عنوان: **نحوه ایجاد فیلدهای محاسباتی در PdfReport**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۳ ۲۱:۵۲
آدرس: www.dotnettips.info
برچسب‌ها: PdfReport

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("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);
                });
            })
            .Export(export =>
            {
                export.ToExcel(footer: "Footer text", header: "&24&U&\"Arial,Regular Bold\" New rpt.",
pageLayoutView: true);
                export.ToXml();
            })
            .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 2, 1, 1 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 دقیقا 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("ف.م.");
                    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 منحصربفرد باشند 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 CellDataها 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("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);
            });
```

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 آن‌را 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).

گزارش جدید ما

ردیف	شماره	نام	نام خانوادگی	ف.م.	موجودی	ف.م.
				نقل از صفحه قبل	۲۳۶,۷۹۱	۷۱۰,۳۷۳
۲۱۵	۲۱۴	نام ۲۱۴	نام خانوادگی ۲۱۴	نام ۲۱۴ - نام خانوادگی ۲۱۴	۱,۲۱۴	۳,۶۴۲
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	۷,۲۹۹	۲۱,۸۹۷
				جمع	۲۴۴,۰۹۰	۷۳۲,۲۷۰