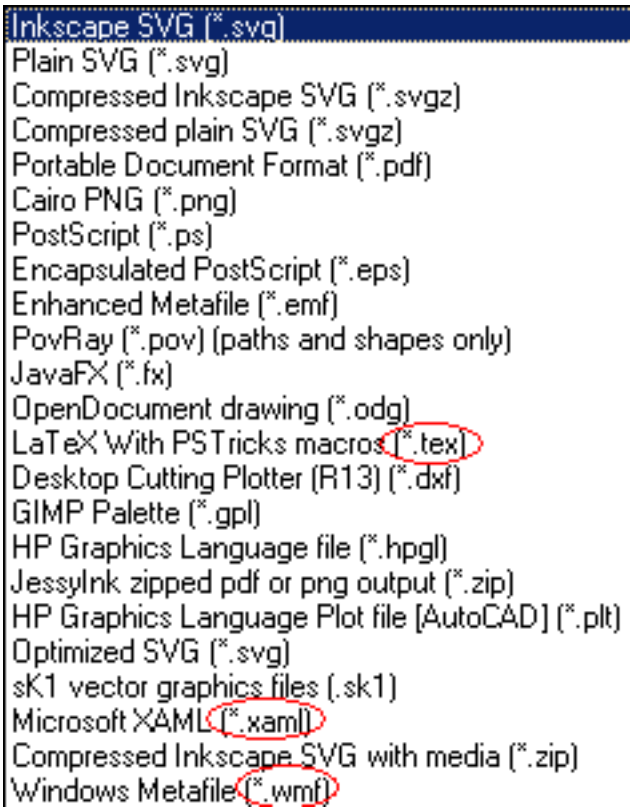
عنوان: **استفاده از گرافیک برداری در iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۳ ۱۹:۴۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: iTextSharp

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:

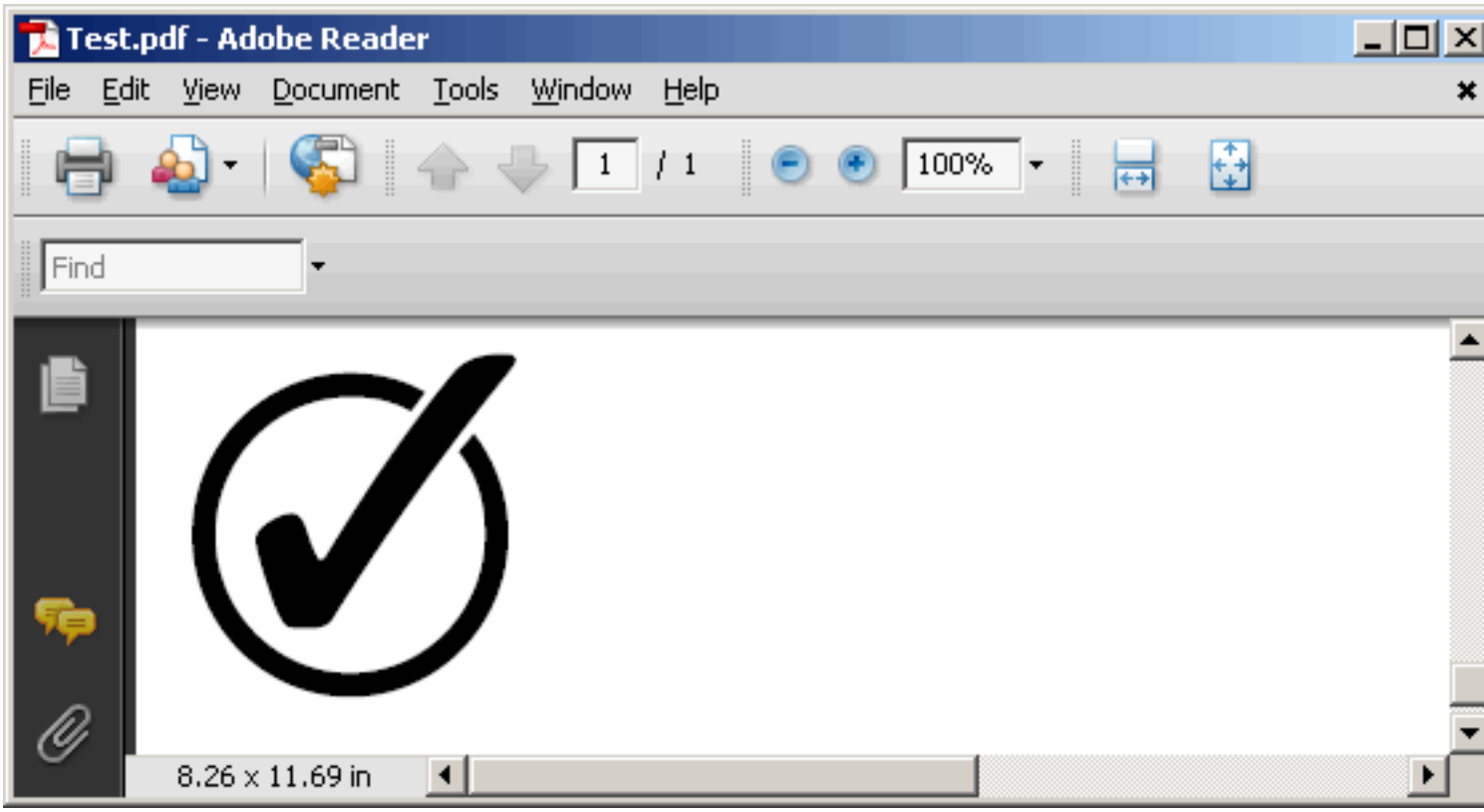
```
%LaTeX with PSTricks extensions
%%Creator: 0.48.0
%%Please note this file requires PSTricks extensions
\psset{xunit=.5pt,yunit=.5pt,runit=.5pt}
\begin{pspicture}(190,190)
{
\newrgbcolor{curcolor}{0 0 0}
\pscustom[linestyle=none,fillstyle=solid,fillcolor=curcolor]
{
\newpath
\moveto(52.73079005,101.89500456)
\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: