FastReport.NET, creating reports from code.

Usually, when programming using the FastReport.NET library, there is need to load the report template from a file, show the user in the preview window, and give the user the chance to change the report template, immediately print or export the report.

However, sometimes there can be a need to create a report template from code. In this article, I will try to look at an example of creating a report from C# code.

To start with, we need to imagine how the report should be. So, what should be in the report:

1. One page.
2. Report header.
3. Page header.
4. Page heading, containing 6 columns.
5. Table with data, received from a demo database (nwind.xml).
6. Page footer.
7. Report footer.

Now, let's go to the creation of the report.

First, we need to create a report, create the data source and register it:

```
report = new Report();
dataset = new DataSet();
dataset.ReadXml(@"..\..\nwind.xml");
report.RegisterData(dataset);
report.GetDataSource("Products").Enabled = true;
```

Here, the products table from the demo database is used as the data source.

In the following code fragment, we will create the report page:

```
ReportPage page = new ReportPage();
page.CreateUniqueName();
page.TopMargin = 10.0f;
page.LeftMargin = 10.0f;
page.RightMargin = 10.0f;
page.BottomMargin = 10.0f;
report.Pages.Add(page);
```

In the second code row, a unique name for the page is created. The method CreateUniqueName () is accessible in any FastReport.NET component. As a result, the report page got the name "Page1". If you want

to give the page a different name, for example “FirstPage”, then, instead of using the CreateUniqueName method, you can use the Name property:

```
page.Name = "FirstPage";
```

In the following four code rows (by using the properties TopMargin, LeftMargin, RightMargin and BottomMargin) the page fields are being given, each one is given 1 cm.Since by default, the page is created with fields of the same size, then in this case using these properties is not obligatory.They are useful, when there is a need to give fields certain sizes not 1 cm. Pay attention that the property saves the size of the fields in millimeters.

In the last code rode, the created page is added to the report.

And let’s create the report header, via the ReportTitle page property:

```
page.ReportTitle = new ReportTitleBand();
page.ReportTitle.CreateUniqueName();
page.ReportTitle.Height = 4.0f * Units.Centimeters;
```

Everything is clear with the first two rows, but the third needs to be understood .In this row the header height is given as 4 cm. Since the property stores height in pixels, then there is need to use Units.Centimeters – number o f pixels in one centimeter.In order to rename, we need to connect to FastReport.Utils.

We create the text object in the header:

```
TextObject titleText = new TextObject();
titleText.CreateUniqueName();
titleText.Left = 1.0f * Units.Centimeters;
titleText.Top = 1.0f * Units.Centimeters;
titleText.Width = 17.0f * Units.Centimeters;
titleText.Height = 2.0f * Units.Centimeters;
titleText.HorzAlign = HorzAlign.Center;
titleText.VertAlign = VertAlign.Center;
titleText.Font = new Font("Chiller", 32.0f, FontStyle.Bold);
titleText.TextColor = Color.DarkGreen;
titleText.FillColor = Color.DarkOrange;
titleText.Border.Color = Color.DarkOrchid;
titleText.Border.Lines = BorderLines.All;
titleText.Border.Width = 4.0f;
titleText.Text = "Report from code demo";
page.ReportTitle.Objects.Add(titleText);
```

Here, the position of the object relative to the report header is indicated via the properties Left and Top, but the size is given with the help of the width and height properties. The following two rows set the horizontal and vertical aligning of the text in the object. The font of the text can be chosen with the Font property.In this case, the name, size and style of the font is shown.The following three rows give the: text color,filled text object color and the color of its border. The last two rows show which border will be visible and its width in pixels.In the last row, the text is given. And the last row of the text object is added to the report header.

Page header is given in the same way. But apart from the page header text, six text objects which are table headings are created in it. Need to point out that, report header and page header are similar in many aspects.The main important difference is that, report header is printed on the first page of the report, and page header on every page.

