

مدتی هست که در حال تهیه کتابخانه‌ی گزارش سازی مبتنی بر iTextSharp هستم و عمده‌ی استفاده از آن برای من تاکنون، تهیه گزارشات باکیفیت PDF فارسی تحت وب بوده؛ هر چند این کتابخانه وابستگی خاصی به فناوری مورد استفاده ندارد و با WinForms، WPF، مشتقات ASP.NET، سرویس‌های WCF و کلا هر جایی که دات نت فریم ورک کامل در دسترس باشد، قابل استفاده است. همچنین به منبع داده خاصی گره نخورده است و حتی می‌توانید از یک anonymously typed list عدم متصل به بانک اطلاعاتی خاصی نیز به عنوان منبع داده آن استفاده کنید.

کتابخانه PdfReport به عمد جهت دات نت 3.5 تهیه شده است تا بازه وسیعی از سیستم عامل‌ها را پوشش دهد. این کتابخانه علاوه بر تبدیل اطلاعات شما به گزارشات مبتنی بر PDF، امکان تهیه خروجی خودکار اکسل (2007 به بعد) را نیز دارد. فایل خروجی آن، به صورت پیوست درون فایل PDF تهیه شده قرار می‌گیرد و جزئی از آن می‌شود. بدیهی است اینبار با کتابخانه گزارش سازی روبرو هستید که با راست به چپ مشکلی ندارد! کتابخانه PdfReport بر پایه کتابخانه‌های معروف سورس باز iTextSharp و EPPlus تهیه شده است. حداقل مزیت استفاده از آن، صرفه جویی در وقت شما جهت آموختن ریزه کاری‌های مرتبط با هر کدام از کتابخانه‌های یاد شده است. برای نمونه جهت فراگیری کار با iTextSharp نیاز است یک کتاب 600 صفحه‌ای به نام iText in action را مطالعه و تمرین کنید. این مورد منهای مسایل و نکات متعدد مرتبط با زبان فارسی است که در این کتاب به آن‌ها اشاره‌ای نشده است. برای تهیه آن، مشکلات متداولی که کاربران مدام در انجمن‌های برنامه نویسی مطرح می‌کنند و ابزارهای موجود عاجز از ارائه راه حلی ساده برای حل آن‌ها هستند، مد نظر بوده و امید است نگرش یک این کتابخانه بتواند بسیاری از این دردها را کاهش دهد. کار با این کتابخانه صرفا با کدنویسی میسر است (code first) و همین مساله انعطاف پذیری قابل توجهی را ایجاد کرده که در طی روزهای آینده با جزئیات بیشتر آن آشنا خواهید شد.

اما چرا PDF؟

استفاده از قالب PDF برای تهیه گزارشات، این مزایا را به همراه دارد:

- دقیقا همان چیزی که مشاهده می‌شود، در هر مکانی قابل چاپ است. با همان کیفیت، همان اندازه صفحه، همان فونت و غیره. به این ترتیب به صفحه بندی بسیار مناسب و بهینه‌ای می‌توان رسید و مشکلات گزارشات HTML ایی وب را ندارد.
- امکان استفاد از فونت‌های شکیل‌تر در آن بدون مشکل و بدون نیاز به نصب بر روی کامپیوتری میسر است؛ چون فونت را می‌توان در فایل PDF نیز قرار داد.
- این فایل در تمام سیستم عامل‌ها پشتیبانی می‌شود. خصوصا اینکه فایل نهایی در تمام کامپیوترها و در انواع و اقسام سیستم عامل‌ها به یک شکل و اندازه نمایش داده خواهد شد. برای مثال اینطور نیست که در ویندوز XP، اعداد آن فارسی نمایش داده شوند و در ویندوز 7 با تنظیمات محلی خاصی، دیگر اینطور نباشد. حتی تحت لینوکس هم اعداد آن فارسی نمایش داده خواهد شد چون فونت مخصوص بکار رفته، در خود فایل PDF قابل قرار دادن است.
- برنامه معروف و رایگان Adobe reader برای خواندن و مشاهده آن کفایت می‌کند و البته کلاینت یکبار باید این برنامه را نصب کند. همچنین از این نوع برنامه‌های رایگان برای مشاهده فایل‌های PDF زیاد است.
- تمام صفحات گزارش را در یک فایل می‌توان داشت و به یکباره تمام آن نیز به سادگی قابل چاپ است. این مشکلی است که با گزارشات تحت وب وجود دارد که نمی‌شود مثلا یک گزارش 100 صفحه‌ای را به یکباره به چاپگر ارسال کرد. به همین جهت عموما کاربران درخواست می‌دهند تا کل گزارش را در یک صفحه HTML نمایش دهید تا ما راحت آنرا چاپ کنیم یا راحت از آن خروجی بگیریم. اما زمانیکه فایل PDF تهیه می‌شود این مشکلات وجود نخواهد داشت و جهت Print بسیار بهینه سازی شده است. تا حدی که الان فرمت برگزیده تهیه کتاب‌های الکترونیکی نیز PDF است. مثلا سایت معروف آمازون امکان فروش نسخه PDF کتاب‌ها را هم پیش بینی کرده است.
- امکان صفحه بندی دقیق به همراه مشخص سازی landscape یا portrait بودن صفحه نهایی میسر است. چیزی که در گزارشات صفحات وب آنچنان معنایی ندارد.
- امکان رمزنگاری اطلاعات در آن پیش بینی شده است. همچنین می‌توان به فایل‌های PDF امضای دیجیتال نیز اضافه کرد. به این ترتیب هرگونه تغییری در محتوای فایل توسط برنامه‌های PDF خوان معتبر گزارش داده شده و می‌توان از صحت اطلاعات ارائه

شده توسط آن اطمینان حاصل کرد.

- از فشرده سازی مطالب، فایل ها و تصاویر قرار داده شده در آن پشتیبانی می کند.

- از گرافیک برداری پشتیبانی می کند.

مجوز استفاده از این کتابخانه:

کار من مبتنی بر LGPL است. به این معنا که به صورت باینری (فایل dll) در هر نوع پروژه ای قابل استفاده است.

اما ... PdfReport از دو کتابخانه دیگر نیز استفاده می کند:

- کتابخانه iTextSharp که دارای مجوز AGPL است. این مجوز رایگان نیست.

- کتابخانه EPPPlus برای تولید فایل های اکسل با کیفیت. مجوز استفاده از این کتابخانه LGPL است و تا زمانیکه به صورت باینری از آن استفاده می کنید، محدودیتی را برای شما ایجاد نخواهد کرد.

کتابخانه PdfReport به صورت سورس باز در CodePlex [قرار گرفته](#) ؛ اما جهت پرسیدن سؤالات، پیشنهادات، ارائه بهبود و غیره می توانید (و بهتر است) از قسمت [مدیریت پروژه مرتبط در سایت جاری](#) نیز استفاده کنید.

نحوه تهیه اولین گزارش، با کتابخانه PdfReport

الف) یک پروژه Class library جدید را شروع کنید. از این جهت که گزارشات PdfReport در انواع و اقسام پروژه های VS.NET قابل استفاده است، می توان از این پروژه Class library به عنوان کلاس های پایه قابل استفاده در انواع و اقسام پروژه های مختلف، بدون نیاز به تغییری در کدهای آن استفاده کرد.

ب) آخرین نگارش فایل های مرتبط با PdfReport را [از اینجا](#) دریافت کنید و سپس ارجاعاتی را به اسمبلی های موجود در بسته آن به پروژه خود اضافه نمایید (ارجاعاتی به PdfReport، iTextSharp و EPPPlus). فایل XML راهنمای کتابخانه نیز به همراه بسته آن می باشد که در حین استفاده از متدها و خواص PdfReport کمک بزرگی خواهد بود.

ج) کلاس های زیر را به آن اضافه کنید:

```
using System.Web;
using System.Windows.Forms;

namespace PdfReportSamples
{
    public static class AppPath
    {
        public static string ApplicationPath
        {
            get
            {
                if (isInWeb)
                    return HttpRuntime.AppDomainAppPath;

                return Application.StartupPath;
            }
        }

        private static bool isInWeb
        {
            get
            {
                return HttpContext.Current != null;
            }
        }
    }
}
```

از این کلاس برای مشخص سازی محل ذخیره سازی فایل های نهایی PDF تولیدی استفاده خواهیم کرد. همانطور که مشاهده می کنید ارجاعاتی را به System.Web.dll و System.Windows.Forms.dll نیاز دارد.

در ادامه کلاس User را جهت ساخت یک منبع داده درون حافظه‌ای تعریف خواهیم کرد:

```
using System;

namespace PdfReportSamples.IList
{
    public class User
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public string LastName { set; get; }
        public long Balance { set; get; }
        public DateTime RegisterDate { set; get; }
    }
}
```

اکنون کلاس اصلی گزارش ما به صورت زیر خواهد بود:

```
using System;
using System.Collections.Generic;
using PdfRpt.Core.Contracts;
using PdfRpt.FluentInterface;

namespace PdfReportSamples.IList
{
    public class IListPdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                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\\tahoma.ttf",
Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.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("گزارش جدید ما");
                });
            })
            .MainTableTemplate(template =>
            {
                template.BasicTemplate(BasicTemplate.ClassicTemplate);
            })
            .MainTablePreferences(table =>
            {
                table.ColumnsWidthsType(TableColumnWidthType.Relative);
                table.NumberOfDataRowsPerPage(5);
            })
            .MainTableDataSource(dataSource =>
            {
                var listOfRows = new List<User>();
                for (int i = 0; i < 200; i++)
                {
                    listOfRows.Add(new User { Id = i, LastName = "نام خانوادگی" + i, Name = "نام " + i,
Balance = i + 1000 });
                }
                dataSource.StronglyTypedList<User>(listOfRows);
            })
            .MainTableSummarySettings(summarySettings =>
            {
                summarySettings.OverallSummarySettings("جمع کل");
                summarySettings.PerviousPageSummarySettings("نقل از صفحه قبل");
            })
        }
    }
}
```

```

summarySettings.PageSummarySettings("جمع صفحه");
})
.MainTableColumns(columns =>
{
    columns.AddColumn(column =>
    {
        column.PropertyName("rowNo");
        column.IsRowNumber(true);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(0);
        column.Width(1);
        column.HeaderCell("#");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.Id);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(1);
        column.Width(2);
        column.HeaderCell("شماره");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.Name);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(2);
        column.Width(3);
        column.HeaderCell("نام");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.LastName);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(3);
        column.Width(3);
        column.HeaderCell("نام خانوادگی");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.Balance);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(4);
        column.Width(2);
        column.HeaderCell("موجودی");
        column.ColumnItemsTemplate(template =>
        {
            template.TextBlock();
            template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
        column.AggregateFunction(aggregateFunction =>
        {
            aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
            aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
    });

})
.MainTableEvents(events =>
{
    events.DataSourceIsEmpty(message: "رکوردی یافت نشد.");
})
.Export(export =>
{
    export.ToExcel();
    export.ToCsv();
    export.ToXml();
})
.Generate(data => data.AsPdfFile(AppPath.ApplicationPath + "\\Pdf\\RptIListSample.pdf"));
}

```

}

و برای استفاده از آن:

```
var rpt = new IListPdfReport().CreatePdfReport();
// rpt.FileName
```



گزارش جدید ما

ردیف	شماره	نام	نام خانوادگی	موجودی
			نقل از صفحه قبل	۲۱۳,۹۱۵
۱۹۶	۱۹۵	نام ۱۹۵	نام خانوادگی ۱۹۵	۱,۱۹۵
۱۹۷	۱۹۶	نام ۱۹۶	نام خانوادگی ۱۹۶	۱,۱۹۶
۱۹۸	۱۹۷	نام ۱۹۷	نام خانوادگی ۱۹۷	۱,۱۹۷
۱۹۹	۱۹۸	نام ۱۹۸	نام خانوادگی ۱۹۸	۱,۱۹۸
۲۰۰	۱۹۹	نام ۱۹۹	نام خانوادگی ۱۹۹	۱,۱۹۹
			جمع صفحه	۵,۹۸۵
			جمع کل	۲۱۹,۹۰۰

برای نمونه، جهت مشاهده نمایش این خروجی در یک برنامه ویندوزی، به مثال‌های همراه سورس پروژه [در مخزن کد آن](#) مراجعه نمایید.

توضیحات بیشتر:

- در قسمت DocumentPreferences، جهت راست به چپ (PdfRunDirection)، اندازه صفحه (PdfPageSize)، جهت صفحه (PageOrientation) و امثال آن تنظیم می‌شوند.
- سپس نیاز است قلم‌های مورد استفاده در گزارش مشخص شوند. در متد DefaultFonts باید دو قلم را معرفی کنید. قلم اول، قلم پیش فرض خواهد بود و قلم دوم برای رفع نواقص قلم اول مورد استفاده قرار می‌گیرد. برای مثال اگر قلم اول فاقد حروف انگلیسی است، به صورت خودکار به قلم دوم رجوع خواهد شد.
- در ادامه در متد PagesFooter، تاریخ درج شده در پایین تمام صفحات مشخص می‌شود. در مورد ساخت Footer سفارشی در قسمت‌های بعدی بحث خواهد شد.
- در متد PagesHeader، متن و تصویر قرار گرفته در Header تمام صفحات گزارش قابل تنظیم است. این مورد نیز قابل سفارشی سازی است که در قسمت‌های بعد به آن خواهیم پرداخت.
- توسط MainTableTemplate، قالب ظاهری ردیف‌های گزارش مشخص می‌شود. یک سری قالب پیش فرض در کتابخانه PdfReport موجود است که توسط متد BasicTemplate آن قابل دسترسی است. در مورد نحوه تعریف قالب‌های سفارشی به مرور در قسمت‌های بعد، بحث خواهد شد.
- در قسمت MainTablePreferences تنظیمات جدول اصلی گزارش تعیین می‌شود. برای مثال چه تعداد ردیف در صفحه نمایش داده شود. اگر این مورد را تنظیم نکنید، به صورت خودکار محاسبه خواهد شد. نحوه تعیین عرض ستون‌های گزارش به کمک متد ColumnsWidthsType مشخص می‌شود که در اینجا حالت نسبی در نظر گرفته شده است.
- منبع داده مورد استفاده توسط متد MainTableDataSource مشخص می‌شود که در اینجا یک لیست جنریک تعیین شده و سپس

توسط متد `StronglyTypedList` در اختیار گزارش ساز جاری قرار می‌گیرد. تعدادی منبع داده پیش فرض در PdfReport وجود دارند که هر کدام را در قسمت‌های بعدی بررسی خواهیم کرد. همچنین امکان تعریف منابع داده سفارشی نیز وجود دارد.

- با کمک متد `MainTableSummarySettings`، برچسب‌های جمع‌های پایین صفحات مشخص می‌شود.
- در قسمت `MainTableColumns`، ستون‌هایی را که علاقمندیم در گزارش ظاهر شوند، قید می‌کنیم. هر ستون باید با یک فیلد یا خاصیت منبع داده متناظر باشد. همچنین همانطور که مشاهده می‌کنید امکان تعیین `Visibility`، عرض و غیره آن نیز مهیا است (قابلیت ساخت گزارشاتی که به انتخاب کاربر، ستون‌های آن ظاهر یا مخفی شوند). در اینجا توسط `callback`‌هایی که در متد `ColumnItemsTemplate` قابل دسترسی هستند، می‌توان اطلاعات را پیش از نمایش فرمت کرد. برای مثال سه رقم جدا کننده به اعداد اضافه کرد (برای نمونه در خاصیت موجودی فوق) و یا توسط متد `AggregateFunction`، می‌توان متد تجمعی مناسبی را جهت ستون جاری مشخص کرد.
- توسط متد `MainTableEvents` به بسیاری از رخدادهای داخلی PdfReport دسترسی خواهیم یافت. برای مثال اگر در اینجا رکوردی موجود نباشد، رخداد `DataSourceIsEmpty` صادر خواهد شد.
- به کمک متد `Export`، خروجی‌های دلخواه مورد نظر را می‌توان مشخص کرد. تعدادی خروجی، مانند اکسل، XML و CSV در این کتابخانه موجود است. امکان سفارشی سازی آن‌ها نیز پیش بینی شده است.
- و نهایتاً توسط متد `Generate` مشخص خواهیم کرد که فایل گزارش کجا ذخیره شود.

لطفاً برای طرح مشکلات و سؤالات خود در رابطه با کتابخانه PdfReport [از این قسمت سایت](#) استفاده کنید.

نظرات خوانندگان

نویسنده: محمد صافدل
تاریخ: ۱۳۹۱/۰۷/۱۳ ۸:۴۲

ممنون آقای نصیری. من با توجه به مقالات گذشته شما در زمینه iTextSharp یک کتابخانه کوچک برای کارهای خودم تهیه کرده بودم ولی محدودیت‌ها و مشکلات زیادی داشت. با توجه به اینکه کیفیت pdfهای سایت شما به نظرم خیلی خوب هستن تصمیم داشتم از شما درباره نحوه ساخت اونها سوال کنم که خودتون زحمتشو کشیدید. قصد دارم با اجازه شما این کتابخانه را جایگزین کنم. باز هم از شما ممنونم

نویسنده: محمد
تاریخ: ۱۳۹۱/۰۷/۱۳ ۹:۱۲

بسیار بسیار ممنون و متشکر
جدا برای گزارشات خصوصاً در وب مشکل داشتم

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۷/۱۳ ۱۱:۴۸

ایول دارین آقای نصیری دی ... از این حرکتتون لذت بردم دی ... خدا اجرت بده دی

نویسنده: mohsen
تاریخ: ۱۳۹۱/۰۷/۱۳ ۱۳:۸

ببخشید آقای نصیری آیا با استفاده از این کتابخانه می‌توان هر گزارشی را با هر شکلی طراحی کرد؟

ممنونم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۳ ۱۳:۲۳

کلمات «هر گزارشی» و «هر شکلی» یعنی چی؟
اگر «هر شکلی» منظور ساخت گزارشات نامنظم است، شاید [این روش](#) استفاده از قالب‌های open office بهتر باشد.

نویسنده: mohsen
تاریخ: ۱۳۹۱/۰۷/۱۳ ۱۳:۳۴

مثلاً بخواهیم هم از این مثالی که فرمودید با استفاده از open office پیاده سازی کردید استفاده کنم و در پایین صفحه یه جدول که به صورت یک Grid اطلاعات را نمایش دهد. مثلاً بخواهیم اطلاعات یک شخص را در بالای صفحه نمایش دهیم و در پایین صفحه تمام درخواست‌های آن کاربر را باید لیست کنیم در این حالت از کدام روش استفاده کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۳ ۱۳:۳۸

- می‌تونید یک هدر سفارشی در PdfReport درست کنید. (چیزی که به همراه آن است یک هدر ساده همه منظوره است)
- می‌تونید از قالب‌های AcroForm مربوط به Open Office در هدر Pdf Report استفاده کنید.
این موارد رو در قسمت‌های بعدی توضیح می‌دم.

نویسنده: mohsen
تاریخ: ۱۳۹۱/۰۷/۱۳ ۱۳:۴۰

ممنونم استاد. پس من مشتاقانه منتظر این مطلبتون هستم. باز هم ممنونم

نویسنده: مجتبی حاجی وندیان
تاریخ: ۱۴:۰ ۱۳۹۱/۰۷/۱۳

استاد نصیری ممنون از تمام زحماتی که می‌کشید؛ در حال حاضر جز تشکر چیزی از دستم بر نیامد.

نویسنده: مسعود زیانی
تاریخ: ۱۴:۵ ۱۳۹۱/۰۷/۱۳

واقعا از زحماتتون تشکر میکنم

خیلی مفید بود.

نویسنده: سیروان عقیفی
تاریخ: ۱۶:۱۷ ۱۳۹۱/۰۷/۱۳

استاد خیلی ممنون واقعا شاهکاره.

نویسنده: حسن زاده
تاریخ: ۱۲:۳۱ ۱۳۹۱/۰۷/۱۴

ممنون مهندس واقعا کار جالبی هست

به نظرتون طراحی یه محیط Designer برای این کتابخانه هم میتونه خوب باشه
اگه در این مورد موافق بودین من هم می‌تونم کمک کنم

نویسنده: وحید نصیری
تاریخ: ۱۳:۱۸ ۱۳۹۱/۰۷/۱۴

یکی از خوبی‌های کارهای سورس باز امکان مشارکت عموم برنامه نویسی‌ها است. اگر علاقمند به مشارکت در آن بودید، می‌تونید
وصله‌های خودتون رو [در اینجا](#) ارسال نمائید تا به پروژه یا مثال‌های آن اضافه شوند. با تشکر

نویسنده: شاهین کیاست
تاریخ: ۱۸:۴۵ ۱۳۹۱/۰۷/۱۵

فوق العادست .

خسته نباشید.

نویسنده: ناصر طاهری
تاریخ: ۲۲:۴۰ ۱۳۹۱/۰۷/۱۵

مثل همیشه عالی

تشکر فراوان دارم از شما

نویسنده: حسین مرادی نیا
تاریخ: ۲۳:۵۱ ۱۳۹۱/۰۷/۱۵

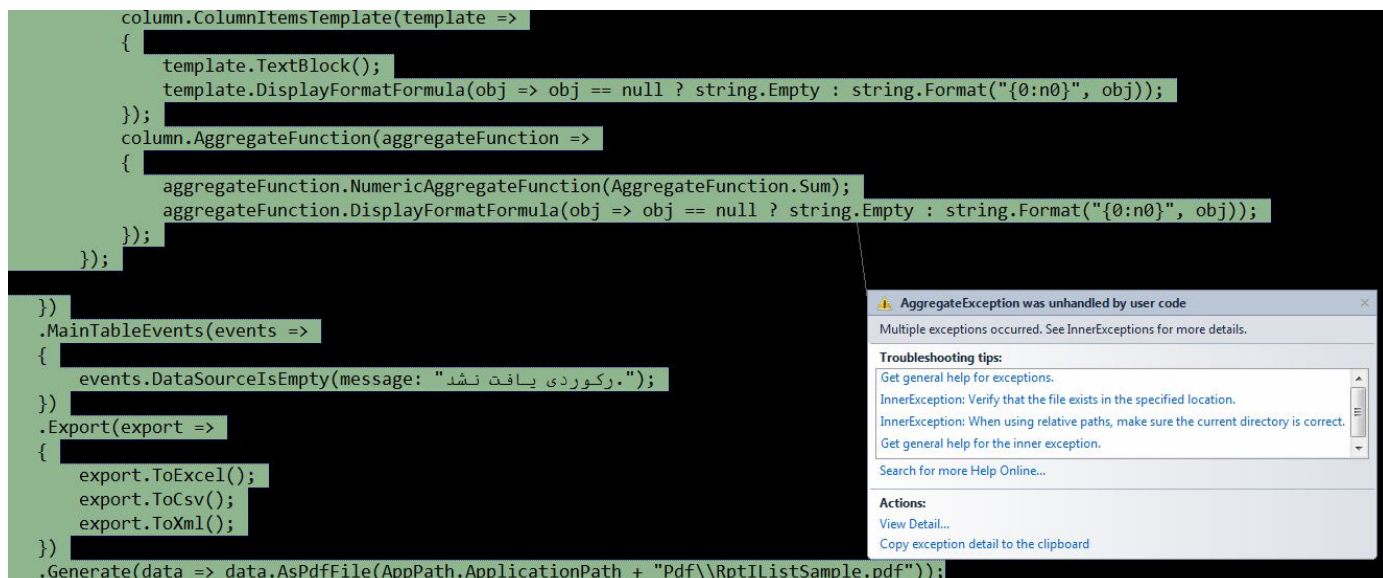
ممنون آقای نصیری

واقعا عالیه

نویسنده: M.B

تاریخ: ۱۰:۴۴ ۱۳۹۱/۰۷/۲۰

استاد من یک پروژه از نوع WebForms ایجاد کردم و سه کلاسی که شما در اینجا ایجاد کردید را عیناً در پروژه ایجاد کردم ولی زمانی که پروژه را اجرا می‌کنم به خطای زیر بر می‌خورم



نویسنده: وحید نصیری

تاریخ: ۱۰:۵۰ ۱۳۹۱/۰۷/۲۰

- لطفاً برای رفع مشکلات مرتبط با PdfReport [از این قسمت](#) سایت استفاده نمائید.
 - برای مشاهده خطای واقعی بر روی لینک view details (ذیل قسمت Actions تصویر فوق) کلیک کنید. در اینجا بهتر می‌توان بررسی کرد که مشکل اصلی چه چیزی بوده است. (ممکن است فونت مورد استفاده در مسیر برنامه شما نباشد، یا دسترسی write نداشته باشید، یا پوشه خروجی pdf در مسیر و ریشه برنامه شما ایجاد نشده (مطابق تنظیمات AppPath انتهای گزارش)، یا هر خطای دیگری که ریز آن در قسمت view details یاد شده، ذکر می‌شود)

نویسنده: پویا امینی

تاریخ: ۱۶:۲۱ ۱۳۹۱/۰۸/۲۹

اگر بخواهیم در قسمت Footer گزارش (هر صفحه)، فقط از </hr> و یک نوشته استفاده کنیم باید از چه تنظیماتی استفاده کنیم؟
 ممنون

نویسنده: وحید نصیری

تاریخ: ۱۶:۳۲ ۱۳۹۱/۰۸/۲۹

لطفاً برجسب PdfReport را در سایت بررسی کنید. به این موضوع پرداخته شده: ([^](#))

نویسنده: molana11

تاریخ: ۱۱:۵۴ ۱۳۹۱/۱۰/۱۶

با سلام. آقای نصیری آیا می‌توان از pdf report در سیلورلایت استفاده کرد؟ با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۶ ۱۲:۳۷

بله (با استفاده از یک سرویس WCF). مثال‌های آن را [دریافت کنید](#) . مراجعه کنید به پوشه [SLPdf](#) آن جهت مشاهده یک نمونه آماده.

نویسنده: molana11
تاریخ: ۱۳۹۱/۱۰/۱۸ ۰:۱۹

آقای نصیری .

با سلام. من قصد دارم گزارشاتم را براساس pdf report ایجاد کنم ولی چند تا مشکل دارم. در صورت امکان راهنمایی کنید:

- 1- آیا می‌شود محتوای فایل پی دی اف را درون سیلورلایت مثلاً در یک گرید یا یک بوردر نمایش داد ؟
 - 2- حجم فایل خروجی در مثالی که شما برای دانلود قرار دادید 400 کیلوبایت بود. آیا می‌توان حجم آنرا پایین آورد؟
 - 3- در مثال شما حدود 20 ثانیه طول کشید تا گزارش ساخته شود. آیا می‌شود این زمان را کمتر کرد؟
- متشکرم از زحمات شما.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۸ ۰:۲۹

- نیاز به نمایش دهنده PDF نوشته شده با سیلورلایت دارید. یک سری کار تجاری [از تلریک](#) و امثال آن ([^](#) , [^](#)) برای اینکار هست.

- حجم فایل نهایی به اندازه کافی فشرده شده است. استفاده از تصاویر یا تعداد صفحات بالا، حجم را بیشتر خواهند کرد به همراه بالا بردن مدت زمان تولید فایل. همچنین یک سری پیوست/خروجی جانبی نیز به فایل اضافه می‌شوند، مانند خروجی اکسل، xml و csv. هر کدام از این‌ها را که مورد نیاز نیستند، در حین تهیه گزارش ذکر نکنید تا فایل نهایی حجم کمتری داشته باشد. بدیهی است تولید هر کدام نیز زمانی را به خود اختصاص خواهند داد.

- مثالی که موجود بود را تست کردم حدود 1 ثانیه بیشتر طول نکشید؛ نه 20 ثانیه.

روشی وجود دارد به نام warmup برای خیلی از کارهای دات نت. در پشت صحنه سیستم، حین اجرای اولیه برنامه یک گزارش خالی را تولید کنید. به این صورت سیستم JIT دات نت مجبور خواهد شد سریعتر وارد عمل شود (نه در زمان نیاز). در دفعه بعد فراخوانی گزارشات، نتیجه کار بسیار سریع خواهد بود.

نویسنده: molana11
تاریخ: ۱۳۹۱/۱۰/۱۸ ۰:۴۱

متشکرم از پاسخ شما.

هنگام ساخته شدن گزارش بلافاصله پنجره دانلود باز می‌شود و درون فریم چیزی نمایش داده نمی‌شود، مشکل از کجاست؟
با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۸ ۰:۴۳

احتمالا دانلود منیجر شما طوری تنظیم شده که لینک‌های PDF را پیش از نمایش در مرورگر، ردیابی می‌کند.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۰/۲۱ ۱۷:۳۰

مهندس نصیری با سلام.

می‌خواستم بابت تهیه pdfReport از شما تشکر کنم.

من در پروژه خودم از pdfReport استفاده کردم. بسیار هم راضی هستم. هم در سیلورلایت و هم در mvc. مزایای code first در ابتدا برایم غیرقابل قبول بود ولی الان code first انتخاب اول منه. همه جوره این کتابخانه عالیه. بسیار بسیار سپاس مهندس جان. یاعلی.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۵:۴۱ ۱۳۹۱/۱۰/۲۳

درون سیلورلایت ، سورس فریم را به مسیر فایل پی دی اف (که توسط دستور AsPdfFile تولید می شود بوسیله WCF) ست میکنم. امکان این وجود دارد که فایل بطور مستقیم درون فریم نمایش داده شود و در جایی ذخیره نشود؟

نویسنده: وحید نصیری
تاریخ: ۱۷:۱۱ ۱۳۹۱/۱۰/۲۳

- لطفا از قسمت پرسش و پاسخ مخصوص این پروژه در سایت برای طرح سؤالات خودتون استفاده کنید.
- شبیه به سؤال شما قبلا مطرح شده.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۷:۲۲ ۱۳۹۱/۱۰/۲۳

متوجه شدم.
متشکرم.

نویسنده: امیر هاشم زاده
تاریخ: ۲۳:۱۳ ۱۳۹۳/۰۵/۲۰

[اینجا](#) راه حلش رو آقای نصیری رو توضیح دادند.

نویسنده: امیر هاشم زاده
تاریخ: ۲۰:۳ ۱۳۹۳/۰۵/۲۲

برچسب های summary، چگونه مقداردهی می شوند؟ و محل آنها چگونه تعیین می شود؟ به عبارت دیگر ارتباط متد MainTableSummarySettings با فیلدهای منبع داده متناظر چگونه است؟

نویسنده: وحید نصیری
تاریخ: ۹:۳۸ ۱۳۹۳/۰۵/۲۲

- به صورت خودکار قبل از اولین ستونی که دارای AggregateFunction است ظاهر خواهند شد.
- همچنین متدهای داخل MainTableSummarySettings دارای پارامترهای دیگری هم هستند (تک پارامتری نیستند) که بتوان توسط آنها محل ظاهر شدن برچسب ها را صریحا مشخص کرد.

نویسنده: امیر هاشم زاده
تاریخ: ۱۰:۱۹ ۱۳۹۳/۰۵/۲۲

1- در صورتیکه چندین ستون عددی داشته باشیم و بخواهیم برای همه آنها برچسب های "نقل از صفحه قبل"، "جمع صفحه"، "جمع کل" داشته باشیم، چگونه امکان پذیر است؟
2- آیا درست متوجه شدم که: جمع ستون دارای AggregateFunction، مقابل برچسب "جمع صفحه" درج می شود؟

نویسنده: وحید نصیری
تاریخ: ۱۰:۳۳ ۱۳۹۳/۰۵/۲۲

- هر ستون عددی که در تعریف آن AggregateFunction درج و مشخص شود، به صورت خودکار دارای فیلدهای ته جمع خواهد بود.
- برچسب ها هم فقط یکبار در هر ردیف مشخص می شوند به صورت خودکار (پیش فرض) یا حالت سفارشی که عنوان شد (مرسوم تمام گزارشات متداول به همین نحو است).

در گزارشات، گاهی از اوقات نیاز خواهد شد تا تعدادی ستون جدید را بر اساس مقادیر فیلدهای موجود در منبع داده گزارش، به صورت پویا محاسبه و تولید کنیم (ایجاد ستون‌هایی که در منبع داده وجود خارجی ندارند). در ادامه با نحوه انجام اینکار در PdfReport آشنا خواهیم شد.

در ابتدا کدهای کامل گزارش این قسمت را در ادامه ملاحظه خواهید کرد (در این کلاس از دو کلاس User و AppPath [قسمت قبل](#) نیز استفاده شده است):

```
using System;
using System.Collections.Generic;
using PdfReportSamples.Models;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;
using PdfRpt.FluentInterface;

namespace PdfReportSamples.CalculatedFields
{
    public class CalculatedFieldsPdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                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(AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
                    Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\verdana.ttf");
            })
            .PagesFooter(footer =>
            {
                footer.DefaultFooter(printDate: DateTime.Now.ToString("MM/dd/yyyy"));
            })
            .PagesHeader(header =>
            {
                header.DefaultHeader(defaultHeader =>
                {
                    defaultHeader.ImagePath(AppPath.ApplicationPath + "\\Images\\01.png");
                    defaultHeader.Message("گزارش جدید ما");
                })
            })
            .MainTableTemplate(template =>
            {
                template.BasicTemplate(BasicTemplate.RainyDayTemplate);
            })
            .MainTablePreferences(table =>
            {
                table.ColumnsWidthsType(TableColumnWidthType.Relative);
            })
            .MainTableDataSource(dataSource =>
            {
                var listOfRows = new List<User>();
                for (int i = 0; i < 220; i++)
                {
                    listOfRows.Add(new User { Id = i, LastName = "نام خانوادگی" + i, Name = "نام" + i,
Balance = i + 1000 });
                }
                dataSource.StronglyTypedList<User>(listOfRows);
            })
            .MainTableEvents(events =>
            {
                events.DataSourceIsEmpty(message: "رکوردی یافت نشد.");
            })
            .MainTableSummarySettings(summary =>
            {
                summary.OverallSummarySettings("جمع");
            })
        }
    }
}
```

```

summary.PreviousPageSummarySettings("نقل از صفحه قبل");
summary.PageSummarySettings("جمع صفحه");
})
.MainTableColumns(columns =>
{
    columns.AddColumn(column =>
    {
        column.PropertyName("rowNo");
        column.IsRowNumber(true);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(0);
        column.Width(1);
        column.HeaderCell("ردیف", captionRotation: 90);
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.Id);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(1);
        column.Width(2);
        column.HeaderCell("شماره");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.Name);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(2);
        column.Width(2);
        column.HeaderCell("نام");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.LastName);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(3);
        column.Width(3);
        column.HeaderCell("نام خانوادگی");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("CF1");
        column.CalculatedField(true,
            list =>
            {
                if (list == null) return string.Empty;
                var name = list.GetSafeStringValueOf<User>(x => x.Name);
                var lastName = list.GetSafeStringValueOf<User>(x => x.LastName);
                return name + " - " + lastName;
            });
        column.HeaderCell("ف.م.");
        column.Width(3);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(4);
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.Balance);
        column.HeaderCell("موجودی");
        column.ColumnItemsTemplate(template =>
        {
            template.TextBlock();
            template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
        column.Width(2);
        column.AggregateFunction(aggregateFunction =>
        {
            aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
            aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
    });
}

```

```

        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(5);
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("CF2");
        column.HeaderCell("ف.م.");
        column.Width(3);
        column.AggregateFunction(aggregateFunction =>
        {
            aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
            aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.ColumnItemsTemplate(template =>
        {
            template.TextBlock();
            template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
        column.CalculatedField(true,
            list =>
            {
                if (list == null) return string.Empty;
                var balance = list.GetValueOf<User>(x => x.Balance);
                return (long)balance * 3;
            });
        column.IsVisible(true);
        column.Order(6);
    });
}

.Export(export =>
{
    export.ToExcel(footer: "Footer text", header: "&24&U&\\"Arial,Regular Bold\\" New rpt.",
pageLayoutView: true);
    export.ToXml1();
})
.Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
"\\Pdf\\RptCalculatedFieldsSample.pdf"));
}
}

```

توضیحات:

- در این مثال از یک قلم سفارشی به نام Iranian Sans استفاده شده که در پوشه bin\fonts [سورس به روز شده](#) پروژه، قابل دریافت است.
- برخلاف [قسمت قبل](#)، از متد table.NumberOfDataRowsPerPage استفاده نشده است. به این ترتیب تعداد ردیف‌ها به صورت خودکار بر اساس اندازه صفحه محاسبه می‌شود.
- منبع داده تعریف شده توسط dataSource.StronglyTypedList، بسیار مناسب است جهت کار با انواع و اقسام ORM‌ها. تفاوتی نمی‌کند از چه ORM ایی استفاده می‌کنید؛ همینقدر که خروجی کار شما یک List باشد، در اینجا قابل استفاده خواهد بود. حتی نیازی هم به بانک اطلاعاتی نیست و در این مثال از یک منبع داده درون حافظه‌ای استفاده شده است.
- توضیحاتی در مورد ColumnsWidthsType :
- برای تعیین عرض ستون‌ها، چهار حالت بر اساس مقادیر enum ایی به نام TableColumnWidthType میسر است:
- الف) Relative : عرض نسبی. به این معنا که اگر سه ستون با عرض‌های 1, 1, 2، تعریف کنید، کل عرض صفحه به 4 قسمت تقسیم می‌شود. از این 4 قسمت، 2 قسمت به ستون اول و یک قسمت به ستون دوم و همچنین یک قسمت به ستون سوم اختصاص خواهد یافت.
- ب) Absolute : در این حالت باید عرض ستون‌ها را دقیقاً بر اساس user space units مشخص کنید.
- ج) FitToContent : سعی خواهد کرد بر اساس طول محتوای یک سلول، عرض بهینه‌ای را محاسبه کند. در این حالت نیازی به قید column.Width نیست.
- د) EquallySized : به صورت خودکار عرض تمام ستون‌ها را یکسان محاسبه می‌کند. در این حالت نیازی به قید column.Width نیست.
- اولین فیلد محاسباتی که در PdfReport به صورت توکار و خودکار در اختیار شما است، فیلد شماره ردیف می‌باشد که به صورت

زیر مشخص می‌شود:

```
column.IsRowNumber(true);
```

بنابراین نیازی نیست تا منبع داده شما شامل یک ستون اضافی به نام ردیف باشد. PdfReport این مورد را به صورت خودکار تولید خواهد کرد.

