عنوان: **گروه بندی اطلاعات و گزارشات Master-Details در PdfReport**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۵ ۱:۲۰
آدرس: www.dotnettips.info
گروه‌ها: PdfReport

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:

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 StronglyTypedListها. 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 آن‌را 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;
                });
```

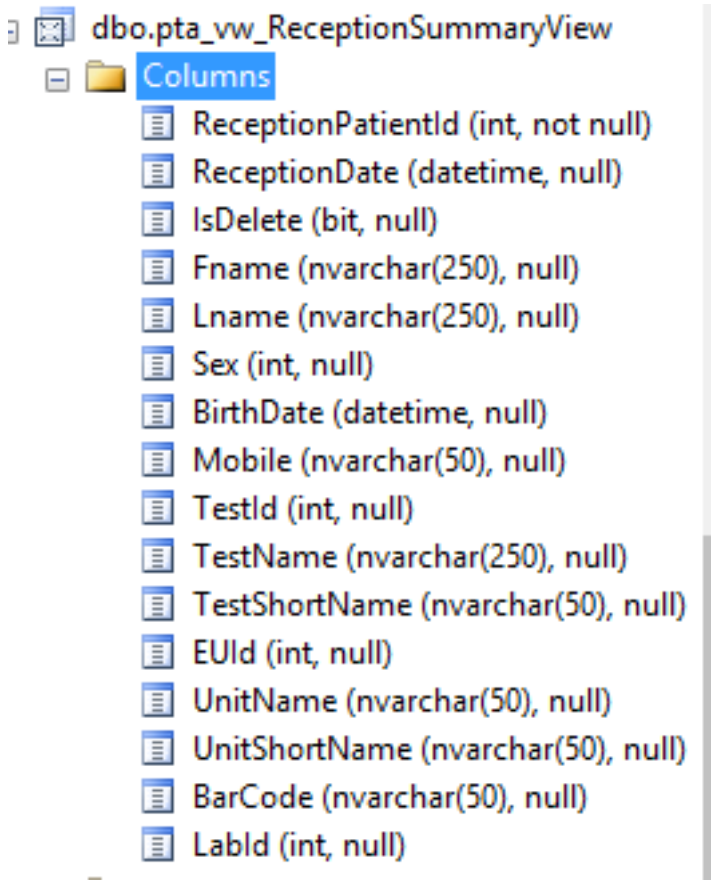
ساختار آن هم بسیار ساده است. توسط newGroupInfo می‌توان به اطلاعات گروه جدید، دسترسی یافت. برای مثال در اینجا اطلاعات والد گروه جدید در حال تهیه، دریافت شده و سپس در ردیف‌های یک جدول دو ستونه درج می‌شود. در ستون اول آن یک برچسب و در ستون دوم، مقدار دریافتی نمایش داده شده است و همینطور الی آخر برای سایر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

نظرات خوانندگان

نویسنده: ح. مرتضوی
تاریخ: ۱۳۹۱/۱۱/۲۷ ۱۸:۲۴

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: ۱۳۹۱/۱۱/۲۷ ۱۸:۳۹

- 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;
                    });
```

- ضمنا الزاما نیازی 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.
- لطفا برای سؤالات بعدی از قسمت پرسش و پاسخ مرتبط با این پروژه در سایت استفاده کنید.

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

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 می‌گیریم.
علت چیست؟
با تشکر

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

- 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 می‌کنید، قابل پردازش هستند یا خیر. یک break point روی آن قرار دهید و محتوای آن‌را بررسی کنید.