Need to pay attention to the code creating the page header text object:

```
TextObject headerText = new TextObject();
headerText.CreateUniqueName();
headerText.Bounds = new RectangleF(0.0f, 0.5f * Units.Centimeters,
    19.0f * Units.Centimeters, 1.0f * Units.Centimeters);
headerText.HorzAlign = HorzAlign.Center;
headerText.VertAlign = VertAlign.Center;
headerText.Font = new Font("Book Antique", 16.0f,
    FontStyle.Bold | FontStyle.Italic);
headerText.TextColor = Color.Teal;
headerText.FillColor = Color.YellowGreen;
headerText.Border.Lines = BorderLines.All;
headerText.Border.TopLine.Color = Color.Indigo;
headerText.Border.LeftLine.Color = Color.Gold;
headerText.Border.RightLine.Color = Color.Gold;
headerText.Border.BottomLine.Color = Color.Indigo;
headerText.Border.TopLine.Width = 3.0f;
headerText.Border.LeftLine.Width = 2.0f;
headerText.Border.RightLine.Width = 2.0f;
headerText.Border.BottomLine.Width = 3.0f;
headerText.Text = TableName + " Table";
page.PageHeader.Objects.Add(headerText);
```

The Bounds property, allows creating a rectangle, which describes object border.that is.it's possible to give the position, width and height of the object with one row. And also a row in which the color and border width of the text object is given.Interesting that, it s possible to set the color of each of the borders separately.

Now we create the databand:

```
DataBand band = new DataBand();
page.Bands.Add(band);
band.CreateUniqueName();
band.DataSource = report.GetDataSource("Products");
band.Height = 0.5f * Units.Centimeters;
```

Only the fourth row needs to be explained, in which the data source for the band is shown.

Further, we create six text objects in the band, which will be connected to the table fields.Let's get the creation of one of the objects, the rest will be created in the same manner:

```
TextObject bandText = new TextObject();
bandText.CreateUniqueName();
bandText.HorzAlign = HorzAlign.Center;
bandText.Bounds = new RectangleF(0.0f * Units.Centimeters, 0.0f,
    1.0f * Units.Centimeters, 0.5f * Units.Centimeters);
bandText.Border.Lines = BorderLines.All;
```

```
bandText.Text = "[Products.ProductID]";
band.AddChild(bandText);
```

The last two rows, the first of which creates a connection to the ProductID field in the Products table.And the second adds the text to the band. Giving the text object parent can also be done in one more way:

```
bandText.Parent = band;
```

The following code fragment creates the pagefooter with the text object in it:

```
page.PageFooter = new PageFooterBand();
page.PageFooter.CreateUniqueName();
page.PageFooter.Height = 0.5f * Units.Centimeters;
TextObject footerText = new TextObject();
footerText.CreateUniqueName();
footerText.HorzAlign = HorzAlign.Right;
footerText.VertAlign = VertAlign.Center;
footerText.Bounds = new RectangleF(0.0f, 0.0f,
    19.0f * Units.Centimeters, 0.5f * Units.Centimeters);
footerText.TextColor = Color.Teal;
footerText.FillColor = Color.YellowGreen;
footerText.Border.Lines = BorderLines.All;
footerText.Border.TopLine.Color = Color.Indigo;
footerText.Border.LeftLine.Color = Color.Gold;
footerText.Border.RightLine.Color = Color.Gold;
footerText.Border.BottomLine.Color = Color.Indigo;
footerText.Border.TopLine.Width = 3.0f;
footerText.Border.LeftLine.Width = 2.0f;
footerText.Border.RightLine.Width = 2.0f;
footerText.Border.BottomLine.Width = 3.0f;
footerText.Text = "Page [Page]";
page.PageFooter.Objects.Add(footerText);
```

Pay attention to the penultimate row.Page footer text (`"Page [Page]"`) is given in it. Here, apart from the Page text, Page – system variable, which stores the current report page is stored, is used.Its use allows to easly number the pages.