- سپس دو فیلد و ستون محاسباتی در گزارش فوق قابل مشاهده هستند:

```
columns.AddColumn(column =>
{
    column.PropertyName("CF1");
    column.CalculatedField(true,
        list =>
        {
            if (list == null) return string.Empty;
            var name = list.GetSafeStringValueOf<User>(x => x.Name);
            var lastName = list.GetSafeStringValueOf<User>(x => x.LastName);
            return name + " - " + lastName;
        });
    column.HeaderCell("م.ف.");
    column.Width(3);
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.IsVisible(true);
    column.Order(4);
});
```

برای اینکه یک فیلد پویا را به مجموعه فیلدهای مهیا شده توسط منبع داده اضافه کنیم، از متد `CalculatedField` با پارامتر اول مساوی `true` استفاده خواهیم کرد. سپس نیاز است نحوه محاسبه این فیلد را مشخص کنیم. امضای متد `CalculatedField` به صورت زیر است:

```
public void CalculatedField(bool isCalculatedField, Func<IList<CellData>, object>
calculatedFieldFormula)
```

به این معنا که توسط آرگومان دوم آن، لیست کلیه مقادیر ردیف جاری در اختیار شما قرار خواهد گرفت. در این بین فرصت خواهیم داشت بر این اساس، فیلد و مقدار جدیدی را تولید کرده و بازگشت دهیم؛ که نمونه‌ای از اینکار را در فیلد محاسباتی `CF1` فوق مشاهده می‌کنید.

باید دقت داشت که نام خواص (`column.PropertyName`) باید منحصریفر باشد و گرنه برنامه با یک استثناء متوقف خواهد شد. اگر ستون معرفی شده متناظر است با یک فیلد یا خاصیت منبع داده، باید `PropertyName` با رعایت کوچکی و بزرگی حروف، معادل فیلد متناظر باشد. اگر ستون تعریف شده یک فیلد محاسباتی است، تنها کافی است یک نام دلخواه غیرتکراری را ذکر کرد. همچنین جهت سهولت کار، در فضای نام `PdfRpt.Core.Helper`، تعدادی متد برای کار با لیستی از `CellData`ها تدارک دیده شده‌اند؛ که نمونه‌ای از آن‌را در اینجا با استفاده از متدهای `GetSafeStringValueOf` ملاحظه می‌کنید.

- فیلد محاسباتی دیگری نیز در این گزارش قابل مشاهده است:

```
columns.AddColumn(column =>
{
    column.PropertyName("CF2");
    column.HeaderCell("م.ف.");
    column.Width(3);
    column.AggregateFunction(aggregateFunction =>
    {
        aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
        aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
    });
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.ColumnItemsTemplate(template =>
    {
        template.TextBlock();
        template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
    });
});
```

```
column.CalculatedField(true,
    list =>
    {
        if (list == null) return string.Empty;
        var balance = list.GetValueOf<User>(x => x.Balance);
        return (long)balance * 3;
    });
column.IsVisible(true);
column.Order(6);
});
```

در اینجا در متد `column.CalculatedField` مقدار فیلد موجودی (`Balance`) ردیف جاری دریافت شده و سپس مقدار دلخواه جدیدی تولید و بازگشت داده شده است.

همچنین توسط متد `column.AggregateFunction` مشخص کرده‌ایم که این ستون جدید نیاز به جمع پایین صفحه هم دارد. به علاوه توسط متد `column.ColumnItemsTemplate` با مشخص سازی `DisplayFormatFormula`، پیش از نمایش اطلاعات فیلد جاری، مقدار آن را دریافت و فرمت کرده‌ایم؛ که در اینجا سه رقم جداکننده اعداد اضافه شده است.

ذکر متد `template.TextBlock` اختیاری است و حالت پیش فرض می‌باشد. قالب‌های دیگری نیز در اینجا تعریف شده‌اند که در مثال‌های قسمت‌های بعدی آن‌ها را بررسی خواهیم کرد (امکان نمایش تصویر، لینک، بارکد و غیره).

 <p>گزارش جدید ما</p>						
ردیف	شماره	نام	نام خانوادگی	ف.م.	موجودی	ف.م.
				نقل از صفحه قبل		
۲۱۵	۲۱۴	نام ۲۱۴	نام خانوادگی ۲۱۴	نام ۲۱۴ - نام خانوادگی ۲۱۴	۱,۲۱۴	۲۳۶,۷۹۱
۲۱۶	۲۱۵	نام ۲۱۵	نام خانوادگی ۲۱۵	نام ۲۱۵ - نام خانوادگی ۲۱۵	۱,۲۱۵	۳,۶۴۵
۲۱۷	۲۱۶	نام ۲۱۶	نام خانوادگی ۲۱۶	نام ۲۱۶ - نام خانوادگی ۲۱۶	۱,۲۱۶	۳,۶۴۸
۲۱۸	۲۱۷	نام ۲۱۷	نام خانوادگی ۲۱۷	نام ۲۱۷ - نام خانوادگی ۲۱۷	۱,۲۱۷	۳,۶۵۱
۲۱۹	۲۱۸	نام ۲۱۸	نام خانوادگی ۲۱۸	نام ۲۱۸ - نام خانوادگی ۲۱۸	۱,۲۱۸	۳,۶۵۴
۲۲۰	۲۱۹	نام ۲۱۹	نام خانوادگی ۲۱۹	نام ۲۱۹ - نام خانوادگی ۲۱۹	۱,۲۱۹	۳,۶۵۷
				جمع صفحه		
				جمع		
					۷,۲۹۹	۲۱,۸۹۷
					۲۴۴,۰۹۰	۷۳۲,۲۷۰

تعدادی از منابع داده پیش فرض PdfReport جهت کار مستقیم با بانک‌های اطلاعاتی مختلف، کوئری نوشتن و نمایش نتایج آن‌ها طراحی شده‌اند.

در این بین با توجه به اینکه دات نت پشتیبانی توکاری از SQL Server دارد، اتصال و استفاده از توانمندی‌های آن نیاز به کتابخانه جانبی خاصی ندارد. اما برای کار با [بانک‌های اطلاعاتی دیگر](#) نیاز خواهد بود تا پروایدر ADO.NET آن‌ها را تهیه و به برنامه اضافه کنیم. چهار نمونه از منابع داده پیش فرضی که در متد MainTableDataSource قابل تعریف هستند به شرح زیر می‌باشند:

```
public void SqlDataReader(string connectionString, string sql, params object[] parametersValues)
//.mdb or .accdb files
public void AccessDataReader(string filePath, string password, string sql, params object[] parametersValues)
public void OdbcDataReader(string connectionString, string sql, params object[] parametersValues)
```

SqlDataReader برای کار با بانک‌های اطلاعاتی SQL Server بهینه سازی شده است.

AccessDataReader قابلیت اتصال به بانک‌های اطلاعاتی اکسس جدید (فایل‌های accdb) و اکسس قدیم (فایل‌های mdb) را دارد. OdbcDataReader یک پروایدر عمومی است که از روز اول دات نت به همراه آن بوده است. برای مثال جهت اتصال به بانک‌های اطلاعاتی فاکس‌پرو می‌تواند مورد استفاده قرار گیرد.

اما ... برای مابقی بانک‌های اطلاعاتی چطور؟

برای سایر بانک‌های اطلاعاتی، منبع داده عمومی زیر تدارک دیده شده است:

```
public void GenericDataReader(string providerName, string connectionString, string sql, params object[] parametersValues)
```

تنها تفاوت آن با نمونه‌های قبل، ذکر providerName آن است. برای مثال جهت اتصال به SQLite ابتدا پروایدر مخصوص ADO.NET [آن را دریافت](#) و به پروژه خود اضافه نمائید. سپس پارامتر providerName فوق را با "System.Data.SQLite" مقدار دهی کنید.

یک نکته:

در تمام منابع داده فوق، امکان نوشتن کوئری‌های پارامتری نیز پیش بینی شده است. فقط باید دقت داشت که پارامترهای معرفی شده باید با @ شروع شوند که یک نمونه از آن را در مثال جاری ملاحظه خواهید نمود.

در ادامه نحوه تهیه گزارش از یک بانک اطلاعاتی SQLite را توسط PdfReport بررسی خواهیم کرد:

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

namespace PdfReportSamples.SQLiteDataReader
{
    public class SQLiteDataReaderPdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                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(AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
                      Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.ttf");
        })
        .PagesFooter(footer =>
        {
            footer.DefaultFooter(DateTime.Now.ToString("MM/dd/yyyy"));
        })
        .PagesHeader(header =>
        {
            header.DefaultHeader(defaultHeader =>
            {
                defaultHeader.RunDirection(PdfRunDirection.RightToLeft);
                defaultHeader.ImagePath(AppPath.ApplicationPath + "\\Images\\01.png");
                defaultHeader.Message("گزارش جدید ما");
            });
        })
        .MainTableTemplate(template =>
        {
            template.BasicTemplate(BasicTemplate.SilverTemplate);
        })
        .MainTablePreferences(table =>
        {
            table.ColumnsWidthsType(TableColumnWidthType.Relative);
            table.NumberOfDataRowsPerPage(5);
        })
        .MainTableDataSource(dataSource =>
        {
            dataSource.GenericDataReader(
                providerName: "System.Data.SQLite",
                connectionString: "Data Source=" + AppPath.ApplicationPath +
"\\data\\blogs.sqlite",
                sql: @"SELECT [url], [name], [NumberOfPosts], [AddDate]
                        FROM [tblBlogs]
                        WHERE [NumberOfPosts]>=@p1",
                parametersValues: new object[] { 10 }
            );
        })
        .MainTableSummarySettings(summarySettings =>
        {
            summarySettings.OverallSummarySettings("جمع کل");
            summarySettings.PreviousPageSummarySettings("نقل از صفحه قبل");
            summarySettings.PageSummarySettings("جمع صفحه");
        })
        .MainTableColumns(columns =>
        {
            columns.AddColumn(column =>
            {
                column.PropertyName("rowNo");
                column.IsRowNumber(true);
                column.CellsHorizontalAlignment(HorizontalAlignment.Center);
                column.IsVisible(true);
                column.Order(0);
                column.Width(1);
                column.HeaderCell("ردیف");
            });

            columns.AddColumn(column =>
            {
                column.PropertyName("url");
                column.CellsHorizontalAlignment(HorizontalAlignment.Center);
                column.IsVisible(true);
                column.Order(1);
                column.Width(2);
                column.HeaderCell("آدرس");
                column.ColumnItemsTemplate(template =>
                {
                    template.Hyperlink(foreColor: System.Drawing.Color.Blue, fontUnderline: true);
                });
            });

            columns.AddColumn(column =>
            {
                column.PropertyName("name");
                column.CellsHorizontalAlignment(HorizontalAlignment.Center);
                column.IsVisible(true);
                column.Order(2);
                column.Width(2);
                column.HeaderCell("نام");
            });
        });

```

```

columns.AddColumn(column =>
{
    column.PropertyName("NumberOfPosts");
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.IsVisible(true);
    column.Order(3);
    column.Width(2);
    column.HeaderCell("تعداد مطلب");
    column.ColumnItemsTemplate(template =>
    {
        template.TextBlock();
        template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
    });
    column.AggregateFunction(aggregateFunction =>
    {
        aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
        aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
    });
});

columns.AddColumn(column =>
{
    column.PropertyName("AddDate");
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.IsVisible(true);
    column.Order(4);
    column.Width(2);
    column.HeaderCell("تاریخ ثبت");
    column.ColumnItemsTemplate(template =>
    {
        template.TextBlock();
        template.DisplayFormatFormula(obj => obj == null ? string.Empty :
PersianDate.ToPersianDateTime((DateTime)obj) /*((DateTime)obj).ToString("dd/MM/yyyy HH:mm")*/);
    });
});

})
.MainTableEvents(events =>
{
    events.DataSourceIsEmpty(message: "There is no data available to display.");
})
.Export(export =>
{
    export.ToExcel();
})
.Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
"\\Pdf\\RptSqlDataReaderSample.pdf"));
}
}
}

```

توضیحات:

- در مثال فوق نحوه استفاده از یک بانک اطلاعاتی SQLite را ملاحظه می‌کنید. این بانک اطلاعاتی نمونه در پوشه `bin\data` [سورس](#) به روز شده پروژه موجود است.

```

dataSource.GenericDataReader(
    providerName: "System.Data.SQLite",
    connectionString: "Data Source=" + AppPath.ApplicationPath +
"\\data\\blogs.sqlite",
    sql: @"SELECT [url], [name], [NumberOfPosts], [AddDate]
FROM [tblBlogs]
WHERE [NumberOfPosts]>=@p1",
    parametersValues: new object[] { 10 }
);

```

فرض بر این است که فایل‌های `System.Data.SQLite.dll` و `SQLite.Interop.dll` را از سایت [SQLite](#) دریافت کرده و سپس ارجاعی را به اسمبلی `System.Data.SQLite.dll` به پروژه خود افزوده‌اید. در مرحله بعد به کمک `GenericDataReader` می‌توان به این پروایدر دسترسی یافت. همانطور که ملاحظه می‌کنید یک کوئری

پارامتری با مقدار پارامتر مساوی 10 جهت تهیه گزارش، تعریف شده است. همچنین باید دقت داشت که اگر پروژه جاری شما مبتنی بر دات نت 4 است، [نیاز خواهید داشت](#) چند سطر زیر را به فایل config برنامه اضافه نمائید تا با SQLite مشکلی نداشته باشد:

```
<?xml version="1.0"?>
<configuration>
  <startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
  </startup>
</configuration>
```

- مرحله بعد نوبت به معرفی ستون‌های گزارش است. هر ستون، معادل یک فیلد معرفی شده در کوئری SQL ارسال شده به GenericDataReader خواهد بود (کوچکی و بزرگی حروف باید در اینجا رعایت شوند).
 - در حین معرفی ستون AddDate، نحوه نمایش و تبدیل تاریخ دریافتی که با فرمت DateTime است را به تاریخ شمسی ملاحظه می‌کنید. متد PersianDate.ToPersianDateTime در فضای نام PdfRpt.Core.Helper قرار دارد. توسط DisplayFormatFormula، فرصت خواهید داشت مقدار متناظر با سلول در حال رندر را پیش از نمایش، به هر نحو دلخواهی فرمت کنید.
 - در ستون url از قالب نمایشی پیش فرض Hyperlink، برای نمایش اطلاعات فیلد جاری به صورت یک لینک قابل کلیک استفاده شده است.

یک نکته:

ذکر قسمت MainTableColumns و تمام تعاریف مرتبط با آن در PdfReports اختیاری است. به این معنا که می‌توانید قسمت گزارش سازی و تعاریف گزارشات برنامه خود را پویا کنید (شبیه به حالت auto generate columns در گریدهای معروف). کوئری‌های SQL متناظر با گزارشات را در بانک اطلاعاتی ذخیره کنید و به گزارش ساز فوق ارسال نمائید. حاصل یک گزارش جدید است.



گزارش جدید ما

ردیف	آدرس	نام	تعداد مطلب	تاریخ ثبت
۱	www.blog۰۱.com	blog۰۱	۱۰	۱۳۹۱/۰۷/۱۴ ۱۲:۲۰
۲	www.blog۰۲.com	blog۰۲	۱۰	۱۳۹۱/۰۷/۱۴ ۱۲:۲۰
۳	www.blog۰۳.com	blog۰۳	۱۰	۱۳۹۱/۰۷/۱۴ ۱۲:۲۰
۴	www.blog۰۴.com	blog۰۴	۱۰	۱۳۹۱/۰۷/۱۴ ۱۲:۲۰
		جمع صفحه	۴۰	
		جمع کل	۴۰	

همانطور که در نکته انتهای قسمت قبل « [کار با بانک‌های اطلاعاتی مختلف در PdfReport](#) » عنوان شد، ذکر قسمت MainTableColumns و تمام تعاریف مرتبط با آن در PdfReports اختیاری است. برای تهیه یک گزارش توسط PdfReport فقط کافی است تا منبع داده را جهت تولید ستون‌های گزارش مشخص کنید.

این مورد انعطاف پذیری زیادی را به همراه خواهد داشت؛ اما ... پس از مدتی این سؤالات مطرح می‌شوند: آیا می‌شود در این ستون‌های خودکار، فیلدهای DateTime، با تاریخ شمسی نمایش داده شوند؟ آیا امکانپذیر است که ستونهای عددی، جمع پایین صفحه داشته باشند؟ و مواردی از این دست که در مورد نحوه مدیریت این نوع ستون‌های خودکار در ادامه بحث خواهد شد.

ابتدا سورس کامل مثال جاری را در ادامه ملاحظه خواهید کرد. تقریباً همان مثال قسمت قبل است که تعاریف ستون‌های آن حذف شده است:

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

namespace PdfReportSamples.AdHocColumns
{
    public class AdHocColumnsPdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                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(AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
                    Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\verdana.ttf");
            })
            .PagesFooter(footer =>
            {
                footer.DefaultFooter(printDate: DateTime.Now.ToString("MM/dd/yyyy"));
            })
            .PagesHeader(header =>
            {
                header.DefaultHeader(defaultHeader =>
                {
                    defaultHeader.ImagePath(AppPath.ApplicationPath + "\\Images\\01.png");
                    defaultHeader.Message("گزارش جدید ما");
                });
            })
            .MainTableTemplate(template =>
            {
                template.BasicTemplate(BasicTemplate.SilverTemplate);
            })
            .MainTablePreferences(table =>
            {
                table.ColumnsWidthsType(TableColumnWidthType.Relative);
            })
            .MainTableDataSource(dataSource =>
            {
                dataSource.GenericDataReader(
                    providerName: "System.Data.SQLite",
                    connectionString: "Data Source=" + AppPath.ApplicationPath +
                    "\\data\\blogs.sqlite",
                    sql: @"SELECT [url] as 'آدرس', [name] as 'نام', [NumberOfPosts] as 'تعداد مطالب',
[AddDate] as 'تاریخ ارسال'
FROM [tblBlogs]
WHERE [NumberOfPosts]>=@p1",
                    parametersValues: new object[] { 10 }
                );
            });
        }
    }
}
```

```

    });
    .MainTableSummarySettings(summary =>
    {
        summary.OverallSummarySettings("جمع کل");
        summary.PreviousPageSummarySettings("نقل از صفحه قبل");
        summary.PageSummarySettings("جمع صفحه");
    })
    .MainTableAdHocColumnsConventions(adHocColumns =>
    {
        //We want sum of the int columns
        adHocColumns.AddTypeAggregateFunction(
            typeof(Int64),
            new AggregateProvider(AggregateFunction.Sum)
            {
                DisplayFormatFormula = obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj)
            });

        //We want to display all of the dateTimes as ShamsiDateTime
        adHocColumns.AddTypeDisplayFormatFormula(
            typeof(DateTime),
            data => { return PersianDate.ToPersianDateTime((DateTime)data); }
        );
        adHocColumns.ShowRowNumberColumn(true);
        adHocColumns.RowNumberColumnCaption("ردیف");
    })
    .MainTableEvents(events =>
    {
        events.DataSourceIsEmpty(message: "There is no data available to display.");
    })
    .Export(export =>
    {
        export.ToExcel();
        export.ToXml();
    })
    .Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
"\Pdf\AdHocColumnsSampleRpt.pdf"));
    }
}

```

توضیحات:

- با توجه به اینکه تعاریف ستون‌ها را حذف کرده‌ایم و به این ترتیب ستون‌ها به صورت خودکار بر اساس فیلدهای معرفی شده در منبع داده تشکیل می‌شوند، نیاز است سر ستون‌ها را بتوانیم فارسی نمایش دهیم. به همین جهت اینبار کوئری SQL ما با استفاده از alias، نامی فارسی را جهت فیلدها تدارک دیده است:

```

SELECT [url] as 'آدرس', [name] as 'نام', [NumberOfPosts] as 'تعداد مطالب', [AddDate] as 'تاریخ ارسال'
FROM [tblBlogs]
WHERE [NumberOfPosts]>=@p1

```

- در مرحله بعد توسط متد `MainTableAdHocColumnsConventions`، یک سری روال را جهت پردازش و نمایش این ستون‌های پویا مشخص می‌کنیم. برای مثال علاقمندیم در این نوع گزارشات هم ستون خودکار ردیف ظاهر شود:

```

adHocColumns.ShowRowNumberColumn(true);
adHocColumns.RowNumberColumnCaption("ردیف");

```

همچنین هر ستونی که نوع داده‌اش `DateTime` بود، از طریق فرمولی که مشخص می‌کنیم، به صورت شمسی نمایش داده شود:

```

adHocColumns.AddTypeDisplayFormatFormula(
    typeof(DateTime),
    data => { return PersianDate.ToPersianDateTime((DateTime)data); }
);

```

به علاوه می‌خواهیم تمام ستون‌هایی از نوع `Int64`، دارای جمع پایین صفحه هم باشند:

```
adHocColumns.AddTypeAggregateFunction(  
    typeof(Int64),  
    new AggregateProvider(AggregateFunction.Sum)  
    {  
        DisplayFormatFormula = obj => obj == null ? string.Empty :  
string.Format("{0:n0}", obj)  
    });
```

نوع int در بانک اطلاعاتی SQLite معادل نوع Int64 در دات نت است.

نظرات خوانندگان

نویسنده: Petek

تاریخ: ۲۱:۱۸ ۱۳۹۱/۰۷/۱۵

با سلام

مهندس اگه یکمی دیگه ادامه بدید دیگه خود من از استفاده از این گزارش سازها مثل StimulReport و ... بی خیال می‌شم و مثل همیشه کارهاتون عالیه . با تشکر

نویسنده: محمد رضا

تاریخ: ۰:۲۲ ۱۳۹۱/۰۷/۲۹

سلام آقای نصیری ، با تشکر از این pdfReport بسیار عالی ، می‌خواستم یک راهنمایی در باره چگونگی استفاده از یک بانک اطلاعاتی sqlServer در این سیستم گزارش گیری بفرمایید
با تشکر و سپاس فراوان .

نویسنده: وحید نصیری

تاریخ: ۰:۲۹ ۱۳۹۱/۰۷/۲۹

مراجعه کنید به مطلب « [کار با بانک‌های اطلاعاتی مختلف در PdfReport](#) »
تمام مباحث و مفاهیم آن یکی است. فقط قسمت منبع داده آن نیاز است تغییر کند:

```
.MainTableDataSource(dataSource =>
{
    dataSource.SqlDataReader(...
```

نویسنده: محمد رضا

تاریخ: ۱۶:۴۶ ۱۳۹۱/۰۷/۲۹

سلام آقای نصیری

من تنظیمات اتصال بانک اطلاعاتی را انجام داده ام . اما باز با مشکل مواجه هستم .
کدهای خودم رو خدمت شما ارسال میکنم . منو راهنمایی بفرمایید.
با تشکر

```
..MainTableDataSource(dataSource =>
{
    dataSource.SqlDataReader(
        providerName: "System.Data.SqlClient",
        connectionString: "Data Source=.\SQLEXPRESS;Initial Catalog=EFDB;Integrated
Security=True",
        sql: @"SELECT [id] as 'شماره', [Title] as 'نام', [Content] as 'تعداد مطالب', [Date]
as 'تاریخ ارسال'
FROM [posts]
WHERE [NumberOfPosts]>=@p1",
        parametersValues: new object[] { 10 }
    );
});
```

نویسنده: وحید نصیری

تاریخ: ۱۶:۵۲ ۱۳۹۱/۰۷/۲۹

- لطفا جهت پرسش و پاسخ ویژه این پروژه، از قسمت مرتبط با آن در سایت استفاده کنید: ([^](#))

- امضای متد ویژه SqlDataReader به صورت زیر است:

```
public void SqlDataReader(string connectionString, string sql, params object[] parametersValues)
```

همانطور که مشاهده می‌کنید نیازی به ذکر providerName ندارد.
- در دفعات بعد (چه در این سایت یا هر جای دیگری)، قسمت «با مشکل مواجه هستم» را بیشتر توضیح دهید. چون در این حالت کلی، اشخاص باید حدس بزنند که مشکل یاد شده چه چیزی بوده.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۸:۳۵ ۱۳۹۱/۱۰/۲۲

آقای نصیری با سلام.
آیا امکان دارد برخی از ستون‌ها را استاتیک و برخی دیگر را adhoc تعریف کنیم البته به غیر از rowNum ؟
با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۹:۳۹ ۱۳۹۱/۱۰/۲۲

سلام؛ ستون‌های استاتیک را هم در دیتاسورس قرار بدید و در حالت adhoc استفاده کنید.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۹:۵۳ ۱۳۹۱/۱۰/۲۲

سلام .
منبع داده من Strongly type می‌باشد . چگونه می‌توانم برای آنها alias name بگذارم. با تشکر.

نویسنده: وحید نصیری
تاریخ: ۲۰:۴۳ ۱۳۹۱/۱۰/۲۲

در این حالت می‌تونید از [data annotations](#) استفاده کنید. به صورت خودکار اعمال خواهند شد.

نویسنده: ایلیا اکبری فرد
تاریخ: ۲۳:۲۲ ۱۳۹۱/۱۰/۲۲

سپاس مهندس. عالی .

عموماً برای نمایش تصاویر در گزارشات، یکی از دو حالت زیر وجود دارد:

الف) مسیر تصاویر موجود در فایل سیستم، در بانک اطلاعاتی ذخیره شده‌اند و قرار است گزارش نهایی در ستونی مشخص شامل این تصاویر باشد.

ب) محتوای بایناری تصاویر در خود بانک اطلاعاتی ذخیره شده‌اند و نیاز است آن‌ها را در گزارشات نمایش دهیم.

هر دو حالت فوق در [PdfReport](#) پشتیبانی می‌شوند و در ادامه نحوه انجام این موارد را بررسی خواهیم کرد.

الف) بارگذاری تصاویر از فایل سیستم

ابتدا مدل زیر را در نظر بگیرید:

```
namespace PdfReportSamples.Models
{
    public class ImageRecord
    {
        public int Id { set; get; }
        public string ImagePath { set; get; }
        public string Name { set; get; }
    }
}
```

توسط آن تعدادی رکورد را که ImagePath آن‌ها فایل‌هایی بر روی سیستم هستند، خواهیم ساخت:

```
using System;
using System.Collections.Generic;
using iTextSharp.text.pdf;
using PdfReportSamples.Models;
using PdfRpt.Core.Contracts;
using PdfRpt.FluentInterface;

namespace PdfReportSamples.ImageFilePath
{
    public class ImageFilePathPdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                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(AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.ttf");
            })
            .PagesFooter(header =>
            {
                footer.DefaultFooter(DateTime.Now.ToString("MM/dd/yyyy"));
            })
            .PagesHeader(header =>
            {
                header.DefaultHeader(defaultHeader =>
                {
                    defaultHeader.ImagePath(AppPath.ApplicationPath + "\\Images\\01.png");
                    defaultHeader.Message("گزارش جدید ما");
                })
            })
            .MainTableTemplate(template =>
```

```

    {
        template.BasicTemplate(BasicTemplate.SnowyPineTemplate);
    })
    .MainTablePreferences(table =>
    {
        table.ColumnsWidthsType(TableColumnWidthType.Relative);
    })
    .MainTableDataSource(dataSource =>
    {
        var listOfRows = new List<ImageRecord>
        {
            new ImageRecord
            {
                Id=1,
                ImagePath = AppPath.ApplicationPath +
                "\\Images\\01.png",
                Name = "Rnd"
            },
            new ImageRecord
            {
                Id=2,
                ImagePath = AppPath.ApplicationPath +
                "\\Images\\02.png",
                Name = "Bug"
            },
            new ImageRecord
            {
                Id=3,
                ImagePath = AppPath.ApplicationPath +
                "\\Images\\03.png",
                Name = "Stuff"
            },
            new ImageRecord
            {
                Id=4,
                ImagePath = AppPath.ApplicationPath +
                "\\Images\\04.png",
                Name = "Sun"
            }
        };
        dataSource.StronglyTypedList(listOfRows);
    })
    .MainTableColumns(columns =>
    {
        columns.AddColumn(column =>
        {
            column.PropertyName("rowNo");
            column.IsRowNumber(true);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(0);
            column.Width(1);
            column.HeaderCell("ردیف");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<ImageRecord>(x => x.Id);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(1);
            column.Width(3);
            column.HeaderCell("شماره");
            column.ColumnItemsTemplate(t => t.Barcode(new Barcode39()));
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<ImageRecord>(x => x.ImagePath);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(2);
            column.Width(3);
            column.HeaderCell("تصویر");
            column.ColumnItemsTemplate(t => t.ImageFilePath(defaultImagePath: string.Empty,
fitImages: false));
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<ImageRecord>(x => x.Name);

```

```

        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(3);
        column.Width(2);
        column.HeaderCell("نام");
    });
}
.MainTableEvents(events =>
{
    events.DataSourceIsEmpty(message: "There is no data available to display.");
})
.Export(export =>
{
    export.ToExcel();
    export.ToXml();
})
.Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
"\\Pdf\\RptImagePathSample.pdf"));
}
}
}

```

توضیحات:

- در متد `MainTableDataSource`، یک سری رکورد دلخواه را بر اساس تعدادی تصویر مشخص ایجاد کرده‌ایم. این تصاویر در پوشه `bin\images` [مثال‌های PdfReport](#) قرار دارند. سپس آن‌ها را توسط متد `dataSource.StronglyTypedList`، در اختیار [PdfReport](#) قرار داده‌ایم.

