Creating reports in FastReport.Net

FastReport.Net library– This is a powerful means of creating reports in the .Net Framework. Contrary to the many similar products, FastReport.Net contains embedded **designer, which is accessible to the end user. This gives the developer the ability to create reports in several methods which we will consider in this article.**

Method 1. In Visual Studio

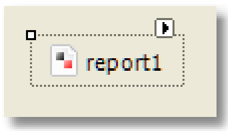
This is the simplest method; it suits better those who have just started mastering FastReport.

We will look at a typical case of using report components in the visual studio environment. During this, the data is taken from a typed source.

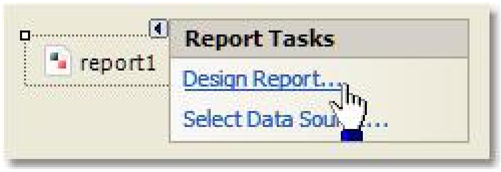
- **create a Windows Forms Application project;**
- **add into it data source (in menu"Data|Add New Data Source...");**
- **switch to the form designer;**
- **add the DataSet component from the "Data" category, onto the form and connect it to the typed data source, which you have created.**

In order to create the report, do the following:

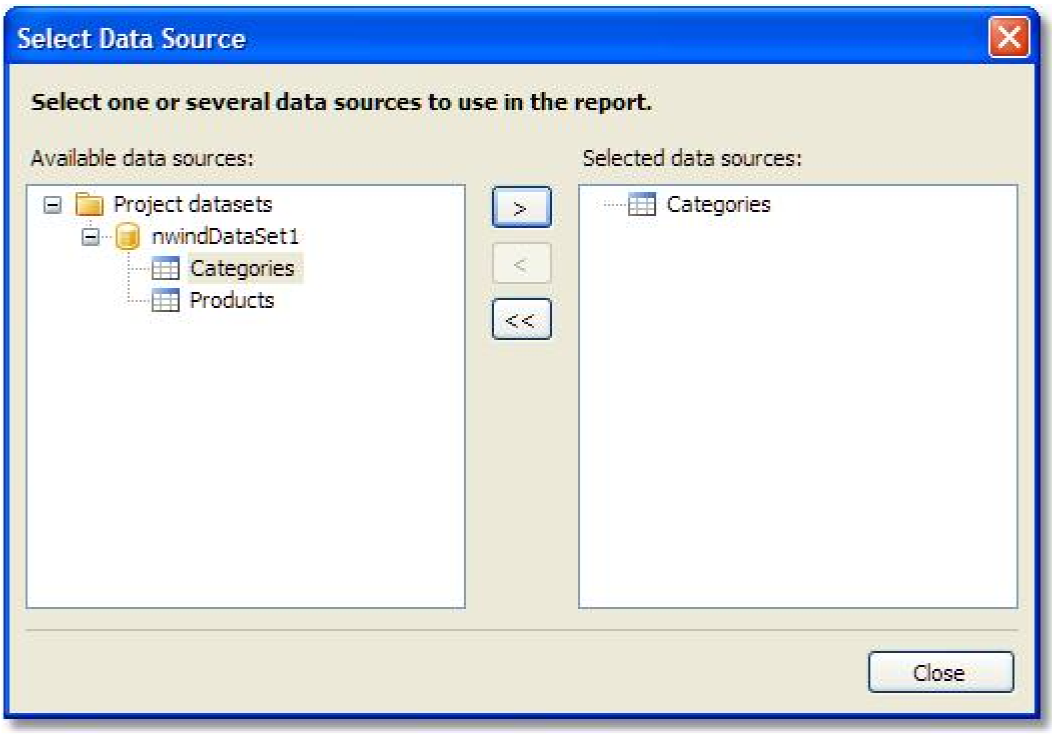
- **place the Report component on the form:**



- **right click the component , or click the "smart tag" button and, select "Design Report...":**



- **you will be asked to select the data source for the report. Select the needed table (one or several):**



- create the report. You can read more about this in the "Users manual ";
- close the report designer;
- add a Button on the form and double click it;
- in the event handler, write:

```
report1.Show();
```

- save the project and run it. When clicking on the button, the report will be built.

This method has got some shortcomings. Firstly, in Visual Studio design-time, you are denied the ability to work with "Live" data. This especially affects working with ASP.NET, and also using business objects as data. This implies that viewing the created report can be done only by running your application.

Secondly, methods of working and saving reports are not optimal. For every report, a different report component needs to be created, reports are saved in application's resource. This can be a problem if there are many reports in your application. However, this lacking can be easily circumvented:

1. Create a report written in the above method;
2. In the report designer, click the "Save" button and save the report in a file;
3. Delete the report component from the form;
4. Use the following code to load and run the report:

```
using (Report report = new Report())
{
  report.Load("your_report.frx");
  report.RegisterData(dataSet1, "dataSet1");
  report.Show();
}
```

Here, you can register data by using the report.RegisterData method. Need to indicate the data source, which was used when creating the report (DataSet or BindingSource), and its name as well.

Method 2. In the Designer.exe application

In the FastReport package, there is a Designer.exe application, which can be used for creating reports. It can be run from the menu:"Run /Programs/FastReports/FastReport.Net/Designer", or by double clicking the report file. Prepared report can be saved in the .frx file.

This method has got some lacking – you cannot work with data identified in your application. It suits better when creating offline reports, which connects to the database on its own. You can add data to the report by using the menu "Data/New datasource…". You can read more about this in the "Users manual ", in the chapter "working with data ".

If your application uses the DataSet/DataTable object for working with data from the database, it is possible to send it to the report, built in the Designer.exe. To do this:

1. **Create a new report in the Designer.exe;**
2. **Add data into it by using menu "**Данные**/**Новый источник данных**…";**
3. **Save the report in the .frx file;**
4. **Write the following code in your application:**

```
using (Report report = new Report())
{
  report.Load("your_report.frx");
  report.RegisterData(dataSet1.Tables["Employees"], "Employees");
  report.Show();
}
```

When registering data by using report.RegisterData, the data called Employees which is contained in the report, will be replaced by the data from your DataTable object. During this, it's important that the structures of the old and new sources are the same.

Method 3. Calling the designer from code

You can call the report designer from your application, which gives you a chance to create a new or change an existing report, without recompiling your application.

To create the report, do the following:

```
using (Report report = new Report())
{
  // register any data, which can be needed in the report
  report.RegisterData(...);
  report.Design();
}
```

This code can be placed in the click event handler of any button in your application. When clicking the button; you run the report designer, create the report and save it in file.After the whole report has been created, the button which calls the designer can be removed.

In order to use in later, do the following:

```
using (Report report = new Report())
{
  report.Load ("your_report.frx");
  // register the same data just like when calling the designer
  report.RegisterData(...);
  report.Show();
}
```

The same "live" data is used when creating and running the report. This means that, when creating the report, you can view the data, and correct the report immediately.

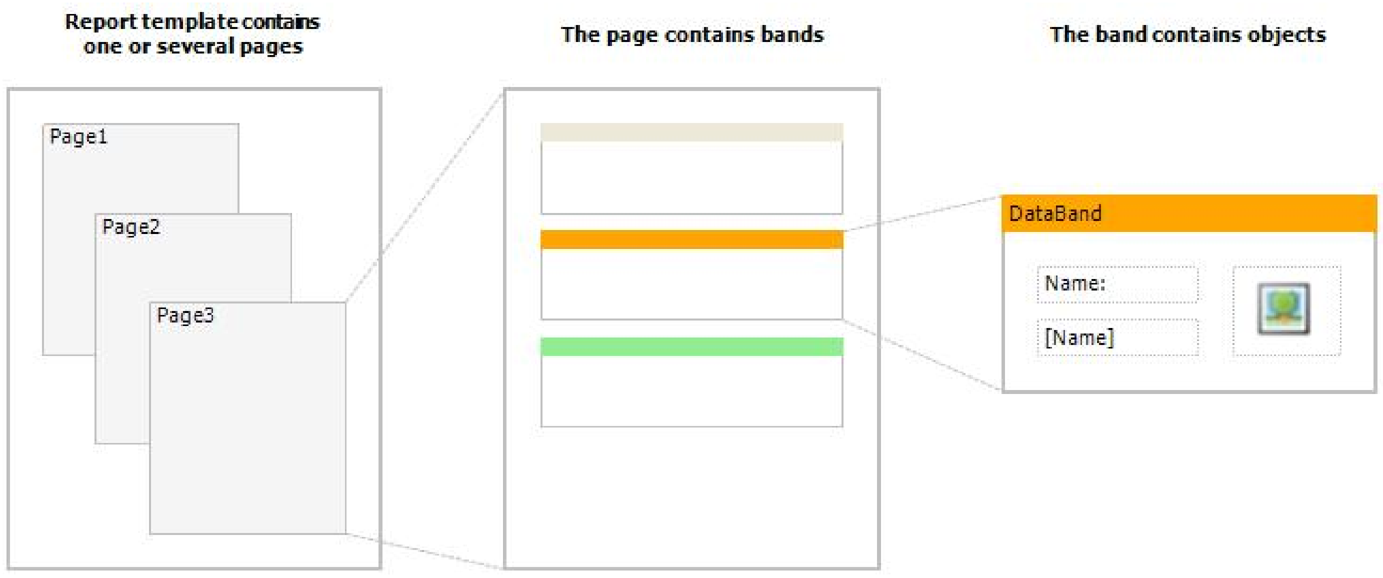
This method hasn't got any lacking. It is recommended to be the main method for working with FastReport fully. So, all the demonstrated reports from Demo.exe were created in itself.

Method 4 Creating a report in code

Many reports, which you will create, have got a structure already known to you. Set of fields in the report, their positions, data grouping and many more –remain unchanged. Such reports can be created by using the designer, in any of the methods written above.

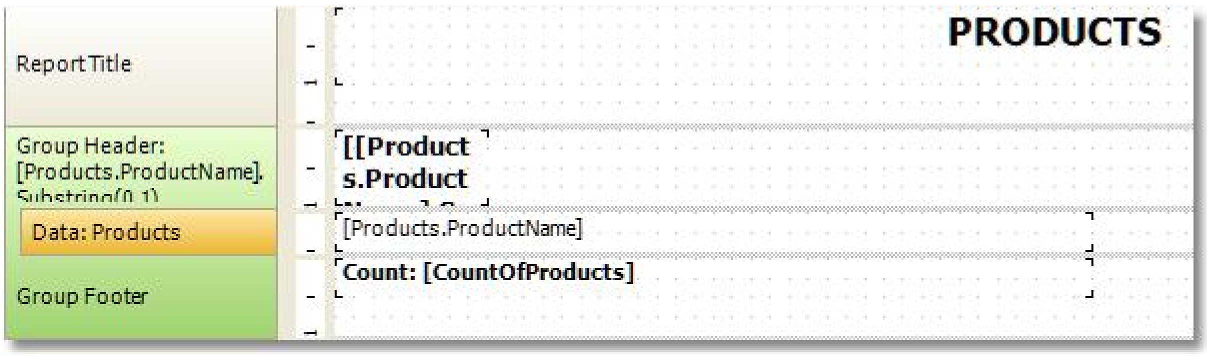
But sometimes the report structure is not known beforehand. For example, you need to create a report, whose interior is configured by the user with the help of the master. In this case, the report can be created with the help of code. This is a laborious method, since there is need to create each report element and configure it properties.