The last code fragment creates the report footer:

```
page.ReportSummary = new ReportSummaryBand();
page.ReportSummary.CreateUniqueName();
page.ReportSummary.Height = 4.0f * Units.Centimeters;
footerText = new TextObject();
footerText.CreateUniqueName();
footerText.Left = 1.0f * Units.Centimeters;
footerText.Top = 1.0f * Units.Centimeters;
footerText.Width = 17.0f * Units.Centimeters;
footerText.Height = 2.0f * Units.Centimeters;
footerText.HorzAlign = HorzAlign.Center;
footerText.VertAlign = VertAlign.Center;
footerText.Font = new Font("Chiller", 32.0f, FontStyle.Bold);
footerText.TextColor = Color.DarkGreen;
footerText.FillColor = Color.DarkOrange;
footerText.Border.Color = Color.DarkOrchid;
footerText.Border.Lines = BorderLines.All;
```

```
footerText.Border.Width = 4.0f;
footerText.Text = "Total Pages [TotalPages#]";
page.ReportSummary.Objects.Add(footerText);
```

Here in the penultimate row uses system variable TotalPages#,in which the total number of pages in the report is saved. The main difference between the report footer and page footer is just the same the difference between the report header and the page header, -page footer is printed on every page and report footer only on the last page.

On figs 1 and 2, report template is shown , and report pages in preview.

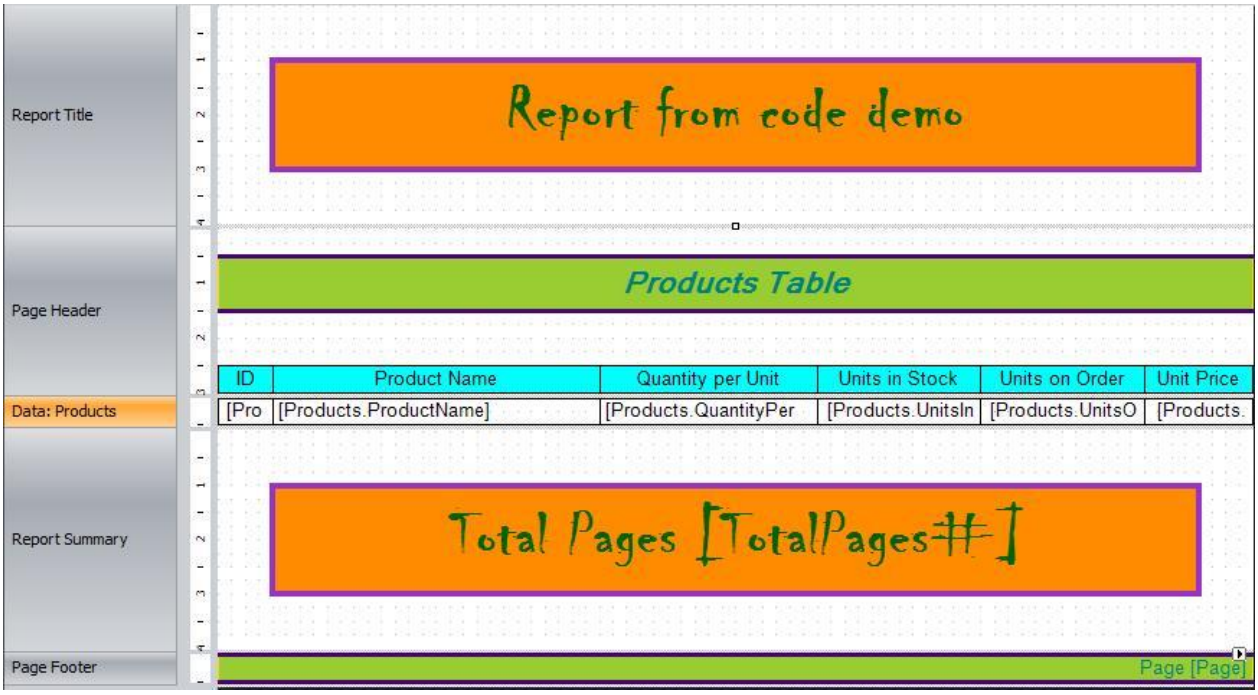
Report Title
Report from code demo
Page Header
Products Table
ID
Product Name
Quantity per Unit
Units in Stock
Units on Order
Unit Price
Data: Products
[Pro
[Products.ProductName]
[Products.QuantityPer
[Products.UnitsIn
[Products.UnitsO
[Products.
Report Summary
Total Pages [TotalPages#]
Page Footer
Page [Page]

