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:

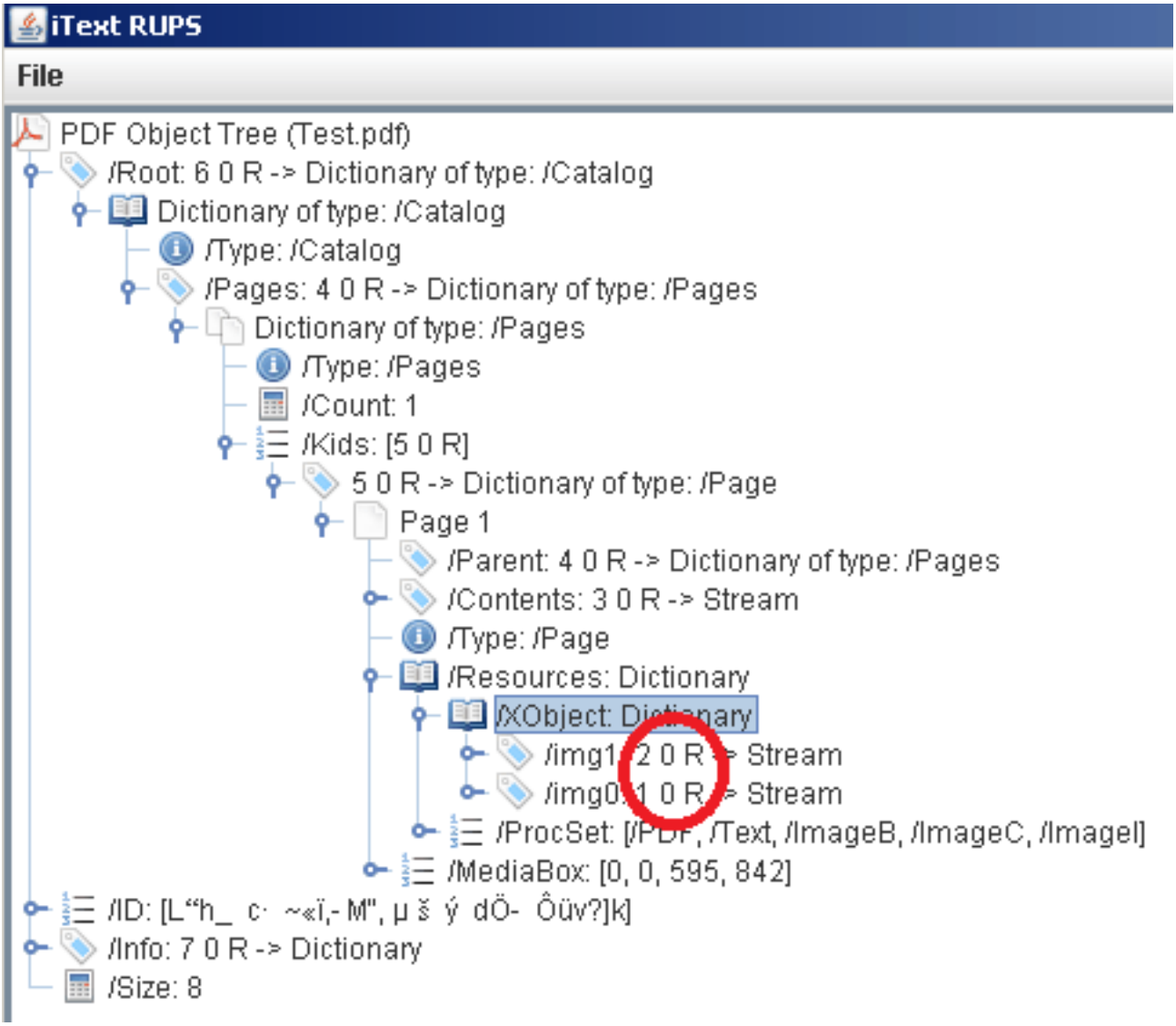
```
private static void CreateTestFile()
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));
                pdfDoc.Open();

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

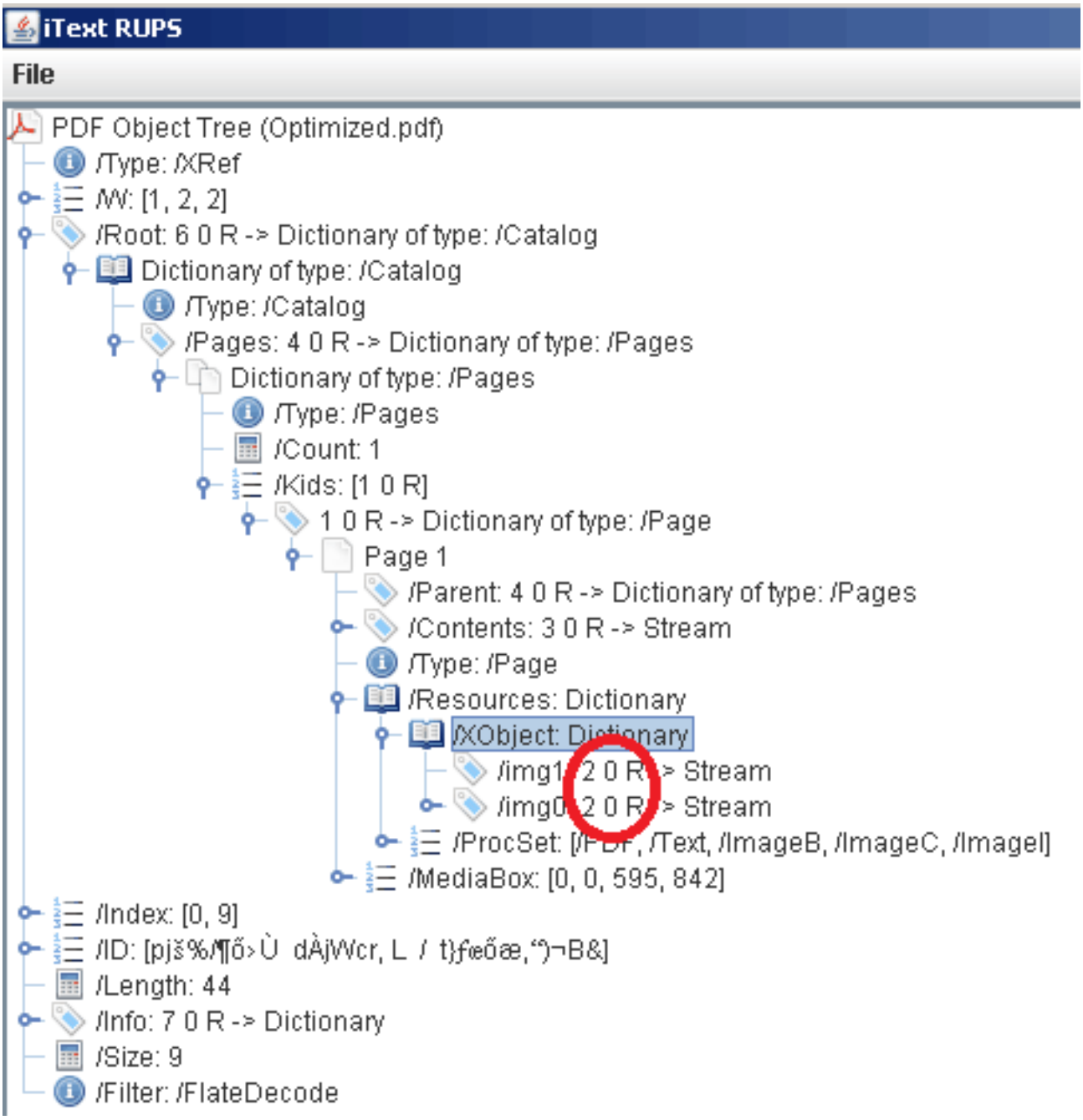
در این فایل دو وهله از تصویر 01.png به صفحه اضافه شده‌اند. بنابراین دقیقا دو تصویر در فایل نهایی تولیدی وجود خواهد داشت.

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.