The report consists of one (often) or several pages having the ReportPage type. Report pages in turn, contain bands. On the band, there are report objects like "Text", "Picture" and others.


Report template contains one or several pages
The page contains bands
The band contains objects
Page1
Page2
Page3
DataBand
Name:
[Name]

Example 1. Creating a new report

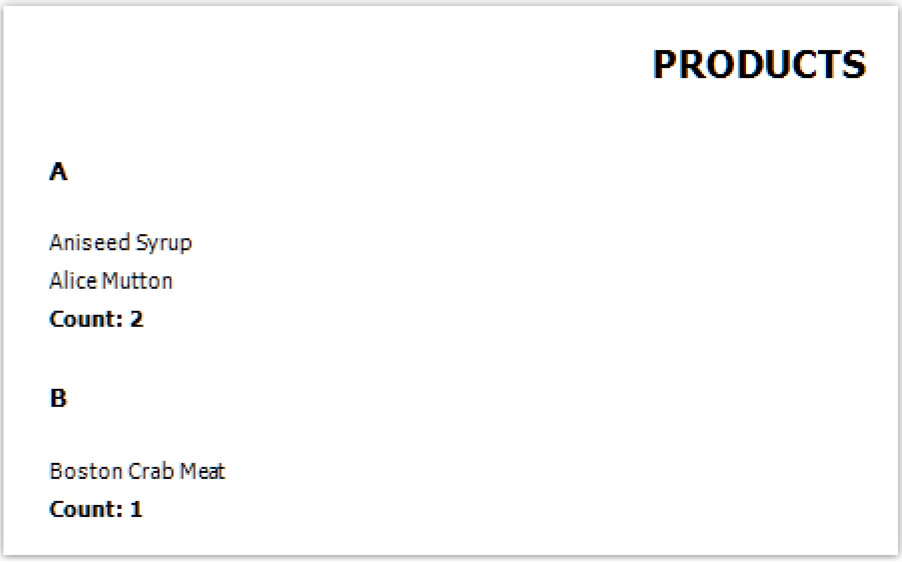
Lets look at an example of creating a report from group having the following type:



```
Report report = new Report();

// register the "Products" table
report.RegisterData(dataSet1.Tables["Products"], "Products");
// enable it to use in a report
report.GetDataSource("Products").Enabled = true;

// create A4 page with all margins set to 1cm
ReportPage page1 = new ReportPage();
page1.Name = "Page1";
report.Pages.Add(page1);

// create a ReportTitle band
page1.ReportTitle = new ReportTitleBand();
page1.ReportTitle.Name = "ReportTitle1";
// set its height to 1.5cm
page1.ReportTitle.Height = Units.Centimeters * 1.5f;

// create group header
GroupHeaderBand group1 = new GroupHeaderBand();
group1.Name = "GroupHeader1";
group1.Height = Units.Centimeters * 1;
// set group condition
group1.Condition = "[Products.ProductName].Substring(0, 1)";
// add group to the page.Bands collection
page1.Bands.Add(group1);

// create group footer
group1.GroupFooter = new GroupFooterBand();
group1.GroupFooter.Name = "GroupFooter1";
group1.GroupFooter.Height = Units.Centimeters * 1;

// create DataBand
DataBand data1 = new DataBand();
data1.Name = "Data1";
data1.Height = Units.Centimeters * 0.5f;
// set data source
data1.DataSource = report.GetDataSource("Products");
// connect databand to a group
group1.Data = data1;

// create "Text" objects

// report title
TextObject text1 = new TextObject();
text1.Name = "Text1";
// set bounds
text1.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 19, Units.Centimeters * 1);
// set text
text1.Text = "PRODUCTS";
// set appearance
text1.HorzAlign = HorzAlign.Center;
text1.Font = new Font("Tahoma", 14, FontStyle.Bold);
// add it to ReportTitle
page1.ReportTitle.Objects.Add(text1);

// group
TextObject text2 = new TextObject();
```

```
text2.Name = "Text2";
text2.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 2, Units.Centimeters * 1);
text2.Text = "[[Products.ProductName].Substring(0, 1)]";
text2.Font = new Font("Tahoma", 10, FontStyle.Bold);
// add it to the GroupHeader
group1.Objects.Add(text2);

// data band
TextObject text3 = new TextObject();
text3.Name = "Text3";
text3.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 10, Units.Centimeters * 0.5f);
text3.Text = "[Products.ProductName]";
text3.Font = new Font("Tahoma", 8);
// add it to DataBand
data1.Objects.Add(text3);

// group footer
TextObject text4 = new TextObject();
text4.Name = "Text4";
text4.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 10, Units.Centimeters * 0.5f);
text4.Text = "Count: [CountOfProducts]";
text4.Font = new Font("Tahoma", 8, FontStyle.Bold);
// add it to GroupFooter
group1.GroupFooter.Objects.Add(text4);

// add a total
Total groupTotal = new Total();
groupTotal.Name = "CountOfProducts";
groupTotal.TotalType = TotalType.Count;
groupTotal.Evaluator = data1;
groupTotal.PrintOn = group1.Footer;
// add it to report totals
report.Dictionary.Totals.Add(groupTotal);

// run the report
report.Show();
```

