عنوان: **تعریف قالب‌های جداول سفارشی و کار با منابع داده‌ای از نوع Anonymous در PdfReport**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۸ ۰:۲۲
آدرس: www.dotnettips.info
برچسب‌ها: PdfReport

تعدادی قالب جدول پیش فرض در PdfReport تعریف شده‌اند، مانند BasicTemplate.RainyDayTemplate، BasicTemplate.SilverTemplate، و غیره. نحوه تعریف این قالب‌ها بر اساس پیاده سازی اینترفیس ITableTemplate است. برای نمونه اگر یک قالب جدید را بخواهیم ایجاد کنیم، تنها کافی است اینترفیس یاد شده را به نحو زیر پیاده سازی نمائیم:

```
using System.Collections.Generic;
using System.Drawing;
using iTextSharp.text;
using PdfRpt.Core.Contracts;

namespace PdfReportSamples.HexDump
{
    public class GrayTemplate : ITableTemplate
    {
        public HorizontalAlignment HeaderHorizontalAlignment
        {
            get { return HorizontalAlignment.Center; }
        }

        public BaseColor AlternatingRowBackgroundColor
        {
            get { return new BaseColor(Color.WhiteSmoke); }
        }

        public BaseColor CellBorderColor
        {
            get { return new BaseColor(Color.LightGray); }
        }

        public IList<BaseColor> HeaderBackgroundColor
        {
            get { return new List<BaseColor> { new BaseColor(ColorTranslator.FromHtml("#990000")), new
BaseColor(ColorTranslator.FromHtml("#e80000")) }; }
        }

        public BaseColor RowBackgroundColor
        {
            get { return null; }
        }

        public IList<BaseColor> PreviousPageSummaryRowBackgroundColor
        {
            get { return new List<BaseColor> { new BaseColor(Color.LightSkyBlue) }; }
        }

        public IList<BaseColor> SummaryRowBackgroundColor
        {
            get { return new List<BaseColor> { new BaseColor(Color.LightSteelBlue) }; }
        }

        public IList<BaseColor> PageSummaryRowBackgroundColor
        {
            get { return new List<BaseColor> { new BaseColor(Color.Yellow) }; }
        }

        public BaseColor AlternatingRowFontColor
        {
            get { return new BaseColor(ColorTranslator.FromHtml("#333333")); }
        }

        public BaseColor HeaderFontColor
        {
            get { return new BaseColor(Color.White); }
        }

        public BaseColor RowFontColor
        {
            get { return new BaseColor(ColorTranslator.FromHtml("#333333")); }
        }

        public BaseColor PreviousPageSummaryRowFontColor
        {
```

```
            get { return new BaseColor(Color.Black); }
        }

        public BaseColor SummaryRowFontColor
        {
            get { return new BaseColor(Color.Black); }
        }

        public BaseColor PageSummaryRowFontColor
        {
            get { return new BaseColor(Color.Black); }
        }

        public bool ShowGridLines
        {
            get { return true; }
        }
    }
}
```

و برای استفاده از آن خواهیم داشت:

```
.MainTableTemplate(template =>
{
      template.CustomTemplate(new GrayTemplate());
})
```

چند نکته:

- در کتابخانه iTextSharp، کلاس رنگ توسط BaseColor تعریف شده است. به همین جهت خروجی رنگ‌ها را در اینجا نیز بر اساس BaseColor مشاهده می‌کنید. اگر نیاز داشتید رنگ‌های تعریف شده در فضای نام استاندارد System.Drawing را به BaseColor تبدیل کنید، فقط کافی است آن‌را به سازنده کلاس BaseColor ارسال نمائید.

- اگر علاقمند هستید که معادل رنگ‌های HTML ایی را در اینجا داشته باشید، می‌توان از متد توکار ColorTranslator.FromHtml استفاده کرد.

- برای تعریف رنگی به صورت شفاف (transparent) آن‌را مساوی null قرار دهید.

- در اینترفیس فوق، تعدادی از خروجی‌ها به صورت IList است. در این موارد می‌توان یک یا دو رنگ را حداکثر معرفی کرد. اگر دو رنگ را معرفی کنید یک گرادیان خودکار از این دو رنگ، تشکیل خواهد شد.

- اگر قالب جدید زیبایی را طراحی کردید، لطفا در این پروژه مشارکت کرده و آن‌را به صورت یک وصله ارائه دهید!

