عنوان: اصلاح Urlها در فایل‌های PDF با استفاده از iTextSharp
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۸ ۱۸:۵
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp, PDF

نحوه ایجاد لینک در فایل‌های PDF به کمک iTextSharp

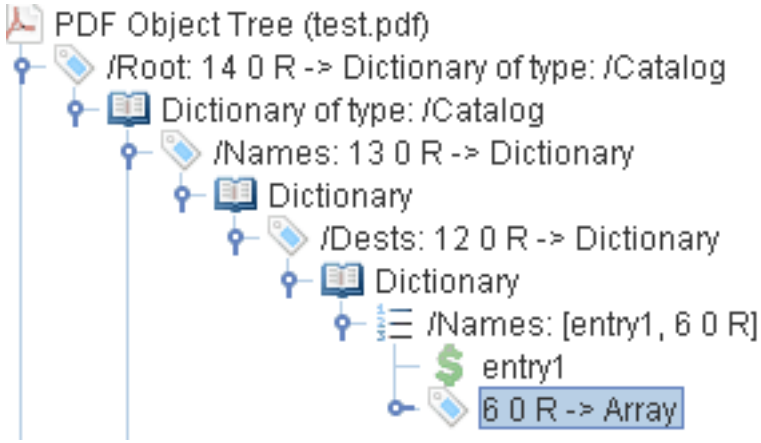
حداقل دو نوع لینک را در فایل‌های PDF می‌توان ایجاد کرد:
الف) لینک به منابع خارجی؛ مانند یک وب سایت
ب) لینک به صفحه‌ای داخل فایل PDF
در ادامه مثالی را مشاهده خواهید نمود که شامل هر دو نوع لینک است:

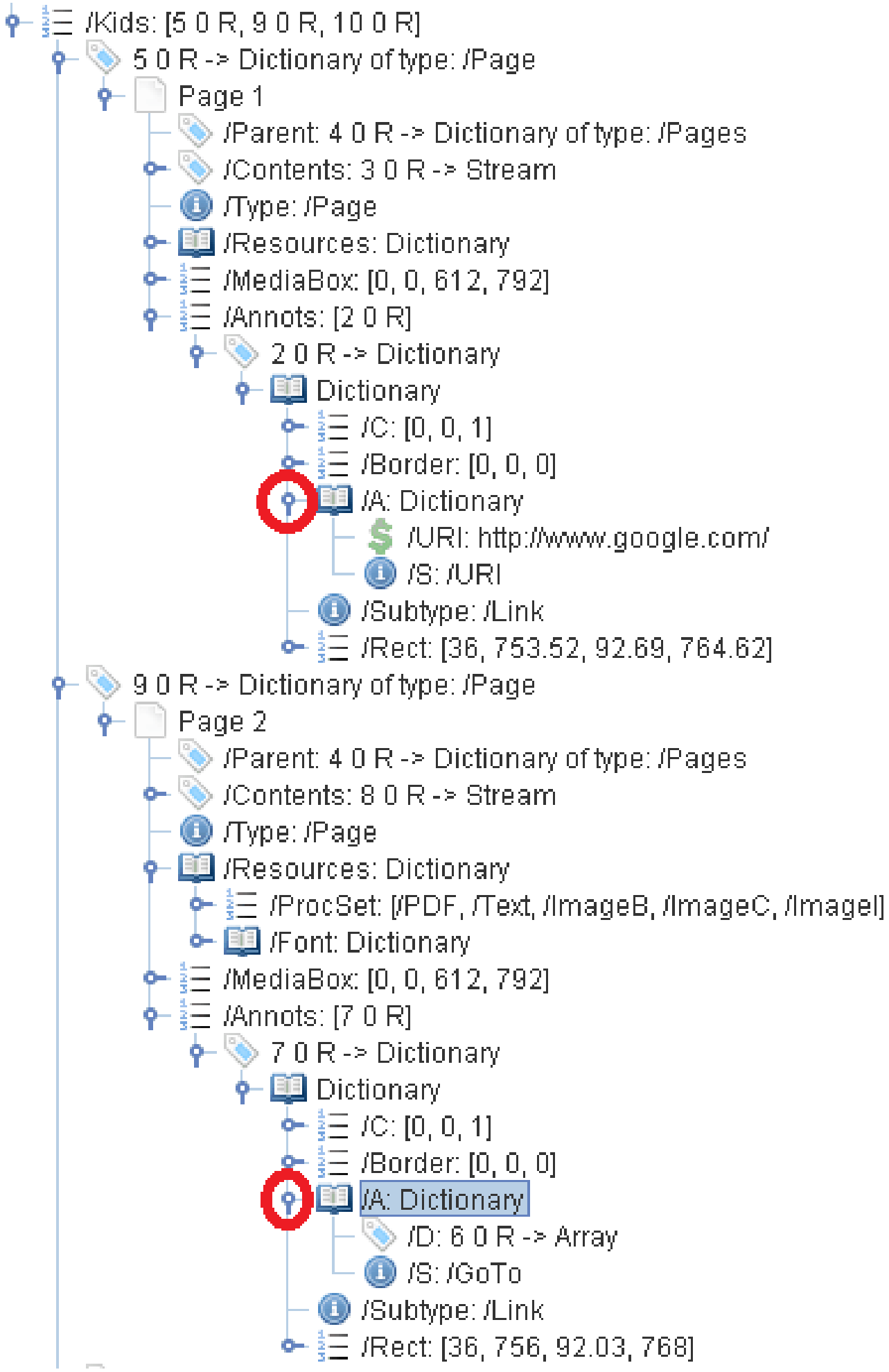
```
void WriteFile()
        {
            using (var doc = new Document(PageSize.LETTER))
            {
                using (var fs = new FileStream("test.pdf", FileMode.Create))
                {
                    using (var writer = PdfWriter.GetInstance(doc, fs))
                    {
                        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));
```

RemovePdfLinks.7z

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۹ ۲۱:۴۲

کدهای نهایی این مطلب را در مخزن کد ذیل می‌توانید دریافت کنید:

- مخزن کد: RemovePdfLinks
- فایل‌های اجرایی