, 0]
/A: Dictionary
/URI: http://www.google.com/
/S: /URI
/Subtype: /Link
/Rect: [36, 753.52, 92.69, 764.62]
9 0 R -> Dictionary of type: /Page
Page 2
/Parent: 4 0 R -> Dictionary of type: /Pages
/Contents: 8 0 R -> Stream
/Type: /Page
/Resources: Dictionary
/ProcSet: [/PDF, /Text, /ImageB, /ImageC, /ImageI]
/Font: Dictionary
/MediaBox: [0, 0, 612, 792]
/Annots: [7 0 R]
7 0 R -> Dictionary
Dictionary
/C: [0, 0, 1]
/Border: [0, 0, 0]
/A: Dictionary
/D: 6 0 R -> Array
/S: /GoTo
/Subtype: /Link
/Rect: [36, 756, 92.03, 768]

در تصویر اول نحوه ذخیره شدن named destinationها را در یک فایل PDF مشاهده می‌کنید.
در تصویر دوم، ساختار دو نوع لینک تعریف شده در صفحات، مشخص هستند. یکی بر اساس Uri کار می‌کند و دیگری بر اساس GoTo.
کاری را که در ادامه قصد داریم انجام دهیم، تبدیل حالت Uri به GoTo است. برای مثال، در ادامه می‌خواهیم لینک مثال فوق را ویرایش کرده و آن‌را تبدیل به لینکی نمائیم که به entry1 اشاره می‌کند. کدهای انجام اینکار را در ادامه ملاحظه می‌کنید:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using iTextSharp.text.pdf;

namespace ReplaceLinks
{
    public class ReplacePdfLinks
    {
        Dictionary<string, PdfObject> _namedDestinations;
        PdfReader _reader;

        public string InputPdf { set; get; }
        public string OutputPdf { set; get; }
        public Func<Uri, string> UriToNamedDestination { set; get; }

        public void Start()
        {
            updatePdfLinks();
            saveChanges();
        }

        private PdfArray getAnnotationsOfCurrentPage(int pageNumber)
        {
            var pageDictionary = _reader.GetPageN(pageNumber);
            var annotations = pageDictionary.GetAsArray(PdfName.ANNOTS);
            return annotations;
        }

        private static bool hasAction(PdfDictionary annotationDictionary)
        {
            return annotationDictionary.Get(PdfName.SUBTYPE).Equals(PdfName.LINK);
        }

        private static bool isUriAction(PdfDictionary annotationAction)
        {
            return annotationAction.Get(PdfName.S).Equals(PdfName.URI);
        }

        private void replaceUriWithLocalDestination(PdfDictionary annotationAction)
        {
            var uri = annotationAction.Get(PdfName.URI) as PdfString;
            if (uri == null)
                return;

            if (string.IsNullOrWhiteSpace(uri.ToString()))
                return;

            var namedDestination = UriToNamedDestination(new Uri(uri.ToString()));
            if (string.IsNullOrWhiteSpace(namedDestination))
                return;

            PdfObject entry;
            if (!_namedDestinations.TryGetValue(namedDestination, out entry))
                return;

            annotationAction.Remove(PdfName.S);
            annotationAction.Remove(PdfName.URI);

            var newLocalDestination = new PdfArray();
            annotationAction.Put(PdfName.S, PdfName.GOTO);
            var xRef = ((PdfArray)entry).First(x => x is PdfIndirectReference);
            newLocalDestination.Add(xRef);
            newLocalDestination.Add(PdfName.FITH);
            annotationAction.Put(PdfName.D, newLocalDestination);
        }
```

```
        private void saveChanges()
        {
            using (var fileStream = new FileStream(OutputPdf, FileMode.Create, FileAccess.Write,
FileShare.None))
            using (var stamper = new PdfStamper(_reader, fileStream))
            {
                stamper.Close();
            }
        }

        private void updatePdfLinks()
        {
            _reader = new PdfReader(InputPdf);
            _namedDestinations = _reader.GetNamedDestinationFromStrings();

            var pageCount = _reader.NumberOfPages;
            for (var i = 1; i <= pageCount; i++)
            {
                var annotations = getAnnotationsOfCurrentPage(i);
                if (annotations == null || !annotations.Any())
                    continue;

                foreach (var annotation in annotations.ArrayList)
                {
                    var annotationDictionary = (PdfDictionary)PdfReader.GetPdfObject(annotation);

                    if (!hasAction(annotationDictionary))
                        continue;

                    var annotationAction = annotationDictionary.Get(PdfName.A) as PdfDictionary;
                    if (annotationAction == null)
                        continue;

                    if (!isUriAction(annotationAction))
                        continue;

                    replaceUriWithLocalDestination(annotationAction);
                }
            }
        }
    }
}
```

توضیح این کدها بدون ارجاع به تصاویر ارائه شده میسر نیست. کار از متد updatePdfLinks شروع می‌شود. با استفاده از متد GetNamedDestinationFromStrings به کلیه named destinationهای تعریف شده دسترسی خواهیم داشت (تصویر اول). در ادامه Annotations هر صفحه دریافت می‌شوند. اگر به تصویر دوم دقت کنید، به ازای هر صفحه یک سری Annot وجود دارد. داخل اشیاء Annotations، لینک‌ها قرار می‌گیرند. در ادامه این لینک‌ها استخراج شده و تنها مواردی که دارای Uri هستند بررسی خواهند شد.

کار تغییر ساختار PDF در متد replaceUriWithLocalDestination انجام می‌شود. در اینجا آدرس استخراجی به استفاده کننده ارجاع شده و named destination مناسبی دریافت می‌شود. اگر این «مقصد نام دار» در مجموعه مقاصد نام دار PDF جاری وجود داشت، خواص لینک قبلی مانند Uri آن حذف شده و با GoTo به آدرس این مقصد جدید جایگزین می‌شود.

در آخر، توسط یک PdfStamper، اطلاعات تغییر کرده را در فایلی جدید ثبت خواهیم کرد.

یک نمونه از استفاده از کلاس فوق به شرح زیر است:

```
new ReplacePdfLinks
            {
                InputPdf = @"test.pdf",
                OutputPdf = "mod.pdf",
                UriToNamedDestination = uri =>
                {
                    if (uri.Host.ToLowerInvariant().Contains("google.com"))
                    {
                        return "entry1";
                    }

                    return string.Empty;
                }
            }.Start();
```

در این مثال، اگر لینکی به آدرس Google.com اشاره کند، ویرایش شده و اینبار به مقصدی داخلی به نام entry1 ختم خواهد شد.

چند نکته تکمیلی

- اگر قصد داشته باشیم تا لینکی را ویرایش کرده اما تنها Uri آن‌را تغییر دهیم، تنها کافی است URI آن‌را به نحو زیر در متد replaceUriWithLocalDestination ویرایش کنیم:

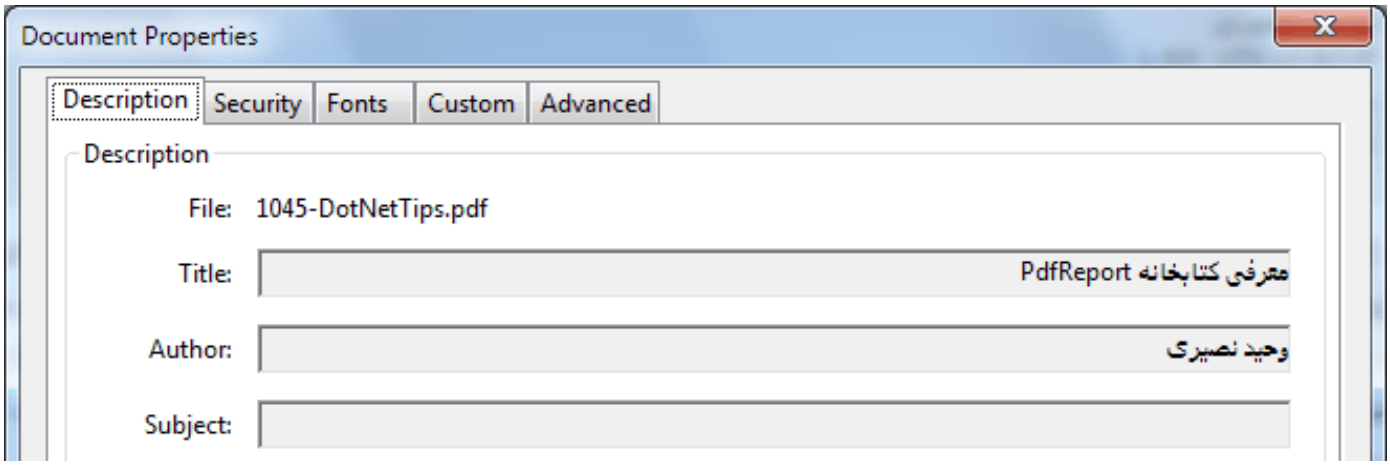
```
annotationAction.Put(PdfName.URI, new PdfString("http://www.bing.com/"));
```

- اگر بجای یک مقصد نام دار، تنها قرار است لینک موجود، به صفحه‌ای مشخص اشاره کند، تغییرات متد replaceUriWithLocalDestination به نحو زیر خواهد بود:

```
newLocalDestination.Add((PdfObject)_reader.GetPageOrigRef(pageNum: 2));
```

عنوان: **تغییر نام دسته جمعی تعدادی فایل PDF بر اساس متادیتای فایل‌ها**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۳ ۸:۵
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp, PDF

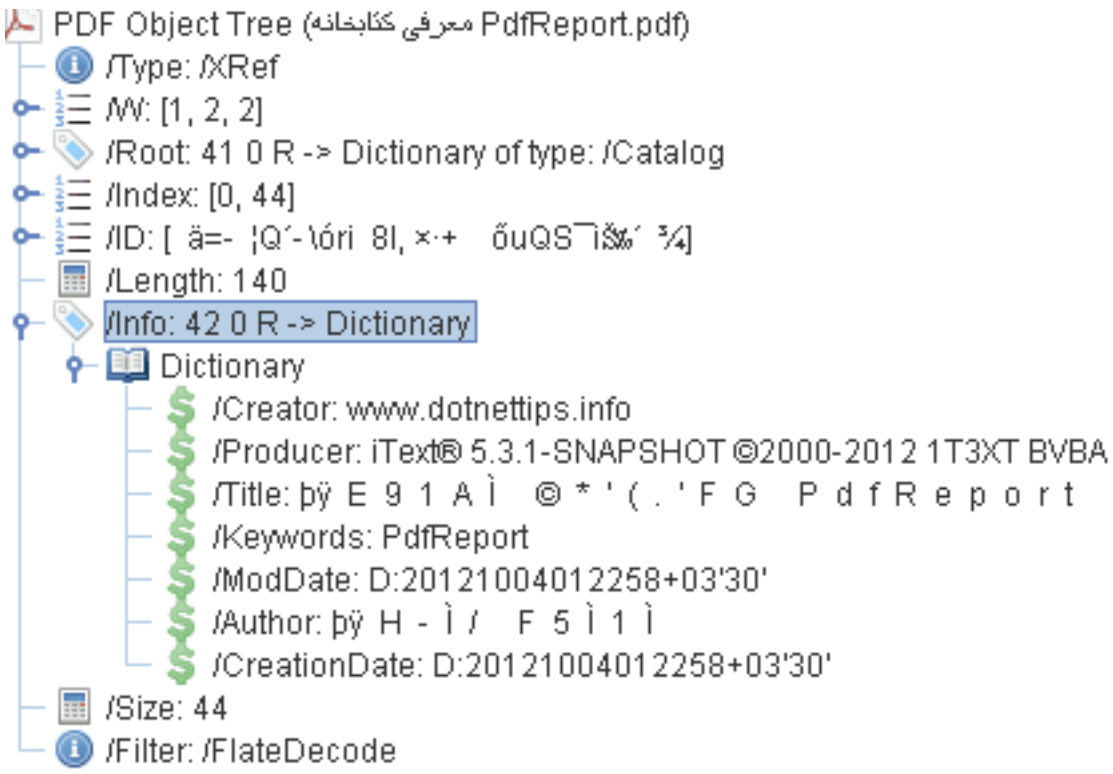
فرض کنید تعداد زیادی فایل PDF را با اسامی نامفهومی داریم. برای نظم بخشیدن و یافتن ساده‌تر مطالب شاید بهتر باشد این فایل‌ها را بر اساس عنوان اصلی ذخیره شده در فایل، تغییر نام دهیم.



امکان خواندن meta data فوق (البته در صورت وجود)، توسط iTextSharp وجود دارد. در ادامه قطعه کد ساده‌ای را ملاحظه می‌کنید که در یک پوشه، تمام فایل‌های PDF را یافته و بر اساس Title یا Subject آن‌ها، فایل موجود را تغییر نام می‌دهد:

```
using System.IO;
using iTextSharp.text.pdf;

namespace BatchRename
{
    class Program
    {
        private static string getTitle(PdfReader reader)
        {
            string title;
            reader.Info.TryGetValue("Title", out title); // Reading PDF file's meta data
            return string.IsNullOrWhiteSpace(title) ? string.Empty : title.Trim();
        }

        private static string getSubject(PdfReader reader)
        {
            string subject;
            reader.Info.TryGetValue("Subject", out subject); // Reading PDF file's meta data
            return string.IsNullOrWhiteSpace(subject) ? string.Empty : subject.Trim();
        }

        static void Main(string[] args)
        {
            var dir = @"D:\Path";
            if (!dir.EndsWith(@"\"))
                dir = dir + @"\";

            foreach (var file in Directory.GetFiles(dir, "*.pdf"))
            {
                var reader = new PdfReader(file);
                var title = getTitle(reader);
                var subject = getSubject(reader);
                reader.Close();

                string newFile = string.Empty;
                if (!string.IsNullOrWhiteSpace(title))
                {
                    newFile = dir + title + ".pdf";
```

```
                }
                else if (!string.IsNullOrWhiteSpace(subject))
                {
                    newFile = dir + subject + ".pdf";
                }