- در مرحله بعد، توسط متد `MainTableColumns`، ستون‌های دلخواه گزارش را ایجاد کرده‌ایم. دو نکته در اینجا مهم هستند:

الف) برای نمایش یک تصویر از فایل سیستم، فقط کافی است از `ColumnItemsTemplate` متناظر با آن استفاده کرد (حالت پیش فرض، نمایش متنی اطلاعات است). برای نمونه قالب پیش فرض `ImagePath` به نحو زیر قابل استفاده است:

```
column.ColumnItemsTemplate(t => t.ImageFilePath(defaultImagePath: string.Empty, fitImages: false));
```

در اینجا در آرگومان اول آن می‌توان مسیر تصویری را مشخص کرد که در صورت موجود نبودن تصویر مشخص شده در منبع داده، بهتر است نمایش داده شود. اگر `string.Empty` وارد شد، از این مورد صرف‌نظر خواهد شد. آرگومان دوم آن مشخص می‌کند که آیا تصویر نمایش داده شده باید با ابعاد سلول متناظر با آن هماهنگ شده و نمایش داده شود یا خیر. ب) در کتابخانه `iTextSharp` که پایه [PdfReport](#) است، امکان تهیه بارکد هم وجود دارد. برای مثال این بارکدها توسط قالب زیر، قابل استفاده خواهند شد:

```
column.ColumnItemsTemplate(t => t.Barcode(new Barcode39()));
```

تصویری از حاصل این گزارش را در ذیل مشاهده می‌کنید:

<div style="text-align: center;">  <p>گزارش جدید ما</p> </div>			
ردیف	شماره	تصویر	نام
۱			Rnd
۲			Bug
۳			Stuff
۴			Sun

(ب) بارگذاری محتوای تصاویر از بانک اطلاعاتی

ابتدا کدهای کامل این مثال را در نظر بگیرید:

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

namespace PdfReportSamples.DbImage
{
    public class DbImagePdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                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(AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
                    Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\verdana.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("گزارش جدید ما");
        });
    })
    .MainTableTemplate(template =>
    {
        template.BasicTemplate(BasicTemplate.AutumnTemplate);
    })
    .MainTablePreferences(table =>
    {
        table.ColumnsWidthsType(TableColumnWidthType.FitToContent);
    })
    .MainTableDataSource(dataSource =>
    {
        dataSource.GenericDataReader(
            providerName: "System.Data.SQLite",
            connectionString: "Data Source=" + AppPath.ApplicationPath + "\\data\\blogs.sqlite",
            sql: @"SELECT [url], [name], [NumberOfPosts], [AddDate], [thumbnail]
                FROM [tblBlogs]
                WHERE [NumberOfPosts]>=@p1",
            parametersValues: new object[] { 10 }
        );
    })
    .MainTableEvents(events =>
    {
        events.DataSourceIsEmpty(message: "There is no data available to display.");
    })
    .MainTableSummarySettings(summary =>
    {
        summary.OverallSummarySettings("جمع کل");
        summary.PreviousPageSummarySettings("نقل از صفحه قبل");
    })
    .MainTableColumns(columns =>
    {
        columns.AddColumn(column =>
        {
            column.PropertyName("rowNo");
            column.IsRowNumber(true);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(0);
            column.HeaderCell("ردیف");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName("url");
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(1);
            column.HeaderCell("آدرس");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName("name");
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(2);
            column.HeaderCell("نام");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName("NumberOfPosts");
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(3);
            column.HeaderCell("تعداد پست");
            column.ColumnItemsTemplate(template =>

```

```

        {
            template.TextBlock();
            template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
        column.AggregateFunction(aggregateFunction =>
        {
            aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
            aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("AddDate");
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(4);
        column.HeaderCell("تاریخ ثبت");
        column.ColumnItemsTemplate(template =>
        {
            template.TextBlock();
            template.DisplayFormatFormula(obj => obj == null ? string.Empty :
((DateTime)obj).ToString("dd/MM/yyyy HH:mm"));
        });
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("thumbnail");
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(5);
        column.HeaderCell("تصویر");
        column.ColumnItemsTemplate(t => t.ByteArrayImage(defaultImagePath:
string.Empty, fitImages: false));
    });
    })
    .Export(export =>
    {
        export.ToXml();
        export.ToExcel();
    })
    .Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
"\\Pdf\\RptDbImageSample.pdf"));
}
}
}

```

توضیحات:

بانک اطلاعاتی SQLite موجود در پوشه bin\data [سورس PdfReport](#)، حاوی فیلدی است به نام thumbnail که در آن محتوای یک سری تصویر ذخیره شده‌اند.

در اینجا توسط کوئری زیر قصد داریم این اطلاعات را خوانده و نمایش دهیم:

```

SELECT [url], [name], [NumberOfPosts], [AddDate], [thumbnail]
FROM [tblBlogs]
WHERE [NumberOfPosts]>=@p1

```

همانطور که ملاحظه می‌کنید، در اینجا نیز تنها کافی است قالب ستون مرتبطی را انتخاب کنیم:

```

column.ColumnItemsTemplate(t => t.ByteArrayImage(defaultImagePath: string.Empty, fitImages:
false));

```

قالب ByteArrayImage، اطلاعات باینری فیلد thumbnail را خوانده و تبدیل به تصاویر قرار داده شده در فایل گزارش نهایی می‌کند. پارامترهای آن، با پارامترهای قالب ImageFilePath که پیشتر توضیح داده شد، یکی هستند.

نظرات خوانندگان

نویسنده: hosseinzadeh
تاریخ: ۱۵:۴۷ ۱۳۹۱/۰۷/۲۲

سپاس

برای قراردادن رشته مرتبط با بارکد زیر عکس تولید شده راهی هست؟

یا اینکه میشه یک ستون از دیتابیس رو به عنوان 2 تا column برای ریپورت تعریف کرد (که در حالت تست من نپذیرفت). یکی مقدار رشته ای و دیگری عکس بارکد رو نمایش بده؟

نویسنده: وحید نصیری
تاریخ: ۱۷:۰۹ ۱۳۹۱/۰۷/۲۲

- می‌تونید یک [قالب ستون سفارشی](#) تعریف کنید. چیزی شبیه به همین مثال است. در یک سلول هم تصویر نمایش داده شده و هم یک رشته.

- برای استفاده از مقدار یک فیلد در دو ستون می‌تونید از [فیلدهای محاسباتی](#) استفاده کنید.

نویسنده: hosseinzadeh
تاریخ: ۲۱:۴۳ ۱۳۹۱/۰۷/۲۲

عالیست، سپاس

گزارشی را در نظر بگیرید با این نیازها:

می‌خواهیم

الف) یک Watermark قطری را بر روی تمام صفحات گزارش ظاهر کنیم.

ب) عددهای درصد پیشرفت یک ستون را به صورت میله‌ای نمایش دهیم.

ج) در هر صفحه بجای اینکه یک جدول، اطلاعات را نمایش دهد و تمام صفحه را پر کند، دو جدول در دو ستون کنار هم اینکار را انجام دهند تا در حین چاپ گزارش، در میزان تعداد صفحات مصرفی صرفه جویی صورت گیرد.

د) مقادیر true با چک مارک و موارد false با علامت ضربدر نمایش داده شوند.

یک چنین شکلی در نهایت مد نظر است:

ردیف	شماره فعالیت	فعالیت	درصد پیشرفت	در جریان
۴۸	Task۴۷	۸۱۵۵	۲۳ %	✓
۴۹	Task۴۸	۱۷۲۵	۳۴ %	✗
۵۰	Task۴۹	۸۲۳۷	۳۱ %	✗
۵۱	Task۵۰	۴۹۷۸	۵۸ %	✗
۵۲	Task۵۱	۳۱۸۱	۳۰ %	✓
۵۳	Task۵۲	۶۰۶۶	۵۷ %	✗
۵۴	Task۵۳	۳۷۵۴	۲۵ %	✗
۵۵	Task۵۴	۵۷۵۶	۹۳ %	✓
۵۶	Task۵۵	۶۳۶۳	۷۷ %	✗
۵۷	Task۵۶	۴۴۶۲	۸۷ %	✗
۵۸	Task۵۷	۹۴۲۸	۹۵ %	✗
۵۹	Task۵۸	۲۸۳۰	۱۵ %	✓
۶۰	Task۵۹	۲۸۸۴	۴۴ %	✗
۶۱	Task۶۰	۶۴۵۸	۲۵ %	✓
۶۲	Task۶۱	۱۵۹۷	۵۹ %	✗
۶۳	Task۶۲	۹۲۱۰	۴۸ %	✗
۶۴	Task۶۳	۵۲۵۹	۵۰ %	✗
۶۵	Task۶۴	۶۸۴۲	۲۸ %	✗
۶۶	Task۶۵	۹۹۳۵	۲۷ %	✗
۶۷	Task۶۶	۲۵۹۰	۴۸ %	✓
۶۸	Task۶۷	۲۵۹۳	۴۲ %	✗
۶۹	Task۶۸	۳۲۵۷	۵۱ %	✓
۷۰	Task۶۹	۲۴۶۳	۴ %	✗
۷۱	Task۷۰	۳۷۸	۳۵ %	✗
۷۲	Task۷۱	۲۱۰۷	۷۰ %	✗
۷۳	Task۷۲	۴۰۸۸	۴۶ %	✗
۷۴	Task۷۳	۴۸۵۸	۶۸ %	✗
۷۵	Task۷۴	۹۵۲۳	۹۱ %	✗
۷۶	Task۷۵	۷۶۵۴	۲۹ %	✓
۷۷	Task۷۶	۹۱۷۶	۲۱ %	✓
۷۸	Task۷۷	۴۲۱۰	۴ %	✓
۷۹	Task۷۸	۶۸۶۸	۴۳ %	✓
۸۰	Task۷۹	۴۷۰۰	۸۱ %	✗
۸۱	Task۸۰	۷۳۷۴	۷۵ %	✗
۸۲	Task۸۱	۱۳۶۷	۱۷ %	✗
۸۳	Task۸۲	۲۸۸۵	۴۵ %	✓
۸۴	Task۸۳	۴۸۲۵	۶۰ %	✓
۸۵	Task۸۴	۹۵۸۷	۷۵ %	✓
۸۶	Task۸۵	۳۴۳۵	۲۳ %	✓
۸۷	Task۸۶	۱۵۲۵	۵۱ %	✗
۸۸	Task۸۷	۹۶۷۵	۲۶ %	✓
۸۹	Task۸۸	۳۳۷۹	۹۵ %	✗
۹۰	Task۸۹	۲۶۱۸	۵۵ %	✗
۹۱	Task۹۰	۲۵۶۳	۹۷ %	✓
۹۲	Task۹۱	۳۳۲۷	۷۲ %	✓
۹۳	Task۹۲	۸۷۰۲	۹۰ %	✓
۹۴	Task۹۳	۲۹۷۵	۵۳ %	✓

ردیف	شماره فعالیت	فعالیت	درصد پیشرفت	در جریان
۱	Task۰	۴۱۱۵	۹۴ %	✗
۲	Task۱	۳۸۷۳	۳ %	✓
۳	Task۲	۴۸۶۹	۲۶ %	✓
۴	Task۳	۴۳۷۳	۷ %	✓
۵	Task۴	۲۴۵۳	۶۲ %	✗
۶	Task۵	۲۵۳۹	۴۱ %	✗
۷	Task۶	۱۷۹۲	۹۰ %	✓
۸	Task۷	۴۷۷۷	۷ %	✓
۹	Task۸	۲۹۲۱	۳۰ %	✓
۱۰	Task۹	۴۵۰۱	۹ %	✓
۱۱	Task۱۰	۴۲۱۹	۲۹ %	✓
۱۲	Task۱۱	۴۳۳۰	۲ %	✓
۱۳	Task۱۲	۳۷۷۴	۷۳ %	✓
۱۴	Task۱۳	۴۱۵۶	۴۹ %	✗
۱۵	Task۱۴	۵۴۹۶	۶۸ %	✓
۱۶	Task۱۵	۹۳۶۱	۵۱ %	✓
۱۷	Task۱۶	۴۱۶۲	۹ %	✓
۱۸	Task۱۷	۷۸۸۱	۶۷ %	✗
۱۹	Task۱۸	۷۶۰۱	۹۲ %	✓
۲۰	Task۱۹	۴۲۹۴	۱۱ %	✓
۲۱	Task۲۰	۱۱۲۹	۵ %	✗
۲۲	Task۲۱	۹۹۰۷	۲ %	✓
۲۳	Task۲۲	۱۷۲۹	۷۹ %	✓
۲۴	Task۲۳	۴۴۲۵	۸۶ %	✓
۲۵	Task۲۴	۴۵۲۸	۷۳ %	✗
۲۶	Task۲۵	۱۳۸۳	۹۲ %	✗
۲۷	Task۲۶	۷۶۴۲	۲۹ %	✓
۲۸	Task۲۷	۱۱۱۲	۴ %	✓
۲۹	Task۲۸	۹۸۰۰	۵۹ %	✓
۳۰	Task۲۹	۲۷۶۶	۶۴ %	✗
۳۱	Task۳۰	۸۴۴۶	۹۴ %	✓
۳۲	Task۳۱	۳۲۲۶	۳۴ %	✗
۳۳	Task۳۲	۲۰۸۴	۵۱ %	✓
۳۴	Task۳۳	۳۲۲۵	۸۴ %	✓
۳۵	Task۳۴	۱۸۴۷	۹۷ %	✗
۳۶	Task۳۵	۳۷۱۱	۳۷ %	✓
۳۷	Task۳۶	۹۶۸۹	۶۹ %	✗
۳۸	Task۳۷	۵۴۲۰	۶۳ %	✓
۳۹	Task۳۸	۹۴۸۶	۱ %	✗
۴۰	Task۳۹	۵۸۴۶	۲ %	✗
۴۱	Task۴۰	۹۵۱۸	۸۸ %	✗
۴۲	Task۴۱	۵۶۸۰	۵۵ %	✓
۴۳	Task۴۲	۱۴۶۵	۱۵ %	✗
۴۴	Task۴۳	۱۲۸۹	۹۲ %	✗
۴۵	Task۴۴	۷۴۷۶	۴۹ %	✗
۴۶	Task۴۵	۱۶۰۸	۹۱ %	✗
۴۷	Task۴۶	۶۲۹۴	۶۷ %	✗

ابتدا کلاس مدل زیر را در نظر بگیرید:

```
namespace PdfReportSamples.Models
{
    public class Task
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public int PercentCompleted { set; get; }
        public bool IsActive { set; get; }
    }
}
```

به این ترتیب یک کلاس فعالیت تعریف شده است که در آن نام فعالیت، درصد پیشرفت و همچنین درجریان بودن آن قابل تنظیم است. از این کلاس جهت تهیه منبع داده گزارش استفاده می‌شود:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using iTextSharp.text;
using PdfReportSamples.Models;
using PdfRpt;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;
using PdfRpt.FluentInterface;

namespace PdfReportSamples.ProgressReport
{
    public class ProgressReportPdfReport
    {
        private IPdfFont getWatermarkFont()
        {
            var watermarkFont = new GenericFontProvider(
                AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
                Environment.GetEnvironmentVariable("SystemRoot") +
                "\\fonts\\verdana.ttf");
            watermarkFont.Color = BaseColor.LIGHT_GRAY;
            watermarkFont.Size = 50;
            return watermarkFont;
        }

        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                doc.Orientation(PageOrientation.Portrait);
                doc.PageSize(PdfPageSize.A4);
                doc.DocumentMetadata(new DocumentMetadata { Author = "Vahid", Application = "PdfRpt",
Keywords = "Test", Subject = "Test Rpt", Title = "Test" });
                doc.DiagonalWatermark(new DiagonalWatermark
                {
                    Text = "نمایش درصد پیشرفت",
                    RunDirection = PdfRunDirection.RightToLeft,
                    Font = getWatermarkFont()
                });
            });
            .DefaultFonts(fonts =>
            {
                fonts.Path(AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
                Environment.GetEnvironmentVariable("SystemRoot") +
                "\\fonts\\verdana.ttf");
            })
            .PagesFooter(header =>
            {
                footer.DefaultFooter(PersianDate.ToPersianDateTime(DateTime.Now, "/", true));
            })
            .PagesHeader(header =>
            {
                header.DefaultHeader(defaultHeader =>
                {
                    defaultHeader.Message("گزارش جدید ما");
                    defaultHeader.ImagePath(AppPath.ApplicationPath + "\\Images\\01.png");
                });
            })
            .MainTableTemplate(template =>
            {
```

```

        template.BasicTemplate(BasicTemplate.SilverTemplate);
    })
    .MainTablePreferences(table =>
    {
        table.ColumnsWidthsType(TableColumnWidthType.Relative);
        table.MultipleColumnsPerPage(new MultipleColumnsPerPage
        {
            ColumnsGap = 20,
            ColumnsPerPage = 2,
            ColumnsWidth = 250,
            IsRightToLeft = true,
            TopMargin = 7
        });
    })
    .MainTableDataSource(dataSource =>
    {
        var listOfRows = new List<Task>();
        var rnd = new Random();
        for (int i = 0; i < 400; i++)
        {
            listOfRows.Add(new Task
            {
                Id = rnd.Next(1000, 10000),
                Name = "Task" + i,
                PercentCompleted = rnd.Next(1, 100),
                IsActive = rnd.Next(0, 2) == 1 ? true : false
            });
        }
        dataSource.StronglyTypedList<Task>(listOfRows);
    })
    .MainTableColumns(columns =>
    {
        columns.AddColumn(column =>
        {
            column.PropertyName("rowNo");
            column.IsRowNumber(true);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(0);
            column.Width(1);
            column.HeaderCell("ردیف", captionRotation: 90);
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<Task>(x => x.Id);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(1);
            column.Width(2);
            column.HeaderCell("شماره فعالیت");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<Task>(x => x.Name);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(2);
            column.Width(3);
            column.HeaderCell("فعالیت");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<Task>(x => x.PercentCompleted);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(3);
            column.Width(3);
            column.HeaderCell("درصد پیشرفت");
            column.ColumnItemsTemplate(template =>
            {
                template.ProgressBar(progressBarColor: Color.SkyBlue, showPercentText: true);
                template.DisplayFormatFormula(obj =>
                {
                    if (obj == null) return "% 0";
                    return "% " + obj.ToString();
                });
            });
        });
    });
}

```

```

        columns.AddColumn(column =>
        {
            column.PropertyName<Task>(x => x.IsActive);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(4);
            column.Width(2);
            column.HeaderCell("در جریان");
            column.ColumnItemsTemplate(template =>
            {
                template.Checkmark(checkmarkFillColor: Color.Green, crossSignFillColor:
Color.DarkRed);
            });
        });
    });
    .MainTableEvents(events =>
    {
        events.DataSourceIsEmpty(message: "There is no data available to display.");
    })
    .Export(export =>
    {
        export.ToExcel();
    })
    .Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
"\\Pdf\\ProgressReportSample.pdf"));
}
}
}

```

توضیحات:

- همانطور که در کدهای فوق ملاحظه می‌کنید، برای تعریف یک watermark قطری در سراسر سند تولیدی، نیاز است در متد DocumentPreferences، تنظیمات DiagonalWatermark را مشخص کرد:

```

doc.DiagonalWatermark(new DiagonalWatermark
{
    Text = "نمایش درصد پیشرفت",
    RunDirection = PdfRunDirection.RightToLeft,
    Font = getWatermarkFont()
});

```

در اینجا Text، متنی است که نمایش داده خواهد شد. تنظیم PdfRunDirection.RightToLeft برای نمایش صحیح متون فارسی الزامی است. همچنین این watermark نیاز به قلم مناسب و متفاوتی نسبت به قلم‌های پیش فرض گزارش نیز دارد:

```

private IPdfFont getWatermarkFont()
{
    var watermarkFont = new GenericFontProvider(
        AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
        Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.ttf");
    watermarkFont.Color = BaseColor.LIGHT_GRAY;
    watermarkFont.Size = 50;
    return watermarkFont;
}

```

قلم‌هایی از جنس IPdfFont را توسط کلاس توکار GenericFontProvider به نحوی که ملاحظه می‌کنید می‌توان ایجاد کرد.

- برای ستون بندی گزارش باید به متد MainTablePreferences رجوع نمود. در اینجا می‌توان تنظیمات دقیق ستون‌های گزارش را مشخص کرد:

```

table.MultipleColumnsPerPage(new MultipleColumnsPerPage
{
    ColumnsGap = 20,
    ColumnsPerPage = 2,
    ColumnsWidth = 250,
    IsRightToLeft = true,

```

```
TopMargin = 7
});
```

برای مثال در اینجا 2 ستون در هر صفحه تعریف شده است (ColumnsPerPage). فاصله بین این ستون‌ها 20 است (ColumnsGap). عرض هر ستون 250 در نظر گرفته شده (ColumnsWidth) و همچنین توسط تنظیم IsRightToLeft، سبب خواهیم شد تا جداول از راست به چپ شروع و در صفحه نمایش داده شوند. (اگر به شماره ردیف‌ها در شکل ابتدای بحث دقت کنید، ردیف 1 در سمت راست صفحه قرار دارد).

- برای نمایش درصد پیشرفت در یک سلول خاص تنها کافی است قالب مخصوص آن را انتخاب و مقدار دهی کنیم:

```
column.ColumnItemsTemplate(template =>
{
    template.ProgressBar(progressBarColor: Color.SkyBlue, showPercentText: true);
    template.DisplayFormatFormula(obj =>
    {
        if (obj == null) return "% 0";
        return "% " + obj.ToString();
    });
});
```

قالب از پیش تعریف شده ProgressBar، مقدار سلول جاری را دریافت و آن را تبدیل به یک میله افقی درصد پیشرفت می‌کند. همچنین در اینجا توسط DisplayFormatFormula، یک علامت درصد هم به متنی که قرار است نمایش داده شود، اضافه کرده‌ایم.

- نمایش چک مارک و علامت ضربدر نیز به همین منوال است. باید قالب مناسبی را برای آن انتخاب و اعمال کرد:

```
column.ColumnItemsTemplate(template =>
{
    template.Checkmark(checkmarkFillColor: Color.Green, crossSignFillColor:
Color.DarkRed);
});
```

قالب Checkmark نیز جزو قالب‌های از پیش تعریف شده PdfReport است و بر اساس گرافیک برداری کار می‌کند.

نظرات خوانندگان

نویسنده:

آرش

تاریخ:

۱۳۹۱/۰۷/۱۶ ۹:۵۰

نمودارهای ستونی و دایره ای هم می‌توان با این ابزار نشان داد؟

نویسنده:

وحید نصیری

تاریخ:

۱۳۹۱/۰۷/۱۶ ۱۰:۲۱

یک مثال در مورد نحوه قرار دادن نمودارهای ms chart (که جزئی از دات نت است) در گزارشات، در قسمت‌های بعد خواهیم داشت.

نویسنده:

وحید نصیری

تاریخ:

۱۳۹۱/۰۷/۱۶ ۱۲:۳۵

ضمناً لطف کنید، در این سایت مطالب خارج از عنوان خاص بحث جاری را مطرح نکنید، چون حذف خواهد شد. برای مدیریت بهتر این مسایل عمومی مرتبط با PdfReport می‌تونید [از اینجا](#) استفاده کنید.

نویسنده:

رحمت اله رضایی

تاریخ:

۱۳۹۲/۱۱/۱۸ ۱۹:۴۹

آیا راهی برای نمایش watermark چند خطی (multiline) هم وجود دارد؟

نویسنده:

وحید نصیری

تاریخ:

۱۳۹۲/۱۱/۱۸ ۲۰:۲۸

خیر. در iTextSharp متد ColumnText.ShowTextAligned فقط از متن یک سطر از زاویه دار فارسی پشتیبانی می‌کند.

صورت مساله:

- می‌خواهیم footer پیش فرض PdfReport را که تاریخ را در یک سمت، و شماره صفحه را در سمتی دیگر نمایش می‌دهد، به عبارت «صفحه x از n» تغییر دهیم.

- می‌خواهیم در Header گزارش بجای Header پیش فرض PdfReport یکی از قالب‌های PDF تهیه شده توسط [Open Office](#) را نمایش دهیم (و یا هر ساختار دیگری را).

تمام اجزای PdfReport جهت امکان اعمال تغییرات کلی و توسعه آن‌ها طراحی شده‌اند؛ قالب‌ها، هدر، فوتر، منابع داده، قالب‌های نمایش سلول‌ها، تعریف توابع تجمعی سفرشی و غیره. جهت سهولت کار، به ازای هر یک از این موارد، پیاده سازی‌های پیش فرضی در PdfReport قرار دارند، امکان اگر مورد رضایت شما نیستند ... از بنیان تغییرشان دهید! (و همچنین اگر مورد جالبی را پیاده سازی کردید، می‌توانید به عنوان یک وصله جدید ارائه دهید تا به پروژه اضافه شود)

ضمناً این مطالب سفرشی سازی نیاز به آشنایی با ساختار iTextSharp را نیز دارند؛ در حد ایجاد یک جدول ساده باید با iTextSharp [آشنا باشید](#).

مدل‌های مورد استفاده:

```
namespace PdfReportSamples.Models
{
    public class Task
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public int PercentCompleted { set; get; }
        public bool IsActive { set; get; }
        public User Assignee { set; get; }
    }
}
```

```
using System;

namespace PdfReportSamples.Models
{
    public class User
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public string LastName { set; get; }
        public long Balance { set; get; }
        public DateTime RegisterDate { set; get; }
    }
}
```

توسط این مدل‌ها قصد داریم تعدادی فعالیت (Task) را که به تعدادی کاربر انتساب یافته است، نمایش دهیم. همچنین نمایش مقادیر خواص تو در تو نیز در اینجا مد نظر است؛ برای مثال ستونی مانند این:

```
column.PropertyName<Task>(x => x.Assignee.Name)
```

کدهای کامل مثال را در ادامه ملاحظه خواهید نمود:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using PdfReportSamples.Models;
using PdfRpt.Core.Contracts;
```



```

using PdfRpt.FluentInterface;

namespace PdfReportSamples.CustomHeaderFooter
{
    public class CustomHeaderFooterPdfReport
    {
        readonly CustomHeader _customHeader = new CustomHeader();
        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\\tahoma.ttf",
Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.ttf");
            })
            .PagesFooter(header =>
            {
                footer.CustomFooter(new CustomFooter(footer.PdfFont, PdfRunDirection.LeftToRight));
            })
            .PagesHeader(header =>
            {
                header.CustomHeader(_customHeader);
            })
            .MainTableTemplate(template =>
            {
                template.BasicTemplate(BasicTemplate.SilverTemplate);
            })
            .MainTablePreferences(table =>
            {
                table.ColumnsWidthsType(TableColumnWidthType.Relative);
                table.MultipleColumnsPerPage(new MultipleColumnsPerPage
                {
                    ColumnsGap = 22,
                    ColumnsPerPage = 2,
                    ColumnsWidth = 250,
                    IsRightToLeft = false,
                    TopMargin = 7
                });
            })
            .MainTableDataSource(dataSource =>
            {
                var rows = new List<Task>();
                var rnd = new Random();
                for (int i = 1; i < 210; i++)
                {
                    rows.Add(new Task
                    {
                        Assignee = new User
                        {
                            Id = i,
                            Name = "user-" + i
                        },
                        IsActive = rnd.Next(0, 2) == 1 ? true : false,
                        Name = "task-" + i
                    });
                }
                dataSource.StronglyTypedList(rows);
            })
            .MainTableColumns(columns =>
            {
                columns.AddColumn(column =>
                {
                    column.PropertyName("rowNo");
                    column.IsRowNumber(true);
                    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
                    column.IsVisible(true);
                    column.Order(0);
                    column.Width(1);
                    column.HeaderCell("#");
                });

                columns.AddColumn(column =>
                {

```

```

        column.PropertyName<Task>(x => x.Name);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(1);
        column.Width(3);
        column.HeaderCell("Task Name");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<Task>(x => x.Assignee.Name); // nested property support
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(2);
        column.Width(3);
        column.HeaderCell("Assignee");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<Task>(x => x.IsActive);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(3);
        column.Width(2);
        column.HeaderCell("Active");
        column.ColumnItemsTemplate(template =>
        {
            template.Checkmark(checkmarkFillColor: Color.Green, crossSignFillColor:
Color.DarkRed);
        });
    });
}
}
.MainTableEvents(events =>
{
    events.DataSourceIsEmpty(message: "There is no data available to display.");
})
.Export(export =>
{
    export.ToExcel();
})
.Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
"\\Pdf\\CustomHeaderFooterPdfReportSample.pdf"));
}
}
}

```

به همراه Header سفرارشی:

```

using System.Collections.Generic;
using iTextSharp.text;
using iTextSharp.text.pdf;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;

namespace PdfReportSamples.CustomHeaderFooter
{
    public class CustomHeader : IPageHeader
    {
        public PdfPTable RenderingGroupHeader(Document pdfDoc, PdfWriter pdfWriter, IList<CellData>
rowdata, IList<SummaryCellData> summaryData)
        {
            return null;
        }

        Image _image;
        public PdfPTable RenderingReportHeader(Document pdfDoc, PdfWriter pdfWriter,
IList<SummaryCellData> summaryData)
        {
            if (_image == null) //cache is empty
            {
                var templatePath = AppPath.ApplicationPath + "\\data\\PdfHeaderTemplate.pdf";
                _image = PdfImageHelper.GetITextSharpImageFromPdfTemplate(pdfWriter, templatePath);
            }

            var table = new PdfPTable(1);
            var cell = new PdfPCell(_image, true) { Border = 0 };
            table.AddCell(cell);
        }
    }
}

```

```

        return table;
    }
}

```

و Footer سفرارشی استفاده شده:

```

using System.Collections.Generic;
using iTextSharp.text;
using iTextSharp.text.pdf;
using PdfRpt.Core.Contracts;

namespace PdfReportSamples.CustomHeaderFooter
{
    public class CustomFooter : IPageFooter
    {
        PdfContentByte _pdfContentByte;
        readonly IPdfFont _pdfRptFont;
        readonly Font _font;
        readonly PdfRunDirection _direction;
        PdfTemplate _template;

        public CustomFooter(IPdfFont pdfRptFont, PdfRunDirection direction)
        {
            _direction = direction;
            _pdfRptFont = pdfRptFont;
            _font = _pdfRptFont.Fonts[0];
        }

        public void ClosingDocument(PdfWriter writer, Document document, IList<SummaryCellData>
columnCellsSummaryData)
        {
            _template.BeginText();
            _template.SetFontAndSize(_pdfRptFont.Fonts[0].BaseFont, 8);
            _template.SetTextMatrix(0, 0);
            _template.ShowText((writer.PageNumber - 1).ToString());
            _template.EndText();
        }

        public void PageFinished(PdfWriter writer, Document document, IList<SummaryCellData>
columnCellsSummaryData)
        {
            var pageSize = document.PageSize;
            var text = "Page " + writer.PageNumber + " / ";
            var textLen = _font.BaseFont.GetWidthPoint(text, _font.Size);
            var center = (pageSize.Left + pageSize.Right) / 2;
            var align = _direction == PdfRunDirection.RightToLeft ? Element.ALIGN_RIGHT :
Element.ALIGN_LEFT;

            ColumnText.ShowTextAligned(
                canvas: _pdfContentByte,
                alignment: align,
                phrase: new Phrase(text, _font),
                x: center,
                y: pageSize.GetBottom(25),
                rotation: 0,
                runDirection: (int)_direction,
                arabicOptions: 0);

            var x = _direction == PdfRunDirection.RightToLeft ? center - textLen : center + textLen;
            _pdfContentByte.AddTemplate(_template, x, pageSize.GetBottom(25));
        }

        public void DocumentOpened(PdfWriter writer, IList<SummaryCellData> columnCellsSummaryData)
        {
            _pdfContentByte = writer.DirectContent;
            _template = _pdfContentByte.CreateTemplate(50, 50);
        }
    }
}

```

البته لازم به ذکر است که تمام این کدها به پوشه Samples سورس پروژه نیز جهت سهولت دسترسی، [اضافه شده‌اند](#).

توضیحات:

برای پیاده سازی Header و Footer سفارشی در PdfReport نیاز خواهید داشت تا دو اینترفیس IPageHeader و IPageFooter را پیاده سازی کنید.

ساختار IPageHeader را در ذیل ملاحظه می کنید:

```
using System.Collections.Generic;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace PdfRpt.Core.Contracts
{
    public interface IPageHeader
    {
        PdfPTable RenderingGroupHeader(Document pdfDoc, PdfWriter pdfWriter, IList<CellData>
newGroupInfo, IList<SummaryCellData> summaryData);

        PdfPTable RenderingReportHeader(Document pdfDoc, PdfWriter pdfWriter, IList<SummaryCellData>
summaryData);
    }
}
```

RenderingGroupHeader مرتبط است به مباحث گروه بندی اطلاعات و گزارشات master-detail که در قسمت های بعد به آنها اشاره خواهد شد. چون در اینجا به آن نیازی نداشتیم، تنها کافی است متد متناظر با آن، null بر گرداند که در کلاس CustomHeader فوق قابل مشاهده است.

متد RenderingReportHeader به ازای تولید هر صفحه جدید، فراخوانی خواهد شد. به عبارتی می توانید در صفحات مختلف، هدرهای مختلفی را نمایش دهید.

خروجی هر دو متد در اینجا یک جدول از نوع PdfPTable است. بنابراین هر نوع ساختار دلخواهی را که علاقمند هستید به شکل یک PdfPTable ایجاد کرده و بازگشت دهید. این جدول در هدر صفحات ظاهر خواهد شد.

برای نمونه در کلاس CustomHeader، یک قالب تهیه شده توسط Open Office توسط متد توکار PdfImageHelper.GetITextSharpImageFromPdfTemplate دریافت و تبدیل به تصویر می شود. این تصویر از نوع تصاویر قابل درک توسط iTextSharp است و نه اینکه واقعا تبدیل به یک تصویر معمولی مثلا از نوع bmp شود. سپس این تصویر، در یک ردیف از جدولی قرار داده شده و این جدول بازگشت داده می شود.

در کل یا توسط کار با PdfPTable می توانید یک هدر غیرپیش فرض را طراحی کنید و یا می توانید توسط ابزارهای بصری مانند Open Office یک قالب خاص را برای آن تهیه کرده و به روشی که ذکر شد و کدهای آنرا ملاحظه می کنید، بارگذاری و استفاده کنید. این قالب ها در مسیر Bin\Data سورس های پروژه قرار داده شده اند.

ساختار IPageFooter به صورت زیر است:

```
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.Collections.Generic;

namespace PdfRpt.Core.Contracts
{
    public interface IPageFooter
    {
        void DocumentOpened(PdfWriter writer, IList<SummaryCellData> columnCellsSummaryData);

        void PageFinished(PdfWriter writer, Document document, IList<SummaryCellData>
columnCellsSummaryData);

        void ClosingDocument(PdfWriter writer, Document document, IList<SummaryCellData>
columnCellsSummaryData);
    }
}
```

برای طراحی یک Footer سفارشی کافی است اینترفیس فوق را پیاده سازی کنید که نمونه ای از آنرا در کدهای کلاس CustomFooter ملاحظه می نمائید.

متد DocumentOpened، با وهله سازی شیء Document فراخوانی می شود.

متد PageFinished هر بار پیش از اتمام کار صفحه جاری و افزوده شدن آن به Document فراخوانی می‌گردد. متد ClosingDocument، در زمان بسته شدن شیء Document فراخوانی خواهد شد.

اگر به امضای این متدها دقت کنید، شیء PdfWriter در اختیار شما قرار گرفته است که توسط آن می‌توان مستقیماً بر روی فایل PDF، محتوایی را قرار داد. شیء Document نیز در دسترس است. مثلاً توسط آن می‌توان اندازه دقیق صفحه را بدست آورد. به علاوه پارامتر columnCellsSummaryData نیز امکان دسترسی به مقادیر ردیف‌های قبلی را در اختیار شما قرار می‌دهد. برای مثال اگر نیاز دارید تا بر اساس مقادیر ستون‌ها و ردیف‌های قبلی، محاسباتی را انجام داده و در پایین صفحات درج کنید، به این ترتیب دسترسی کاملی به آن‌ها، خواهید داشت.

استفاده از این کلاس‌های سفرارشی نیز همواره به شکل زیر خواهد بود:

```
readonly CustomHeader _customHeader = new CustomHeader();
//...
.PagesFooter(footer =>
{
    footer.CustomFooter(new CustomFooter(footer.PdfFont, PdfRunDirection.LeftToRight));
})
.PagesHeader(header =>
{
    header.CustomHeader(_customHeader);
})
```

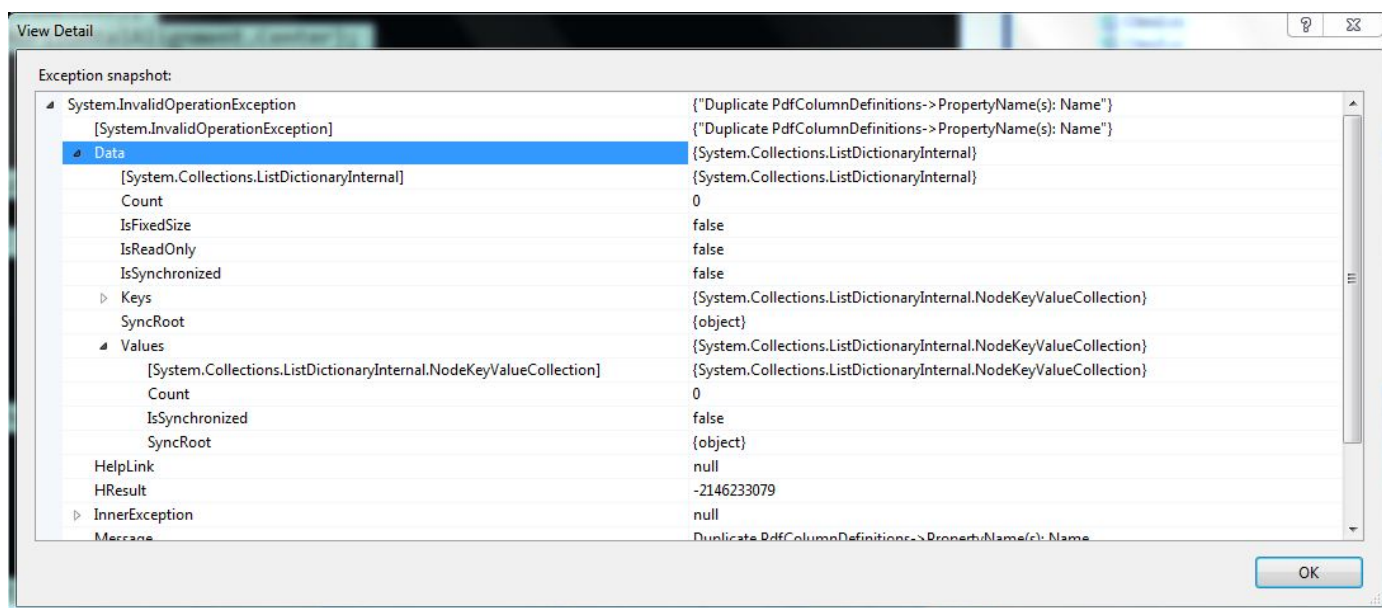
کلاً در PdfReport هر جایی متدی به نام CustomXYZ را مشاهده کردید، این متد یک اینترفیس را دریافت می‌کند. به عبارتی این امکان را خواهید داشت تا از متدهای پیش فرض مهیا صرف‌نظر کرده و مطابق نیاز، نسبت به پیاده سازی و استفاده از وهله جدیدی از این اینترفیس تعریف شده، اقدام کنید.

نظرات خوانندگان

نویسنده: mohsen

تاریخ: ۱۳۹۱/۰۷/۲۷ ۰:۴۸

با سلام؛ من برای پیاده سازی این مثال ابتدا خواستم که دقیقاً مثالی که شما اینجا ذکر کردید را بنویسم ولی متأسفانه به خطای زیر برخورددم ممنون میشم اگر راهنماییم کنید



نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۷/۲۷ ۱:۳

این مورد رو اخیرا اضافه کردم. لطفاً [نگارش 1.2](#) رو دریافت کنید تا خواص تو در تو را بدون مشکل بتوانید استفاده کنید. همچنین بهتر است از [NuGet استفاده کنید](#) تا از به روز رسانی‌ها بهتر مطلع شوید.

نویسنده: پویا امینی

تاریخ: ۱۳۹۱/۰۷/۲۹ ۰:۴۰

با سلام خدمت جناب نصیری، ببخشید شما فرمودید

می‌خواهیم در Header گزارش بجای Header پیش فرض PdfReport یکی از قالب‌های PDF تهیه شده توسط [Open Office](#) را نمایش دهیم (و یا هر ساختار دیگری را).

ولی در مثالی که در اینجا زدید در قسمت Header یک جدول ایجاد کردید حال اگر من بخواهم واقعاً از فایلی که با استفاده از OpenOffice ایجاد کرده‌ام و با استفاده از این روش مقدار TextBox‌های اون رو پر کرده‌ام استفاده کنم باید چه تغییری ایجاد کنم. ممنونم

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۷/۲۹ ۰:۵۱

خروجی نهایی متد `public PdfTable RenderingReportHeader` یک جدول است. به همین جهت تعریف یک جدول ساده رو مشاهده کردید (که داخل آن این قالب قرار گرفته). اما فایل [PdfHeaderTemplate.pdf](#) ذکر شده در آن، واقعا یک فایل قالب Open Office است. فایل odt آن هم در پوشه [Bin/Data](#) سورسها موجود است.

نویسنده: پویا امینی
تاریخ: ۱۳۹۱/۰۷/۲۹ ۱:۳۱

جناب نصیری من زمانی که فایل [25817](#) رو از این [آدرس](#) دانلود می‌کنم و کل Solution رو اجرا می‌کنم بهم خطای زیر رو میده

Error List			
7 Errors 0 Warnings 0 Messages			
Description	File	L	C
2 Metadata file 'C:\Users\Administrator\Desktop\vahid\Samples\PdfReportSamples\bin\Debug\PdfReportSamples.dll' could not be found			WebAppTests
6 Metadata file 'C:\Users\Administrator\Desktop\vahid\Samples\PdfReportSamples\bin\Debug\PdfReportSamples.dll' could not be found	CSC		DemosBrowser
1 Metadata file 'C:\Users\Administrator\Desktop\vahid\Lib\bin\Debug\PdfRpt.dll' could not be found			PdfReportSamples
3 Metadata file 'C:\Users\Administrator\Desktop\vahid\Lib\bin\Debug\PdfRpt.dll' could not be found			WebAppTests
7 Metadata file 'C:\Users\Administrator\Desktop\vahid\Lib\bin\Debug\PdfRpt.dll' could not be found	CSC		DemosBrowser
5 Importing key file "key.pfx" was canceled.			PdfRpt
4 Cannot import the following key file: key.pfx. The key file may be password protected. To correct this, try to import the certificate again or manually install the certificate to the Strong Name CSP with the following key container name: VS_KEY_5705A10B541BBA03			PdfRpt

و وقتی که از پوشه Samples فایل مربوط به WebApp رو باز می‌کنم باز هم قادر به اجرای اون نیستم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۹ ۹:۳۰

[اینجا پاسخ دادم](#) به این سؤال

نویسنده: a.g
تاریخ: ۱۳۹۲/۰۳/۱۵ ۱۵:۱۵

سلام
من نیاز دارم تا در Page Header و Group Header از قالب تهیه شده توسط Open Office استفاده کنم. قالب هایی که تهیه شدن، یه سری فیلد دارن که موقع ساخت گزارش باید پر بشن.
چطور باید فیلدهای موجود در قالب رو بعد از لود مقدار دهی کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۵ ۱۶:۵۴

در مورد جزئیات نحوه مقدار دهی فیلدهای این نوع قالبها مراجعه کنید به مطلب « [ساخت یک گزارش ساز به کمک iTextSharp و Open Office](#) ».

بعد از آشنایی، متد `GetITextSharpImageFromAcroForm` تعریف شده در PdfReport هم راه ساده‌تر پر کردن این نوع فیلدها است.