Prepared report will be like this:



PRODUCTS

A

Aniseed Syrup
Alice Mutton
Count: 2

B

Boston Crab Meat
Count: 1

Example 2. Adding elements to a report

In some cases, there is no need to create a report in code from "zero" –it's enough to make some changes to an existing report template. The following example shows, how to add a ReportTitle band having one "Text" object to an existing report:

```
Report report = new Report();
// load the report from a file
report.Load("myreport.frx");

// register the "Products" table
report.RegisterData(dataSet1.Tables["Products"], "Products");

// get the first report page
ReportPage page1 = report.Pages[0] as ReportPage;

// create ReportTitle band
page1.ReportTitle = new ReportTitleBand();
page1.ReportTitle.Name = "ReportTitle1";
// set its height to 1.5cm
page1.ReportTitle.Height = Units.Centimeters * 1.5f;

// create the Text object on the title
TextObject text1 = new TextObject();
text1.Name = "Text1";
// set bounds
text1.Bounds = new RectangleF(0, 0,
  Units.Centimeters * 19, Units.Centimeters * 1);
// set text
text1.Text = "PRODUCTS";
// set appearance
text1.HorzAlign = HorzAlign.Center;
text1.Font = new Font("Tahoma", 14, FontStyle.Bold);
// add it to ReportTitle
page1.ReportTitle.Objects.Add(text1);
```

Example 3. Creating a dialogue

The following example shows how to add a dialogue form having an "Ok" button to an existing report:

```
Report report = new Report();
// load the report from a file
report.Load("myreport.frx");

// create the dialog page
DialogPage dialog = new DialogPage();
dialog.Name = "Form1";
report.Pages.Add(dialog);

// create OK button
ButtonControl btnOk = new ButtonControl();
btnOk.Name = "btnOk";
btnOk.Location = new Point(12, 12);
btnOk.Size = new Size(75, 25);
btnOk.Text = "OK";
btnOk.DialogResult = DialogResult.OK;

// add it to the dialog
dialog.Controls.Add(btnOk);
```

Example 4. Creating a DataBase connection and DataSouce

The following example shows how to add a database connection to an existing report:

```
Report report = new Report();
// load the report from a file
report.Load("myreport.frx");

// create the connection object
MsAccessDataConnection conn = new MsAccessDataConnection();
conn.Name = "Connection1";
conn.DataSource = @"d:\crosstest.mdb";
report.Dictionary.Connections.Add(conn);

// create all connection tables and views
conn.CreateAllTables();
// enable the "cross" table
report.GetDataSource("cross").Enabled = true;
```