تهیه یک منبع داده ناشناس

مثال زیر را در نظر بگیرید. در اینجا قصد داریم معادل Ascii اطلاعات Hex را تهیه کنیم:

```
using System;
using System.Collections;
using System.Linq;

namespace PdfReportSamples.HexDump
{
    public static class PrintHex
    {
        public static char ToSafeAscii(this int b)
        {
            if (b >= 32 && b <= 126)
            {
                return (char)b;
            }
            return '_';
        }

        public static IEnumerable HexDump(this byte[] data)
        {
            int bytesPerLine = 16;
            return data
                        .Select((c, i) => new { Char = c, Chunk = i / bytesPerLine })
                        .GroupBy(c => c.Chunk)
                        .Select(g =>
                                  new
```

```
                                {
                                    Hex = g.Select(c => String.Format("{0:X2} ",
c.Char)).Aggregate((s, i) => s + i),
                                    Chars = g.Select(c =>
ToSafeAscii(c.Char).ToString()).Aggregate((s, i) => s + i)
                                })
                        .Select((s, i) =>
                                    new
                                    {
                                        Offset = String.Format("{0:d6}", i * bytesPerLine),
                                        Hex = s.Hex,
                                        Chars = s.Chars
                                    });
        }
    }
}
```

نکته مهم این منبع داده، خروجی IEnumerable آن و Select نهایی عبارت LINQ ایی است که مشاهده می‌کنید. در اینجا اطلاعات به یک شیء ناشناس با اعضای Offset، Hex و Chars نگاشت شده‌اند.
مفهوم فوق از دات نت 3 به بعد تحت عنوان anonymous types در دسترس است. توسط این قابلیت می‌توان یک شیء را بدون نیاز به تعریف ابتدایی آن ایجاد کرد. این نوع‌های ناشناس توسط واژه‌های کلیدی new و var تولید می‌شوند. کامپایلر به صورت خودکار برای هر anonymous type یک کلاس ایجاد می‌کند.

نکته‌ای مهم حین کار با کلاس‌های ناشناس:
کلاس‌های ناشناس به صورت خودکار توسط کامپایلر تولید می‌شوند و ... از نوع internal هم تعریف خواهند شد. به عبارتی در اسمبلی‌های دیگر قابل استفاده نیستند. البته می‌توان توسط ویژگی InternalsVisibleTo: assembly ، تعاریف internal یک اسمبلی را دراختیار اسمبلی دیگری نیز گذاشت. ولی درکل باید به این موضوع دقت داشت و اگر قرار است منبع داده‌ای به این نحو تعریف شود، بهتر است داخل همان اسمبلی تعاریف گزارش باشد.

برای نمایش این نوع اطلاعات حاصل از کوئری‌های LINQ می‌توان از منبع داده پیش فرض AnonymousTypeList به نحو زیر استفاده کرد:

```
using System;
using System.Text;
using PdfRpt.Core.Contracts;
using PdfRpt.FluentInterface;

namespace PdfReportSamples.HexDump
{
    public class HexDumpPdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.LeftToRight);
                doc.Orientation(PageOrientation.Portrait);
                doc.PageSize(PdfPageSize.A4);
                doc.DocumentMetadata(new DocumentMetadata { Author = "Vahid", Application = "PdfRpt",
Keywords = "Test", Subject = "Test Rpt", Title = "Test" });
            })
            .DefaultFonts(fonts =>
            {
                fonts.Path(Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\COUR.ttf",
                    Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\tahoma.TTF");
            })
            .PagesFooter(footer =>
            {
                footer.DefaultFooter(DateTime.Now.ToString("MM/dd/yyyy"));
            })
            .PagesHeader(header =>
            {
                header.DefaultHeader(defaultHeader =>
                {
                    defaultHeader.ImagePath(AppPath.ApplicationPath + "\\Images\\01.png");
                    defaultHeader.Message("Hex Dump");
                });
            })
```

```
            .MainTableTemplate(template =>
            {
                template.CustomTemplate(new GrayTemplate());
            })
            .MainTablePreferences(table =>
            {
                table.ColumnsWidthsType(TableColumnWidthType.Relative);
            })
            .MainTableDataSource(dataSource =>
            {
                var data = Encoding.UTF8.GetBytes("The quick brown fox jumps over the lazy dog.");
                var list = data.HexDump();
                dataSource.AnonymousTypeList(list);
            })
            .MainTableColumns(columns =>
            {
                columns.AddColumn(column =>
                {
                    column.PropertyName("Offset");
                    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
                    column.IsVisible(true);
                    column.Order(0);
                    column.Width(0.5f);
                    column.HeaderCell("Offset");
                });

                columns.AddColumn(column =>
                {
                    column.PropertyName("Hex");
                    column.CellsHorizontalAlignment(HorizontalAlignment.Left);
                    column.IsVisible(true);
                    column.Order(1);
                    column.Width(2.5f);
                    column.HeaderCell("Hex");
                });

                columns.AddColumn(column =>
                {
                    column.PropertyName("Chars");
                    column.CellsHorizontalAlignment(HorizontalAlignment.Left);
                    column.IsVisible(true);
                    column.Order(2);
                    column.Width(1f);
                    column.HeaderCell("Chars");
                });
            })
            .MainTableEvents(events =>
            {
                events.DataSourceIsEmpty(message: "There is no data available to display.");
            })
            .Generate(data => data.AsPdfFile(AppPath.ApplicationPath + "\\Pdf\\HexDumpSampleRpt.pdf"));
        }
    }
}
```