```
public static iTextSharp.text.Image GetITextSharpImageFromAcroForm(
    this PdfWriter pdfWriter,
    string pdfTemplateFilePath,
    IList<CellData> data,
    Action<IList<CellData>, AcroFields, PdfStamper> onFillAcroForm,
    IList<iTextSharp.text.Font> fonts,
```

```
int pageNumber = 1)
```

یک چنین امضایی داره تعریف شده در فضای نام [PdfRpt.Core.Helper](#).

نویسنده: عباس قربانی
تاریخ: ۲۳:۴۴ ۱۳۹۲/۰۳/۱۵

ممنون، واقعا لطف کردید

نویسنده: ali
تاریخ: ۱۸:۲۲ ۱۳۹۲/۰۳/۲۵

سلام

اول باید تشکر کنم بابت این ابزار که زحمتشو کشیدید.
میخواستم ببینم چطور میشه به AcroForm رو داخل footer گذاشت؟

نویسنده: وحید نصیری
تاریخ: ۱۸:۳۹ ۱۳۹۲/۰۳/۲۵

[در مثال هدر و فوتر سفارشی](#) یک نمونه استفاده از AcroForm به عنوان header هست. نکته مهم آن نحوه بازگشت این قالب به فرمت تصویر برداری قابل استفاده در iTextSharph است. سپس [در مثال InlineProviders](#) یک روش ساده تر افزودن محتویات دلخواه به فوتر صفحه معرفی شده در متد `inlineFooter.AddPageFooter`.

نویسنده: مهرداد
تاریخ: ۰:۲۷ ۱۳۹۲/۰۵/۰۵

با تشکر از شما
میشه نمونه دیگری برای استفاده سفارشی از فوتر هم بگذارید
چون مثلا من میخوام چند جمله در صفحه آخر فقط نمایش بدم
مثل امضا مدیر و امضا کاربر
ولی هر روشی انجام میدم باز جواب نمیده بهم و این کار را انجام نمیده
باید چه کاری بکنم
متشکرم

نویسنده: وحید نصیری
تاریخ: ۰:۵۹ ۱۳۹۲/۰۵/۰۵

[از رخدادها باید استفاده کنید](#).

نویسنده: مهرداد
تاریخ: ۱:۲ ۱۳۹۲/۰۵/۰۵

متشکرم
و یک سوال دیگه
من در قسمت هدر سایت گرید و تبیل طراحی میکنم
میخوام برادر هیچ کدوم از گریدها نداشته باشه اما گرید اصلی میشه
اما گریدهای داخلی نمیشه؟!
آیا باید کار دیگه ای کنم؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۰۵

- برای پرسش و پاسخ‌های متفرقه در مورد این کتابخانه لطفاً از [قسمت مخصوص آن در سایت](#) استفاده کنید.
- یک قسمت به [طراحی رنگ گرید اصلی](#) اختصاص دارد.
- پس از آن می‌تونید [قالب شفاف](#) هم مثلاً ایجاد کنید (یا هر حالت دلخواه دیگری). روش استفاده:

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

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 نمایید.
- 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 Hex، Offset 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 [assembly: InternalsVisibleTo](#)، تعریف internal یک اسمبلی را در اختیار 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 در نظر 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 .
میخواهم با این کتابخانه از کنترل jqGrid در mvc خروجی pdf تهیه کنم. به عبارت بهتر میخواهم یک کلاس بسازم که بصورت generic باشد. هر نوع jqGrid ی رو که بهش دادم برام تبدیل به pdf کنه. نخواه 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](#)

صورت مساله:

- لیستی از حقوق کارکنان را داریم. در گزارش نهایی آن نیاز است عدد حقوق کارکنانی با مبلغ کمتر از 1000، با رنگی دیگر نمایش داده شوند.

همچنین در این گزارش هر ردیفی که در ماه 7 واقع شده نیز ظاهر عدد سلول مربوط به آن ماه، به رنگ قهوه‌ای و زمینه زرد تغییر یابد.

- در ستون مشخصات افراد این گزارش، نیاز است تصویر کارمند به همراه نام او در ذیل این تصویر (داخل یک سلول) نمایش داده شوند.

چیزی شبیه به این گزارش!

 <p>گزارش جدید ما</p>			
ردیف	شخص	ماه	مبلغ
		نقل از ستون قبل	۲۵,۴۶۰
۲۵	 شخص ۲۴	۷	۱,۵۳۷
۲۶	 شخص ۲۵	۷	۱,۹۵۹
۲۷	 شخص ۲۶	۱۰	۱,۳۹۰
	 شخص ۲۷		
ردیف	شخص	ماه	مبلغ
		نقل از ستون قبل	۱۵,۴۶۳
۱۷	 شخص ۱۶	۱۰	۴۹۱
۱۸	 شخص ۱۷	۷	۱,۹۴۳
۱۹	 شخص ۱۸	۴	۱,۷۸۷

مورد اول در گزارشات، اصطلاحاً به conditional formatting معروف است و مورد دوم مرتبط است به تهیه قالب‌های سفارشی، بجای استفاده از قالب‌های سلول‌های پیش فرض PdfReport؛ که در ادامه نحوه انجام این موارد را بررسی خواهیم کرد.

ابتدا سورس کامل این مثال را ملاحظه نمایید:

```
using System;
```



```

using iTextSharp.text;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;
using PdfRpt.FluentInterface;

namespace PdfReportSamples.CustomCellTemplate
{
    public class CustomCellTemplatePdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                doc.Orientation(PageOrientation.Portrait);
                doc.PageSize(PdfPageSize.A4);
                doc.DocumentMetadata(new DocumentMetadata { Author = "Vahid", Application = "PdfRpt",
Keywords = "Test", Subject = "Test Rpt", Title = "Test" });
                doc.Compression(new CompressionSettings
                {
                    CompressionLevel = CompressionLevel.BestCompression,
                    EnableCompression = true
                });
            });
            .DefaultFonts(fonts =>
            {
                fonts.Path(AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
                    Environment.GetEnvironmentVariable("SystemRoot") + "\\fonts\\verdana.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("گزارش جدید ما");
                });
            })
            .MainTableTemplate(template =>
            {
                template.BasicTemplate(BasicTemplate.SnowyPineTemplate);
            })
            .MainTablePreferences(table =>
            {
                table.ColumnsWidthsType(TableColumnWidthType.Relative);
                table.MultipleColumnsPerPage(new MultipleColumnsPerPage
                {
                    ColumnsGap = 20,
                    ColumnsPerPage = 2,
                    ColumnsWidth = 250,
                    IsRightToLeft = true,
                    TopMargin = 7
                });
            })
            .MainTableDataSource(dataSource =>
            {
                var table = new System.Data.DataTable("لیست حقوق");
                table.Columns.Add("شخص", typeof(string));
                table.Columns.Add("ماه", typeof(int));
                table.Columns.Add("مبلغ", typeof(decimal));

                var rnd = new Random();
                for (int i = 0; i < 200; i++)
                    table.Rows.Add("شخص " + i, rnd.Next(1, 12), rnd.Next(400, 2000));

                dataSource.DataTable(table);
            })
            .MainTableEvents(events =>
            {
                events.DataSourceIsEmpty(message: "There is no data available to display.");
                events.CellCreated(args =>
                {
                    //change the background color of the cell based on the value
                    if (args.RowType == RowType.DataTableRow && args.Cell.RowData.Value != null &&
args.Cell.RowData.Value is decimal)
                    {
                        if ((decimal)args.Cell.RowData.Value <= 1000)
                            args.Cell.BasicProperties.BackgroundColor = BaseColor.CYAN;
                    }
                });
            });
        }
    }
}

```

```

    });
    });
    .MainTableSummarySettings(summary =>
    {
        summary.OverallSummarySettings("جمع کل");
        summary.PageSummarySettings("جمع صفحه");
        summary.PreviousPageSummarySettings("نقل از ستون قبل");
    })
    .MainTableColumns(columns =>
    {
        columns.AddColumn(column =>
        {
            column.PropertyName("rowNo");
            column.IsRowNumber(true);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(0);
            column.Width(1);
            column.HeaderCell("ردیف");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName("شخص");
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(1);
            column.Width(3);
            column.HeaderCell("شخص");
            column.ColumnItemsTemplate(t => t.CustomTemplate(new MyCustomCellTemplate()));
        });

        columns.AddColumn(column =>
        {
            column.PropertyName("ماه");
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(2);
            column.Width(2);
            column.HeaderCell("ماه");
            column.ColumnItemsTemplate(template =>
            {
                template.TextBlock();
                template.ConditionalFormatFormula(list =>
                {
                    var cellValue = int.Parse(list.GetSafeStringValueOf("ماه", nullValue:
"0"));
                    if (cellValue == 7)
                    {
                        return new CellBasicProperties
                        {
                            PdfFontStyle = DocumentFontStyle.Bold |
DocumentFontStyle.Underline,
                            FontColor = new BaseColor(System.Drawing.Color.Brown),
                            BackgroundColor = new BaseColor(System.Drawing.Color.Yellow)
                        };
                    }
                    return new CellBasicProperties { PdfFontStyle = DocumentFontStyle.Normal
};
                });
            });
        });

        columns.AddColumn(column =>
        {
            column.PropertyName("مبلغ");
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(3);
            column.Width(2);
            column.HeaderCell("مبلغ");
            column.ColumnItemsTemplate(template =>
            {
                template.TextBlock();
                template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
            });
            column.AggregateFunction(aggregateFunction =>
            {
                aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
            });
        });
    });
}

```

```

        aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
    });
    });
    .Export(export =>
    {
        export.ToXml();
        export.ToExcel();
    })
    .Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
"\\Pdf\\RptDataTableSample.pdf"));
}
}
}

```

به همراه قالب سلول سفارشی آن:

```

using System;
using System.Collections.Generic;
using iTextSharp.text;
using iTextSharp.text.pdf;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;

namespace PdfReportSamples.CustomCellTemplate
{
    public class MyCustomCellTemplate : IColumnItemsTemplate
    {
        Random _rnd = new Random();

        public void CellRendered(PdfPCell cell, Rectangle position, PdfContentByte[] canvases,
CellAttributes attributes)
        {
        }

        public CellBasicProperties BasicProperties { set; get; }
        public Func<IList<CellData>, CellBasicProperties> ConditionalFormatFormula { set; get; }

        public PdfPCell RenderingCell(CellAttributes attributes)
        {
            var pdfCell = new PdfPCell();
            var table = new PdfPTable(1) { RunDirection = PdfWriter.RUN_DIRECTION_RTL };

            var filePath = AppPath.ApplicationPath + "\\Images\\" + _rnd.Next(1, 5).ToString("00") +
".png";
            var photo = PdfImageHelper.GetITextSharpImageFromImageFile(filePath);
            table.AddCell(new PdfPCell(photo, fit: false)
            {
                Border = 0,
                VerticalAlignment = Element.ALIGN_BOTTOM,
                HorizontalAlignment = Element.ALIGN_CENTER
            });

            var name = attributes.RowData.TableRowData.GetSafeStringValueOf("شخص");
            table.AddCell(new PdfPCell(attributes.BasicProperties.PdfFont.FontSelector.Process(name))
            {
                Border = 0,
                HorizontalAlignment = Element.ALIGN_CENTER
            });

            pdfCell.AddElement(table);

            return pdfCell;
        }
    }
}

```

توضیحات:

- در این مثال از منبع داده‌ای از نوع DataTable استفاده شده است؛ که نحوه بکارگیری آن‌را در متد MainTableDataSource
- ملاحظه می‌کنید. ستون‌های تعریف شده در MainTableColumns نیز بر اساس ستون‌های DataTable مشخص شده‌اند.
- در متد DocumentPreferences، نحوه مشخص سازی فشرده سازی نهایی فایل PDF را ملاحظه می‌کنید. این مورد از مزایای

استفاده از فایل‌های PDF است.

- برای اعمال فرمت شرطی اطلاعات در PdfReport دو روش وجود دارد.
الف) استفاده از متد MainTableEvents و کار کردن با رخدادهای تعریف شده در آن مانند CellCreated. در اینجا می‌توان در نحوه رندر شدن یک سلول دخالت کرد:

```
events.CellCreated(args =>
{
    //change the background color of the cell based on the value
    if (args.RowType == RowType.DataTableRow && args.Cell.RowData.Value != null &&
args.Cell.RowData.Value is decimal)
    {
        if ((decimal)args.Cell.RowData.Value <= 1000)
            args.Cell.BasicProperties.BackgroundColor = BaseColor.CYAN;
    }
});
```

برای مثال در تعاریف فوق، اگر نوع ردیف، از نوع ردیف‌های اطلاعاتی جدول باشد، مقدار آن دریافت شده و بر اساس شرطی مشخص، برای نمونه رنگ پس زمینه آن سلول تغییر داده می‌شود.
ب) همانطور که در قسمت تعریف ستون «ماه» ملاحظه می‌کنید، توسط متد template.ConditionalFormatFormula نیز، امکان فرمت شرطی اطلاعات فراهم شده است. در اینجا می‌توان به لیست اطلاعات سلول‌های ردیف جاری دسترسی یافت و سپس بر اساس آن تصمیم‌گیری کرد.

- جهت تعریف قالب‌های سفارشی سلول‌ها کافی است اینترفیس IColumnItemsTemplate را پیاده‌سازی کنیم؛ که نمونه‌ای از آن را در کدهای MyCustomCellTemplate فوق ملاحظه می‌کنید. در اینجا فرصت خواهید داشت هر شکل و طرح متنوعی را تهیه کرده و به صورت یک PdfPCell بازگشت دهید. برای نمونه در مثال فوق، یک جدول را در سلول تعریف شده قرار داده‌ایم. این جدول یک ستون دارد و هر سلولی که به آن اضافه خواهد شد، یک ردیف را تشکیل خواهد داد. در ردیف اول آن تصویر قرار گرفته و در ردیف دوم آن مقدار سلول جاری.

در مطلب «[تولید پویای ستون‌ها در PdfReport](#)» عنوان شد که ذکر قسمت MainTableColumns و تمام تعاریف مرتبط با آن در PdfReports اختیاری است. همچنین به کمک متد MainTableAdHocColumnsConventions می‌توان بر اساس نوع‌های داده‌ای، بر روی نحوه نمایش ستون‌ها تاثیر گذاشت. برای مثال هرجایی DateTime مشاهده شد، به صورت خودکار تبدیل به تاریخ شمسی شود.

روش دیگری که این روزها در اکثر فریم‌های دات نتی مرسوم شده است، استفاده از Data Annotations جهت انتساب یک سری متادیتا به خاصیت‌های تعریف شده کلاس‌ها است. برای مثال ASP.NET MVC از این قابلیت زیاد استفاده می‌کند (در تولید پویای کد، یا اعتبار سنجی‌های سمت سرور و کاربر).

به همین جهت برای سازگاری بیشتر PdfReport با مدل‌های اینگونه فریم ورک‌ها، اکثر ویژگی‌ها و Data Annotations متداول را نیز می‌توان در PdfReport بکار برد. همچنین تعدادی ویژگی سفارشی نیز تعریف شده است، که در ادامه به بررسی آن‌ها خواهیم پرداخت.

آشنایی با مدل‌های بکار رفته در مثال جاری:

```
using System.ComponentModel;

namespace PdfReportSamples.Models
{
    public enum JobTitle
    {
        [Description("Grunt")]
        Grunt,

        [Description("Programmer")]
        Programmer,

        [Description("Analyst Programmer")]
        AnalystProgrammer,

        [Description("Project Manager")]
        ProjectManager,

        [Description("Chief Information Officer")]
        ChiefInformationOfficer,
    }
}
```

در اینجا یک enum، جهت تعیین سمت شغلی تعریف شده است. برای اینکه بتوان خروجی مطلوبی را در گزارشات شاهد بود، می‌توان از ویژگی Description، جهت تعیین مقدار نمایشی آن‌ها نیز استفاده کرد و این تعاریف در PdfReport خوانده و اعمال می‌شوند.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using PdfReportSamples.Models;
using PdfRpt.Aggregates.Numbers;
using PdfRpt.ColumnsItemsTemplates;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;
using PdfRpt.DataAnnotations;

namespace PdfReportSamples.DataAnnotations
{
    public class Person
    {
        [IsVisible(false)]
        public int Id { get; set; }

        [DisplayName("User name")]
        //Note: If you don't specify the ColumnItemsTemplate, a new TextBlockField() will be used
    }
}
```

```

automatically.
[ColumnItemsTemplate(typeof(TextBlockField))]
public string Name { get; set; }

[DisplayName("Job title")]
public JobTitle JobTitle { set; get; }

[DisplayName("Date of birth")]
[DisplayFormat(DataFormatString = "{0:MM/dd/yyyy}")]
public DateTime DateOfBirth { get; set; }

[DisplayName("Date of death")]
[DisplayFormat(NullDisplayText = "-", DataFormatString = "{0:MM/dd/yyyy}")]
public DateTime? DateOfDeath { get; set; }

[DisplayFormat(DataFormatString = "{0:n0}")]
[CustomAggregateFunction(typeof(Sum))]
public int Salary { get; set; }

[IsCalculatedField(true)]
[DisplayName("Calculated Field")]
[DisplayFormat(DataFormatString = "{0:n0}")]
[AggregateFunction(AggregateFunction.Sum)]
public string CalculatedField { get; set; }

[CalculatedFieldFormula("CalculatedField")]
public static Func<IList<CellData>, object> CalculatedFieldFormula =
    list =>
    {
        if (list == null) return string.Empty;
        var salary = (int)list.GetValueOf<Person>(x =>
            x.Salary);

        return salary * 0.8;
    }; //Note: It's a static field, not a property.
}
}

```

مدل فوق جهت مقدار دهی اطلاعات یک شخص تعریف شده است.

- اگر قصد ندارید خاصیتی در این بین، در گزارشات ظاهر شود، از ویژگی `IsVisible` با مقدار `false` استفاده کنید.
- از ویژگی `DisplayName` جهت تعیین برچسب‌های سرستون‌ها استفاده خواهد شد.
- ذکر ویژگی `ColumnItemsTemplate` اختیاری است و اگر عنوان نشود به صورت خودکار از `TextBlockField` استفاده خواهد شد. اما اگر نیاز به استفاده از قالب‌های ستون‌های سفارشی و یا حتی قالب‌های پیش فرض دیگری که متنی نیستند، وجود دارد، می‌توانید از ویژگی `ColumnItemsTemplate` به همراه نوع کلاس مورد نظر استفاده نمایید. کلاس‌های پیش فرض قالب‌های ستون‌ها در `PdfReport` در پوشه `Lib\ColumnsItemsTemplates` سورس آن قرار دارند.
- برای تعیین نحوه فرمت اطلاعات در اینجا می‌توان از ویژگی `DisplayFormat` استفاده کرد. این ویژگی در اسمبلی `System.ComponentModel.DataAnnotations.dll` دات نت تعریف شده است؛ که در اینجا نمونه‌ای از استفاده از آن را برای تعیین نحوه نمایش تاریخ، ملاحظه می‌کنید. توسط این ویژگی حتی می‌توان مشخص ساخت (توسط پارامتر `NullDisplayText`) که اگر اطلاعاتی `null` بود، بجای آن چه عبارتی نمایش داده شود.
- اگر علاقمند به اعمال تابعی تجمعی به ستونی خاص هستید، از ویژگی `CustomAggregateFunction` استفاده کنید. پارامتر آن نوع کلاس تابع مورد نظر است. یک سری تابع تجمعی پیش فرض در فضای نام `PdfRpt.Aggregates.Numbers` قرار دارند. البته امکان تهیه انواع سفارشی آن‌ها نیز پیش بینی شده است که در قسمت‌های بعد به آن خواهیم پرداخت.
- امکان تعریف خواص محاسباتی نیز پیش بینی شده است. برای این منظور دو کار را باید انجام داد:
 - الف) ویژگی `IsCalculatedField` را با مقدار `true` بر روی خاصیت مورد نظر اعمال کنید.
 - ب) هم نام خاصیت محاسباتی افزوده شده به کلاس جاری، ویژگی `CalculatedFieldFormula` را بر روی یک فیلد استاتیک عمومی در آن کلاس به نحوی که ملاحظه می‌کنید (مطابق امضای فیلد `CalculatedFieldFormula` فوق)، تعریف نمایید. (علت این است که نمی‌توان توسط ویژگی‌ها از `delegates` استفاده کرد و این محدودیت ذاتی وجود دارد)

در ادامه کدهای منبع داده فرضی مثال جاری ذکر شده است:

```

using System;
using System.Collections.Generic;
using PdfReportSamples.Models;

```

```

namespace PdfReportSamples.DataAnnotations
{
    public static class PersonnelDataSource
    {
        public static IList<Person> CreatePersonnelList()
        {
            return new List<Person>
            {
                new Person
                {
                    Id = 1,
                    Name = "Edward",
                    DateOfBirth = new DateTime(1900, 1, 1),
                    DateOfDeath = new DateTime(1990, 10, 15),
                    JobTitle = JobTitle.ChiefInformationOfficer,
                    Salary = 5000
                },
                new Person
                {
                    Id = 2,
                    Name = "Margaret",
                    DateOfBirth = new DateTime(1950, 2, 9),
                    DateOfDeath = null,
                    JobTitle = JobTitle.AnalystProgrammer,
                    Salary = 4000
                },
                new Person
                {
                    Id = 3,
                    Name = "Grant",
                    DateOfBirth = new DateTime(1975, 6, 13),
                    DateOfDeath = null,
                    JobTitle = JobTitle.Programmer,
                    Salary = 3500
                }
            };
        }
    }
}

```

در پایان، نحوه استفاده از منبع داده فوق جهت تامین یک گزارش، به نحو زیر می‌باشد:

```

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

namespace PdfReportSamples.DataAnnotations
{
    public class DataAnnotationsPdfReport
    {
        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\\tahoma.ttf",
Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.ttf");
            })
            .PagesFooter(footer =>
            {
                footer.DefaultFooter(printDate: DateTime.Now.ToString("MM/dd/yyyy"));
            })
            .PagesHeader(header =>
            {
                header.DefaultHeader(defaultHeader =>
                {
                    defaultHeader.ImagePath(AppPath.ApplicationPath + "\\Images\\01.png");
                    defaultHeader.Message("new rpt.");
                    defaultHeader.RunDirection(PdfRunDirection.LeftToRight);
                });
            });
        }
    }
}


```

```

    })
    .MainTableTemplate(template =>
    {
        template.BasicTemplate(BasicTemplate.ClassicTemplate);
    })
    .MainTablePreferences(table =>
    {
        table.ColumnsWidthsType(TableColumnWidthType.FitToContent);
    })
    .MainTableDataSource(dataSource =>
    {
        dataSource.StronglyTypedList(PersonnelDataSource.CreatePersonnelList());
    })
    .MainTableEvents(events =>
    {
        events.DataSourceIsEmpty(message: "There is no data available to display.");
    })
    .MainTableSummarySettings(summary =>
    {
        summary.OverallSummarySettings("Total");
        summary.PageSummarySettings("Page Summary");
        summary.PreviousPageSummarySettings("Previous Page Summary");
    })
    .MainTableAdHocColumnsConventions(adHocColumns =>
    {
        adHocColumns.ShowRowNumberColumn(true);
        adHocColumns.RowNumberColumnCaption("#");
    })
    .Export(export =>
    {
        export.ToExcel();
        export.ToXml();
    })
    .Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
    "\\Pdf\\DataAnnotationsSampleRpt.pdf"));
    }
}

```

همانطور که مشخص است، از ذکر متد MainTableColumns به علت استفاده از DataAnnotations صرفنظر شده و PdfReport این تعاریف را بر اساس ویژگی‌های خواص کلاس شخص دریافت می‌کند. تنها از متد MainTableAdHocColumnsConventions جهت مشخص سازی اینکه نیاز به نمایش ستون ردیف می‌باشد، استفاده کرده‌ایم.

 <p>new rpt.</p>						
#	User name	Job title	Date of birth	Date of death	Salary	Calculated Field
1	Edward	Chief Information Officer	01/01/1900	10/15/1990	5,000	4,000
2	Margaret	Analyst Programmer	02/09/1950	-	4,000	3,200
3	Grant	Programmer	06/13/1975	-	3,500	2,800
Page Summary					12,500	10,000
Total					12,500	10,000

در حالت کلی، هر شیءایی را که بتوان تبدیل به تصویر کرد، قابلیت قرارگیری در یک فایل PDF را هم خواهد داشت. از این نمونه می‌توان به اشیاء [MSChart](#) اشاره کرد که از دات نت 4 جزئی از کتابخانه‌های اصلی دات نت شده‌اند و البته برای دات نت سه و نیم نیز به صورت جداگانه قابل دریافت است.

در ادامه مثالی را بررسی خواهیم کرد که بر اساس ردیف‌های گزارش آن، یک نمودار، به انتهای گزارش اضافه خواهد شد. بدهای کامل این مثال را در ذیل مشاهده می‌کنید:

```
using System;
using System.Collections.Generic;
using PdfReportSamples.Models;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;
using PdfRpt.FluentInterface;

namespace PdfReportSamples.ChartImage
{
    public class ChartImagePdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            var chart = new MSChartHelper
            {
                AxisXTitle = "User",
                AxisYTitle = "Balance",
                ChartTitle = "Users Balance",
                AxisTitleFont = new System.Drawing.Font("Tahoma", 12f),
                LabelStyleFont = new System.Drawing.Font("Tahoma", 10f),
                ChartTitleFont = new System.Drawing.Font("Arial", 16f,
System.Drawing.FontStyle.Bold),
                LegendsFont = new System.Drawing.Font("Tahoma", 12f)
            };

            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                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(string.Format("{0}\\fonts\\irsans.ttf", AppPath.ApplicationPath),
string.Format("{0}\\fonts\\verdana.ttf",
Environment.GetEnvironmentVariable("SystemRoot")));
            })
            .PagesFooter(footer =>
            {
                footer.DefaultFooter(printDate: DateTime.Now.ToString("MM/dd/yyyy"));
            })
            .PagesHeader(header =>
            {
                header.DefaultHeader(defaultHeader =>
                {
                    defaultHeader.ImagePath(AppPath.ApplicationPath + "\\Images\\01.png");
                    defaultHeader.Message("گزارش جدید ما");
                });
            })
            .MainTableTemplate(template =>
            {
                template.BasicTemplate(BasicTemplate.ClassicTemplate);
            })
            .MainTablePreferences(table =>
            {
                table.ColumnsWidthsType(TableColumnWidthType.Relative);
            })
            .MainTableDataSource(dataSource =>
            {
                var listOfRows = new List<User>();
                for (var i = 0; i < 7; i++)
```

```

        {
            listOfRows.Add(new User { Id = i, LastName = "نام خانوادگی " + i, Name = "نام " + i,
Balance = (i * 50) + 1000 });
        }
        dataSource.StronglyTypedList(listOfRows);
    })
    .MainTableEvents(events =>
    {
        events.DataSourceIsEmpty(message: "There is no data available to display.");
        events.DocumentOpened(args =>
        {
            chart.ChartInit(width: (int)args.PdfWriter.PageSize.Width - 100, height: 250);
        });
        events.RowAdded(args =>
        {
            if (args.RowType == RowType.DataTableRow)
            {
                var name = args.TableRowData.GetValueOf<User>(x => x.Name);
                if (name == null) return;

                var balance = args.TableRowData.GetValueOf<User>(x => x.Balance);
                if (balance == null) return;

                chart.AddXY(name, balance);
            }
        });
        events.DocumentClosing(args =>
        {
            chart.AddChartToPage(args.PdfDoc);
            chart.FreeResources();
        });
    })
    .MainTableSummarySettings(summary =>
    {
        summary.OverallSummarySettings("جمع");
        summary.PreviousPageSummarySettings("نقل از صفحه قبل");
    })
    .MainTableColumns(columns =>
    {
        columns.AddColumn(column =>
        {
            column.PropertyName("rowNo");
            column.IsRowNumber(true);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(0);
            column.Width(1);
            column.HeaderCell("ردیف", captionRotation: 90);
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<User>(x => x.Id);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(1);
            column.Width(2);
            column.HeaderCell("شماره");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<User>(x => x.Name);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(2);
            column.Width(2);
            column.HeaderCell("نام");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<User>(x => x.LastName);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(3);
            column.Width(3);
            column.HeaderCell("نام خانوادگی");
        });

        columns.AddColumn(column =>

```

```

        {
            column.PropertyName<User>(x => x.Balance);
            column.HeaderCell("موجودی");
            column.ColumnItemsTemplate(template =>
            {
                template.TextBlock();
                template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
            });
            column.Width(2);
            column.AggregateFunction(aggregateFunction =>
            {
                aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
                aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
            });
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(4);
        });
    });
    .Generate(data => data.AsPdfFile(AppPath.ApplicationPath + "\\Pdf\\RptChartSample.pdf"));
}
}
}

```

برای تهیه این گزارش و افزودن نمودار به آن، از کلاس کمکی ذیل استفاده شده است:

```

using System.Drawing;
using System.IO;
//It's part of the .NET 4.0+ now.
using System.Windows.Forms.DataVisualization.Charting;
using iTextSharp.text;
using iTextSharp.text.pdf;
using PdfRpt.Core.Helper;

namespace PdfReportSamples.ChartImage
{
    public class MSChartHelper
    {
        // MS Chart learning tutorials:
        // http://weblogs.asp.net/scottgu/archive/2010/02/07/built-in-charting-controls-vs-2010-and-
net-4-series.aspx
        Chart _chart;

        public System.Drawing.Font AxisTitleFont { set; get; }

        public string AxisXTitle { set; get; }

        public string AxisYTitle { set; get; }

        public string ChartTitle { set; get; }

        public System.Drawing.Font ChartTitleFont { set; get; }

        public System.Drawing.Font LabelStyleFont { set; get; }

        public System.Drawing.Font LegendsFont { set; get; }

        public void AddChartToPage(Document pdfDoc,
            int scalePercent = 100,
            float spacingBefore = 20,
            float spacingAfter = 10,
            float widthPercentage = 80)
        {
            using (var chartimage = new MemoryStream())
            {
                _chart.SaveImage(chartimage, ChartImageFormat.Bmp); //BMP gives the best compression
result

                var iTextSharpImage =
PdfImageHelper.GetITextSharpImageFromByteArray(chartimage.GetBuffer());
                iTextSharpImage.ScalePercent(scalePercent);
                iTextSharpImage.Alignment = Element.ALIGN_CENTER;

                var table = new PdfPTable(1)
                {
                    WidthPercentage = widthPercentage,

```

```

        SpacingBefore = spacingBefore,
        SpacingAfter = spacingAfter
    };
    table.AddCell(iTextSharpImage);

    pdfDoc.Add(table);
}
}

public void AddXY(object xValue, params object[] yValue)
{
    _chart.Series[0].Points.AddXY(xValue, yValue);
}

public void ChartInit(int width, int height)
{
    _chart = new Chart
    {
        Width = width,
        Height = height,
        AntiAliasing = AntiAliasingStyles.All,
        TextAntiAliasingQuality = TextAntiAliasingQuality.High,
        Palette = ChartColorPalette.BrightPastel,
        BackColor = ColorTranslator.FromHtml("#F3DFC1"),
        BackGradientStyle = GradientStyle.TopBottom
    };

    setBorder();
    setTitles();
    setChartAreas();
    setLegends();
    setSeries();
}

public void FreeResources()
{
    if (_chart != null && !_chart.IsDisposed)
        _chart.Dispose();
}

private void setBorder()
{
    _chart.BorderSkin.SkinStyle = BorderSkinStyle.Emboss;
    _chart.BorderlineWidth = 2;
    _chart.BorderlineColor = Color.FromArgb(181, 64, 1);
    _chart.BorderlineDashStyle = ChartDashStyle.Solid;
}

private void setChartAreas()
{
    _chart.ChartAreas.Add("ChartArea1");
    _chart.ChartAreas[0].AxisX.Title = AxisXTitle;
    _chart.ChartAreas[0].AxisY.Title = AxisYTitle;
    _chart.ChartAreas[0].AxisX.TitleFont = AxisTitleFont;
    _chart.ChartAreas[0].AxisY.TitleFont = AxisTitleFont;
    _chart.ChartAreas[0].AxisX.LabelStyle.Font = LabelStyleFont;
    _chart.ChartAreas[0].AxisX.LabelStyle.Angle = -90;
    _chart.ChartAreas[0].BackColor = Color.White;
    _chart.ChartAreas[0].AxisX.LineColor = Color.FromArgb(64, 64, 64);
    _chart.ChartAreas[0].AxisX.MajorGrid.LineColor = Color.FromArgb(64, 64, 64);
    _chart.ChartAreas[0].AxisY.LineColor = Color.FromArgb(64, 64, 64);
    _chart.ChartAreas[0].AxisY.MajorGrid.LineColor = Color.FromArgb(64, 64, 64);
}

private void setLegends()
{
    _chart.Legends.Add("Default");
    _chart.Legends[0].LegendStyle = LegendStyle.Row;
    _chart.Legends[0].IsTextAutoFit = false;
    _chart.Legends[0].DockedToChartArea = "ChartArea1";
    _chart.Legends[0].Docking = Docking.Bottom;
    _chart.Legends[0].IsDockedInsideChartArea = false;
    _chart.Legends[0].BackColor = Color.Transparent;
    _chart.Legends[0].Font = LegendsFont;
}

private void setSeries()
{
    _chart.Series.Add("");
    _chart.Series[0].ChartType = SeriesChartType.Column;
    _chart.Series[0].Palette = ChartColorPalette.EarthTones;
}

```

```

        _chart.Series[0].IsValueShownAsLabel = true;
        _chart.Series[0].IsVisibleInLegend = false;
    }

    private void setTitles()
    {
        _chart.Titles.Add(ChartTitle);
        _chart.Titles[0].Font = ChartTitleFont;
        _chart.Titles[0].TextStyle = TextStyle.Shadow;
        _chart.Titles[0].ShadowOffset = 3;
        _chart.Titles[0].ShadowColor = Color.FromArgb(32, 0, 0);
        _chart.Titles[0].Alignment = ContentAlignment.TopCenter;
        _chart.Titles[0].ForeColor = Color.FromArgb(26, 59, 105);
    }
}

```

توضیحات:

- استفاده از MSChart در اینجا از این جهت مناسب است که فراگیری کار کردن با آن عمومی بوده و در پروژه‌های وب و ویندوز کاربرد دارد و احتمالاً هم اکنون با نحوه کار کردن با آن آشنا هستید، زیرا [از سال 2010](#) به دات نت اضافه شده است.
- در این بین تنها متد جدید و مهم کلاس MSChartHelper، متد AddChartToPage آن است. به کمک متد chart.SaveImage می‌توان تصویر نهایی معادل یک نمودار را در حافظه ذخیره کرد. سپس با استفاده از متد PdfImageHelper.GetITextSharpImageFromByteArray، این تصویر موجود در حافظه را به معادل قابل استفاده آن در iTextSharp تبدیل کرده و به صفحه اضافه خواهیم کرد.
- در کدهای اصلی تولید گزارش، مقدار دهی کلاس کمکی MSChartHelper در قسمت رخدادهای قابل استفاده PdfReport در متد MainTableEvents آن انجام شده است:
- در روال رویدادگردان DocumentOpened، بر اساس عرض واقعی صفحه، عرض نمودار را مشخص می‌کنیم:

```

events.DocumentOpened(args =>
{
    chart.ChartInit(width: (int)args.PdfWriter.PageSize.Width - 100, height: 250);
});

```

- سپس در روال رویدادگردان RowAdded، فرصت خواهیم داشت به اطلاعات در حال افزوده شدن به گزارش دسترسی داشته باشیم. این اطلاعات را به متد افزودن XY نمودار ارسال خواهیم کرد:

```

events.RowAdded(args =>
{
    if (args.RowType == RowType.DataTableRow)
    {
        var name = args.TableRowData.GetValueOf<User>(x => x.Name);
        if (name == null) return;

        var balance = args.TableRowData.GetValueOf<User>(x => x.Balance);
        if (balance == null) return;

        chart.AddXY(name, balance);
    }
});

```

- و در آخر، پیش از بسته شدن فایل PDF تولیدی (DocumentClosing)، نمودار نهایی را به صفحه اضافه کرده و منابع مرتبط با آن را آزاد خواهیم کرد:

```

events.DocumentClosing(args =>
{
    chart.AddChartToPage(args.PdfDoc);
    chart.FreeResources();
});

```

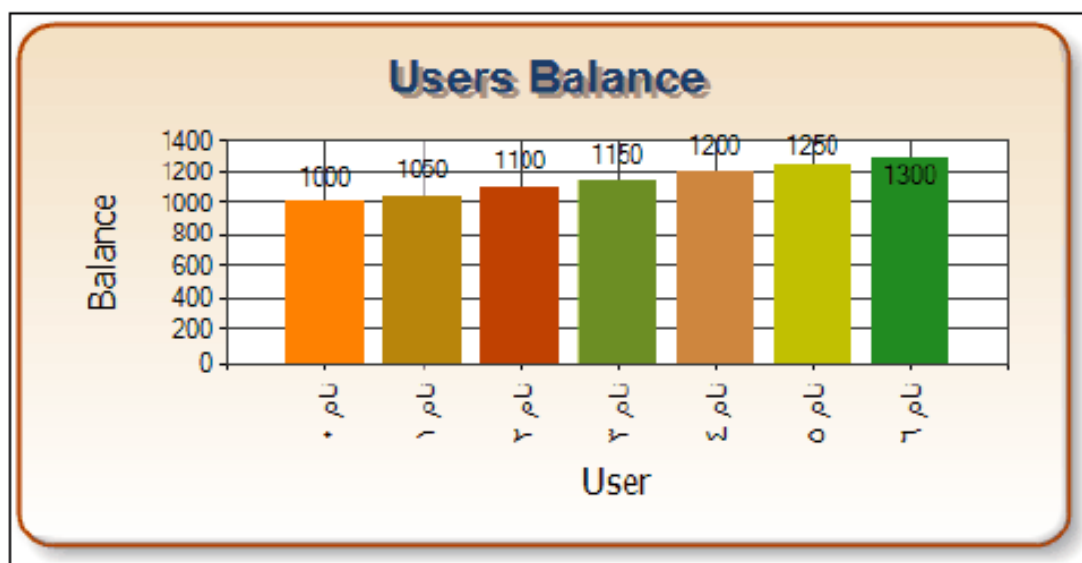
بنابراین اگر این سؤال عمومی وجود دارد که آیا می‌توان در این بین، به ابتدا و انتهای گزارش اشیایی را افزود، روش کلی آن را در

روال‌های فوق ملاحظه می‌کنید. توسط args.PdfDoc و args.PdfWriter می‌توان به زیرساخت iTextSharp دسترسی یافت.



گزارش جدید ما

ردیف	شماره	نام	نام خانوادگی	موجودی
۱	۰	نام ۰	نام خانوادگی ۰	۱,۰۰۰
۲	۱	نام ۱	نام خانوادگی ۱	۱,۰۵۰
۳	۲	نام ۲	نام خانوادگی ۲	۱,۱۰۰
۴	۳	نام ۳	نام خانوادگی ۳	۱,۱۵۰
۵	۴	نام ۴	نام خانوادگی ۴	۱,۲۰۰
۶	۵	نام ۵	نام خانوادگی ۵	۱,۲۵۰
۷	۶	نام ۶	نام خانوادگی ۶	۱,۳۰۰
جمع				۸,۰۵۰



نظرات خوانندگان

نویسنده: الهام
تاریخ: ۱۳۹۱/۰۷/۲۲ ۸:۲۵

سلام آقای نصیری

مثال مفیدی بود. من یک مشکلی داشتم با ms-chart میخوامستم از شما راهنمایی بگیرم.

آیا میشه اعداد داخل ms-chart رو به فارسی تبدیل کرد؟

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۰:۱۸

بله. از فونت‌های فارسی تغییر یافته استفاده کنید. برای مثال فونت `irsans` ایی که در پوشه `bin` مثال‌های پروژه هست، از این نوع است. فونت‌های معمولی رو در یک [فونت ادیتور](#) باز می‌کنند و بجای اعداد انگلیسی آن، معادل اعداد فارسی موجود در همان فایل فونت را کپی و پیست می‌کنند. این روش از قدیم برای ساخت گزارشات فارسی کاربرد داشته. ضمناً اگر فونت مورد نظر بر روی سرور نصب نیست، به این صورت هم قابل بارگذاری است:

```
var privateFontCollection = new PrivateFontCollection();
privateFontCollection.AddFontFile(fontPath);
var fontFamily = privateFontCollection.Families[0];
var font = new Font(fontFamily);
```

شکل زیر را که شبیه به یک فاکتور فروش است در نظر بگیرید:

 <p>گزارش جدید ما</p>				
ردیف	شماره	نام	نام خانوادگی	موجودی
۱	۰	نام ۰	نام خانوادگی ۰	۱,۰۰۰
۲	۱	نام ۱	نام خانوادگی ۱	۱,۰۰۱
۳	۲	نام ۲	نام خانوادگی ۲	۱,۰۰۲
۴	۳	نام ۳	نام خانوادگی ۳	۱,۰۰۳
۵	۴	نام ۴	نام خانوادگی ۴	۱,۰۰۴
۶	۵	نام ۵	نام خانوادگی ۵	۱,۰۰۵
۷	۶	نام ۶	نام خانوادگی ۶	۱,۰۰۶
			جمع صفحه	۷,۰۲۱
			جمع کل	۷,۰۲۱
			مالیات	۱۵۴
			عوارض	۱۲۶
			جمع کل	۷,۳۰۱
			قابل پرداخت	هفت هزار و سیصد و یک ریال

نکته‌ای که در اینجا مدنظر است، دسترسی به عدد جمع آخر گزارش و سپس بر اساس آن، ساخت دو ستون اضافی ذیل جدول اصلی گزارش است که موارد مالیات، عوارض، جمع کل و مبلغ به حروف را نسبت به مثال‌های قبلی، اضافه‌تر دارد.

در ادامه کدهای کامل این مثال را مشاهده می‌کنید. همچنین این کد و کلاس‌های وابسته به آن مانند User و TransparentTemplate به سورس‌های کتابخانه PdfReport [نیز اضافه شده‌اند](#).

```
using System;
using System.Collections.Generic;
using iTextSharp.text;
using iTextSharp.text.pdf;
using PdfReportSamples.Models;
using PdfReportSamples.Templates;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;
using PdfRpt.FluentInterface;

namespace PdfReportSamples.Tax
{
    public class TaxPdfReport
    {
        public IPdfReportData CreatePdfReport()
        {
            return new PdfReport().DocumentPreferences(doc =>
            {
                doc.RunDirection(PdfRunDirection.RightToLeft);
                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(AppPath.ApplicationPath + "\\fonts\\irsans.ttf",
                    Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.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("گزارش جدید ما");
        });
    })
    .MainTableTemplate(template =>
    {
        template.CustomTemplate(new TransparentTemplate());
    })
    .MainTablePreferences(table =>
    {
        table.ColumnsWidthsType(TableColumnWidthType.Relative);
    })
    .MainTableDataSource(dataSource =>
    {
        var listOfRows = new List<User>();
        for (int i = 0; i < 7; i++)
        {
            listOfRows.Add(new User { Id = i, LastName = "نام خانوادگی " + i, Name = "نام " + i,
Balance = i + 1000 });
        }
        dataSource.StronglyTypedList<User>(listOfRows);
    })
    .MainTableSummarySettings(summarySettings =>
    {
        summarySettings.OverallSummarySettings("جمع کل");
        summarySettings.PreviousPageSummarySettings("نقل از صفحه قبل");
        summarySettings.PageSummarySettings("جمع صفحه");
    })
    .MainTableColumns(columns =>
    {
        columns.AddColumn(column =>
        {
            column.PropertyName("rowNo");
            column.IsRowNumber(true);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(0);
            column.Width(1);
            column.HeaderCell("ردیف");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<User>(x => x.Id);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(1);
            column.Width(2);
            column.HeaderCell("شماره");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<User>(x => x.Name);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.IsVisible(true);
            column.Order(2);
            column.Width(3);
            column.HeaderCell("نام");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName<User>(x => x.LastName);

```

```

        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(3);
        column.Width(3);
        column.HeaderCell("نام خانوادگی");
    });

    columns.AddColumn(column =>
    {
        column.PropertyName<User>(x => x.Balance);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.IsVisible(true);
        column.Order(4);
        column.Width(2);
        column.HeaderCell("موجودی");
        column.ColumnItemsTemplate(template =>
        {
            template.TextBlock();
            template.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
        column.AggregateFunction(aggregateFunction =>
        {
            aggregateFunction.NumericAggregateFunction(AggregateFunction.Sum);
            aggregateFunction.DisplayFormatFormula(obj => obj == null ? string.Empty :
string.Format("{0:n0}", obj));
        });
    });

    });
    .MainTableEvents(events =>
    {
        events.DataSourceIsEmpty(message: "There is no data available to display.");

        events.MainTableAdded(args =>
        {
            var balanceData = args.LastOverallAggregateValueOf<User>(u => u.Balance);
            var balance = double.Parse(balanceData,
System.Globalization.NumberStyles.AllowThousands);

            var others = Math.Round(balance * 1.8 / 100);
            var tax = Math.Round(balance * 2.2 / 100);
            var total = balance + tax + others;

            var taxTable = new PdfPTable(args.Table); // Create a clone of the MainTable's
structure

            taxTable.AddSimpleRow(
                null /* null = empty cell */, null, null,
                (data, cellProperties) =>
                {
                    data.Value = "مالیات";
                    cellProperties.PdfFont = args.PdfFont;
                    cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
                },
                (data, cellProperties) =>
                {
                    data.Value = string.Format("{0:n0}", tax);
                    cellProperties.PdfFont = args.PdfFont;
                    cellProperties.BorderColor = BaseColor.LIGHT_GRAY;
                    cellProperties.ShowBorder = true;
                }
            );

            taxTable.AddSimpleRow(
                null, null, null,
                (data, cellProperties) =>
                {
                    data.Value = "عوارض";
                    cellProperties.PdfFont = args.PdfFont;
                    cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
                },
                (data, cellProperties) =>
                {
                    data.Value = string.Format("{0:n0}", others);
                    cellProperties.PdfFont = args.PdfFont;
                    cellProperties.BorderColor = BaseColor.LIGHT_GRAY;
                    cellProperties.ShowBorder = true;
                }
            );

            taxTable.AddSimpleRow(
                null, null, null,

```

```

        (data, cellProperties) =>
        {
            data.Value = "جمع کل";
            cellProperties.PdfFont = args.PdfFont;
            cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
        },
        (data, cellProperties) =>
        {
            data.Value = string.Format("{0:n0}", total);
            cellProperties.PdfFont = args.PdfFont;
            cellProperties.BorderColor = BaseColor.LIGHT_GRAY;
            cellProperties.ShowBorder = true;
        });

taxTable.AddSimpleRow(
    null, null, null,
    (data, cellProperties) =>
    {
        data.Value = "قابل پرداخت";
        cellProperties.PdfFont = args.PdfFont;
        cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
    },
    (data, cellProperties) =>
    {
        data.Value = total.NumberToText(Language.Persian) + " ریال";
        cellProperties.PdfFont = args.PdfFont;
        cellProperties.BorderColor = BaseColor.LIGHT_GRAY;
        cellProperties.ShowBorder = true;
        cellProperties.PdfFontStyle = DocumentFontStyle.Bold;
    });

args.PdfDoc.Add(taxTable);
});
})
.Export(export =>
{
    export.ToExcel();
})
.Generate(data => data.AsPdfFile(AppPath.ApplicationPath + "\\Pdf\\TaxReportSample.pdf"));
}
}
}

```

توضیحات:

تنها تفاوت این مثال با مثال‌های قبلی، کدهای مرتبط با متد `events.MainTableAdded` می‌باشند. توسط متد `args.LastOverallAggregateValueOf` می‌توان به مقدار نهایی متد تجمعی تعریف شده برای یک ستون خاص دسترسی یافت:

```

var balanceData = args.LastOverallAggregateValueOf<User>(u => u.Balance);
var balance = double.Parse(balanceData, System.Globalization.NumberStyles.AllowThousands);

```

سپس بر این اساس، امکان محاسبه مالیات و عوارض میسر می‌شود:

```

var others = Math.Round(balance * 1.8 / 100);
var tax = Math.Round(balance * 2.2 / 100);
var total = balance + tax + others;

```

در ادامه نیاز داریم تا یک جدول جدید را ذیل جدول اصلی ایجاد کنیم. نکته مهم این جدول جدید، هماهنگی عرض ستون‌های آن با ستون‌های جدول اصلی است. به همین منظور می‌توان از خاصیت `args.Table` جهت دسترسی به خواص جدول اصلی استفاده کرد و جدول جدیدی را ایجاد نمود:

```
var taxTable = new PdfPTable(args.Table);
```

از اینجا به بعد دیگر به عهده خودتان است. می‌توانید از دانش `iTextSharp` خود استفاده کرده و ردیف‌های این جدول جدید را پر کنید. یا اینکه می‌توانید از متد کمکی توکار `AddSimpleRow` به نحو زیر استفاده نمایید:

```
taxTable.AddSimpleRow(
```

```

null /* null = empty cell */, null, null,
(data, cellProperties) =>
{
    data.Value = "مالیات";
    cellProperties.PdfFont = args.PdfFont;
    cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
},
(data, cellProperties) =>
{
    data.Value = string.Format("{0:n0}", tax);
    cellProperties.PdfFont = args.PdfFont;
    cellProperties.BorderColor = BaseColor.LIGHT_GRAY;
    cellProperties.ShowBorder = true;
});

```

با توجه به اینکه قصد نداریم در سه ستون اول این جدول جدید، عنصری را نمایش دهیم، آن‌ها را با null مقدار دهی کرده و سپس ستون برچسب و ستون مقدار را اضافه می‌کنیم (آرگومان‌های این متد به صورت params تعریف شده‌اند. بنابراین هر تعداد ستون که نیاز باشد قابل تعریف است).

با مقدار دهی data، مقدار مورد نظر در آن سلول ثبت می‌گردد. با مقدار دهی خواص cellProperties، نوع قلم، جهت قرارگیری و سایر تنظیماتی را که ملاحظه می‌کنید، می‌توان اعمال کرد.

و در آخر لازم است که این جدول جدید را به شیء Document اضافه کنیم تا نمایش داده شود:

```
args.PdfDoc.Add(taxTable);
```

یک نکته:

متد NumberToText جزئی از کتابخانه PdfReport (تعریف شده در فضای نام PdfRpt.Core.Helper) است و برای نمایش رقم به حروف می‌تواند مورد استفاده قرار گیرد:

```
total.NumberToText(Language.Persian)
```

نظرات خوانندگان

نویسنده: mrm

تاریخ: ۲۲:۴۷ ۱۳۹۱/۰۷/۳۰

آقای نصیری با سلام

میخواستم ببینم آیا میتونم از pdfreport برای تولید برگه امتحان تستی استخراج شده از بانک سوال استفاده کنم و آیا شما توصیه میکنید یا نه؟
برگه شامل یک سربرگ و حاشیه هم باید باشه و سوالات تستی با گزینه هاشون توی اون باید به شکل مرتبی برای چاپ حاضر بشن
خیلی ممنون

نویسنده:

وحید نصیری

تاریخ: ۲۳:۱۰ ۱۳۹۱/۰۷/۳۰

سلام؛

بله. ولی فکر نمیکنم این بله به درد شما بخورد. به همین جهت [در این قسمت](#) مخصوص سایت، شمای بانک (ساختار بانک)، به علاوه شکل نهایی مدنظر را به صورت یک بازخورد جدید ارسال کنید. من کدهای گزارش رو براتون به شکل یک مثال پیوست می‌کنم.

نویسنده:

م دانا

تاریخ: ۱۸:۲۲ ۱۳۹۱/۱۱/۱۴

با سلام و تشکر از زحمات شما در ارتباط با کتابخانه PDFReport.
متأسفانه متد AddSimpleRow را در مثال فوق نمی‌شناسد! چه دلیلی می‌تواند داشته باشد؟

نویسنده:

وحید نصیری

تاریخ: ۱۸:۳۹ ۱۳۹۱/۱۱/۱۴

آخرین تغییرات [این مثال](#) رو در اینجا می‌تونید مشاهده کنید. فقط در در آن PdfPTable تبدیل به PdfGrid شده.

نویسنده:

پویا

تاریخ: ۱۵:۳۹ ۱۳۹۲/۰۲/۲۴

با عرض سلام.

فرض می‌کنیم که هدر ما از سه قسمت هدر و جدول (همان جدولی که برای فاکتور در اینجا در نظر گرفته شده است) و فوتر تشکیل شده است حال چگونه می‌توانم فونت جدول را تعیین کنم یعنی این که سایز اون را تغییر دهم یا اینکه خود فونت را تغییر دهم. ممنون

نویسنده:

وحید نصیری

تاریخ: ۱۶:۵۲ ۱۳۹۲/۰۲/۲۴

- فونتی که در قسمت DefaultFonts تعریف می‌شود (که در اینجا متدهای تنظیم اندازه و رنگ نیز وجود دارند) در حقیقت فونت اصلی جدول گزارش است. اگر هدر و فوتر، فونتی رو معرفی نکنه، این فونت برای آن‌ها هم استفاده خواهد شد (در حالت‌های DefaultHeader و DefaultFooter). اما می‌شود هدر و فوتر سفارشی هم تعریف کرد؛ با هر نوع طراحی و هر نوع فونت دلخواهی که صلاح می‌دونید (مراجعه کنید به مثال [InlineProviders مجموعه مثال‌ها](#)، که در آن خاصیت PdfFont مستقل هم وجود دارد).
- به علاوه در اینجا حتی می‌شود بنابر شرایط و مقادیر سلول‌ها، فونت و رنگ خاصی را به مقادیر یک سلول اعمال کرد. نمونه‌اش

در مثال [CustomCellTemplate](#) وجود دارد.

- در این حالت‌های خاص باید IPdfFont را پیاده سازی کنید. یک نمونه پروایدر ساده ساز در اینجا به نام کلاس GenericFontProvider برای اینکار تدارک دیده شده. مثالی در این مورد: [InjectCustomRows](#) جهت مقدار دهی .CellBasicProperties

اگر به بانک اطلاعاتی مثال‌های [همراه سورس‌های](#) PdfReport در مسیر Bin\Data\blogs.sqlite مراجعه کنید، دو جدول والدین و فرزندان هم در آن وجود دارند:

tblKids	
Id	
ParentId	
BirthDate	
Name	
tblParents	
Id	
BirthDate	
Name	
LastName	

بر این اساس قصد داریم رابطه یک به چند فوق را گروه بندی شده نمایش دهیم:



Family rpt

Name: Parent1
 Last Name: LM1
 Birth Date: 2002/10/05 12:20:08 ب.ظ

#	Child Name	BirthDate
1	Kid26	1986/10/05 12:20:08 ب.ظ
2	Kid36	1976/10/05 12:20:08 ب.ظ
3	Kid40	1972/10/05 12:20:08 ب.ظ
4	Kid69	1943/10/05 12:20:08 ب.ظ
5	Kid73	1939/10/05 12:20:08 ب.ظ
6	Kid77	1935/10/05 12:20:08 ب.ظ
7	Kid8	2004/10/05 12:20:08 ب.ظ
8	Kid9	2003/10/05 12:20:08 ب.ظ

Name: Parent10
 Last Name: LM10
 Birth Date: 1912/10/05 12:20:08 ب.ظ

#	Child Name	BirthDate
1	Kid20	1992/10/05 12:20:08 ب.ظ
2	Kid31	1981/10/05 12:20:08 ب.ظ
3	Kid34	1978/10/05 12:20:08 ب.ظ
4	Kid42	1970/10/05 12:20:08 ب.ظ

(البته این اعداد و اطلاعات، به صورت اتفاقی تولید شده‌اند و الزامی ندارد که والد متولد 2002 هنوز والد شده باشد؛ یا اینکه فرزندی متولد 2003 داشته باشد!)

بنابراین صورت مساله ما به این ترتیب خواهد بود:

بر اساس اطلاعات دو جدول والدین و فرزندان فوق، اطلاعات نهایی را در جداول مجزایی بر اساس والدین و فرزندان آن‌ها گروه بندی نمائید.

سورس کامل این مثال را در ادامه مشاهده می‌کنید:

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

namespace PdfReportSamples.MasterDetails
{
    public class MasterDetailsPdfReport
    {
        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\\arial.ttf",
                    Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.ttf");
    })
    .PagesHeader(header =>
    {
        header.CustomHeader(new MasterDetailsHeaders { PdfRptFont = header.PdfFont });
    })
    .PagesFooter(footer =>
    {
        footer.DefaultFooter(DateTime.Now.ToString("MM/dd/yyyy"));
    })
    .MainTableTemplate(t => t.BasicTemplate(BasicTemplate.SilverTemplate))
    .MainTablePreferences(table =>
    {
        table.ColumnsWidthsType(TableColumnWidthType.Relative);
        table.GroupsPreferences(new GroupsPreferences
        {
            GroupType = GroupType.HideGroupingColumns,
            RepeatHeaderRowPerGroup = true,
            ShowOneGroupPerPage = false,
            SpacingBeforeAllGroupsSummary = 5f,
            NewGroupAvailableSpacingThreshold = 170
        });
    })
    .MainTableDataSource(dataSource =>
    {
        dataSource.GenericDataReader(
            providerName: "System.Data.SQLite",
            connectionString: "Data Source=" + AppPath.ApplicationPath + "\\data\\blogs.sqlite",
            sql: @"select
                tblParents.BirthDate as ParentBirthDate,
                tblParents.Name as ParentName,
                tblParents.LastName as ParentLastName,
                tblKids.Name as KidName,
                tblKids.BirthDate as KidBirthDate
            from tblParents
                left outer join tblKids
                    on tblKids.ParentId = tblParents.Id
            order by
                tblParents.Name,
                tblParents.LastName,
                tblKids.Name"
        );
    })
    .MainTableColumns(columns =>
    {
        columns.AddColumn(column =>
        {
            column.PropertyName("rowNo");
            column.IsRowNumber(true);
            column.CellsHorizontalAlignment(HorizontalAlignment.Left);
            column.IsVisible(true);
            column.Order(0);
            column.Width(1);
            column.HeaderCell("#");
        });

        columns.AddColumn(column =>
        {
            column.PropertyName("ParentBirthDate");
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.Order(1);
            column.Width(2);
            column.HeaderCell("ParentBirthDate");
            column.Group(true,
                (val1, val2) =>
                {
                    var date1 = (DateTime)val1;
                    var date2 = (DateTime)val2;
                    return date1.Year == date2.Year && date1.Month == date2.Month && date1.Day ==
date2.Day;
                });
        });

        columns.AddColumn(column =>
        {
            column.PropertyName("ParentName");
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.Order(2);
            column.Width(2);

```

```

        column.HeaderCell("ParentName");
        column.Group(true,
            (val1, val2) =>
            {
                return val1.ToString() == val2.ToString();
            });
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("ParentLastName");
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.Order(3);
        column.Width(2);
        column.HeaderCell("ParentLastName");
        column.Group(true,
            (val1, val2) =>
            {
                return val1.ToString() == val2.ToString();
            });
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("KidName");
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.Order(4);
        column.Width(2);
        column.HeaderCell("Child Name");
        column.IsVisible(true);
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("KidBirthDate");
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.Order(5);
        column.Width(2);
        column.HeaderCell("BirthDate");
        column.IsVisible(true);
    });
    })
    .MainTableEvents(events =>
    {
        events.DataSourceIsEmpty(message: "There is no data available to display.");
    })
    .Export(e => e.ToExcel())
    .Generate(data => data.AsPdfFile(AppPath.ApplicationPath +
        "\\Pdf\\RptMasterDetailsSample.pdf"));
    }
}

```

به همراه سر ستون‌های مجزای هر گروه و صفحه:

```

using System.Collections.Generic;
using iTextSharp.text;
using iTextSharp.text.pdf;
using PdfRpt.ColumnsItemsTemplates;
using PdfRpt.Core.Contracts;
using PdfRpt.Core.Helper;

namespace PdfReportSamples.MasterDetails
{
    public class MasterDetailsHeaders : IPageHeader
    {
        public IPdfFont PdfRptFont { set; get; }

        public PdfPTable RenderingGroupHeader(Document pdfDoc, PdfWriter pdfWriter, IList<CellData>
            newGroupInfo, IList<SummaryCellData> summaryData)
        {
            var parentName = newGroupInfo.GetSafeStringValueOf("ParentName");
            var parentLastName = newGroupInfo.GetSafeStringValueOf("ParentLastName");
            var parentBirthDate = newGroupInfo.GetSafeStringValueOf("ParentBirthDate");

            var table = new PdfPTable(relativeWidths: new[] { 1f, 5f }) { WidthPercentage = 100 };
            table.AddSimpleRow(
                (cellData, cellProperties) =>

```

```

        {
            cellData.Value = "Name:";
            cellProperties.PdfFont = PdfRptFont;
            cellProperties.PdfFontStyle = DocumentFontStyle.Bold;
            cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
        },
        (cellData, cellProperties) =>
        {
            cellData.Value = parentName;
            cellProperties.PdfFont = PdfRptFont;
            cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
        });
    table.AddSimpleRow(
        (cellData, cellProperties) =>
        {
            cellData.Value = "Last Name:";
            cellProperties.PdfFont = PdfRptFont;
            cellProperties.PdfFontStyle = DocumentFontStyle.Bold;
            cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
        },
        (cellData, cellProperties) =>
        {
            cellData.Value = parentLastName;
            cellProperties.PdfFont = PdfRptFont;
            cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
        });
    table.AddSimpleRow(
        (cellData, cellProperties) =>
        {
            cellData.Value = "Birth Date:";
            cellProperties.PdfFont = PdfRptFont;
            cellProperties.PdfFontStyle = DocumentFontStyle.Bold;
            cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
        },
        (cellData, cellProperties) =>
        {
            cellData.Value = parentBirthDate;
            cellProperties.PdfFont = PdfRptFont;
            cellProperties.HorizontalAlignment = HorizontalAlignment.Left;
        });
    return table.AddBorderToTable(borderColor: BaseColor.LIGHT_GRAY, spacingBefore: 5f);
}

public PdfPTable RenderingReportHeader(Document pdfDoc, PdfWriter pdfWriter,
    IList<SummaryCellData> summaryData)
{
    var table = new PdfPTable(numColumns: 1) { WidthPercentage = 100 };
    table.AddSimpleRow(
        (cellData, cellProperties) =>
        {
            cellData.CellTemplate = new ImageFilePathField();
            cellData.Value = AppPath.ApplicationPath + "\\Images\\01.png";
            cellProperties.HorizontalAlignment = HorizontalAlignment.Center;
        });
    table.AddSimpleRow(
        (cellData, cellProperties) =>
        {
            cellData.Value = "Family rpt";
            cellProperties.PdfFont = PdfRptFont;
            cellProperties.PdfFontStyle = DocumentFontStyle.Bold;
            cellProperties.HorizontalAlignment = HorizontalAlignment.Center;
        });
    return table.AddBorderToTable();
}
}
}

```

توضیحات:

- منبع داده مورد استفاده در اینجا از نوع GenericDataReader است؛ جهت خواندن رکوردهای بانک اطلاعاتی SQLite ذکر شده در ابتدای بحث. (دو مثال دیگر هم به پوشه [مثالهای سورسهای PdfReport](#) اضافه شده‌اند به نام‌های Grouping و WrapGroupsInColumns که به همین موضوع گروه بندی می‌پردازند؛ البته با استفاده از StronglyTypedList ها. ولی درکل مفاهیم و اصول آنها یکی است.)

```

select
    tblParents.BirthDate as ParentBirthDate,
    tblParents.Name as ParentName,
    tblParents.LastName as ParentLastName,

```

```
tblKids.Name as KidName,
tblKids.BirthDate as KidBirthDate
from tblParents
    left outer join tblKids
        on tblKids.ParentId = tblParents.Id
    order by
        tblParents.Name,
        tblParents.LastName,
        tblKids.Name
```

در کوئری فوق (و کلا گروه بندی اطلاعات) دو نکته حائز اهمیت است:

الف) چون قرار است اطلاعات بر اساس مشخصات والدین و فرزندان آنها گروه بندی شود، نیاز است حتما `order by` و مرتب سازی رکوردها قید گردد.

ب) در PdfReport نمی‌توانید در خواص معرفی شده جهت تعریف ستون‌ها، از نام‌های تکراری استفاده کنید. برای رفع این مشکل استفاده از Alias پیشنهاد می‌شود؛ مانند:

```
tblParents.Name as ParentName,
tblKids.Name as KidName,
```

- مشخص سازی خاصیت و ستونی که قرار است در گروه بندی شرکت کند بسیار ساده است:

```
column.Group(true,
    (val1, val2) =>
    {
        return val1.ToString() == val2.ToString();
    });
```

در اینجا به کمک متد `Group`، قابلیت گروه بندی بر روی این ستون فعال شده و سپس باید فرمولی را جهت مشخص سازی حد و مرز گروه مشخص کنیم. برای مثال در اینجا اگر مقادیر ردیف جاری (`val2`) و ردیف قبلی (`val1`) یکسان نبودند، یعنی گروه خاتمه یافته و گروه جدیدی شروع می‌شود (به همین جهت عنوان شد که مرتب سازی اطلاعات ضروری است).

- تنظیم دیگری را که در اینجا می‌توان ذکر کرد، مورد ذیل است:

```
table.GroupsPreferences(new GroupsPreferences
{
    GroupType = GroupType.HideGroupingColumns,
    RepeatHeaderRowPerGroup = true,
    ShowOneGroupPerPage = false,
    SpacingBeforeAllGroupsSummary = 5f,
    NewGroupAvailableSpacingThreshold = 170
});
```

به این ترتیب می‌توان مشخص کرد که آیا باید ستون‌های دخیل در گروه بندی، در گزارش نمایش داده شوند یا خیر (`GroupType.HideGroupingColumns`)، آیا سر ستون هر جدول، به ازای هر گروه باید تکرار شود؟ (`RepeatHeaderRowPerGroup`)، آیا در هر صفحه یک گروه نمایش داده شود (`ShowOneGroupPerPage`) یا اینکه گروه‌ها به صورت متوالی در صفحات درج شوند. توسط `SpacingBeforeAllGroupsSummary`، فاصله جمع نهایی تمام گروه‌ها از آخرین گروه نمایش داده شده مشخص می‌شود. به کمک `NewGroupAvailableSpacingThreshold` مشخص می‌کنیم که در چه فاصله‌ای از انتهای صفحه، گروه جدیدی نباید درج شود و این گروه باید به صفحه بعدی منتقل شده و از آنجا شروع شود.

- اگر به تصویر ابتدای مطلب دقت کرده باشید، علاوه بر هدر صفحه، هر گروه نیز یک هدر مجزا دارد. برای طراحی آن باید اینترفیس `IPageHeader` را پیاده سازی کرد که نمونه‌ای از آن را در کلاس `MasterDetailsHeaders` فوق مشاهده می‌کنید.