                if (!string.IsNullOrWhiteSpace(newFile))
                    File.Move(file, newFile);
            }
        }
    }
}
```



در قطعه کد فوق علت مراجعه به reader.Info، بر اساس ساختار یک فایل PDF است. در Dictionary به نام Info (تصویر فوق)، در یک سری کلید مشخص، اطلاعاتی مانند تهیه کننده، عنوان و غیره درج می‌شوند. به این ترتیب با استفاده از شیء PdfReader، فایل را گشوده، این متادیتا را خوانده و سپس بر اساس آن می‌توان فایل را تغییر نام داد.

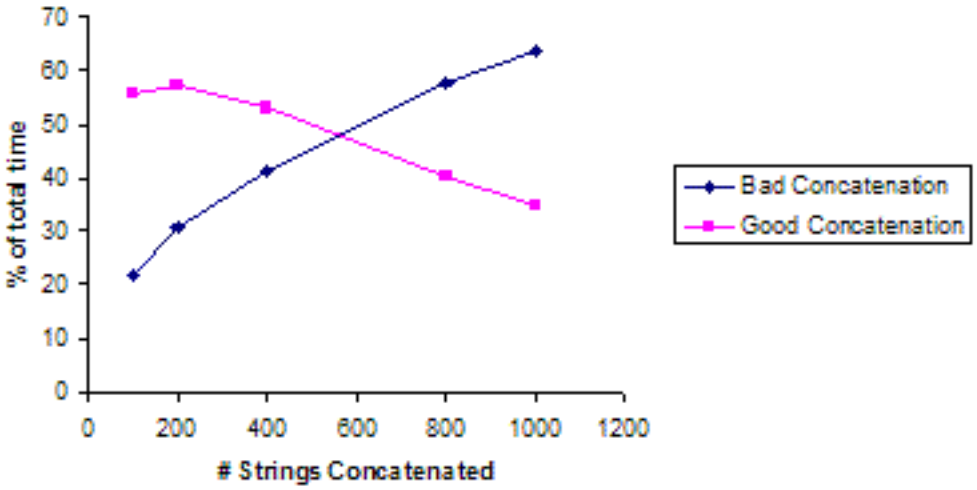
نظرات خوانندگان

نویسنده: علیرضا نوری
تاریخ: ۱۳۹۱/۱۰/۲۴ ۲:۳۱

با وجود اینکه می‌دونم برای سادگی خیلی مسائل رو در نظر نگرفتید، اما چون این کد برای آموزشه، بهتره که خوب نوشته بشه. برای مثال Concat کردن چند رشته در .Net کار درستی نیست. علاوه بر اون، بهتره که از تابع Path.Combine برای اتصال دو مسیر استفاده بشه.
مرسی

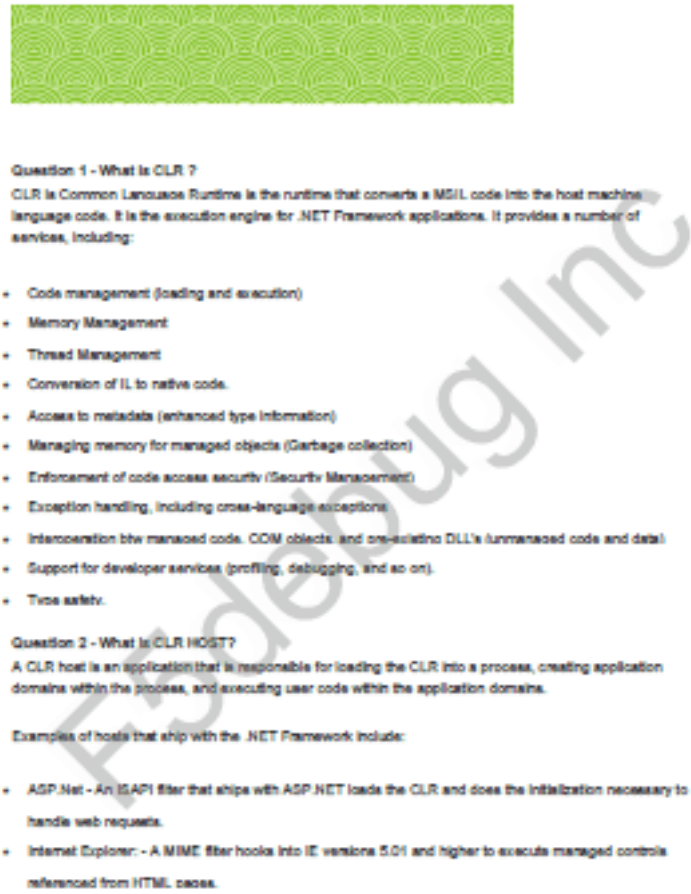
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۴ ۱۰:۵۵

بستگی داره چه تعداد رشته رو قراره با هم جمع بزنید. اگر 10هزار مورد است، استفاده از StringBuilder می‌تونه مفید باشه؛ اما اگر فقط سه قطعه است (مانند تشکیل newFile در مثال بالا)، تفاوتی را در کارآیی احساس نخواهید کرد ؛ ضمن اینکه سربار تولید و وهله سازی StringBuilder برای اتصال فقط سه قطعه با هم می‌تونه تا چهار برابر حالت‌های معمولی باشه (مراجعه کنید به قسمت ابتدای نمودار مقاله Performance considerations for strings ؛ در این نمودار، تفاوت پس از اتصال 600 قطعه به هم، خودش رو نشون می‌ده.)



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:



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.

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

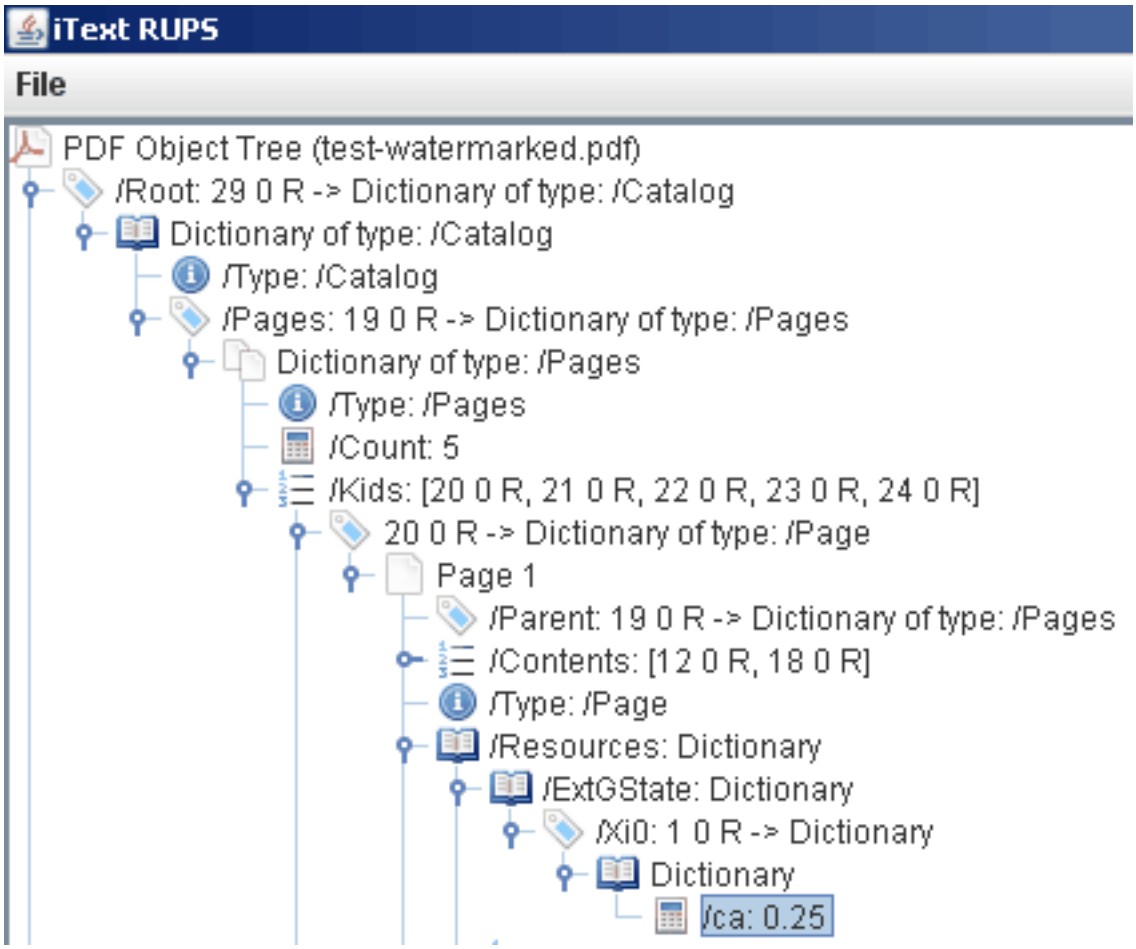
```
private static void addWatermark(string pdfFile, string watermarkedFile, string watermarkText)
        {
            FontFactory.Register("c:\\windows\\fonts\\tahoma.ttf");
            var tahoma = FontFactory.GetFont("Tahoma", BaseFont.IDENTITY_H, 40, Font.NORMAL,
BaseColor.BLACK);

            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?

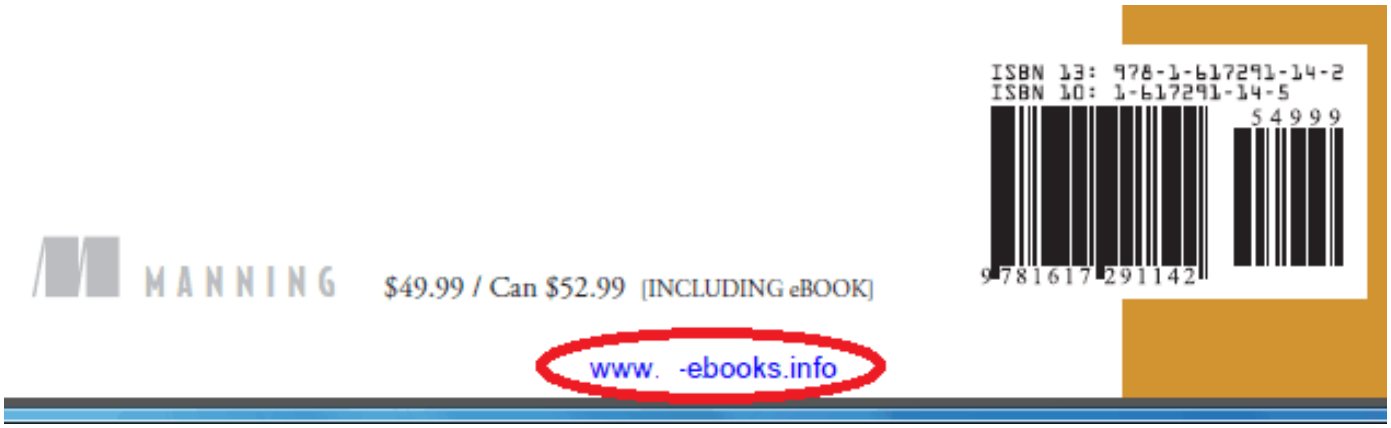
A CLR host is an application that is responsible for loading the CLR into a process, creating application domains within the process, and executing user code within the application domains.

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.

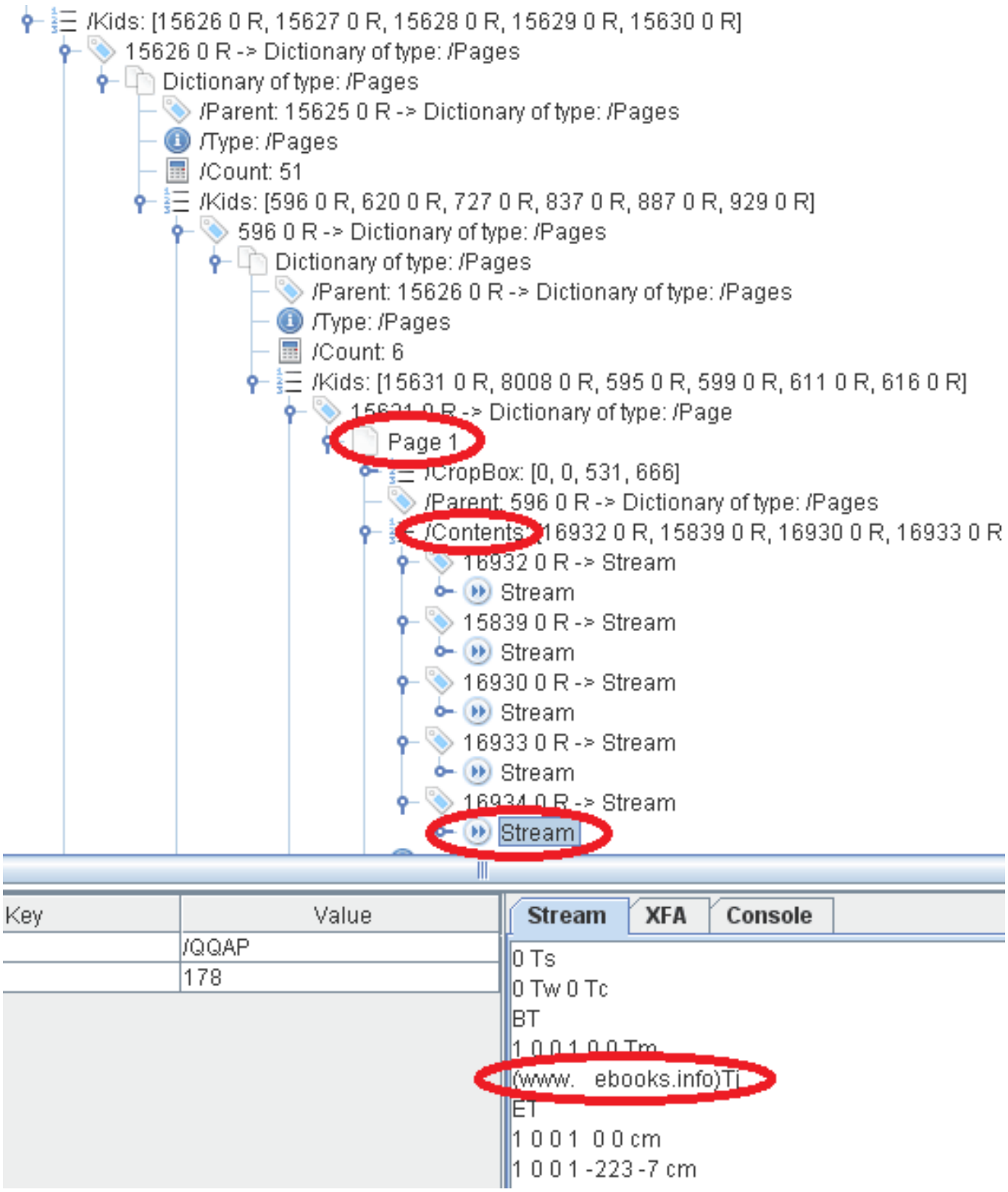
عنوان: **حذف لایه‌های جدید اضافه شده به فایل‌های PDF توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۳۱ ۸:۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp, Watermark

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.

عنوان: **تبدیل HTML فارسی به PDF با استفاده از افزونه‌ی XMLWorker کتابخانه‌ی iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۲۴ ۱۵:۱۰
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp, PdfReport, PDF

پیشتر مطلبی را در مورد « تبدیل HTML به PDF با استفاده از کتابخانه‌ی iTextSharp » در این سایت مطالعه کرده‌اید. این مطلب از افزونه HTMLWorker کتابخانه iTextSharp استفاده می‌کند که ... مدتی است توسط نویسندگان این مجموعه منسوخ شده اعلام گردیده و دیگر پشتیبانی نمی‌شود.
کتابخانه جایگزین آن‌را افزونه XMLWorker معرفی کرده‌اند که توانایی پردازش CSS و HTML بهتر و کاملتری را نسبت به HTMLWorker ارائه می‌دهد. این کتابخانه نیز همانند HTMLWorker پشتیبانی توکاری از متون راست به چپ و یونیکد فارسی، ندارد و نیاز است برای نمایش صحیح متون فارسی در آن، نکات خاصی را اعمال نمود که در ادامه بحث آن‌ها را مرور خواهیم کرد.

ابتدا برای دریافت آخرین نگارش‌های iTextSharp و افزونه XMLWorker آن به آدرس‌های ذیل مراجعه نمائید:
http://sourceforge.net/projects/itextsharp/files/itextsharp
http://sourceforge.net/projects/itextsharp/files/xmlworker

تهیه یک UnicodeFontProvider

Encoding پیش فرض قلم‌ها در XMLWorker مساوی BaseFont.CP1252 است؛ که از حروف یونیکد پشتیبانی نمی‌کند. برای رفع این نقیصه نیاز است یک منبع تامین قلم سفارشی را برای آن ایجاد نمود:

```
public class UnicodeFontProvider : FontFactoryImp
    {
        static UnicodeFontProvider()
        {
            // روش صحیح تعریف فونت
            var systemRoot = Environment.GetEnvironmentVariable("SystemRoot");
            FontFactory.Register(Path.Combine(systemRoot, "fonts\\tahoma.ttf"));
            // ثبت سایر فونت‌ها در اینجا
            //FontFactory.Register(Path.Combine(Environment.CurrentDirectory, "fonts\\irsans.ttf"));
        }

        public override Font GetFont(string fontname, string encoding, bool embedded, float size, int
style, BaseColor color, bool cached)
        {
            if (string.IsNullOrWhiteSpace(fontname))
                return new Font(Font.FontFamily.UNDEFINED, size, style, color);
            return FontFactory.GetFont(fontname, BaseFont.IDENTITY_H, BaseFont.EMBEDDED, size, style,
color);
        }
    }
```

قلم‌های مورد نیاز را در سازنده کلاس به نحوی که مشاهده می‌کنید، ثبت نمائید.
مابقی مسایل آن خودکار خواهد بود و هر زمانیکه نیاز به قلم خاصی از طرف XMLWorker وجود داشت، به متد GetFont فوق مراجعه کرده و اینبار قلمی با BaseFont.IDENTITY_H را دریافت می‌کند. IDENTITY_H در استاندارد PDF، جهت مشخص ساختن encoding قلم‌هایی با پشتیبانی از یونیکد بکار می‌رود.

تهیه منبع تصاویر

در XMLWorker اگر تصاویر با http شروع نشوند (دریافت تصاویر وب آن خودکار است)، آن تصاویر را از مسیری که توسط پیاده سازی کلاس AbstractImageProvider مشخص خواهد شد، دریافت می‌کند که نمونه‌ای از پیاده سازی آن‌را در ذیل مشاهده می‌کنید:

```
public class ImageProvider : AbstractImageProvider
    {
        public override string GetImageRootPath()
        {
```

```
            var path = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
            return path + "\\"; // مهم است که این مسیر به بک اسلش ختم شود تا درست کار کند
        }
    }
```

نحوه تعریف یک فایل CSS خارجی

```
public static class XMLWorkerUtils
    {
        /// <summary>
        /// نحوه تعریف یک فایل سی اس اس خارجی
        /// </summary>
        public static ICssFile GetCssFile(string filePath)
        {
            using (var stream = new FileStream(filePath, FileMode.Open, FileAccess.Read,
FileShare.ReadWrite))
            {
                return XMLWorkerHelper.GetCSS(stream);
            }
        }
    }
```

برای مسیردهی یک فایل CSS در کتابخانه XMLWorker می‌توان از کلاس فوق استفاده کرد.

تبدیل المان‌های HTML پردازش شده به یک لیست PDF ایی

تهیه مقدمات فارسی سازی و نمایش راست به چپ اطلاعات در کتابخانه XMLWorker از اینجا شروع می‌شود. در حالت پیش فرض کار آن، المان‌های HTML به صورت خودکار Parse شده و به صفحه اضافه می‌شوند. به همین دلیل دیگر فرصت اعمال خواص RTL به المان‌های پردازش شده دیگر وجود نخواهد داشت و به صورت توکار نیز این مسایل درنظر گرفته نمی‌شود. به همین دلیل نیاز است که در حین پردازش المان‌های HTML و تبدیل آن‌ها به معادل المان‌های PDF، بتوان آن‌ها را جمع آوری کرد که نحوه انجام آن‌را با پیاده سازی اینترفیس IElementHandler در ذیل مشاهده می‌کنید:

```
/// <summary>
    /// معادل پی دی افی المان‌های اچ تی ام ال را جمع آوری می‌کند
    /// </summary>
    public class ElementsCollector : IElementHandler
    {
        private readonly Paragraph _paragraph;

        public ElementsCollector()
        {
            _paragraph = new Paragraph
            {
                Alignment = Element.ALIGN_LEFT  // سبب می‌شود تا در حالت راست به چپ از سمت راست
صفحه شروع شود
            };
        }

        /// <summary>
        /// این پاراگراف حاوی کلیه المان‌های متن است
        /// </summary>
        public Paragraph Paragraph
        {
            get { return _paragraph; }
        }

        /// <summary>
        /// بجای اینکه خود کتابخانه اصلی کار افزودن المان‌ها را به صفحات انجام دهد
        /// قصد داریم آن‌ها را ابتدا جمع آوری کرده و سپس به صورت راست به چپ به صفحات نهایی اضافه کنیم
        /// </summary>
        /// <param name="htmlElement"></param>
        public void Add(IWritable htmlElement)
        {
            var writableElement = htmlElement as WritableElement;
            if (writableElement == null)
                return;
```

```
            foreach (var element in writableElement.Elements())
            {
                fixNestedTablesRunDirection(element);
                _paragraph.Add(element);
            }
        }

        /// <summary>
        /// نیاز است سلول‌های جداول تو در توی پی دی اف نیز راست به چپ شوند
        /// </summary>
        private void fixNestedTablesRunDirection(IElement element)
        {
            var table = element as PdfPTable;
            if (table == null)
                return;

            table.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
            foreach (var row in table.Rows)
            {
                foreach (var cell in row.GetCells())
                {
                    cell.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
                    foreach (var item in cell.CompositeElements)
                    {
                        fixNestedTablesRunDirection(item);
                    }
                }
            }
        }
    }
```

این کلاس کلیه المان‌های دریافتی را به یک پاراگراف اضافه می‌کند. همچنین اگر به جدولی در این بین برخورد، مباحث RTL آن‌را نیز اصلاح خواهد نمود.

یک مثال کامل از نحوه کنار هم قرار دادن پیشنیازهای تهیه شده

خوب؛ تا اینجا یک سری پیشنیاز را تهیه کردیم، اما XMLWorker از وجود آن‌ها بی‌خبر است. برای معرفی آن‌ها باید به نحو ذیل عمل کرد:

```
using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("test.pdf",
FileMode.Create));
                pdfWriter.RgbTransparencyBlending = true;
                pdfDoc.Open();


                var html = @"<span style='color:blue; font-family:tahoma;'><b>آزمایش</b></span>
                                کتابخانه <i>iTextSharp</i> <u>جهت بررسی فارسی نویسی</u>
                            <table style='color:blue; font-family:tahoma;'
border='1'><tr><td>eeمتن</td></tr></table>
                            <code>This is a code!</code>
                            <br/>
                            <img src='av-13489.jpg' />
                            ";

                var cssResolver = new StyleAttrCSSResolver();
                // cssResolver.AddCss(XMLWorkerUtils.GetCssFile(@"c:\path\pdf.css"));
                cssResolver.AddCss(@"code
                                    {
                                       padding: 2px 4px;
                                       color: #d14;
                                       white-space: nowrap;
                                       background-color: #f7f7f9;
                                       border: 1px solid #e1e1e8;
                                    }",
                                    "utf-8", true);

                // کار جمع آوری المان‌های ترجمه شده به المان‌های پی دی اف را انجام می‌دهد
                var elementsHandler = new ElementsCollector();

                var htmlContext = new HtmlPipelineContext(new CssAppliersImpl(new
UnicodeFontProvider()));
```

```
                htmlContext.SetImageProvider(new ImageProvider());
                htmlContext.CharSet(Encoding.UTF8);

htmlContext.SetAcceptUnknown(true).AutoBookmark(true).SetTagFactory(Tags.GetHtmlTagProcessorFactory());
                var pipeline = new CssResolverPipeline(cssResolver,
                                                       new HtmlPipeline(htmlContext, new
ElementHandlerPipeline(elementsHandler, null)));
                var worker = new XMLWorker(pipeline, parseHtml: true);
                var parser = new XMLParser();
                parser.AddListener(worker);
                parser.Parse(new StringReader(html));

                // با هندلر سفارشی که تهیه کردیم تمام المان‌های اچ تی ام ال به المان‌های پی دی اف تبدیل
شدند
                // الان تنها کافی کافی است تا این‌ها را در یک جدول راست به چپ محصور کنیم تا درست
نمایش داده شوند
                var mainTable = new PdfPTable(1) { WidthPercentage = 100, RunDirection =
PdfWriter.RUN_DIRECTION_RTL };
                var cell = new PdfPCell
                {
                    Border = 0,
                    RunDirection = PdfWriter.RUN_DIRECTION_RTL,
                    HorizontalAlignment = Element.ALIGN_LEFT
                };
                cell.AddElement(elementsHandler.Paragraph);
                mainTable.AddCell(cell);

                pdfDoc.Add(mainTable);
            }

            Process.Start("test.pdf");
```

نحوه تعریف inline css یا نحوه افزودن یک فایل css خارجی را نیز در ابتدای این مثال مشاهده می‌کنید.
UnicodeFontProvider باید به HtmlPipelineContext شناسانده شود.
ImageProvider توسط متد SetImageProvider به HtmlPipelineContext معرفی می‌شود.
ElementsCollector سفارشی ما در قسمت CssResolverPipeline باید به سیستم تزریق شود.
پس از آن XMLWorker را وادار می‌کنیم تا HTML را Parse کرده و معادل المان‌های PDF ایی آن‌را تهیه کند؛ اما آن‌ها را به صورت خودکار به صفحات فایل PDF نهایی اضافه نکند. در این بین ElementsCollector ما این المان‌ها را جمع آوری کرده و در نهایت، پاراگراف کلی حاصل از آن‌را به یک جدول با RUN_DIRECTION_RTL اضافه می‌کنیم. حاصل آن نمایش صحیح متون فارسی است.

کدهای مثال فوق را از آدرس ذیل نیز می‌توانید دریافت کنید:

XMLWorkerRTLsample.cs

به روز رسانی

کلیه نکات مطلب فوق را به همراه بهبودهای مطرح شده در نظرات آن، در پروژه‌ی ذیل می‌توانید به صورت یکجا دریافت و بررسی کنید:

XMLWorkerRTLsample.zip

نظرات خوانندگان

نویسنده: ح م
تاریخ: ۱۳۹۲/۰۸/۱۶ ۹:۱۸

همه‌ی فونت‌ها و استایل‌ها را هم که پیوست می‌کنم، برای برخی از کدهای HTML ، خروجی pdf سفید(خالی) است.
راهی وجود دارد که خطاهای پارسر دست کم در حالت Debug نشان داده شوند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۶ ۱۰:۱۱

بله. یک کلاس لاگر سفارشی درست کنید:

```
public class CustomLogger : iTextSharp.text.log.ILogger
    {
        public iTextSharp.text.log.ILogger GetLogger(Type klass)
        {
            return this;
        }

        public iTextSharp.text.log.ILogger GetLogger(string name)
        {
            return this;
        }

        public bool IsLogging(iTextSharp.text.log.Level level)
        {
            return true;
        }

        public void Warn(string message)
        {
            System.Diagnostics.Trace.TraceWarning(message);
        }

        public void Trace(string message)
        {
            System.Diagnostics.Trace.TraceInformation(message);
        }

        public void Debug(string message)
        {
            System.Diagnostics.Trace.TraceInformation(message);
        }

        public void Info(string message)
        {
            System.Diagnostics.Trace.TraceInformation(message);
        }

        public void Error(string message)
        {
            System.Diagnostics.Trace.TraceError(message);
        }

        public void Error(string message, Exception e)
        {
            System.Diagnostics.Trace.TraceError(message + System.Environment.NewLine + e);
        }
    }
```

بعد در ابتدای اجرای برنامه آن‌را ثبت کنید:

```
iTextSharp.text.log.LoggerFactory.GetInstance().SetLogger(new CustomLogger());
```

خروجی‌ها در پنجره دیباگ VS.NET نمایش داده می‌شوند.

نویسنده: مهرداد
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۱:۱۳

سلام دوست عزیز
من کد بالا رو تست کردم ظاهرا تگ‌های div رو نشون نمیده و از بین می‌بره !

تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۱:۳۱

- نام این کتابخانه Worker **XML** هست. یعنی HTML شما باید معتبر باشد و تگ‌های آن همانند یک فایل XML درست تشکیل و باز و بسته شده باشند؛ چیزی مثل XHTML ها.
- می‌توانید از کتابخانه HTML Agility pack برای درست کردن XHTML استفاده کنید:

```
var sb = new StringBuilder();
var stringWriter = new StringWriter(sb);
var doc = new HtmlDocument
            {
                OptionOutputAsXml = true,
                OptionCheckSyntax = true,
                OptionFixNestedTags = true,
                OptionAutoCloseOnEnd = true,
                OptionDefaultStreamEncoding = Encoding.UTF8
            };
doc.LoadHtml(htmlContent);
doc.Save(stringWriter);
var xhtml = sb.ToString();
```

خاصیت OptionOutputAsXml آن‌را true کنید تا در حد توانش مشکلات HTML شما را برطرف و یک خروجی XHTML را تولید کند.
- سایر مشکلات آن‌را بهتر است در mailing لیست آن‌ها به همراه ارائه مثال قابل بازتولیدی ارسال کنید.

نویسنده: جلال
تاریخ: ۱۳۹۲/۱۲/۱۵ ۸:۵۰

با سلام؛ من خیلی دنبال کلاس HtmlDocument گشتم، اما نه توی net. پیدا کردم و نه توی سایت خودتون، میتونید راهنماییم کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۵ ۹:۲۵

«می‌توانید از کتابخانه HTML Agility pack استفاده کنید»

```
PM> Install-Package HtmlAgilityPack
```

نویسنده: جلال
تاریخ: ۱۳۹۲/۱۲/۱۵ ۱۱:۴

من کدی که فرمودید رو اضافه کردم، همچنین، کد Html هم Valid هستش، و کلا با div ساخته شده، اما pdf خروجی سفید هستش.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۵ ۱۲:۳۵

برای رفع مشکل محو شدن Div، کدهای کلاس ElementsCollector مطلب جاری را به نحو زیر تغییر دهید:

```
public void Add(IWritable htmlElement)
        {
```

```
            var writableElement = htmlElement as WritableElement;
            if (writableElement == null)
                return;

            foreach (var element in writableElement.Elements())
            {
                var div = element as PdfDiv;
                if (div != null)
                {
                    foreach (var divChildElement in div.Content)
                    {
                        fixNestedTablesRunDirection(divChildElement);
                        _paragraph.Add(divChildElement);
                    }
                }
                else
                {
                    fixNestedTablesRunDirection(element);
                    _paragraph.Add(element);
                }
            }
        }
```

نویسنده: سمیه
تاریخ: ۱۳۹۳/۰۱/۱۹ ۱۵:۳۹

سلام؛ ضمن تشکر از مطلب مفیدتان من نمونه کدهایی که در قسمت پایین قرار داده بودید، دانلود کردم. همچنین آخرین نگارش‌های iTextSharp و افزونه XMLWorker را از لینک هایی معرفی شده دانلود و dll هایشان را به پروژه ام اضافه کرده ام، ولی با وجود این به فضای نام iTextSharp.tool خطا می‌دهد و آن را نمی‌شناسد. می‌شه لطفا من راهنمایی کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۱۹ ۱۷:۲۰

پروژه شما باید ارجاعاتی را به دو فایل itextsharp.dll و itextsharp.xmlworker.dll داشته باشد.

```
PM> Install-Package iTextSharp
PM> Install-Package itextsharp.xmlworker
```

نویسنده: هیمن صادقی
تاریخ: ۱۳۹۳/۰۲/۲۵ ۱۹:۰

درود
با سپاس از مطالب که در سایت قرار دادید یک مشکل داشتم
من کد رو در پروژه قرار دارم اما کد زیر که قرار متن راست به چپ کنه کار نمی‌کنه

```
_paragraph = new Paragraph
  {
Alignment = Element.ALIGN_LEFT  // سبب می‌شود تا در حالت راست به چپ از سمت راست صفحه شروع شود
  };
```

و کد زیر هم کار نمی‌کنه

```
fixNestedTablesRunDirection(element);
```

اگر لطف کنید من رو راهنمایی کنید

نویسنده: هیمن صادقی
تاریخ: ۱۳۹۳/۰۲/۲۵ ۲۲:۲۸

درود؛ پیوست پیام قبلی که گفتم کد کار نمی‌کنه: rar.1

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۶ ۰:۲۹

- XML Worker از تمام امکانات CSS پشتیبانی نمی‌کند. لیست موارد پشتیبانی شده در اینجا (رنگ‌های سبز)
- در کد شما float: left و float: right دارید که مطابق لینک داده شده فعلا پشتیبانی نمی‌شود.
- نکته‌ی تکمیلی « برای رفع مشکل محو شدن Div، کدهای کلاس ElementsCollector مطلب جاری را به نحو زیر تغییر دهید » را هم اضافه نکرده‌اید.
- کد fixNestedTablesRunDirection مطلب جاری در کدهای شما به نمونه‌ای که cell.RunDirection = PdfWriter.RUN_DIRECTION_RTL ندارد، تغییر پیدا کرده. بنابراین کار نخواهد کرد.

نویسنده: هیمن صادقی
تاریخ: ۱۳۹۳/۰۲/۲۶ ۱:۱۹

ممنون از شما

تابع fixNestedTablesRunDirection در خط

```
if (table == null)
                return;
```

خاتمه پیدا می‌کند و کدی را که برداشتم تاثیر بر کد نداره. زمانیکه به صورت دستی کد زیر را به متن اضافه می‌کنیم

```
paragraph.Add("Data")
```

کار میکنه یعنی راست به چپ را درست می‌کند. اما زمانی که فایل html بهش میدم چپ به راست می‌باشد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۶ ۲:۹

متد Add را به این صورت اصلاح کنید تا جهت Paragraph ها را هم درست کند:

```
public void Add(IWritable htmlElement)
        {
            var writableElement = htmlElement as WritableElement;
            if (writableElement == null)
                return;

            foreach (var element in writableElement.Elements())
            {
                if (element is PdfDiv)
                {
                    var div = element as PdfDiv;
                    foreach (var divChildElement in div.Content)
                    {
                        fixNestedTablesRunDirection(divChildElement);
                        _paragraph.Add(divChildElement);
                    }
                }
                else if(element is Paragraph)
                {
                    var paragraph = element as Paragraph;
                    paragraph.Alignment = Element.ALIGN_LEFT;
                    _paragraph.Add(element);
                }
                else
                {
                    fixNestedTablesRunDirection(element);
                    _paragraph.Add(element);
                }
            }
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۶ ۲:۱۲

یک نکته‌ی مهم

از خروجی GetBuffer استریم نباید استفاده شود:

```
return File(memoryStream.GetBuffer(), "application/pdf", "Test.pdf");
```

باید از ToArray استفاده کنید تا حاوی اضافات بافر نباشد (نمایش پیغام ذخیره تغییرات در adobe reader به همین دلیل اضافات است):

```
return File(memoryStream.ToArray(), "application/pdf", "Test.pdf");
```

در این حالت حجم فایل نهایی هم نصف خواهد بود.

نویسنده: الیاس سربند
تاریخ: ۱۳۹۳/۰۳/۰۷ ۱۷:۳۸

سلام و خسته نباشید. میشه از این روش توی ASP.Net استفاده کرد؟ اگر آره در مورد دستور آخر Process.Start چه باید کرد؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۰۷ ۱۷:۵۶

- مثال پیوست شده کمی بالاتر یک مثال ASP.NET MVC است.
- Process.Start را حذف کنید؛ نیازی نیست.
- به قسمت new FileStream آن دقت کنید. اینجا مسیر یک فایل را می‌شود مشخص کرد. فایل نهایی تولید شده در این مسیر نوشته می‌شود. از آن مسیر در برنامه‌های وب و ویندوز می‌توان استفاده کرد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۰۷ ۱۸:۳

به روز رسانی

کلیه نکات مطلب فوق را به همراه بهبودهای مطرح شده در نظرات آن، در پروژه‌ی ذیل می‌توانید به صورت یکجا دریافت و بررسی کنید:

XMLWorkerRTLsample.zip

نویسنده: مصطفی سلطانی
تاریخ: ۱۳۹۳/۰۶/۰۱ ۱۲:۲۳

با سلام
با تشکر از مطلب مفیدتان
من پروژه نمونه شما را دانلود کردم ولی داخل جدول مشکل راست به چپ فارسی را مشاهده می‌کنم. مثلا لغت "متن" به صورت "ن ت م" نشان داده می‌شود.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۰۱ ۱۳:۱۷

ابتدای متد Add فایل ElementsCollector.cs آن‌را به صورت زیر اصلاح کنید:

```
public void Add(IWritable htmlElement)
```

```
        {
            var writableElement = htmlElement as WritableElement;
            if (writableElement == null)
                return;

            foreach (var element in writableElement.Elements())
            {
                if (element is NoNewLineParagraph)
                {
                    var noNewLineParagraph = element as NoNewLineParagraph;
                    foreach (var item in noNewLineParagraph)
                    {
                        fixNestedTablesRunDirection(item);
                        _paragraph.Add(item);
                    }
                }
                else if (element is PdfDiv)
```

عنوان: **تهیه پردازنده‌های سفارشی برای افزونه XMLWorker کتابخانه iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۲۴ ۱۳:۵
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

پیشتر مطلب « تهیه پردازنده‌های سفارشی برای HTMLWorker کتابخانه iTextSharp » را در این سایت مطالعه کرده‌اید. از آنجائیکه افزونه HTMLWorker منسوخ شده است و دیگر پشتیبانی نخواهد شد، باید کدهای فعلی را به افزونه XMLWorker منتقل کرد. مقدمه‌ای را در این زمینه در مطلب « تبدیل HTML فارسی به PDF با استفاده از افزونه‌ی XMLWorker کتابخانه‌ی iTextSharp » می‌توانید مطالعه نمائید.
در ادامه قصد داریم همان امکان پشتیبانی از تصاویر base64 مدفون شده در صفحات HTML را به کتابخانه XMLWorker نیز اضافه کنیم.

تهیه پردازنده‌های سفارشی تگ‌های HTML جهت افزونه XMLWorker

در اینجا کار با ارث بری از کلاس AbstractTagProcessor شروع می‌شود. سپس کافی است تا متد End این کلاس پایه را override کرده و تگ سفارشی خود را پردازش کنیم. نمونه‌هایی از این نوع پردازنده‌ها را در پوشه itextsharp.xmlworker\iTextSharp\tool\xml\html سورس‌های iTextSharp می‌توانید ملاحظه کنید و جهت ایده گرفتن بسیار مناسب می‌باشند.
یکی از این پردازنده‌های پیش فرض موجود در افزونه XMLWorker کار پردازش تگ‌های img را انجام می‌دهد و در کلاس iTextSharp.tool.xml.html.Image قرار گرفته است. این پردازنده به صورت پیش فرض تصاویر base64 را پردازش نمی‌کند. برای رفع این مشکل می‌توان به نحو ذیل عمل کرد:

```
public class CustomImageTagProcessor : iTextSharp.tool.xml.html.Image
    {
        public override IList<IElement> End(IWorkerContext ctx, Tag tag, IList<IElement>
currentContent)
        {
            IDictionary<string, string> attributes = tag.Attributes;
            string src;
            if (!attributes.TryGetValue(HTML.Attribute.SRC, out src))
                return new List<IElement>(1);

            if (string.IsNullOrEmpty(src))
                return new List<IElement>(1);

            if (src.StartsWith("data:image/", StringComparison.InvariantCultureIgnoreCase))
            {
                // data:[<MIME-type>][;charset=<encoding>][;base64],<data>
                var base64Data = src.Substring(src.IndexOf(",") + 1);
                var imagedata = Convert.FromBase64String(base64Data);
                var image = iTextSharp.text.Image.GetInstance(imagedata);

                var list = new List<IElement>();
                var htmlPipelineContext = GetHtmlPipelineContext(ctx);
                list.Add(GetCssAppliers().Apply(new
Chunk((iTextSharp.text.Image)GetCssAppliers().Apply(image, tag, htmlPipelineContext), 0, 0, true), tag,
htmlPipelineContext));
                return list;
            }
            else
            {
                return base.End(ctx, tag, currentContent);
            }
        }
    }
```

با ارث بری از کلاس پردازنده پیش فرض تگ‌های تصاویر یا iTextSharp.tool.xml.html.Image شروع و سپس متد End آن‌را تحریف کرده‌ایم.
در اینجا اگر src یک تگ img با الگوی تصاویر base64 شروع شده باشد، تصویر آن استخراج و تبدیل به وهله‌ای از تصاویر استاندارد iTextSharp می‌شود. سپس این تصویر در اختیار htmlPipelineContext قرار داده خواهد شد و یا اگر این تصویر از نوع base64 نباشد، همان متد base.End کلاس پایه، جهت پردازش آن کفایت می‌کند.

استفاده از یک پردازنده تگ سفارشی HTML افزونه XMLWorker

تا اینجا یک پردازنده جدید تصاویر را ایجاد کرده‌ایم. برای اینکه XMLWorker را از وجود آن مطلع کنیم، نیاز است آن‌را به درون htmlPipeline این افزونه تزریق نمائیم که کدهای کامل آن‌را در ذیل مشاهده می‌کنید:

```
using (var doc = new Document(PageSize.A4))
            {
                var writer = PdfWriter.GetInstance(doc, new FileStream("test.pdf", FileMode.Create));
                doc.Open();
                var html = @"<img
src='
U29mdHdhcmUAQWRvYmUgSW1hZ2VSZWFkeXHJZTwAAAGAUExURdSmeJp2SHlbQIRoSUg2J499a8KebqeHZuGufBEVJPz7+3NWPVxGMdu
whPXEktnX1mtROLq7t5WDc2VMNv3LmKB8TMSidMbFxLGlmXlhSMSddpJUL+y8i3VlVqedlOzr6gUIF2lXRLCLY4ZyXLyYaYhtUYiJhJ
FyU1dBLLiVZnlwZrWRY/Hx8b+2rbySaJh9YqeooDw4NygnKvvJlpyblzksIUhGRryYckc7MPjGlKODX5x8VVA8K+azgM3FvDInHK2JW
2ZbUOHh4Xt2cFpaWKeAUM6kel1RRJmUjo5vSrWzrJJ1WFhLQCQmMuK1iJiMgmthWPPCkOm3hEtBOunm5LCNXnJtZquEXmNkYvG+i7Ct
q+y5hrWRbKqSeaN/WqmFVYFgQh8aGOa4isWkd8mcby4vONDNy0AwI5h2U19JMxkdLzIuL1JBMjQ3P5Z6Ve6/j93c2+Xi34KAfJ5/Xvj
4+O/u7sSKVJd4Wo6QjXE+IeOwfQcNJoBeQ8Gdbf/Mmf///5GX6NEAAAcrSURBVHja3JbpX9pIGMchiWkgEaOBtaGinBLEyopFBeMqtY
KI4kGt2lILFsUoXa3WdZcc/dd3JheHAvaz7/Z5Ec2Q7/yeaw7Lz/9klv8rfnM+Orz5cXLjZsL+67h9eCq9Vaxvzc6v3W6+/TX85kN6i
xdokkQQCaE5vrg28Qv4a2yFQcpSi/HzH6efi+/UaEAwWAtepuvv3tw/B//hqZGQqDFSmyHC7v0z8EldlZQQEgTfMgF23h8/T+gEhQGr
cQYrMBKVtvfDb4qU/j3DMK3SdIKWsNs++M1iS8R8W/gULyG1771w+/stQWpTpFpzByb09MRHEwaoxUxToGtaZiBrE72cXzMyhcDiIRg
CHxJPIxKt5aF23gMf0iquz8BJmAAFpUStxvG0xIA3arcHPsvrJM1wvFTDeEGQeKCewCo1jgRDwKuJrrh9C3osIfyiz+NboZFKxU0xJE
YmeJbBhPoKiKyMDXfHd0mJWSETnoKiKCmgSioFDKFr4T1lbn/fgkHf+PGu+A+A12imMqdAqzNUXlFCFP+gOD41CKJBcCB4bKSnOmitB
5VWSgnMrSjhCnu8D1hoS1xP/KcH1BhZdGi4c4VNAh/I5PGyRjdQqje+A6YXPIpup/DhHlMUh44f1hAJ6x77z3OwVjG/0ml7Ot4gOWnx
vkfbALw+2EnPGc43ojWk3qNt7hdpiSp0ajcMukHQPB/4o3vPf8TKQgc+pqXdkpEtgGewE7THel/j66dtdBLA1XAYRXK8AGbxC/6RHvj
bCuOE0Kklk8lcg/+OicaJcOhfTflTVYCHuYvX3XH7QCxcUAol9i6VursLha+VfcLPHwamZjfSAgxi6QId6oFnC5awsjdoWYjFPrOlB3
QONAtJjrwsetiq2jkzgfc9nPdklJBDyXvGj+Zf+jIKe7pPoNFoOHwyoyaQKFcD9z3wzbwSGnT6fCMB9u5UmWMLYwTJQo5QC2AB6r122
ukBJeVWnA6HIwlLnp/bI/w5wI3tJR3LjcZMbvVzL/xHwOG+M6s2mFeSjRm0QRyDYnyCOEv/0fOYGM/vha4N3J1S5hoZhCAcYBro/AwV
63NIjafuzL4rLSjOZYKeIT45j9XUnQTs/Y7Inbqp/pABeIPBqsTystr0/pd9T9jprZIGO9CHa4gTPHairxr/eP/rwai+YdzlWQfALSH
u4qTxfHxiQKVTaBINvfCjDFo1Fmzjor/zP+0BNXdgxSTdqRe5w0bT2hq+293mdWDOSJ5DWbgwd4uGpSPxXW5WGzGddhYWHsDRguqpO5
x9jjq4HY3BnjtcRRGGe/Xqn38YC6SraVt84jnXwo0FgC8kOK7s+mv91St6RhVnZ72Vqeln4EM+cFY43SHgdj584c9ormdFbx3Jbk73v
9PuvNCCvx67ntPzlmG2xUvUhQpZz9roxHdwXx4e7Yb/fdXc7o81PFcUxW2ry+Wy5miM4gQkEAh0uxKfXWbdLXs1XGxZURRnXZpZrVbX
egT/rUvm571itnncQPctWZso2hAdd61GIzIuf32y5zduL0VxtwQPWG2vB7QP0OKKVaejOI7L8lP4+S3r+wY+zSZfGPvGPlFlt8FQ3BC
PQPYpfOjWs3QHtMVLJqmU0NLe9XVhsBpOwyER0+D1oE534t8Hsn/KctwLokxUgeunD6FwCA2xMGtAPAdhjkr55afwoaksGpHlAKTnWU
K9ZIAt15k/U+mK5voSuoI9Vre/fZPOBcFQKg4+PXsXg7urVra0Stvqmud4mTp4hN/s+lAIy8ErIC7Oz8aITzqegYkUL4tawQ+ivEvud
P7Gt6SPpCpewJ8BfN+pb/aq71dG2kjayLuJ3/vC+gB+EBe9Xm/8KEQs67hShMmgIRsNylFuFe9UL1IGHXHNAtr77ZYN7htNB8LxJmCn
yaBZULpJ6/g4ZZQCX83FAS1u3675xnTaX/GKFdLl+gIaDZeFpU78rS9oDnzZEmHstqPJKc9n90LJPThyBUZIVRtMv8Q1v9Xx8bzxigd
dWo1t7yZ//zgSCwRiK6CO0PUD2OR4hMnhHfiPtYiJr4a8Jj4MbHNe7UC4RtTfc5wsd+DD6RbxxTZ8chtkrcJGIlqX41GqTVzFp3wmfm
CNi5rNT74Z3nwHi2BjZW11AtdzgvxIfSBl4l/Klzr+bfLvzSNYA1u9xTfmz8f4lLmA5HWfgV8eTa7BEohxox1xeZ1F5Ef4fTrYnL4oG
jb7QZ3JVgk2W4KJPMZvmWbo9KWJ27QsXKHm3DkhJT/Gs6z55lo0abV5wCSL5txL/CMa4PYPUXN+5qwTj68aXwa5MP4Efj/VDA4TW3BV
3PQMp7Wlgnfg555mcPFO8RbXMbXv8Oh6pG3J7IRM8bq3Q/zKLFqUQ3GteNYvbepG1XG57O0Qt9Hmd1bOKC1qbZH/zbK78FWzYMJ2aZo
XPq7kr8ZvORr+iUSjJzQb/Gpa5l8BBgBZTppAyfsf0wAAAABJRU5ErkJggg==' width='62' height='80' style='float:
left; margin-right: 28px;' />";

                var tagProcessors = (DefaultTagProcessorFactory)Tags.GetHtmlTagProcessorFactory();
                tagProcessors.RemoveProcessor(HTML.Tag.IMG); // remove the default processor
                tagProcessors.AddProcessor(HTML.Tag.IMG, new CustomImageTagProcessor()); // use our new
processor

                var cssResolver = new StyleAttrCSSResolver();
                cssResolver.AddCss(@"code { padding: 2px 4px; }", "utf-8", true);
                var charset = Encoding.UTF8;
                var hpc = new HtmlPipelineContext(new CssAppliersImpl(new XMLWorkerFontProvider()));
                hpc.SetAcceptUnknown(true).AutoBookmark(true).SetTagFactory(tagProcessors); // inject
the tagProcessors
                var htmlPipeline = new HtmlPipeline(hpc, new PdfWriterPipeline(doc, writer));
                var pipeline = new CssResolverPipeline(cssResolver, htmlPipeline);
                var worker = new XMLWorker(pipeline, true);
                var xmlParser = new XMLParser(true, worker, charset);
                xmlParser.Parse(new StringReader(html));
            }
            Process.Start("test.pdf");
```

در اینجا ابتدا لیست پردازنده‌های پیش فرض افزونه XMLWorker را دریافت و سپس پردازنده تگ img آن‌را حذف و با نمونه جدید خود جایگزین کرده‌ایم. در ادامه این لیست تغییر یافته به درون HtmlPipelineContext تزریق شده‌است تا بجای DefaultTagProcessorFactory اصلی مورد استفاده قرار گیرد.

نظرات خوانندگان

نویسنده: ایمان دارابی
تاریخ: ۱۳۹۲/۰۷/۲۷ ۱۵:۱

با سلام؛ معمولا تگ عکس حاوی مسیر فایل عکس در صورتی که در فایل pdf عکس معمولا embed می‌شه دو تا راه حل به نظر من می‌رسه یکی دانلود عکس‌ها و قرار دادن آنها در پوشه تمپ که پردازنده فوق باید اونو هندل کنه که البته برای افزایش سرعت باید از درخواست‌های غیر هم زمان استفاده کرد چون ممکنه مجبور به دانلود چندین عکس در یک رشته html باشیم و راه حل دوم عدم ذخیره فایل عکس و تبدیل آن به رشته که در مثال شما دیده می‌شه شما کدوم روش را توصیه می‌کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۲۷ ۱۵:۱۴

تمام این روش‌ها پشتیبانی می‌شوند. اگر src تصویر
1- مسیر لوکال هست، در مطلب مقدماتی استفاده از XMLWorker از کلاس ImageProvider تهیه شده استفاده کنید.
2- URL و مسیر وب است، خود iTextSharp به صورت خودکار آن‌را دانلود می‌کند.
3- base64 است، از راه حل مطلب جاری استفاده کنید.
از لحاظ سرعت کار، 3 سریعترین است؛ بعد 1 و در آخر 2.

نهایتا در هر سه حالت، عکس در فایل PDF مدفون می‌شود و نیازی به تنظیم خاصی ندارد.