توضیحات:

در اینجا منبع داده بر اساس کلاس‌های کمکی که تعریف کردیم، به نحو زیر مشخص شده است:

```
.MainTableDataSource(dataSource =>
            {
                var data = Encoding.UTF8.GetBytes("The quick brown fox jumps over the lazy dog.");
                var list = data.HexDump();
                dataSource.AnonymousTypeList(list);
            })
```

و سپس برای معرفی ستون‌های متناظر با این منبع داده ناشناس، فقط کافی است آن‌ها را به صورت رشته‌ای معرفی کنیم:

```
column.PropertyName("Offset");
//...
column.PropertyName("Hex");
//...
column.PropertyName("Chars");
```

Hex Dump

Offset	Hex	Chars
000000	54 68 65 C2 A0 71 75 69 63 6B C2 A0 62 72 6F 77	The__quick__brow
000016	6E C2 A0 66 6F 78 C2 A0 6A 75 6D 70 73 C2 A0 6F	n__fox__jumps__o
000032	76 65 72 C2 A0 74 68 65 C2 A0 6C 61 7A 79 C2 A0	ver__the__lazy__
000048	64 6F 67 2E	dog.

نکته‌ای در مورد خواص تودرتو:

در حین استفاده از AnonymousTypeList امکان تعریف خواص تو در تو نیز وجود دارد. برای مثال فرض کنید که Select نهایی به شکل زیر تعریف شده است و در اینجا OrderInfoData نیز خود یک شیء است:

```
.Select(x => new
{
   OrderInfo = x.OrderInfoData
})
```

برای استفاده از یک چنین منبع داده‌ای، ذکر مسیر خاصیت تودرتوی مورد نظر نیز مجاز است:

```
column.PropertyName("OrderInfo.Price");
```

نظرات خوانندگان

نویسنده: مجتبی کاویانی
تاریخ: ۱۳۹۱/۰۷/۱۸ ۰:۳۹

ممنون از مطالب مفیدتون
آیا سطرها با متون طولانی خودکار بزرگتر می‌شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۸ ۰:۴۵

- برای TableColumnWidthType حالت Fit to content هم درنظر گرفته شده که سعی خواهد کرد بر اساس طول محتوای مطالب تمام ستون‌ها و عرض صفحه، عرض ستون‌ها را به صورت خودکار تنظیم کند.
- برای Height یک ردیف، بله. این مورد خودکار است و نیازی به تنظیم ندارد.

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۲:۸

ممنون از کتابخانه pdfReport .
میخوام با این کتابخانه از کنترل jqGrid در mvc خروجی pdf تهیه کنم. به عبارت بهتر میخوام یک کلاس بسازم که بصورت generic باشه. هر نوع jqGrid ی رو که بهش دادم برام تبدیل به pdf کنه. نخواد که برای هر grid یک کلاس بسازم .
با تشکر
لطفا منو راهنمایی کنید ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۶ ۱۲:۲۶

این کتابخانه وابسته به MVC یا WinForms و امثال آن نیست. بر اساس دیتاسورس شما کار می‌کند و سایر تنظیماتی که با کدنویسی مشخص می‌کنید.
یکبار یک قالب کلی برای آن تهیه کنید. سپس از روش تولید پویای ستون‌ها استفاده کنید:
الف) تولید پویای ستون‌ها در حالت استفاده از SQL خام
ب) تولید پویای ستون‌ها در حالت استفاده از ORMها