```
public PdfPTable RenderingGroupHeader(Document pdfDoc, PdfWriter pdfWriter, IList<CellData>
newGroupInfo, IList<SummaryCellData> summaryData)
{
    var parentName = newGroupInfo.GetSafeStringValueOf("ParentName");
    var parentLastName = newGroupInfo.GetSafeStringValueOf("ParentLastName");
```

```
var parentBirthDate = newGroupInfo.GetSafeStringValueOf("ParentBirthDate");  
var table = new PdfPTable(relativeWidths: new[] { 1f, 5f }) { WidthPercentage = 100 };  
table.AddSimpleRow(  
    (cellData, cellProperties) =>  
    {  
        cellData.Value = "Name:";  
        cellProperties.PdfFont = PdfRptFont;  
        cellProperties.PdfFontStyle = DocumentFontStyle.Bold;  
        cellProperties.HorizontalAlignment = HorizontalAlignment.Left;  
    },  
    (cellData, cellProperties) =>  
    {  
        cellData.Value = parentName;  
        cellProperties.PdfFont = PdfRptFont;  
        cellProperties.HorizontalAlignment = HorizontalAlignment.Left;  
    }  
);
```

ساختار آن هم بسیار ساده است. توسط newGroupInfo می‌توان به اطلاعات گروه جدید، دسترسی یافت. برای مثال در اینجا اطلاعات والد گروه جدید در حال تهیه، دریافت شده و سپس در ردیف‌های یک جدول دو ستونه درج می‌شود. در ستون اول آن یک برجسب و در ستون دوم، مقدار دریافتی نمایش داده شده است و همینطور الی آخر برای سایر ردیف‌ها.

نظرات خوانندگان

نویسنده:

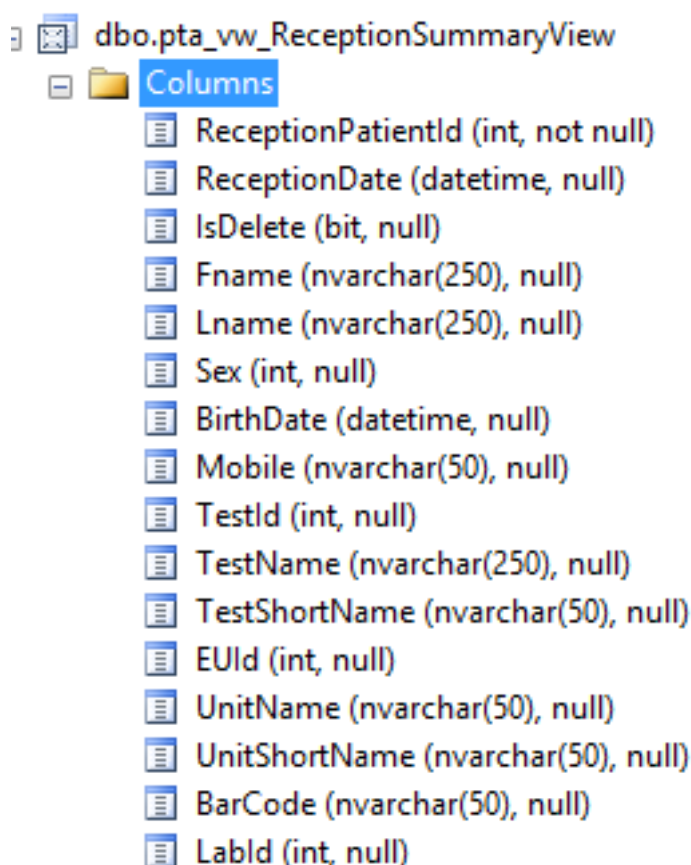
ح. مرتضوی

تاریخ:

۱۸:۲۴ ۱۳۹۱/۱۱/۲۷

سلام.

من یک View در SQL دارم که چند جدول مختلف را با هم Join میزند.



رکوردها باید با فیلد ReceptionPatientId گروه بندی شوند.

کد برنامه:

```
return new PdfReport().DocumentPreferences(doc =>
{
    doc.RunDirection(PdfRunDirection.LeftToRight);
    doc.Orientation(PageOrientation.Portrait);
    doc.PageSize(PdfPageSize.A4);
    doc.DocumentMetadata(new DocumentMetadata
    {
        Author = username,
        Application = "",
        Keywords = "Executive Unit Work List Report",
        Subject = "Executive Unit Work List Report",
        Title = "Executive Unit Work List Report"
    });
});
```

```

    })
    .DefaultFonts(fonts => fonts.Path(Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\tahoma.ttf",
                                Environment.GetEnvironmentVariable("SystemRoot") +
"\\fonts\\verdana.ttf"))
    .PagesFooter(footer => footer.DefaultFooter(DateTime.Now.ToLongPersianDate()))
    .PagesHeader(header =>
    {
        header.CustomHeader(new UnitWorkListHeader { PdfRptFont = header.PdfFont });
    })
    .MainTableTemplate(template => template.BasicTemplate(BasicTemplate.RainyDayTemplate))
    .MainTablePreferences(table =>
    {
        table.ColumnsWidthsType(TableColumnWidthType.Relative);
        table.GroupsPreferences(new GroupsPreferences
        {
            GroupType = GroupType.HideGroupingColumns,
            RepeatHeaderRowPerGroup = true,
            ShowOneGroupPerPage = false,
            SpacingBeforeAllGroupsSummary = 5f,
            NewGroupAvailableSpacingThreshold = 170
        });
    })
    .MainTableDataSource(dataSource =>
        dataSource.DataTable(new ReceptionPatientBl(context).
            GetReceptionSummaryView(_rcpIds).
            ToDataSet(false).Tables[0]))
    .MainTableColumns(columns =>
    {
        columns.AddColumn(column =>
        {
            column.PropertyName("rowNo");
            column.IsRowNumber(true);
            column.CellsHorizontalAlignment(HorizontalAlignment.Left);
            column.IsVisible(true);
            column.Order(0);
            column.Width(1);
            column.HeaderCell("#");
        });
        columns.AddColumn(column =>
        {
            column.PropertyName("ReceptionPatientId");
            column.IsRowNumber(false);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.Order(3);
            column.Width(2);
            column.HeaderCell("ReceptionPatientId");
            column.Group(true,
                (val1, val2) =>
                {
                    return val1 == val2;
                });
        });
        columns.AddColumn(column =>
        {
            column.PropertyName("ReceptionDate");
            column.IsRowNumber(false);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.Order(3);
            column.Width(2);
            column.HeaderCell("ReceptionDate");
            column.Group(true,
                (val1, val2) =>
                {
                    return val1 == val2;
                });
        });
        columns.AddColumn(column =>
        {
            column.PropertyName("IsDelete");
            column.IsRowNumber(false);
            column.CellsHorizontalAlignment(HorizontalAlignment.Center);
            column.Order(3);
            column.Width(2);
            column.HeaderCell("IsDelete");
            column.Group(true,
                (val1, val2) =>
                {
                    return val1 == val2;
                });
        });
    });

```

```

    });
});

columns.AddColumn(column =>
{
    column.PropertyName("Fname");
    column.IsRowNumber(false);
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.Order(3);
    column.Width(2);
    column.HeaderCell("Fname");
    column.Group(true,
        (val1, val2) =>
        {
            return val1 == val2;
        });
});

columns.AddColumn(column =>
{
    column.PropertyName("Lname");
    column.IsRowNumber(false);
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.Order(3);
    column.Width(2);
    column.HeaderCell("Lname");
    column.Group(true,
        (val1, val2) =>
        {
            return val1 == val2;
        });
});

columns.AddColumn(column =>
{
    column.PropertyName("Sex");
    column.IsRowNumber(false);
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.Order(3);
    column.Width(2);
    column.HeaderCell("Sex");
    column.Group(true,
        (val1, val2) =>
        {
            return val1 == val2;
        });
});

columns.AddColumn(column =>
{
    column.PropertyName("BirthDate");
    column.IsRowNumber(false);
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.Order(3);
    column.Width(2);
    column.HeaderCell("BirthDate");
    column.Group(true,
        (val1, val2) =>
        {
            return val1 == val2;
        });
});

columns.AddColumn(column =>
{
    column.PropertyName("Mobile");
    column.IsRowNumber(false);
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.Order(3);
    column.Width(2);
    column.HeaderCell("Mobile");
    column.Group(true,
        (val1, val2) =>
        {
            return val1 == val2;
        });
});

columns.AddColumn(column =>
{
    column.PropertyName("LabId");

```



```

        column.IsRowNumber(false);
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.Order(3);
        column.Width(2);
        column.HeaderCell("LabId");
        column.Group(true,
            (val1, val2) =>
            {
                return val1.ToString() == val2.ToString();
            }
        ));
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("TestName");
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.Order(4);
        column.Width(2);
        column.HeaderCell("نام تست");
        column.IsVisible(true);
    });

    columns.AddColumn(column =>
    {
        column.PropertyName("TestShortName");
        column.CellsHorizontalAlignment(HorizontalAlignment.Center);
        column.Order(5);
        column.Width(2);
        column.HeaderCell("نام اختصاری تست");
        column.IsVisible(true);
    });
    })
    .MainTableEvents(events => events.DataSourceIsEmpty(message: "There is no data available to
display."))
    // .Export(export =>
    // {
    //     export.ToExcel();
    // })
    .Generate(data =>
    {
        var fileName = "ExecutiveWorkListReport.pdf";
        fileName = HttpUtility.UrlEncode(fileName, Encoding.UTF8);
        data.FlushInBrowser(fileName);
    });
}

```

اما خروجی به صورت زیر است:

گزارش لیست کاری واحدهای اجرایی

Reception Id: 1531
 Mobile: 09123687629
 Unit Name: نامشخص

#	نام تست	نام اختصاری تست
1	T3	T3

Reception Id: 1531
 Mobile: 09123687629
 Unit Name: نامشخص

#	نام تست	نام اختصاری تست
1	T4	T4

Reception Id: 1531
 Mobile: 09123687629
 Unit Name: Human

#	نام تست	نام اختصاری تست
1	Testosterone	Testo

همانطور که مشاهده می‌کنید بدون توجه به فیلد ID سه مرتبه این گروه تکرار شده است.

ممکن است راهنمایی فرمایید مشکل کار کجاست؟

نویسنده: وحید نصیری
 تاریخ: ۱۸:۳۹ ۱۳۹۱/۱۱/۲۷

- اگر روی یک فیلد قرار است گروه بندی شود، فقط یکبار column.Group را تعریف کنید. مابقی تعاریف column.Group باید حذف شوند. بنابراین اگر پایه گروه بندی، فیلد ReceptionPatientId است، تعاریف مرتبط با آن را نگه دارید و سطر column.Group مابقی رو حذف کنید. همچنین فرض هم بر این خواهد بود که اطلاعات شما بر اساس فیلدهای شرکت کننده در گروه بندی پیشتر sort شده‌اند.

```
column.Group((val1, val2) =>
{
    return (int)val1 == (int)val2;
});
```

- ضمناً الزامی نیست یک view رو تبدیل به datatable کنید برای استفاده در اینجا. [دیتاسورس کار با sql server](#) هم وجود دارد برای کارآیی و سرعت بیشتر.
 - لطفاً برای سؤالات بعدی [از قسمت پرسش و پاسخ مرتبط](#) با این پروژه در سایت استفاده کنید.

نویسنده: ناصر پورعلی
 تاریخ: ۱۱:۴۶ ۱۳۹۳/۰۲/۱۶

با سلام

در این مثال در صورتی که تنظیم گروه بندی

```
GroupType = GroupType.IncludeGroupingColumns
```

باشد و بخواهیم فیلد تاریخ که بر اساس آن گروه بندی نیز انجام شده، را نشان دهیم و البته به تاریخ فارسی تبدیل کنیم با استفاده از کد زیر :

```
column.ColumnItemsTemplate(template =>
{
    template.TextBlock();
    template.DisplayFormatFormula(obj =>
DateTimeHelper.ToPersianShortDateString((DateTime)obj));
});
```

خطای Specified cast is not valid می گیریم.

علت چیست؟

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۱:۵۸ ۱۳۹۳/۰۲/۱۶

- لطفا برای سؤالات بعدی [از قسمت پرسش و پاسخ مرتبط](#) با این پروژه در سایت استفاده کنید.
+ یعنی تاریخ مدنظر معتبر نیست (obj دریافتی از نوع DateTime نیست). بهتر است از [DateTime.TryParse](#) استفاده کنید تا مشخص شود، اطلاعاتی که با آن کار می کنید، قابل پردازش هستند یا خیر. یک break point روی آن قرار دهید و محتوای آن را بررسی کنید.

نمایش علائم مختلف در گزارشات و تهیه لیست قلم‌های نصب شده در سیستم توسط PdfReport

عنوان:

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۷/۲۶ ۰:۳۴

آدرس: www.dotnettips.info

برچسب‌ها: PdfReport

دو مثال جدید به سورس‌های PdfReport اضافه شده است:

الف) `Samples\PdfReportSamples\ZapfDingbatsSymbols`

تعاریف قلم توکاری به نام Adobe Zapf Dingbats در iTextSharp وجود دارد که جهت نمایش انواع و اقسام علائم در فایل‌های PDF می‌تواند بکار گرفته شود. این قلم توکار توسط قالبی به نام Symbol در PdfReport قابل استفاده است:

```
column.ColumnItemsTemplate(template =>
{
    template.Symbol(data =>
    {
        if (Enum.IsDefined(typeof(AdobeZapfDingbats), data))
        {
            return (AdobeZapfDingbats)data;
        }
        return AdobeZapfDingbats.BallotX;
    });
});
```

در اینجا data مقدار سلول جاری پیش از رندر شدن است. بر این اساس تنها کافی است انتخابی را انجام داده و یکی از مقادیر enum ای به نام AdobeZapfDingbats را بازگردانیم.

دریافت فایل PDF خروجی حاصل:

[ZapfDingbatsSymbols.pdf](#)

ب) `Samples\PdfReportSamples\PersianFontsListToPdf`

در این مثال لیست تمام فونت‌های شروع شده با b که در سیستم نصب شده‌اند، تهیه می‌شود. برای اینکار یک قالب سفارشی سلول به نام `FontsListCellTemplate` تهیه شده است. ساختار آن هم بسیار ساده است. بر اساس اطلاعات ردیف جاری، متن و نام قلم مورد نظر را دریافت کرده و اطلاعات نهایی را نمایش می‌دهد.

دریافت فایل PDF خروجی حاصل:

[FontsListToPdfSample.pdf](#)

پیشنیازها

[تهیه گزارشات Crosstab به کمک LINQ](#)

[تهیه گزارشات Crosstab به کمک LINQ - قسمت دوم](#)

در دنیای واقعی، تهیه گزارشات به سادگی که در اکثر نرم افزارهای گزارش ساز موجود پیش بینی شده است، نیست. در این نوع نرم افزارها، بر اساس یک طراح بصری، تعدادی ستون مشخص، ایجاد شده، منبع داده‌ای به این ستون‌ها متصل و نهایتاً گزارش تهیه می‌شود. اما اگر همین گزارش دارای تعداد ستون‌های متغیری باشد، اغلب این برنامه‌ها ناکارآمد خواهند بود. برای مثال لیست حضور و غیاب دانش آموزان را در نظر بگیرید.

اگر معلمی بخواهد سه روز در هفته را گزارش بگیرد، گزارش نیاز به سه ستون خواهد داشت. اگر 20 روز قبل مد نظر باشد، 20 ستون و همینطور الی آخر.

یا نمونه‌ی دیگری از این دست، گزارش حضور و غیاب پرسنل است. در اینجا بر اساس تعداد باری که شخص کارت می‌زند، ورود و خروج او محاسبه می‌شود. این مورد هم تعداد ثابتی نیست. ممکن است یک نفر 8 بار در طول روز کارت بزند، یک نفر فقط دو بار. به علاوه جمع ساعات هم اینجا دیگر عددی نیست و نیاز به فرمول خاصی دارد.


روش حل این نوع مسایل را در PdfReport [در مثال‌های جدید](#) ذیل می‌توانید مشاهده کنید:
[DynamicCrosstab](#) (یک گزارش Crosstab با تعداد ستون متغیر)



Students Rpt.

#	Id	Name	Day 14	Day 15	Day 16	Day 17	Day 18	IsPresent	IsNotPresent
1	1	Student 1	✓	✓	✗	✓	✓	4	1
2	2	Student 2	✓	✓	✗	✓	✓	4	1
3	3	Student 3	✗	✗	✓	✗	✗	1	4
4	4	Student 4	✗	✗	✗	✓	✗	1	4
5	5	Student 5	✗	✓	✗	✗	✗	1	4
6	6	Student 6	✗	✗	✗	✓	✓	2	3
7	7	Student 7	✓	✗	✗	✓	✗	2	3
8	8	Student 8	✓	✓	✗	✓	✗	3	2
9	9	Student 9	✗	✓	✗	✗	✗	1	4
10	10	Student 10	✓	✓	✓	✓	✓	5	0
11	11	Student 11	✓	✗	✗	✗	✓	2	3
12	12	Student 12	✗	✓	✓	✓	✓	4	1
13	13	Student 13	✓	✗	✗	✓	✗	2	3
14	14	Student 14	✓	✗	✓	✗	✗	2	3
15	15	Student 15	✓	✗	✗	✓	✗	2	3
16	16	Student 16	✗	✗	✓	✓	✓	3	2
17	17	Student 17	✓	✓	✗	✗	✓	3	2
18	18	Student 18	✓	✓	✗	✓	✗	3	2
19	19	Student 19	✗	✗	✓	✓	✗	2	3
20	20	Student 20	✗	✗	✗	✗	✗	0	5


(2) [WorkedHours](#) (یک گزارش Crosstab با تعداد ستون متغیر به علاوه تابع تجمعی سفارشی محاسبه جمع ساعات اشخاص)



Worked Hours Rpt.

#	Id	Name	Date	In 1	Out 1	In 2	Out 2	In 3	Out 3	WorkedHours
1	2	Emp 2	12/05/2011	07:30	09:10	10:52	16:20			07:08
2	1	Emp 1	12/05/2011	08:09	10:10	11:02	14:20	16:30	18:09	06:58
3	2	Emp 2	12/06/2011	07:35	07:55	08:55	18:20			09:45
4	1	Emp 1	12/06/2011	08:10	17:10					09:00
Page Summary										32:51
Summary										32:51

(3) [ExtraHeadingCells](#) (یک گزارش Crosstab به همراه ردیف‌های header اضافی و نحوه تعریف آن)



Our new Rpt

Contacts List							
#	Person			Phones			
	Id	Name	Last Name	Home	Office	Cell	Fax
1	1	John	Doe	(305) 555-1111	(305) 555-2222	(305) 555-3333	
2	2	Jane	Doe	(305) 555-1111	(305) 555-4444	(305) 555-5555	
3	3	Jerome	Doe	(305) 555-6666	(305) 555-2222	(305) 555-7777	
4	4	Joel	Smith			(305) 555-7777	(305) 555-6666

(4) [ExpensesCrosstab](#) (یک گزارش Crosstab کلاسیک)



Our new rpt.

#	Year	Month	Computer Department	Math Department	Physics Department	Total
1	2011	11	100	200	150	450
2	2011	10	75	150	130	355
3	2011	9	90	95	100	285
Page Summary			265	445	380	1,090
Summary			265	445	380	1,090

(5) [PdfA](#) (یک گزارش Crosstab که به صورت استاندارد PdfA تهیه شده است. PdfA حالت خاصی از استاندارد PDF است که برای مستند سازی عموماً مورد استفاده قرار می‌گیرد. رمزنگاری اطلاعات در آن ممنوع است. تصاویر بکارگرفته شده نباید شفاف باشند. قلم‌های مورد استفاده حتماً باید در فایل مدفون شوند و مواردی از این دست)

You are viewing this document in **PDF/A mode.**

#	Sales Person	Corolla		Camry		Prius		Total	
		SalePrice	Count	SalePrice	Count	SalePrice	Count	SalePrice	Count
1	Chris	6,500	2	7,000	2	2,000	1	15,500	5
2	Brian	2,100	2	4,000	1	2,000	1	8,100	4
3	Jason	1,000	1	1,100	1	3,000	2	5,100	4
Page Summary		9,600	5	12,100	4	7,000	4	28,700	13
Grand Total		9,600	5	12,100	4	7,000	4	28,700	13

نظرات خوانندگان

نویسنده: مهران

تاریخ: ۱۴:۱۷ ۱۳۹۱/۱۰/۱۲

سلام

فوق العاده حرفه ای و دارای انعطاف پذیری بالایی است!

من قبلا با telerik reporting کار کردم که بیشتر به با ویزارد کار می‌کنه، ولی خوب این روش code first شاید اولش یه کم شلوغ و پیچیده به نظر بیاد ولی ابزار بسیار کارآمدیست.

یک سوال:

DisplayName را چگونه روی ستونهای گزارشات crosstab اعمال نماییم؟

نویسنده: وحید نصیری

تاریخ: ۱۴:۵۵ ۱۳۹۱/۱۰/۱۲

دو نوع crosstab داریم. در حالت معمول مانند [ExpensesCrosstab](#) که ستونهای خروجی مشخص هستند می‌تونید از همان متد column.PropertyName استفاده کنید و به همراه سایر تنظیمات دیگر ستون. در حالت پیشرفته crosstab پویا که تعداد ستونهای خروجی مشخص نیستند و هربار می‌تواند متغیر باشد مثل [گزارش ساعات کاری](#)، یک روش این است که نام خاصیت‌ها را کمی واضح‌تر انتخاب کرد. فارسی هم مجاز است:

```
list.Pivot(x =>
    new
    {
        = نام ستون
```

نویسنده: مهران

تاریخ: ۱۲:۷ ۱۳۹۱/۱۰/۱۷

سلام

نام ستون در linq اعمال می‌شود ولی در خروجی گزارش به زیر نمایش داده می‌شود:
نام ستون

نویسنده: وحید نصیری

تاریخ: ۱۳:۲۵ ۱۳۹۱/۱۰/۱۷

- shift+space یا نیم فاصله اینجا استفاده شده بود.

- راه بهتر: مثال DynamicCrosstab رو کمی تغییر دادم. به events.RowStarted آن دقت کنید. در این روال رخداد گردان می‌تونید در هر نوع گزارشی در PdfReport، برچسب‌های سطر هدر جدول تولیدی رو پیش از درج در فایل گزارش، کلا تغییر بدید: (^)
- اگر سؤالی در مورد PdfReport دارید لطفاً [در قسمت سؤال و جواب‌های آن](#) مطرح کنید.

نویسنده: مهران

تاریخ: ۱۳:۴۸ ۱۳۹۱/۱۰/۱۷

با تشکر فراوان از آقای نصیری، مشکل رفع شد.

با استفاده از اشیاء Com همراه با Acrobat SDK می‌توان تمام صفحات یک فایل PDF را تبدیل به تصویر کرد. این SDK به همراه نگارش کامل Adobe Acrobat نیز بر روی سیستم نصب می‌شود و یا می‌توان آن را به صورت جداگانه از سایت Adobe دریافت کرد.

<http://www.adobe.com/devnet/acrobat/downloads.html>

پس از آن، برای تبدیل صفحات یک فایل PDF به تصویر، مراحل زیر باید طی شود:

الف) وهله سازی از شیء AcroExch.PDDoc

در صورتیکه SDK یاد شده بر روی سیستم نصب نباشد، این وهله سازی با شکست مواجه خواهد شد و همچنین باید دقت داشت که این SDK به همراه نگارش رایگان Adobe reader ارائه نمی‌شود.

ب) گشودن فایل PDF به کمک شیء Com وهله سازی شده (pdfDoc.Open)

ج) دریافت اطلاعات صفحه مورد نظر (pdfDoc.AcquirePage)

د) کپی این اطلاعات به درون clipboard ویندوز (pdfPage.CopyToClipboard)

به این ترتیب به یک تصویر Bmp قرار گرفته شده در clipboard ویندوز خواهیم رسید

ه) مرحله بعد تغییر ابعاد و ذخیره سازی این تصویر نهایی است.

کدهای زیر، روش انجام این مراحل را بیان می‌کنند:

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.IO;
using System.Runtime.InteropServices;
using System.Threading;
using System.Windows.Forms;
using Acrobat; //Add a Com Object ref. to "Adobe Acrobat 10.0 Type Library" => Program
Files\Adobe\Acrobat 10.0\Acrobat\acrobat.tlb
using Microsoft.Win32;

namespace PdfThumbnail.Lib
{
    public static class PdfToImage
    {
        const string AdobeObjectsErrorMessage = "Failed to create the PDF object.";
        const string BadFileErrorMessage = "Failed to open the PDF file.";
        const string ClipboardError = "Failed to get the image from clipboard.";
        const string SdkError = "This operation needs the Acrobat
        SDK(http://www.adobe.com/devnet/acrobat/downloads.html), which is combined with the full version of
        Adobe Acrobat.";

        public static byte[] PdfPageToPng(string pdfFilePath, int thumbWidth = 600, int thumbHeight =
        750, int pageNumber = 0)
        {
            byte[] imageData = null;
            runJob((pdfDoc, pdfRect) =>
            {
                imageData = pdfPageToPng(thumbWidth, thumbHeight, pageNumber, pdfDoc, pdfRect);
            }, pdfFilePath);
            return imageData;
        }

        public static void AllPdfPagesToPng(Action<byte[], int, int> dataCallback, string pdfFilePath,
        int thumbWidth = 600, int thumbHeight = 750)
        {
            runJob((pdfDoc, pdfRect) =>
            {
                var numPages = pdfDoc.GetNumPages();
                for (var pageNumber = 0; pageNumber < numPages; pageNumber++)
                {
                    var imageData = pdfPageToPng(thumbWidth, thumbHeight, pageNumber, pdfDoc,
                    pdfRect);
                    dataCallback(imageData, pageNumber + 1, numPages);
                }
            }, pdfFilePath);
        }
    }
}
```

```

    }

    static void runJob(Action<CAcroPDDoc, CAcroRect> job, string pdfFilePath)
    {
        if (!File.Exists(pdfFilePath))
            throw new InvalidOperationException(BadFileErrorMessage);

        var acrobatPdfDocType = Type.GetTypeFromProgID("AcroExch.PDDoc");
        if (acrobatPdfDocType == null || !isAdobeSdkInstalled)
            throw new InvalidOperationException(SdkError);

        var pdfDoc = (CAcroPDDoc)Activator.CreateInstance(acrobatPdfDocType);
        if (pdfDoc == null)
            throw new InvalidOperationException(AdobeObjectsErrorMessage);

        var acrobatPdfRectType = Type.GetTypeFromProgID("AcroExch.Rect");
        var pdfRect = (CAcroRect)Activator.CreateInstance(acrobatPdfRectType);

        var result = pdfDoc.Open(pdfFilePath);
        if (!result)
            throw new InvalidOperationException(BadFileErrorMessage);

        job(pdfDoc, pdfRect);

        releaseComObjects(pdfDoc, pdfRect);
    }

    public static byte[] ResizeImage(this Image image, int thumbWidth, int thumbHeight)
    {
        var srcWidth = image.Width;
        var srcHeight = image.Height;
        using (var bmp = new Bitmap(thumbWidth, thumbHeight, PixelFormat.Format32bppArgb))
        {
            using (var gr = Graphics.FromImage(bmp))
            {
                gr.SmoothingMode = SmoothingMode.HighQuality;
                gr.PixelOffsetMode = PixelOffsetMode.HighQuality;
                gr.CompositingQuality = CompositingQuality.HighQuality;
                gr.InterpolationMode = InterpolationMode.High;

                var rectDestination = new Rectangle(0, 0, thumbWidth, thumbHeight);
                gr.DrawImage(image, rectDestination, 0, 0, srcWidth, srcHeight,
GraphicsUnit.Pixel);

                using (var memStream = new MemoryStream())
                {
                    bmp.Save(memStream, ImageFormat.Png);
                    return memStream.ToArray();
                }
            }
        }
    }

    static bool isAdobeSdkInstalled
    {
        get
        {
            return Registry.ClassesRoot.OpenSubKey("AcroExch.PDDoc", writable: false) != null;
        }
    }

    private static Bitmap pdfPageToBitmap(int pageNumber, CAcroPDDoc pdfDoc, CAcroRect pdfRect)
    {
        var pdfPage = (CAcroPDPage)pdfDoc.AcquirePage(pageNumber);
        if (pdfPage == null)
            throw new InvalidOperationException(BadFileErrorMessage);

        var pdfPoint = (CAcroPoint)pdfPage.GetSize();

        pdfRect.Left = 0;
        pdfRect.Right = pdfPoint.x;
        pdfRect.Top = 0;
        pdfRect.Bottom = pdfPoint.y;

        pdfPage.CopyToClipboard(pdfRect, 0, 0, 100);

        Bitmap pdfBitmap = null;
        var thread = new Thread(() =>
        {
            var data = Clipboard.GetDataObject();
            if (data != null && data.GetDataPresent(DataFormats.Bitmap))

```

```

        pdfBitmap = (Bitmap)data.GetData(DataFormats.Bitmap);
    });
    thread.SetApartmentState(ApartmentState.STA);
    thread.Start();
    thread.Join();

    Marshal.ReleaseComObject(pdfPage);

    return pdfBitmap;
}

private static byte[] pdfPageToPng(int thumbWidth, int thumbHeight, int pageNumber, CAcroPDDoc pdfDoc, CAcroRect pdfRect)
{
    var pdfBitmap = pdfPageToBitmap(pageNumber, pdfDoc, pdfRect);
    if (pdfBitmap == null)
        throw new InvalidOperationException(ClipboardError);

    var pdfImage = pdfBitmap.GetThumbnailImage(thumbWidth, thumbHeight, null, IntPtr.Zero);
    // (+ 7 for template border)
    var imageData = pdfImage.ResizeImage(thumbWidth + 7, thumbHeight + 7);
    return imageData;
}

private static void releaseComObjects(CAcroPDDoc pdfDoc, CAcroRect pdfRect)
{
    pdfDoc.Close();
    Marshal.ReleaseComObject(pdfRect);
    Marshal.ReleaseComObject(pdfDoc);
}
}
}

```

و برای استفاده از آن خواهیم داشت:

```

using System;
using System.IO;
using System.Windows.Forms;
using PdfThumbnail.Lib;

namespace PdfThumbnail
{
    class Program
    {
        static void Main(string[] args)
        {
            var pdfPath = Application.StartupPath + @"\test.pdf";
            PdfToImage.AllPdfPagesToPng((pageImageData, pageNumber, numPages) =>
            {
                Console.WriteLine("Page {0}/{1}", pageNumber, numPages);
                File.WriteAllBytes(string.Format("{0}\\page-{1}.png", Application.StartupPath,
                pageNumber), pageImageData);
            }, pdfPath);
        }
    }
}

```

کدهای این قسمت را از اینجا نیز می‌توانید دریافت کنید:

[PdfThumbnail.zip](#)

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۲۰ ۰:۲۰

امکان دریافت SDK فوق با IP ایرانی نیست. آنرا [از اینجا](#) هم می‌توانید دریافت کنید.

نویسنده: molana11
تاریخ: ۱۳۹۲/۰۱/۲۲ ۱۴:۴۷

با سلام.
می‌توان از این مثال در وب نیز استفاده کرد؟
باتشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۲ ۱۵:۳۰

- اگر سرور دار خودتون هستید و می‌تونید روی سرور وابستگی‌های مربوط به نگارش کامل Adobe Acrobat را نصب کنید و همچنین برنامه در حالت Full trust اجرا می‌شود؛ بله.
- راه حل‌های دیگری هم هستند که در قسمت اشتراک‌های سایت [مطرح شدند](#) .

نویسنده: molana11
تاریخ: ۱۳۹۲/۰۱/۲۲ ۱۵:۱۰

مهندس جان.
کیفیت تصویر تولیدی را چگونه می‌توان بالا برد؟
باتشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۲ ۱۶:۲۹

- کیفیت بالایی داره.
- حداکثر از ResizeImage استفاده نکنید (مستقیماً از Bitmap تولیدی استفاده کنید) یا آنرا تغییر دهید. در کل متد ResizeImage هم بر اساس تولید تصویر با کیفیت بالا تنظیم شده.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۶:۳۲

با سلام . روش دیگری که نیاز به نصب کتابخانه جانبی دیگر نداشته باشد و بتوان از آن تحت وب نیز استفاده کرد (برای هر دو ورژن x64 و x86) را چه توصیه میکنید؟ با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۷:۰۶

[کتابخانه‌ی GhostScript هم هست](#) . یک سری ابزار دیگر هم [در اینجا](#) لیست شدند.

فرض کنید که لیستی از کاربران را به همراه نام و تصاویر آن‌ها داریم. قصد داریم این اطلاعات را در یک سلول نمایش دهیم و نه اینکه هر کدام را در سلول‌های جداگانه‌ای قرار دهیم. [روش متداول انجام اینکار](#) تعریف یک قالب سلول سفارشی با پیاده سازی اینترفیس IColumnItemsTemplate است. راه میانبری نیز برای حل این مساله وجود دارد:

```
columns.AddColumn(column =>
{
    column.PropertyName("User");
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.IsVisible(true);
    column.Order(1);
    column.Width(3);
    column.HeaderCell("User");
    column.CalculatedField(list =>
    {
        var user = list.GetSafeStringValueOf("User");
        var photo = new Uri(list.GetSafeStringValueOf("Photo"));
        var image = string.Format("<img src='{0}' />", photo);
        return
            @"<table style='width: 100%;'>
                <tr>
                    <td>" + user + @"</td>
                </tr>
                <tr>
                    <td>" + image + @"</td>
                </tr>
            </table>";
    });
    column.ColumnItemsTemplate(template =>
    {
        template.Html(); // Using iTextSharp's limited HTML to PDF capabilities
    });
});
(HTMLWorker class).
});
```

می‌توان از قابلیت‌های محدود تبدیل HTML به PDF موجود در کلاس [HTMLWorker](#) استفاده کرد. البته نباید انتظار زیادی از این کلاس داشت، اما برای اینگونه مقاصد بسیار مفید است. در اینجا به کمک یک CalculatedField، مقدار جدید سلول را که یک جدول HTMLایی است، به منبع داده مورد استفاده تزریق می‌کنیم. سپس با استفاده از قالب Html، آن‌را پردازش و نمایش خواهیم داد. کدهای کامل این مثال را در اینجا می‌توانید ملاحظه کنید: ([^](#))

ابتدا مثال ساده زیر را در نظر بگیرید:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace OptimizeImageSizes
{
    class Program
    {
        static void Main(string[] args)
        {
            test1();
            test2();
        }

        private static void test2()
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test2.pdf",
                    FileMode.Create));
                pdfDoc.Open();

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

            Process.Start("test2.pdf");
        }

        private static void test1()
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test1.pdf",
                    FileMode.Create));
                pdfDoc.Open();

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

            Process.Start("test1.pdf");
        }
    }
}
```

در اینجا یک تصویر به نام myImage.png به دو طریق، به صفحه‌ای اضافه شده است:

الف) در متد test1، یک وهله از آن تهیه و دو بار به صفحه اضافه شده است.

ب) در متد test2، به نحوی متداول، هربار که نیاز به نمایش تصویری بوده، یک وهله جدید از تصویر تهیه و اضافه شده است.

نکته‌ی مهم در اینجا، حجم نهایی دو فایل حاصل است:

حجم فایل test2.pdf دقیقاً دو برابر حجم فایل test1.pdf است. علت هم به این بر می‌گردد که هر وهله جدیدی از شیء Image،

صرفنظر از محتوای آن، توسط iTextSharp به صورت جداگانه‌ای در فایل pdf نهایی ثبت خواهد شد.

این مورد خصوصاً در تهیه گزارشاتی که تصویری را در پشت صفحه صفحات نمایش می‌دهد یا در هدر صفحه یک تصویر مشخص و ثابتی قرار گرفته است و نیاز است این تصویر در تمام صفحات تکرار شود، بسیار مهم است و در صورت عدم رعایت نکته تهیه یک

وهله از تصاویری تکراری، می تواند حجم فایل را بی جهت تا چندمگابایت افزایش دهد.

نظرات خوانندگان

نویسنده: محمدرضا
تاریخ: ۹:۴۸ ۱۳۹۱/۰۸/۲۸

سلام

آیا این امکان برای تصویر به کاربرده شده در htmlheader هم فراهم است؟

نویسنده: وحید نصیری
تاریخ: ۱۰:۵۲ ۱۳۹۱/۰۸/۲۸

مطلب فوق در مورد iTextSharp بود.

در PdfReport متد header.CacheHeader وجود دارد که کل هدر را کش می‌کند (حالت پیش فرض است) و اشیاء آن را یکبار محاسبه و به صفحات اضافه خواهد کرد (در آخرین نگارش موجود در SVN). به این ترتیب فرقی نمی‌کند که هدر سفارشی است یا هر نوع پیش فرض دیگری. برای تمام آن‌ها کش توکار اعمال خواهد شد. اگر خواستید به ازای صفحات مختلف هدرهای مختلفی داشته باشید نیاز است header.CacheHeader را false کنید. در این حالت بهینه سازی صورت نخواهد گرفت.

[در نگارش 1.6](#) ، قالب سلول جدیدی به نام MonthCalendar اضافه شده است که امکان نمایش تقویم ماهیانه شمسی و میلادی را فراهم می‌کند. در ادامه نحوه استفاده از آن را بررسی خواهیم کرد. کدهای کامل این مثال را از اینجا نیز می‌توانید دریافت کنید: ([^](#))
(فرض کنید اطلاعات حضور و غیاب کارمندان را به نحو زیر در اختیار دارید:

```
namespace PdfReportSamples.Models
{
    public class UserWorkedHours
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public int DayNumber { set; get; }
        public int Month { set; get; }
        public int Year { set; get; }
        public string Description { set; get; }
    }
}
```

و برای نمونه منبع داده فرضی ما نیز به صورت زیر است (تعدادی روز، به همراه ساعات کارکرد):

```
private static List<UserWorkedHours> createUsersWorkedHours()
{
    var usersWorkedHours = new List<UserWorkedHours>();
    for (int i = 1; i < 11; i++)
    {
        for (int j = 1; j < 28; j++)
        {
            usersWorkedHours.Add(new UserWorkedHours
            {
                Id = i,
                Name = "کارمند " + i,
                Year = 1391, // مشخص می‌شود
                Month = i,
                DayNumber = j,
                Description = i % 2 == 0 ? "05:00" : "08:00"
            });
        }
    }
    return usersWorkedHours;
}
```

در این منبع داده فرضی، متن Description ذیل شماره روز، در تقویم ماهیانه نمایش داده خواهد شد.
سلولی که قرار است قالب MonthCalendar را نمایش دهد نیاز به شیءایی از نوع PdfRpt.Calendar.CalendarData دارد که به نحو زیر تعریف شده است:

```
using System.Collections.Generic;

namespace PdfRpt.Calendar
{
    public class CalendarData
    {
        public int Month { set; get; }
        public int Year { set; get; }
        public IList<DayInfo> MonthDaysInfo { set; get; }
    }
}
```

