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)
```