Fig 1.


Report from code demo
Products Table
Products Table
Total Pages 2

Fig 2.

Adding dialogue window to a Report.

We looked at an example of creating a report from code.However, the possibilities of working with reports from coed in FastReport.NET is not limited. Nothing disturbs us from modifying reports which already exist.Lets examine this by looking at an example of adding a dialogue window into a report, and this on its own is an interesting task.

In order to concentrate on the addition of the dialogue, we will create a very simple report containing empty pages in the designer. To do this, simply click menu File New, then select blank report in the drop down menu. The report is ready, you can save it.

And now, about the dialogue window. Its form is shown in fig 3.

Fig 3

Here everything is simple as well.The function for this window is as follows:

1. The window appears before the report preview window shows up.
2. If you click OK, then preview will be shown.
3. If you click Cancel, then we won't see the preview window.

Window creation code:

```
report.Load("report.frx");

DialogPage dialog = new DialogPage();

dialog.Name = "Dialog1";
dialog.Form.Size = new Size(125, 125);
report.Pages.Add(dialog);
```

The first code loads the report from the file which we have created before.For simplicity, the report was saved in the application folder, and otherwise, the load function will have to be given the full folder file.The following three rows create the dialogue window, gives it the name and size. The last row adds the form to the report page list.

And now, the OK button:

```
ButtonControl buttonOK = new ButtonControl();
buttonOK.Name = "ButtonOK";
buttonOK.Location = new Point(25, 25);
buttonOK.Size = new Size(75, 25);
buttonOK.Text = "OK";
buttonOK.DialogResult = DialogResult.OK;
dialog.Controls.Add(buttonOK);
```

Everything is simple here. We create a button, give it a name, position in the window, size and the text on the button. The last two rows are more interesting.The first one determines the standard behavior of the OK button. The second adds the button to the dialogue window.

The cancel button is created in the same way:

```
ButtonControl buttonCancel = new ButtonControl();
buttonCancel.Name = "ButtonCancel";
buttonCancel.Location = new Point(25, 55);
buttonCancel.Size = new Size(75, 25);
buttonCancel.Text = "Cancel";
buttonCancel.DialogResult = DialogResult.Cancel;
dialog.Controls.Add(buttonCancel);
```

Need to point out that, the DialogPage and ButtonControl classes,are located within the FastReport.Dialog area.That is why,in order to have access to it, need to coneect to it by via the word using, or by showing the name.

Adding a connection to a report.

Also, you can add a new connection to the report. We will look at this possibility by using an empty report from the previous report.

We need to add the connection to th edemo database(demo.mdb), which is conatained in Fast Reports products called FastReport Studio.

And so, the code:

```
report.Load("report.frx");

MsAccessDataConnection connection = new MsAccessDataConnection();
connection.Name = "Connection1";
connection.DataSource = "demo.mdb";
report.Dictionary.Connections.Add(connection);
connection.CreateAllTables();
report.GetDataSource("customer").Enabled = true;
```

There isnt anything complicated here. We load the report. Create the connection, show its name, path to the database (which is located in the application folder), register the connection in the report. The penultimate row loads the whole table.And the last row makes the customer table accesible in the report.

The MsAccessDataConnection class is located in FastReport.Data.