```
namespace PdfRpt.Calendar
{
    public class DayInfo
```

```

{
    public int DayNumber { set; get; }
    public int Month { set; get; }
    public int Year { set; get; }

    public string Description { set; get; }
    public bool ShowDescriptionInFooter { set; get; }
}

```

این ساختار بر اساس اطلاعات یک ماه و روزهای آن است. متن Description در صورت false بودن ShowDescriptionInFooter ذیل شماره روز نمایش داده خواهد شد، در غیراینصورت در پایان ماه به شکل یک سطر جدید نمایش داده می‌شود. در اینجا روزهای ماه و سال بر اساس نوع تقویم معنا خواهند شد.

اکنون نیاز است تا اطلاعات منبع داده خود را به CalendarData نگاشت کنیم تا بتوان از آن در قالب سلول جدید MonthCalendar استفاده کرد. انجام اینکار با استفاده از امکانات LINQ به نحو زیر است:

```

public static IList<UserMonthCalendar> CreateDataSource()
{
    var usersWorkedHours = createUsersWorkedHours();
    // Mapping a list of normal Users WorkedHours to a list of Users + CalendarData
    return usersWorkedHours
        .GroupBy(x => new
            {
                Id = x.Id,
                Name = x.Name
            })
        .Select(
            x => new UserMonthCalendar
            {
                Id = x.Key.Id,
                Name = x.Key.Name,
                // Calendar's cell data type should be
                PdfRpt.Calendar.CalendarData
                MonthCalendarData = new CalendarData
                {
                    Year = x.First().Year,
                    Month = x.First().Month,
                    MonthDaysInfo = x.ToList().Select(y => new DayInfo
                    {
                        Description = y.Description,
                        ShowDescriptionInFooter = false,
                        DayNumber = y.DayNumber
                    }).ToList()
                })
            }).ToList();
}

```

UserMonthCalendar، شامل ستون‌هایی است که قرار است در گزارش ما ظاهر شوند:

```

using PdfRpt.Calendar;

namespace PdfReportSamples.Models
{
    public class UserMonthCalendar
    {
        public int Id { set; get; }
        public string Name { set; get; }
        // Calendar's cell data type should be CalendarData
        public CalendarData MonthCalendarData { set; get; }
    }
}

```

ستون اول، شماره شخص، ستون دوم شامل نام شخص و ستون سوم، شامل اطلاعات یک ماه شخص است. برای نمایش این اطلاعات توسط PdfReport، دو ستون اول یاد شده نکته خاصی ندارند، اما نحوه تعریف ستون تقویم ماهیانه آن به صورت زیر خواهد بود:

```

columns.AddColumn(column =>
{
    // Calendar's cell data type should be PdfRpt.Calendar.CalendarData
    column.PropertyName<UserMonthCalendar>(x => x.MonthCalendarData);
    column.CellsHorizontalAlignment(HorizontalAlignment.Center);
    column.IsVisible(true);
    column.Order(3);
    column.Width(3);
    column.HeaderCell("تقویم ماهیانه");
    column.ColumnItemsTemplate(itemsTemplate =>
    {
        itemsTemplate.MonthCalendar(new CalendarAttributes
        {
            CalendarType = CalendarType.PersianCalendar,
            UseLongDayNamesOfWeek = true,
            Padding = 3,
            DescriptionHorizontalAlignment = HorizontalAlignment.Center,
            SplitRows = true,
            CellsCustomizer = info =>
            {
                if (info.Year == 1391 && info.Month == 1 && info.DayNumber == 1)
                {
                    info.NumberCell.BackgroundColor = new
BaseColor(System.Drawing.Color.LimeGreen);
                    var phrase = info.NumberCell.Phrase;
                    foreach (var chunk in phrase.Chunks)
                        chunk.Font.Color = new BaseColor(System.Drawing.Color.Yellow);
                }
            }
        });
    });
});
});

```

توسط CalendarAttributes می‌توان یک سری از خواص تقویم نمایش داده شده را تغییر داد. برای مثال CalendarType مشخص می‌کند که نوع تقویم شمسی است یا میلادی؛ UseLongDayNamesOfWeek برای نمایش نام روزها به صورت کامل «شنبه» یا «ش» (نام کوتاه شده آن) بکار می‌رود. SplitRows مشخص می‌کند که اگر تقویم در یک صفحه جا نهد، به صفحه بعد منتقل شود یا تا جایی که ممکن است در صفحه جاری اطلاعات آن نمایش داده شده و سپس مابقی را در صفحه بعد ترسیم کند (مقدار true آن). به علاوه توسط CellsCustomizer می‌توان فرمت کردن شرطی اطلاعات را انجام داد. برای مثال در اینجا اگر روز مورد نظر، روز اول سال 91 باشد، رنگ زمینه سلول و رنگ متن عدد آن تغییر خواهد کرد.



گزارش ساعات کارکرد کارکنان

#	شماره	نام	تقویم ماهیانه																																																																																				
فروردین ۱۳۹۱																																																																																							
			<table><tr><th>شنبه</th><th>یک شنبه</th><th>دوشنبه</th><th>سه شنبه</th><th>چهارشنبه</th><th>پنج شنبه</th><th>جمعه</th></tr><tr><td></td><td></td><td></td><td>۱</td><td>۲</td><td>۳</td><td>۴</td></tr><tr><td></td><td></td><td></td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td></tr><tr><td></td><td></td><td>۵</td><td>۶</td><td>۷</td><td>۸</td><td>۹</td></tr><tr><td></td><td></td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td></tr><tr><td></td><td></td><td>۱۲</td><td>۱۳</td><td>۱۴</td><td>۱۵</td><td>۱۶</td></tr><tr><td></td><td></td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td></tr><tr><td></td><td></td><td>۱۹</td><td>۲۰</td><td>۲۱</td><td>۲۲</td><td>۲۳</td></tr><tr><td></td><td></td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td><td>۸:۰۰</td></tr><tr><td></td><td></td><td>۲۶</td><td>۲۷</td><td>۲۸</td><td>۲۹</td><td>۳۰</td></tr><tr><td></td><td></td><td>۸:۰۰</td><td>۸:۰۰</td><td></td><td></td><td>۳۱</td></tr></table>	شنبه	یک شنبه	دوشنبه	سه شنبه	چهارشنبه	پنج شنبه	جمعه				۱	۲	۳	۴				۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰			۵	۶	۷	۸	۹			۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰			۱۲	۱۳	۱۴	۱۵	۱۶			۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰			۱۹	۲۰	۲۱	۲۲	۲۳			۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰			۲۶	۲۷	۲۸	۲۹	۳۰			۸:۰۰	۸:۰۰			۳۱							
شنبه	یک شنبه	دوشنبه	سه شنبه	چهارشنبه	پنج شنبه	جمعه																																																																																	
			۱	۲	۳	۴																																																																																	
			۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰																																																																																	
		۵	۶	۷	۸	۹																																																																																	
		۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰																																																																																	
		۱۲	۱۳	۱۴	۱۵	۱۶																																																																																	
		۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰																																																																																	
		۱۹	۲۰	۲۱	۲۲	۲۳																																																																																	
		۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰	۸:۰۰																																																																																	
		۲۶	۲۷	۲۸	۲۹	۳۰																																																																																	
		۸:۰۰	۸:۰۰			۳۱																																																																																	
۱	۱	کارمند ۱																																																																																					
اردیبهشت ۱۳۹۱																																																																																							
			<table><tr><th>شنبه</th><th>یک شنبه</th><th>دوشنبه</th><th>سه شنبه</th><th>چهارشنبه</th><th>پنج شنبه</th><th>جمعه</th></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>۱</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>۵:۰۰</td></tr><tr><td></td><td></td><td>۲</td><td>۳</td><td>۴</td><td>۵</td><td>۶</td></tr><tr><td></td><td></td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td></tr><tr><td></td><td></td><td>۹</td><td>۱۰</td><td>۱۱</td><td>۱۲</td><td>۱۳</td></tr><tr><td></td><td></td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td></tr><tr><td></td><td></td><td>۱۶</td><td>۱۷</td><td>۱۸</td><td>۱۹</td><td>۲۰</td></tr><tr><td></td><td></td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td></tr><tr><td></td><td></td><td>۲۳</td><td>۲۴</td><td>۲۵</td><td>۲۶</td><td>۲۷</td></tr><tr><td></td><td></td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td><td>۵:۰۰</td></tr><tr><td></td><td></td><td>۳۰</td><td>۳۱</td><td></td><td></td><td></td></tr></table>	شنبه	یک شنبه	دوشنبه	سه شنبه	چهارشنبه	پنج شنبه	جمعه							۱							۵:۰۰			۲	۳	۴	۵	۶			۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰			۹	۱۰	۱۱	۱۲	۱۳			۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰			۱۶	۱۷	۱۸	۱۹	۲۰			۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰			۲۳	۲۴	۲۵	۲۶	۲۷			۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰			۳۰	۳۱			
شنبه	یک شنبه	دوشنبه	سه شنبه	چهارشنبه	پنج شنبه	جمعه																																																																																	
						۱																																																																																	
						۵:۰۰																																																																																	
		۲	۳	۴	۵	۶																																																																																	
		۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰																																																																																	
		۹	۱۰	۱۱	۱۲	۱۳																																																																																	
		۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰																																																																																	
		۱۶	۱۷	۱۸	۱۹	۲۰																																																																																	
		۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰																																																																																	
		۲۳	۲۴	۲۵	۲۶	۲۷																																																																																	
		۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰	۵:۰۰																																																																																	
		۳۰	۳۱																																																																																				
۲	۲	کارمند ۲																																																																																					

نظرات خوانندگان

نویسنده: یزدان
تاریخ: ۱۱:۱۳ ۱۳۹۲/۰۱/۱۷

سلام

مهندس نصیری لینکی برای دانلود یکجای مثالهای PDF Repoert هست ؟

نویسنده: وحید نصیری
تاریخ: ۱۱:۳۷ ۱۳۹۲/۰۱/۱۷

مراجعه کنید به [قسمت مخصوص این پروژه در سایت](#) . مشخصات پروژه و محل سورس کنترل آن که حاوی تمام مثالها است، ذکر شده.

پیشتر در سایت جاری مطلبی را در مورد « [بهینه سازی حجم فایل PDF تولیدی در حین کار با تصاویر در iTextSharp](#) » مطالعه کرده اید. خلاصه آن به این نحو است که می توان در یک فایل PDF، ده ها تصویر را که تنها به یک فایل فیزیکی اشاره می کنند قرار داد. به این ترتیب حجم فایل نهایی تا حد بسیار قابل ملاحظه ای کاهش می یابد. البته آن مطلب در مورد تولید یک فایل PDF جدید صدق می کند. اما در مورد فایل های PDF موجود و از پیش آماده شده چگونه؟

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 می کنید:

```
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 قابلیت های سطح پایین PdfReader استفاده شده است. یک فایل PDF از پیش آماده، توسط این شیء گشوده شده و سپس محتویات تصاویر آن یافت می شوند. در ادامه هش MD5 آن ها محاسبه و با یکدیگر مقایسه می شوند. اگر هش تکراری یافت شد، یعنی تصویر یافت شده تکراری است و این فایل قابلیت بهینه سازی و کاهش حجم (قابل ملاحظه ای) را دارا می باشد.

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 مجدداً 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(PRIIndirectReference 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();
    }
}
```

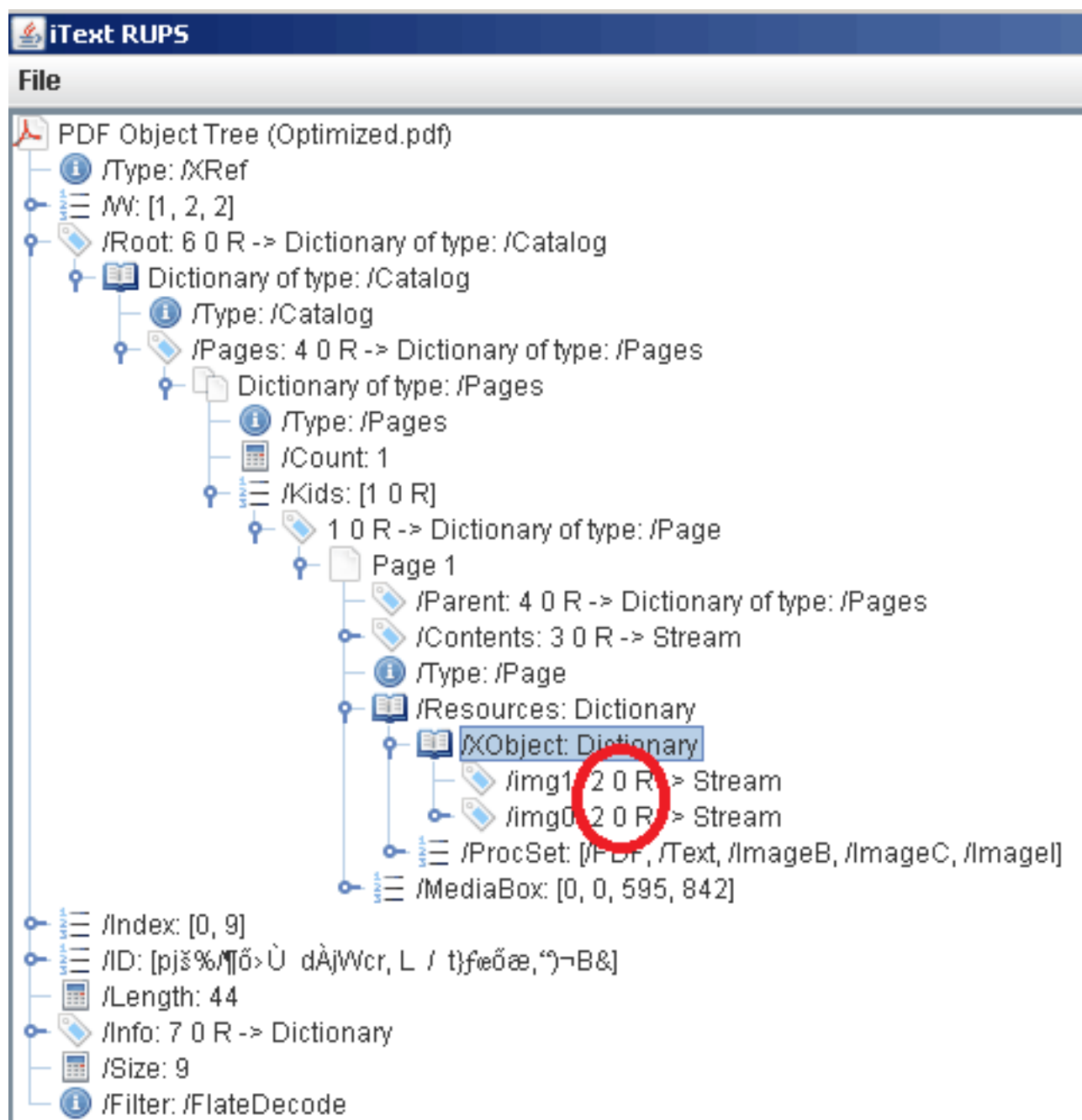
به نظر در نگارش iTextSharp 5.3.4 نویسندگان این کتابخانه اندکی فراموش کرده‌اند که باید تعدادی متد دیگر را نیز override کنند! به همین جهت کلاس PdfSmartCopy2 را مشاهده می‌کنید (اگر از نگارش‌های پایین‌تر استفاده می‌کنید، نیازی به آن نیست). استفاده از آن هم ساده است. در متد RemoveDuplicateObjects، ابتدا هر 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. احتمالاً برنامه‌های PDF compressor تجاری را در گوشه و کنار اینترنت دیده‌اید. متد RemoveDuplicateObjects دقیقاً همان کار را انجام می‌دهد.

اگر علاقمند هستید که متد فوق را آزمایش کنید یک فایل جدید PDF را به صورت زیر ایجاد نمایید:

```
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 png.01 به صفحه اضافه شده‌اند. بنابراین دقیقاً دو تصویر در فایل نهایی تولیدی وجود خواهد داشت.



اینبار دو تصویر داریم که هر دو به یک stream اشاره می‌کنند. تصاویر فوق به کمک برنامه **iText RUPS** تهیه شده‌اند.

برنامه رایگان Adobe reader یک سری خط فرمان دارد که توسط آن‌ها می‌توان فایل‌های PDF را مستقیماً به چاپگر ارسال کرد. در ادامه قطعه کدی را ملاحظه خواهید کرد که انجام اینکار را کپسوله می‌کند:

```
using System;
using System.Diagnostics;
using System.IO;
using System.Management;
using Microsoft.Win32;

namespace PdfFilePrinter
{
    /// <summary>
    /// Executes the Adobe Reader and prints a file while suppressing the Acrobat print
    /// dialog box, then terminating the Reader.
    /// </summary>
    public class AcroPrint
    {
        /// <summary>
        /// The Adobe Reader or Adobe Acrobat path such as 'C:\Program Files\Adobe\Adobe Reader
        X\AcroRd32.exe'.
        /// If it's not specified, the InstalledAdobeReaderPath property value will be used.
        /// </summary>
        public string AdobeReaderPath { set; get; }

        /// <summary>
        /// Returns the default printer name.
        /// </summary>
        public string DefaultPrinterName
        {
            get
            {
                var query = new ObjectQuery("SELECT * FROM Win32_Printer");
                using (var searcher = new ManagementObjectSearcher(query))
                {
                    foreach (var mo in searcher.Get())
                    {
                        if (((bool?)mo["Default"]) ?? false)
                            return mo["Name"] as string;
                    }
                }
                return string.Empty;
            }
        }

        /// <summary>
        /// The name and path of the PDF file to print.
        /// </summary>
        public string PdfFilePath { set; get; }

        /// <summary>
        /// Name of the printer such as '\\PrintServer\HP LaserJet'.
        /// If it's not specified, the DefaultPrinterName property value will be used.
        /// </summary>
        public string PrinterName { set; get; }

        /// <summary>
        /// Returns the HKEY_CLASSES_ROOT\Software\Adobe\Acrobat\Exe value.
        /// If AcroRd32.exe does not exist, returns string.Empty
        /// </summary>
        public string InstalledAdobeReaderPath
        {
            get
            {
                var acroRd32Exe = Registry.ClassesRoot.OpenSubKey(@"Software\Adobe\Acrobat\Exe",
                writable: false);
                if (acroRd32Exe == null)
                    return string.Empty;

                var exePath = acroRd32Exe.GetValue(string.Empty) as string;
                if (string.IsNullOrEmpty(exePath))
                    return string.Empty;
            }
        }
    }
}
```

```

        exePath = exePath.Trim(new[] { ' ' });
        return File.Exists(exePath) ? exePath : string.Empty;
    }
}

/// <summary>
/// Executes the Adobe Reader and prints a file while suppressing the Acrobat print
/// dialog box, then terminating the Reader.
/// </summary>
/// <param name="timeout">The amount of time, in milliseconds, to wait for the associated
process to exit. The maximum is the largest possible value of a 32-bit integer, which represents
infinity to the operating system.</param>
public void PrintPdfFile(int timeout = Int32.MaxValue)
{
    if (!File.Exists(PdfFilePath))
        throw new ArgumentException(PdfFilePath + " does not exist.");

    var args = string.Format("/N /T \"{0}\" \"{1}\"", PdfFilePath, getPrinterName());
    var process = startAdobeProcess(args);
    if (!process.WaitForExit(timeout))
        process.Kill();
}

private Process startAdobeProcess(string arguments = "")
{
    var startInfo = new ProcessStartInfo
    {
        FileName = this.getExePath(),
        Arguments = arguments,
        CreateNoWindow = true,
        ErrorDialog = false,
        UseShellExecute = false,
        Verb = "print"
    };

    return Process.Start(startInfo);
}

private string getPrinterName()
{
    var printer = PrinterName;
    if (string.IsNullOrEmpty(printer))
        printer = DefaultPrinterName;

    if (string.IsNullOrEmpty(printer))
        throw new ArgumentException("Please set the PrinterName.");

    return printer;
}

private string getExePath()
{
    var exePath = AdobeReaderPath;
    if (string.IsNullOrEmpty(exePath) || !File.Exists(exePath))
        exePath = InstalledAdobeReaderPath;

    if (string.IsNullOrEmpty(exePath))
        throw new ArgumentException("Please set the full path of the AcroRd32.exe or
Acrobat.exe.");

    return exePath;
}
}
}

```

توضیحات:

استفاده ابتدایی از کلاس فوق به نحو زیر است:

```

new AcroPrint
{
    PdfFilePath = @"D:\path\test.pdf"
}.PrintPdfFile();

```

به این ترتیب فایل PDF ذکر شده به چاپگر پیش فرض سیستم ارسال می‌شود.

ملاحظات:

- کدهای فوق نیاز به ارجاعی به اسمبلی استاندارد System.Management.dll نیز دارند.
- اگر علاقمند بودید که چاپگر خاصی را معرفی کنید (برای مثال یک چاپگر تعریف شده در شبکه)، می‌توانید خاصیت PrinterName را مقدار دهی نمائید.
- محل نصب Adobe reader از رجیستری ویندوز استخراج می‌شود. اما اگر محل نصب برنامه استاندارد نبود، نیاز است خاصیت AdobeReaderPath مقدار دهی گردد.
- تحت هر شرایطی برنامه Adobe reader ظاهر خواهد شد؛ حتی اگر در حین آغاز پروسه سعی در مخفی کردن پنجره آن نمائید.
- اینکار به عمد جهت مسایل امنیتی در این برنامه در نظر گرفته شده است تا کاربر بداند که پروسه چاپ آغاز شده است.

نظرات خوانندگان

نویسنده: ایمان محمدی
تاریخ: ۰۲۴ ۱۳۹۱/۱۱/۲۴

فکر کنم در فایل‌های pdf قابلیت وجود داره که دستور پرینت رو می‌تونه ذخیره کنه تا بهنگام باز شدن فایل دستور پرینت (یا پرینت دیالوگ) اجرا بشه. کاربرد این حالت برای وب ممکنه مناسب باشه ، شاید بد نیست چنین قابلیتی در پروژه PdfReport داشته باشید.

نویسنده: وحید نصیری
تاریخ: ۰۳۷ ۱۳۹۱/۱۱/۲۴

وجود داره. این تنظیم رو اضافه کنید:

```
return new PdfReport().DocumentPreferences(doc =>
{
    //...
    doc.PrintingPreferences(new PrintingPreferences
    {
        ShowPrintDialogAutomatically = true
    });
})
```

نویسنده: ایمان محمدی
تاریخ: ۱:۱۷ ۱۳۹۱/۱۱/۲۴

چه خوب ، نکته جالب اینه pdf خوان داخلی کروم هم از این قابلیت پشتیبانی میکنه و برای یک برنامه وب می‌تونه تداعی کننده print perview یک برنامه ویندوزی باشه.

نویسنده: علی پناهی
تاریخ: ۱۳:۴۲ ۱۳۹۳/۰۱/۲۰

با مقدار Timeout مشکل دارم، مثلا مقدار 10 که میدم در یک سیستم قدیمی چاپ انجام نمیشه و برای اطمینان از چاپ 50 که میدم ، چون دو برگه برای چاپ دارم یکی برای مشتری یکی برای مغازه دار دو دقیقه طول میکشه.

نویسنده: محسن خان
تاریخ: ۱۴:۴۴ ۱۳۹۳/۰۱/۲۰

مقدار پیش فرضش که در متد مشخص شده، برای تمام سیستم‌ها کافی هست.

نویسنده: علی پناهی
تاریخ: ۷:۱۱ ۱۳۹۳/۰۱/۲۱

اصلاحیه اعداد من 10000 و 50000 می‌باشد.
من زمان تعیین کرده بودم که کاربر پنجره را نبندد و خودش بسته بشود،
پس باید به کاربر بگویم خودش پنجره Adobe Reader را ببندد.
اگر این مورد را بشود به صورت دیگه ای حل کرد، بهتر و کاربردی تره.

تبدیل بی عیب و نقص یک فایل PDF (انواع و اقسام آن‌ها) به متن قابل درک بسیار مشکل است. در ادامه بررسی خواهیم کرد که چرا.

برخلاف تصور عموم، ساختار یک صفحه PDF شبیه به یک صفحه فایل Word نیست. این صفحات درحقیقت نوعی Canvas برای نقاشی هستند. در این بوم نقاشی، شکل، تصویر، متن و غیره در مختصات خاصی قرار خواهند گرفت. حتی کلمه «متن» می‌تواند به صورت سه حرف در سه مختصات خاص یک صفحه PDF نقاشی شود. برای درک بهتر این مورد نیاز است سورس یک صفحه PDF را بررسی کرد.

نحوه استخراج سورس یک صفحه PDF

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace TestReaders
{
    class Program
    {
        static void writePdf()
        {
            using (var document = new Document(PageSize.A4))
            {
                var writer = PdfWriter.GetInstance(document, new FileStream("test.pdf",
                    FileMode.Create));
                document.Open();

                document.Add(new Paragraph("Test"));

                PdfContentByte cb = writer.DirectContent;
                BaseFont bf = BaseFont.CreateFont();
                cb.BeginText();
                cb.SetFontAndSize(bf, 12);
                cb.MoveText(88.66f, 367);
                cb.ShowText("ld");
                cb.MoveText(-22f, 0);
                cb.ShowText("Wor");
                cb.MoveText(-15.33f, 0);
                cb.ShowText("llo");
                cb.MoveText(-15.33f, 0);
                cb.ShowText("He");
                cb.EndText();

                PdfTemplate tmp = cb.CreateTemplate(250, 25);
                tmp.BeginText();
                tmp.SetFontAndSize(bf, 12);
                tmp.MoveText(0, 7);
                tmp.ShowText("Hello People");
                tmp.EndText();
                cb.AddTemplate(tmp, 36, 343);
            }

            Process.Start("test.pdf");
        }

        private static void readPdf()
        {
            var reader = new PdfReader("test.pdf");
            int intPageNum = reader.NumberOfPages;
            for (int i = 1; i <= intPageNum; i++)
            {
                byte[] contentBytes = reader.GetPageContent(i);
                File.WriteAllBytes("page-" + i + ".txt", contentBytes);
            }
            reader.Close();
        }
    }
}
```

```
static void Main(string[] args)
{
    writePdf();
    readPdf();
}
}
```

فایل PDF تولیدی حاوی سه عبارت کامل و مفهوم می‌باشد:

Test

Hello World
Hello People

اگر علاقمند باشید که سورس واقعی صفحات یک فایل PDF را مشاهده کنید، نحوه انجام آن توسط کتابخانه iTextSharp به صورت فوق است.

هرچند متد `GetPageContent` آرایه‌ای از بایت‌ها را بر می‌گرداند، اما اگر حاصل نهایی را در یک ادیتور متنی باز کنیم، قابل مطالعه و خواندن است. برای مثال، سورس مثال فوق (محتوای فایل `page-1.txt` تولید شده) به نحو زیر است:

```
q
BT
36 806 Td
0 -18 Td
/F1 12 Tf
(Test)Tj
0 0 Td
ET
Q
BT
/F1 12 Tf
```

```
88.66 367 Td
(ld)Tj
-22 0 Td
(Wor)Tj
-15.33 0 Td
(llo)Tj
-15.33 0 Td
(He)Tj
ET
q 1 0 0 1 36 343 cm /Xf1 Do Q
```

و تفسیر این عملگرها به این ترتیب است:

```
SaveGraphicsState(); // q
BeginText(); // BT
MoveTextPos(36, 806); // Td
MoveTextPos(0, -18); // Td
SelectFontAndSize("/F1", 12); // Tf
ShowText("(Test)"); // Tj
MoveTextPos(0, 0); // Td
EndTextObject(); // ET
RestoreGraphicsState(); // Q
BeginText(); // BT
SelectFontAndSize("/F1", 12); // Tf
MoveTextPos(88.66, 367); // Td
ShowText("(ld)"); // Tj
MoveTextPos(-22, 0); // Td
ShowText("(Wor)"); // Tj
MoveTextPos(-15.33, 0); // Td
ShowText("(llo)"); // Tj
MoveTextPos(-15.33, 0); // Td
ShowText("(He)"); // Tj
EndTextObject(); // ET
SaveGraphicsState(); // q
TransMatrix(1, 0, 0, 1, 36, 343); // cm
XObject("/Xf1"); // Do
RestoreGraphicsState(); // Q
```

همانطور که ملاحظه می‌کنید کلمه Test به مختصات خاصی انتقال داده شده و سپس به کمک اطلاعات فونت F1، ترسیم می‌شود. تا اینجا استخراج متن از فایل‌های PDF ساده به نظر می‌رسد. باید به دنبال Tj گشت و حروف مرتبط با آن‌را ذخیره کرد. اما در مورد «ترسیم» عبارات hello world و hello people اینطور نیست. عبارت hello world به حروف متفاوتی تقسیم شده و سپس در مختصات مشخصی ترسیم می‌گردد. عبارت hello people به صورت یک شیء ذخیره شده در قسمت منابع فایل PDF، بازیابی و نمایش داده می‌شود و اصلاً در سورس صفحه جاری وجود ندارد.

این تازه قسمتی از نحوه عملکرد فایل‌های PDF است. در فایل‌های PDF می‌توان قلم‌ها را مدفون ساخت. همچنین این قلم‌ها نیز تنها زیر مجموعه‌ای از قلم اصلی مورد استفاده هستند. برای مثال اگر عبارت Test قرار است نمایش داده شود، فقط اطلاعات T، e و s در فایل نهایی PDF قرار می‌گیرند. به علاوه امکان تغییر کلی شماره Glyph متناظر با هر حرف نیز توسط PDF writer وجود دارد. به عبارتی الزامی نیست که مشخصات اصلی فونت حتماً حفظ شود.

شاید بعضی از PDFهای فارسی را دیده باشید که پس از کپی متن آن‌ها در برنامه Adobe reader و سپس paste آن در جایی دیگر، متن حاصل قابل خواندن نیست. علت این است که نحوه ذخیره سازی قلم مورد استفاده کاملاً تغییر کرده است و برای بازیابی متن اینگونه فایل‌ها، استفاده از OCR ساده‌ترین روش است. برای نمونه در این قلم جدید مدفون شده، دیگر شماره کاراکتر 0x41 مساوی A نیست. بنابر سلیقه PDF writer این شماره به Glyph دیگری انتساب داده شده و چون قلم و مشخصات هندسی Glyph مورد استفاده در فایل PDF ذخیره می‌شود، برای نمایش این نوع فایل‌ها هیچگونه مشکلی وجود ندارد. اما متن آن‌ها به سادگی قابل بازیابی نیست.

عموما در برنامه‌های وب برای نمایش فایل‌های پویای باینری تولید شده، یا ابتدا آن‌ها را بر روی سخت دیسک ذخیره کرده و مسیر نهایی را به نحوی به کاربر نمایش می‌دهند و یا فایل را بدون ذخیره سازی، در مرورگر کاربر اصطلاحا Flush می‌کنند. حالت Flush سبب نمایش صفحه دیالوگ ذخیره سازی فایل گردیده و در همینجا Response خاتمه خواهد یافت. برای نمونه در اینجا توسط متد inMemoryFile، یک فایل PDF در حافظه تشکیل شده و سپس به صورت یک Byte Array بازگشت داده می‌شود. در ادامه کار، این اطلاعات در مرورگر کاربر Flush خواهد شد:

```
using System.IO;
using System.Net.Mime;
using System.Web;

namespace WebApplication
{
    public class PdfHandler : IHttpHandler
    {
        private static byte[] inMemoryFile()
        {
            //تولید پویای فایل در حافظه و یا حتی خواندن از یک نمونه موجود
            return File.ReadAllBytes(@"D:\path\DynamicCrosstabSampleRpt.pdf");
        }

        public void ProcessRequest(HttpContext context)
        {
            var pdf = inMemoryFile();

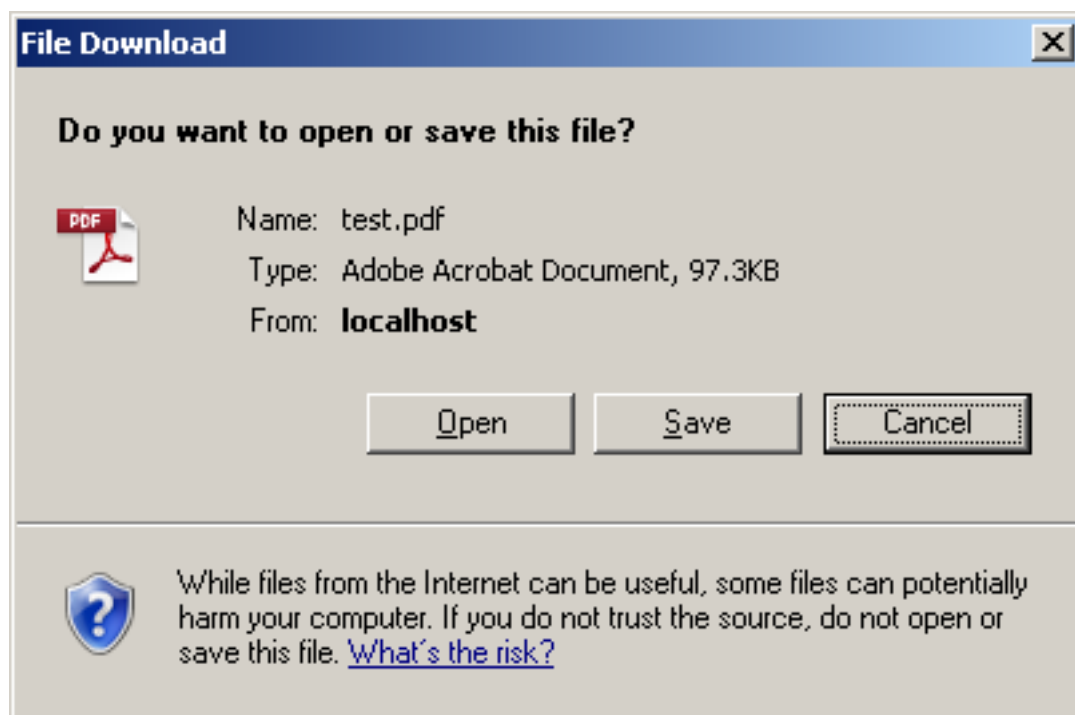
            context.Response.Cache.SetCacheability(HttpCacheability.NoCache);
            context.Response.ContentType = MediaTypeNames.Application.Pdf;
            context.Response.AddHeader("Content-Length", pdf.Length.ToString());
            context.Response.AddHeader("content-disposition", "attachment;filename=test.pdf");
            context.Response.Buffer = true;
            context.Response.Clear();
            context.Response.OutputStream.Write(pdf, 0, pdf.Length);
            context.Response.OutputStream.Flush();
            context.Response.OutputStream.Close();
            context.Response.End();
        }

        public bool IsReusable
        {
            get { return false; }
        }
    }
}
```

و برای نمایش آن در یک iframe در صفحه:

```
<iframe width="100%" src="PdfHandler.ashx" height="200px"></iframe>
```

نتیجه کار، نمایش صفحه دیالوگ ذخیره سازی فایل به کاربر است:



سؤال: فرض کنید Adobe reader بر روی سیستم نصب است و مرورگر با استفاده از Active-X آن می‌تواند این نوع فایل‌ها را نمایش دهد. آیا راهی وجود دارد تا بجای نمایش save popup dialog، این فایل توسط مرورگر نمایش داده شود؟
پاسخ: بلی. در کدهای فوق تنها کافی است یک سطر آن تغییر کند:

```
Response.AddHeader("content-disposition", "inline;filename=test.pdf");
```

در اینجا تنها نحوه مقدار دهی content-disposition تفاوت کرده است. حالت attachment سبب نمایش save popup dialog می‌شود و مقدار inline، فایل را در مرورگر نمایش خواهد داد.
اینبار اگر برنامه را اجرا کنیم، iframe ایی که به PdfHandler.ashx اشاره می‌کند، فایل PDF را در صفحه نمایش می‌دهد.

نظرات خوانندگان

نویسنده:

آرمان

تاریخ:

۱۱:۴۸ ۱۳۹۱/۱۲/۲۸

عالی بود مهندس. در صورتی که inline باشه ولی برنامه ای برای استفاده از پی دی اف نباشه یا مرورگر مجاز به استفاده نباشه باز صفحه دانلود دیده میشه؟

نویسنده:

وحید نصیری

تاریخ:

۱۲:۱۲ ۱۳۹۱/۱۲/۲۸

بله. صفحه file download فوق باز می‌شود.

نویسنده:

امیر هاشم زاده

تاریخ:

۲۲:۱۴ ۱۳۹۳/۰۵/۲۰

آیا امکان عدم دریافت فایل مذکور توسط دانلود منیجرها وجود دارد؟

نویسنده:

وحید نصیری

تاریخ:

۲۲:۱۹ ۱۳۹۳/۰۵/۲۰

با یک پسوند دیگر که نمی‌شناسند تست کنید؛ مثلاً filename=test.abc

نویسنده:

افتابی

تاریخ:

۰:۱۶ ۱۳۹۳/۰۵/۳۱

متشکر؛ خوب اگه من بخوام با استفاده از PDFReport گزارش درست کنم و توی خود view نمایش بدم، این کدها رو توی کدوم کلاس بگذارم؟

نویسنده:

وحید نصیری

تاریخ:

۰:۲۴ ۱۳۹۳/۰۵/۳۱

به صورت توکار لحاظ شده:

```
.Generate(data =>
{
    fileName = HttpUtility.UrlEncode(fileName, Encoding.UTF8);
    data.FlushInBrowser(fileName, FlushType.Inline);
}); // creating an in-memory PDF file
```

FlushType. Inline آن همان مطلب جاری است.

پیشتر مطلبی را در مورد « [تبدیل HTML به PDF با استفاده از کتابخانه‌ی iTextSharp](#) » در این سایت مطالعه کرده‌اید. این مطلب از افزونه HTMLWorker کتابخانه iTextSharp استفاده می‌کند که ... مدتی است توسط نویسندگان این مجموعه منسوخ شده اعلام گردیده و دیگر پشتیبانی نمی‌شود.

کتابخانه جایگزین آن را افزونه XMLWorker معرفی کرده‌اند که توانایی پردازش CSS و HTML بهتر و کاملتری را نسبت به HTMLWorker ارائه می‌دهد. این کتابخانه نیز همانند HTMLWorker پشتیبانی توکاری از متون راست به چپ و یونیکد فارسی، ندارد و نیاز است برای نمایش صحیح متون فارسی در آن، نکات خاصی را اعمال نمود که در ادامه بحث آن‌ها را مرور خواهیم کرد.

ابتدا برای دریافت آخرین نگارش‌های iTextSharp و افزونه XMLWorker آن به آدرس‌های ذیل مراجعه نمائید:

<http://sourceforge.net/projects/itextsharp/files/itextsharp>

<http://sourceforge.net/projects/itextsharp/files/xmlworker>

تهیه یک UnicodeFontProvider

Encoding پیش فرض قلم‌ها در XMLWorker مساوی BaseFont.CP1252 است؛ که از حروف یونیکد پشتیبانی نمی‌کند. برای رفع این نقیصه نیاز است یک منبع تامین قلم سفارشی را برای آن ایجاد نمود:

```
public class UnicodeFontProvider : FontFactoryImp
{
    static UnicodeFontProvider()
    {
        // روش صحیح تعریف فونت
        var systemRoot = Environment.GetEnvironmentVariable("SystemRoot");
        FontFactory.Register(Path.Combine(systemRoot, "fonts\\tahoma.ttf"));
        // ثبت سایر فونت‌ها در اینجا
        //FontFactory.Register(Path.Combine(Environment.CurrentDirectory, "fonts\\irsans.ttf"));
    }

    public override Font GetFont(string fontname, string encoding, bool embedded, float size, int style, BaseColor color, bool cached)
    {
        if (string.IsNullOrEmpty(fontname))
            return new Font(Font.FontFamily.UNDEFINED, size, style, color);
        return FontFactory.GetFont(fontname, BaseFont.IDENTITY_H, BaseFont.EMBEDDED, size, style, color);
    }
}
```

قلم‌های مورد نیاز را در سازنده کلاس به نحوی که مشاهده می‌کنید، ثبت نمائید.

باقی مسایل آن خودکار خواهد بود و هر زمانیکه نیاز به قلم خاصی از طرف XMLWorker وجود داشت، به متد GetFont فوق مراجعه کرده و اینبار قلمی با BaseFont.IDENTITY_H را دریافت می‌کند. IDENTITY_H در استاندارد PDF، جهت مشخص ساختن encoding قلم‌هایی با پشتیبانی از یونیکد بکار می‌رود.

تهیه منبع تصاویر

در XMLWorker اگر تصاویر با http شروع نشوند (دریافت تصاویر وب آن خودکار است)، آن تصاویر را از مسیری که توسط پیاده سازی کلاس AbstractImageProvider مشخص خواهد شد، دریافت می‌کند که نمونه‌ای از پیاده سازی آن را در ذیل مشاهده می‌کنید:

```
public class ImageProvider : AbstractImageProvider
{
    public override string GetImageRootPath()
    {

```

```

    var path = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
    return path + "\\"; // مهم است که این مسیر به یک اسلش ختم شود تا درست کار کند
}
}

```

نحوه تعریف یک فایل CSS خارجی

```

public static class XMLWorkerUtils
{
    /// <summary>
    /// نحوه تعریف یک فایل سی اس اس خارجی
    /// </summary>
    public static ICssFile GetCssFile(string filePath)
    {
        using (var stream = new FileStream(filePath, FileMode.Open, FileAccess.Read,
        FileShare.ReadWrite))
        {
            return XMLWorkerHelper.GetCSS(stream);
        }
    }
}

```

برای مسیریابی یک فایل CSS در کتابخانه XMLWorker می‌توان از کلاس فوق استفاده کرد.

تبدیل المان‌های HTML پردازش شده به یک لیست PDF ایی

تهیه مقدمات فارسی سازی و نمایش راست به چپ اطلاعات در کتابخانه XMLWorker از اینجا شروع می‌شود. در حالت پیش فرض کار آن، المان‌های HTML به صورت خودکار Parse شده و به صفحه اضافه می‌شوند. به همین دلیل دیگر فرصت اعمال خواص RTL به المان‌های پردازش شده دیگر وجود نخواهد داشت و به صورت توکار نیز این مسایل در نظر گرفته نمی‌شود. به همین دلیل نیاز است که در حین پردازش المان‌های HTML و تبدیل آن‌ها به معادل المان‌های PDF، بتوان آن‌ها را جمع آوری کرد که نحوه انجام آن‌را با پیاده سازی اینترفیس IElementHandler در ذیل مشاهده می‌کنید:

```

/// <summary>
/// معادل پی دی افی المان‌های اچ تی ام ال را جمع آوری می‌کند
/// </summary>
public class ElementsCollector : IElementHandler
{
    private readonly Paragraph _paragraph;

    public ElementsCollector()
    {
        _paragraph = new Paragraph
        {
            Alignment = Element.ALIGN_LEFT // سبب می‌شود تا در حالت راست به چپ از سمت راست
        };
    }

    /// <summary>
    /// این پاراگراف حاوی کلیه المان‌های متن است
    /// </summary>
    public Paragraph Paragraph
    {
        get { return _paragraph; }
    }

    /// <summary>
    /// بجای اینکه خود کتابخانه اصلی کار افزودن المان‌ها را به صفحات انجام دهد
    /// قصد داریم آن‌ها را ابتدا جمع آوری کرده و سپس به صورت راست به چپ به صفحات نهایی اضافه کنیم
    /// </summary>
    /// <param name="htmlElement"></param>
    public void Add(IWritable htmlElement)
    {
        var writableElement = htmlElement as WritableElement;
        if (writableElement == null)
            return;
    }
}

```

```

        foreach (var element in writableElement.Elements())
        {
            fixNestedTablesRunDirection(element);
            _paragraph.Add(element);
        }
    }

    /// <summary>
    /// نیاز است سلول‌های جداول تو در تو پی دی اف نیز راست به چپ شوند
    /// </summary>
    private void fixNestedTablesRunDirection(IElement element)
    {
        var table = element as PdfPTable;
        if (table == null)
            return;

        table.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
        foreach (var row in table.Rows)
        {
            foreach (var cell in row.GetCells())
            {
                cell.RunDirection = PdfWriter.RUN_DIRECTION_RTL;
                foreach (var item in cell.CompositeElements)
                {
                    fixNestedTablesRunDirection(item);
                }
            }
        }
    }
}

```

این کلاس کلیه المان‌های دریافتی را به یک پاراگراف اضافه می‌کند. همچنین اگر به جدولی در این بین برخورد، مباحث RTL آن‌را نیز اصلاح خواهد نمود.

یک مثال کامل از نحوه کنار هم قرار دادن پیشنیازهای تهیه شده

خوب؛ تا اینجا یک سری پیشنیاز را تهیه کردیم، اما XMLWorker از وجود آن‌ها بی‌خبر است. برای معرفی آن‌ها باید به نحو ذیل عمل کرد:

```

using (var pdfDoc = new Document(PageSize.A4))
{
    var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("test.pdf",
        FileMode.Create));
    pdfWriter.RgbTransparencyBlending = true;
    pdfDoc.Open();

    var html = @"<span style='color:blue; font-family:tahoma;'><b>آزمایش</b></span>
        <i>iTextSharp</i> <u>فارسی نویسی</u>
        <table style='color:blue; font-family:tahoma;
border='1'><tr><td>متن</td></tr></table>
        <code>This is a code!</code>
        <br/>
        <img src='av-13489.jpg' />
        ";

    var cssResolver = new StyleAttrCSSResolver();
    // cssResolver.AddCss(XMLWorkerUtils.GetCssFile(@"c:\path\pdf.css"));
    cssResolver.AddCss(@"code
    {
        padding: 2px 4px;
        color: #d14;
        white-space: nowrap;
        background-color: #f7f7f9;
        border: 1px solid #e1e1e8;
    }",
        "utf-8", true);

    // کار جمع آوری المان‌های ترجمه شده به المان‌های پی دی اف را انجام می‌دهد
    var elementsHandler = new ElementsCollector();

    var htmlContext = new HtmlPipelineContext(new CssApppliersImpl(new
        UnicodeFontProvider()));
}

```

```

htmlContext.SetImageProvider(new ImageProvider());
htmlContext.CharSet(Encoding.UTF8);

htmlContext.SetAcceptUnknown(true).AutoBookmark(true).SetTagFactory(Tags.GetHtmlTagProcessorFactory());
var pipeline = new CssResolverPipeline(cssResolver,
                                     new HtmlPipeline(htmlContext, new
ElementHandlerPipeline(elementsHandler, null)));
var worker = new XMLWorker(pipeline, parseHtml: true);
var parser = new XMLParser();
parser.AddListener(worker);
parser.Parse(new StringReader(html));

// با هندلر سفارشی که تهیه کردیم تمام المان‌های اچ تی ام ال به المان‌های پی دی اف تبدیل
// الان تنها کافی است تا این‌ها را در یک جدول راست به چپ محصور کنیم تا درست
// نمایش داده شوند
var mainTable = new PdfPTable(1) { WidthPercentage = 100, RunDirection =
PdfWriter.RUN_DIRECTION_RTL };
var cell = new PdfPCell
{
    Border = 0,
    RunDirection = PdfWriter.RUN_DIRECTION_RTL,
    HorizontalAlignment = Element.ALIGN_LEFT
};
cell.AddElement(elementsHandler.Paragraph);
mainTable.AddCell(cell);

pdfDoc.Add(mainTable);
}

Process.Start("test.pdf");

```

نحوه تعریف inline css یا نحوه افزودن یک فایل css خارجی را نیز در ابتدای این مثال مشاهده می‌کنید.

UnicodeFontProvider باید به HtmlPipelineContext شناسانده شود.

ImageProvider توسط متد SetImageProvider به HtmlPipelineContext معرفی می‌شود.

ElementsCollector سفارشی ما در قسمت CssResolverPipeline باید به سیستم تزریق شود.

پس از آن XMLWorker را وادار می‌کنیم تا HTML را Parse کرده و معادل المان‌های PDF ایی آنرا تهیه کند؛ اما آن‌ها را به صورت خودکار به صفحات فایل PDF نهایی اضافه نکند. در این بین ElementsCollector ما این المان‌ها را جمع آوری کرده و در نهایت، پاراگراف کلی حاصل از آن‌ها را به یک جدول با RUN_DIRECTION_RTL اضافه می‌کنیم. حاصل آن نمایش صحیح متون فارسی است.

کدهای مثال فوق را از آدرس ذیل نیز می‌توانید دریافت کنید:

[XMLWorkerRTLsample.cs](#)

به روز رسانی

کلیه نکات مطلب فوق را به همراه بهبودهای مطرح شده در نظرات آن، در پروژه‌ی ذیل می‌توانید به صورت یکجا دریافت و بررسی کنید:

[XMLWorkerRTLsample.zip](#)

نظرات خوانندگان

نویسنده: ح م

تاریخ: ۹:۱۸ ۱۳۹۲/۰۸/۱۶

همه‌ی فونت‌ها و استایل‌ها را هم که پیوست می‌کنم، برای برخی از کدهای HTML، خروجی pdf سفید(خالی) است. راهی وجود دارد که خطاهای پارسر دست کم در حالت Debug نشان داده شوند؟

نویسنده: وحید نصیری

تاریخ: ۱۰:۱۱ ۱۳۹۲/۰۸/۱۶

بله. یک کلاس لاگر سفارشی درست کنید:

```
public class CustomLogger : iTextSharp.text.log.ILogger
{
    public iTextSharp.text.log.ILogger GetLogger(Type klass)
    {
        return this;
    }

    public iTextSharp.text.log.ILogger GetLogger(string name)
    {
        return this;
    }

    public bool IsLogging(iTextSharp.text.log.Level level)
    {
        return true;
    }

    public void Warn(string message)
    {
        System.Diagnostics.Trace.TraceWarning(message);
    }

    public void Trace(string message)
    {
        System.Diagnostics.Trace.TraceInformation(message);
    }

    public void Debug(string message)
    {
        System.Diagnostics.Trace.TraceInformation(message);
    }

    public void Info(string message)
    {
        System.Diagnostics.Trace.TraceInformation(message);
    }

    public void Error(string message)
    {
        System.Diagnostics.Trace.TraceError(message);
    }

    public void Error(string message, Exception e)
    {
        System.Diagnostics.Trace.TraceError(message + System.Environment.NewLine + e);
    }
}
```

بعد در ابتدای اجرای برنامه آنرا ثبت کنید:

```
iTextSharp.text.log.LoggerFactory.GetInstance().SetLogger(new CustomLogger());
```

خروجی‌ها در پنجره دیباگ VS.NET نمایش داده می‌شوند.

نویسنده: مهرداد
تاریخ: ۲۱:۱۳ ۱۳۹۲/۰۸/۲۴

سلام دوست عزیز
من کد بالا رو تست کردم ظاهراً تگ‌های div رو نشون نمیده و از بین می‌بره !

تشکر

نویسنده: وحید نصیری
تاریخ: ۲۱:۳۱ ۱۳۹۲/۰۸/۲۴

- نام این کتابخانه XML Worker هست. یعنی HTML شما باید معتبر باشد و تگ‌های آن همانند یک فایل XML درست تشکیل و باز و بسته شده باشند؛ چیزی مثل XHTML ها.
- می‌توانید از کتابخانه [HTML Agility pack](#) برای درست کردن XHTML استفاده کنید:

```
var sb = new StringBuilder();
var stringWriter = new StringWriter(sb);
var doc = new HtmlDocument
{
    OptionOutputAsXml = true,
    OptionCheckSyntax = true,
    OptionFixNestedTags = true,
    OptionAutoCloseOnEnd = true,
    OptionDefaultStreamEncoding = Encoding.UTF8
};
doc.LoadHtml(htmlContent);
doc.Save(stringWriter);
var xhtml = sb.ToString();
```

- خاصیت OptionOutputAsXml آنرا true کنید تا در حد توانش مشکلات HTML شما را برطرف و یک خروجی XHTML را تولید کند.
- [سایر مشکلات](#) آنرا بهتر است در [mailing لیست آن‌ها](#) به همراه ارائه مثال قابل بازتولیدی ارسال کنید.

نویسنده: جلال
تاریخ: ۸:۵۰ ۱۳۹۲/۱۲/۱۵

با سلام؛ من خیلی دنبال کلاس HtmlDocument گشتم، اما نه توی .net پیدا کردم و نه توی سایت خودتون، میتونید راهنمایی کنید؟

نویسنده: وحید نصیری
تاریخ: ۹:۲۵ ۱۳۹۲/۱۲/۱۵

«می‌توانید از کتابخانه [HTML Agility pack](#) استفاده کنید»

```
PM> Install-Package HtmlAgilityPack
```

نویسنده: جلال
تاریخ: ۱۱:۴ ۱۳۹۲/۱۲/۱۵

من کدی که فرمودید رو اضافه کردم، همچنین، کد HTML هم Valid هستش، و کلاً با div ساخته شده، اما pdf خروجی سفید هستش.

نویسنده: وحید نصیری
تاریخ: ۱۲:۳۵ ۱۳۹۲/۱۲/۱۵

برای رفع مشکل محو شدن Div، کدهای کلاس ElementsCollector مطلب جاری را به نحو زیر تغییر دهید:

```
public void Add(IWritable htmlElement)
{
```

```

var writableElement = htmlElement as WritableElement;
if (writableElement == null)
    return;

foreach (var element in writableElement.Elements())
{
    var div = element as PdfDiv;
    if (div != null)
    {
        foreach (var divChildElement in div.Content)
        {
            fixNestedTablesRunDirection(divChildElement);
            _paragraph.Add(divChildElement);
        }
    }
    else
    {
        fixNestedTablesRunDirection(element);
        _paragraph.Add(element);
    }
}
}

```

نویسنده: سمیه

تاریخ: ۱۵:۳۹ ۱۳۹۳/۰۱/۱۹

سلام! ضمن تشکر از مطلب مفیدتان من نمونه کدهایی که در قسمت پایین قرار داده بودید، دانلود کردم. همچنین آخرین نگارش‌های iTextSharp و افزونه XMLWorker را از لینک‌هایی معرفی شده دانلود و dll هایشان را به پروژه ام اضافه کرده ام، ولی با وجود این به فضای نام iTextSharp.tool خطا می‌دهد و آن را نمی‌شناسد. می‌شه لطفاً من راهنمایی کنید؟

نویسنده: وحید نصیری

تاریخ: ۱۷:۲۰ ۱۳۹۳/۰۱/۱۹

پروژه شما باید ارجاعاتی را به دو فایل itextsharp.dll و itextsharp.xmlworker.dll داشته باشد.

```

PM> Install-Package iTextSharp
PM> Install-Package itextsharp.xmlworker

```

نویسنده: هیمن صادقی

تاریخ: ۱۹:۰۰ ۱۳۹۳/۰۲/۲۵

درود

با سپاس از مطالب که در سایت قرار دادید یک مشکل داشتم
من کد رو در پروژه قرار دارم اما کد زیر که قرار متن راست به چپ کنه کار نمی‌کنه

```

_paragraph = new Paragraph
{
    Alignment = Element.ALIGN_LEFT // سبب می‌شود تا در حالت راست به چپ از سمت راست صفحه شروع شود
};

```

و کد زیر هم کار نمی‌کنه

```
fixNestedTablesRunDirection(element);
```

اگر لطف کنید من رو راهنمایی کنید

نویسنده: هیمن صادقی

تاریخ: ۲۲:۲۸ ۱۳۹۳/۰۲/۲۵

درود؛ پیوست پیام قبلی که گفتم کد کار نمی‌کنه: [rar.1](#)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۶ ۰:۲۹

- XML Worker از تمام امکانات CSS پشتیبانی نمی‌کند. لیست موارد پشتیبانی شده [در اینجا \(رنگ‌های سبز\)](#)
- در کد شما float: right و float: left دارید که مطابق لینک داده شده فعلاً پشتیبانی نمی‌شود.
- نکته‌ی تکمیلی « [برای رفع مشکل محو شدن Div، کدهای کلاس ElementsCollector مطلب جاری را به نحو زیر تغییر دهید](#) » را هم اضافه نکرده‌اید.
- کد cell.RunDirection = fixNestedTablesRunDirection مطلب جاری در کدهای شما به نمونه‌ای که PdfWriter.RUN_DIRECTION_RTL ندارد، تغییر پیدا کرده. بنابراین کار نخواهد کرد.

نویسنده: هومن صادقی
تاریخ: ۱۳۹۳/۰۲/۲۶ ۱:۱۹

نمونه از شما
تابع fixNestedTablesRunDirection در خط

```
if (table == null)
    return;
```

خاتمه پیدا می‌کند و کدی را که برداشتم تاثیر بر کد نداره. زمانیکه به صورت دستی کد زیر را به متن اضافه می‌کنیم

```
paragraph.Add("Data")
```

کار می‌کنه یعنی راست به چپ را درست می‌کند. اما زمانی که فایل html بهش میدم چپ به راست می‌باشد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۶ ۲:۰۹

متد Add را به این صورت اصلاح کنید تا جهت Paragraph ها را هم درست کند:

```
public void Add(IWritable htmlElement)
{
    var writableElement = htmlElement as WritableElement;
    if (writableElement == null)
        return;

    foreach (var element in writableElement.Elements())
    {
        if (element is PdfDiv)
        {
            var div = element as PdfDiv;
            foreach (var divChildElement in div.Content)
            {
                fixNestedTablesRunDirection(divChildElement);
                _paragraph.Add(divChildElement);
            }
        }
        else if (element is Paragraph)
        {
            var paragraph = element as Paragraph;
            paragraph.Alignment = Element.ALIGN_LEFT;
            _paragraph.Add(element);
        }
        else
        {
            fixNestedTablesRunDirection(element);
            _paragraph.Add(element);
        }
    }
}
```

نویسنده: وحید نصیری
تاریخ: ۲:۱۲ ۱۳۹۳/۰۲/۲۶

یک نکته‌ی مهم

از خروجی GetBuffer استریم نباید استفاده شود:

```
return File(memoryStream.GetBuffer(), "application/pdf", "Test.pdf");
```

باید از ToArray استفاده کنید تا حاوی اضافات بافر نباشد (نمایش پیغام ذخیره تغییرات در adobe reader به همین دلیل اضافات است):

```
return File(memoryStream.ToArray(), "application/pdf", "Test.pdf");
```

در این حالت حجم فایل نهایی هم نصف خواهد بود.

نویسنده: الیاس سررند
تاریخ: ۱۷:۳۸ ۱۳۹۳/۰۳/۰۷

سلام و خسته نباشید. میشه از این روش توی ASP.Net استفاده کرد؟ اگر آره در مورد دستور آخر Process.Start چه باید کرد؟ ممنون

نویسنده: وحید نصیری
تاریخ: ۱۷:۵۶ ۱۳۹۳/۰۳/۰۷

- مثال پیوست شده [کمی بالاتر](#) یک مثال ASP.NET MVC است.

- Process.Start را حذف کنید؛ نیازی نیست.

- به قسمت new FileStream آن دقت کنید. اینجا مسیر یک فایل را می‌شود مشخص کرد. فایل نهایی تولید شده در این مسیر نوشته می‌شود. از آن مسیر در برنامه‌های وب و ویندوز می‌توان استفاده کرد.

نویسنده: وحید نصیری
تاریخ: ۱۸:۳ ۱۳۹۳/۰۳/۰۷

به روز رسانی

کلیه نکات مطلب فوق را به همراه بهبودهای مطرح شده در نظرات آن، در پروژه‌ی ذیل می‌توانید به صورت یکجا دریافت و بررسی کنید:

[XMLWorkerRTLSample.zip](#)

نویسنده: مصطفی سلطانی
تاریخ: ۱۲:۲۳ ۱۳۹۳/۰۶/۰۱

با سلام

با تشکر از مطلب مفیدتان

من پروژه نمونه شما را دانلود کردم ولی داخل جدول مشکل راست به چپ فارسی را مشاهده می‌کنم. مثلاً لغت "متن" به صورت "ن ت م" نشان داده می‌شود.

نویسنده: وحید نصیری
تاریخ: ۱۳:۱۷ ۱۳۹۳/۰۶/۰۱

ابتدای متد Add فایل ElementsCollector.cs آن را به صورت زیر اصلاح کنید:

```
public void Add(IWritable htmlElement)
```

```
{
    var writableElement = htmlElement as WritableElement;
    if (writableElement == null)
        return;

    foreach (var element in writableElement.Elements())
    {
        if (element is NoNewLineParagraph)
        {
            var noNewLineParagraph = element as NoNewLineParagraph;
            foreach (var item in noNewLineParagraph)
            {
                fixNestedTablesRunDirection(item);
                _paragraph.Add(item);
            }
        }
        else if (element is PdfDiv)
```

پیشنیازها

- صفحه بندی و مرتب سازی خودکار اطلاعات به کمک jqGrid در ASP.NET MVC
- فعال سازی و پردازش جستجوی پویای jqGrid در ASP.NET MVC
- سفارشی سازی عناصر صفحات پویای افزودن و ویرایش رکوردهای jqGrid در ASP.NET MVC
- آشنایی با کتابخانه‌ی PDF Report

اضافه کردن دکمه‌ی خروجی به jqGrid

برای تهیه خروجی از jqGrid نیاز است بدانیم، اکنون در چه صفحه‌ای از اطلاعات قرار داریم؟ بر روی چه ستونی، مرتب سازی صورت گرفته‌است؟ بر روی کدام فیلدها با چه مقادیری جستجو انجام شده‌است؟ تا ... بتوانیم بر این مبنا، منبع داده‌ی موجود را فیلتر کرده و لیست نهایی را تبدیل به گزارش کنیم. گزارشی که دقیقاً با اطلاعاتی که کاربر در صفحه مشاهده می‌کند، تطابق داشته باشد.

خوشبختانه تمام این سؤالات توسط متد توکار excelExport در سمت سرور قابل دریافت است:

```
@section Scripts
{
    <script type="text/javascript">
    $(document).ready(function () {
        $('#list').jqGrid({
            caption: "آزمایش ششم",
            // مانند قبل
        }).navGrid(
            // مانند قبل
        ).jqGrid('navButtonAdd', '#pager', {
            caption: "", buttonicon: "ui-icon-print", title: "خروجی پی دی اف",
            onClickButton: function () {
                $('#list').jqGrid('excelExport', { url: '@Url.Action("GetProducts",
                    "Home")' });
            }
        });
    });
    </script>
}
```

8.000.000	1393/04/07	نام 9	9	9
خروجی پی دی اف	1393/04/06	نام 10	10	10

نمایش 1 - 10 از 500 صفحه 1 از 50 < > >> >>>

در اینجا توسط متد navButtonAdd یک دکمه‌ی جدید را اضافه کرده‌ایم که کلیک بر روی آن سبب فراخوانی متد excelExport و ارسال اطلاعات گزارش به url تنظیم شده‌است. باید دقت داشت که این اطلاعات از طریق Http Get به سرور ارسال می‌شوند و دقیقاً اجزای آن همان اجزای جستجوی پویای jqGrid است:

```
public ActionResult GetProducts(string sidx, string sord, int page, int rows,
    bool _search, string searchField, string searchString,
    string searchOper, string filters, string oper)
```

با این تفاوت که یک oper نیز به مجموعه‌ی پارامترهای ارسالی به سرور اضافه شده‌است. این oper در اینجا با excel مقدار دهی می‌شود.

البته چون تعداد این پارامترها بیش از اندازه شده‌است، بهتر است آن‌ها را تبدیل به یک کلاس کرد:

```
namespace jqGrid06.Models
{
    public class JqGridRequest
    {
        public string sidx { set; get; }
        public string sord { set; get; }
        public int page { set; get; }
        public int rows { set; get; }
        public bool _search { set; get; }
        public string searchField { set; get; }
        public string searchString { set; get; }
        public string searchOper { set; get; }
        public string filters { set; get; }
        public string oper { set; get; }
    }
}
```

و متد جستجوی پویا را به نحو ذیل بازنویسی نمود:

```
public ActionResult GetProducts(JqGridRequest request)
{
    var list = ProductDataSource.LatestProducts;

    var pageIndex = request.page - 1;
    var pageSize = request.rows;
    var totalRecords = list.Count;
    var totalPages = (int)Math.Ceiling(totalRecords / (float)pageSize);

    var productsQuery = list.AsQueryable();

    productsQuery = new JqGridSearch().ApplyFilter(productsQuery, request, this.Request.Form);
    productsQuery = productsQuery.OrderBy(request.sidx + " " + request.sord);

    if (string.IsNullOrEmpty(request.oper))
    {
        productsQuery = productsQuery
            .Skip(pageIndex * pageSize)
            .Take(pageSize);
    }
    else if (request.oper == "excel")
    {
        productsQuery = productsQuery
            .Skip(pageIndex * pageSize);
    }

    var productsList = productsQuery.ToList();

    if (!string.IsNullOrEmpty(request.oper) && request.oper == "excel")
    {
        new ProductsPdfReport().CreatePdfReport(productsList);
    }

    var productsData = new JqGridData
    {
        Total = totalPages,
        Page = request.page,
        Records = totalRecords,
        Rows = (productsList.Select(product => new JqGridRowData
        {
            Id = product.Id,
            RowCells = new List<string>
            {
                product.Id.ToString(CultureInfo.InvariantCulture),
                product.Name,
                product.AddDate.ToPersianDate(),
                product.Price.ToString(CultureInfo.InvariantCulture)
            }
        })).ToArray()
    };
};
```

```
return Json(productsData, JsonRequestBehavior.AllowGet);
}
```

توضیحات:

اکثر قسمت‌های این متد با متدی که در مطلب « [فعال سازی و پردازش جستجوی پویای jqGrid در ASP.NET MVC](#) » مشاهده کردید یکی است؛ برای مثال order by با استفاده از کتابخانه‌ی Dynamic LINQ به صورت پویا عمل می‌کند و متد ApplyFilter، کار تهیه where پویا را انجام می‌دهد. فقط در اینجا بررسی و پردازش پارامتر oper نیز اضافه شده‌است. اگر این پارامتر مقدار دهی شده باشد، یعنی نیاز است کل اطلاعات را واکنشی کرد؛ زیرا می‌خواهیم گزارش گیری کنیم و نه اینکه صرفاً اطلاعات یک صفحه را به کاربر بازگشت دهیم. همچنین در اینجا List نهایی فیلتر شده به یک گزارش Pdf Report ارسال می‌شود. این گزارش چون نهایتاً اطلاعات را در مرورگر کاربر Flush می‌کند، کار به اجرای سایر قسمت‌ها نخواهد رسید و همینجا گزارش نهایی تهیه می‌شود.



کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

[jqGrid06.7z](#)

نظرات خوانندگان

نویسنده: سروش
تاریخ: ۹:۳۸ ۱۳۹۳/۰۴/۱۷

با سلام من هنگام اجرای پروژه با خطای زیر روبرو میشم

```
Could not load file or assembly 'System.Web.Mvc, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35' or one of its dependencies. The system cannot find the file specified.
```

نویسنده: وحید نصیری
تاریخ: ۹:۵۴ ۱۳۹۳/۰۴/۱۷

- ابتدا به اینترنت وصل شوید.
- سپس در خط فرمان پاورشل نیوگت دستور زیر را اجرا کنید:

```
PM> update-package -reinstall
```

به این صورت بسته‌های MVC 5 آن به صورت صحیح به پروژه اضافه می‌شوند.

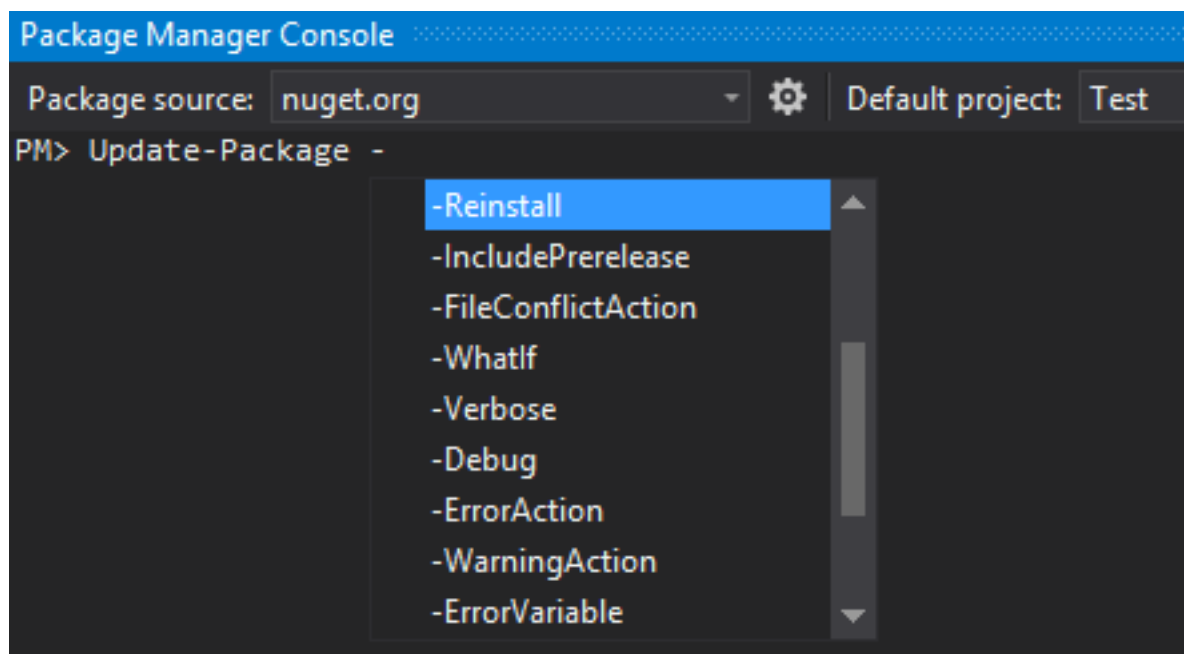
نویسنده: سروش
تاریخ: ۱۷:۵۷ ۱۳۹۳/۰۴/۱۷

متأسفانه با تایپ فرمان بالا پیغام زیر صادر می‌گردد

```
Update-Package : A parameter cannot be found that matches parameter name 'reinstall'.  
At line:1 char:26  
+ update-package -reinstall <<<<  
+ CategoryInfo          : InvalidArgument: (:) [Update-Package], ParameterBindingException  
+ FullyQualifiedErrorId : NamedParameterNotFound,NuGet.PowerShell.Commands.UpdatePackageCommand
```

نویسنده: وحید نصیری
تاریخ: ۱۹:۰۰ ۱۳۹۳/۰۴/۱۷

دریافت آخرین نگارش نیوگت از اینجا [^](#) و [^](#)



نویسنده: سروش
تاریخ: ۱۱:۴۶ ۱۳۹۳/۰۴/۱۸

با سلام و ادب

در پروژه خروجی Excel بوسیله EPPPlus موجود نبود میشه راهنمایی کنید

نویسنده: وحید نصیری
تاریخ: ۱۱:۵۲ ۱۳۹۳/۰۴/۱۸

اگر به تصویر آخر دقت کنید، خروجی اکسل کتابخانه‌ی [Pdf Report](#) در قسمت پیوست‌های فایل PDF تولیدی قرار می‌گیرد.

نویسنده: داوود
تاریخ: ۱۵:۲۷ ۱۳۹۳/۰۴/۲۷

با سلام

به نظر فابل zip این مطلب دچار مشکل است بعد از دانلود نتوانستم آن را اکستراکت کنم
درضمن پسوند عجیبی دارد (jqGrid06.7z)
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۷:۱۵ ۱۳۹۳/۰۴/۲۷

از برنامه‌ی [7zip](#) یا [winrar](#) استفاده کنید.

نویسنده: جواد وکیلی
تاریخ: ۱۵:۷ ۱۳۹۳/۰۵/۱۴

یه مشکلی که این گرید با راست به چپ دارد نمایش اشتباه تعداد رکوردها هنگامی که از هزار بیشتر می‌شود

5	نام 9	9	9
5	نام 10	10	10
نمایش 1 - 10 از 500 >> صفحه 1 از 150			

تصویر بالا تعداد هزار و پانصد می باشد .

نویسنده: وحید نصیری
تاریخ: ۱۵:۵۱ ۱۳۹۳/۰۵/۱۴

[پیش فرض های](#) آن قابل سفارشی سازی است:

```
$.jgrid.formatter.integer.thousandsSeparator = ',';
$.jgrid.formatter.number.thousandsSeparator = ',';
$.jgrid.formatter.currency.thousandsSeparator = ',';
```

این سطرها را پیش از تعریف گرید قرار دهید.