

Order ID	Date	Ship To	Ship Address	Freight
10248	7/4/1996	Vins et alcools Chevalier	59 rue de l'Abbaye	32.38
10249	7/5/1996	Toms Spezialitäten	Luisenstr. 48	11.61
10250	7/8/1996	Hanari Carnes	Rua do Paço, 67	65.83
10251	7/8/1996	Victuailles en stock	2, rue du Commerce	41.34
10252	7/9/1996	Suprêmes délices	Boulevard Tirou, 255	51.30
10253	7/10/1996	Hanari Carnes	Rua do Paço, 67	58.17
10254	7/11/1996	Chop-suey Chinese	Hauptstr. 31	22.98
10255	7/12/1996	Richter Supermarkt	Starenweg 5	148.33
10256	7/15/1996	Wellington Importadora	Rua do Mercado, 12	13.97

Image: 1.0

Introduction

We hear this all the time, “Two birds with one stone.” What if I say, “Four birds with one stone”? I am sure four sound much better than two. So, what are my four birds and one stone?

My four birds are four distinct different outputs generated using source *NorthWind->Orders (SQL Server 2000)* and my stone is single physical *Ms Reporting Services .rdlc* file, which I am using as template to produce different outputs. This particular figure of speech is perfectly applicable to the technique, which I am going to share with you.

The application of this technique is not something new; we all have done same or similar while dealing with reporting of data. What is new here is the approach, which I can call to reuse of report (as we commonly reuse the code).

Let us discuss a practical scenario here. If I ask you, what kind of output you see in (image 1.0); you would probably say a simple report listing orders information. Well, you guessed it right. What will you do if the end-users want same report using data grouped by *CustomerID*(image 1.1)? In most cases, you might end up writing a new report. In this article, I will demonstrate how to reuse the report to produce the demanded output without the need of writing a new report.

I assume the reader of this article is comfortable using Visual Studio 2005, C#, Sql Server 2000 and Windows Forms. Basic understanding of how report designer works is helpful to work with attached code.

Three button technique

To make life little more interesting, I added three extra buttons: *Orders by Customer*, *Orders by City* and *Orders by Country* to user interface. These three extra buttons on the user interface do not have any built in magic; they are just helping me demonstrate the technique. So, without further ado:

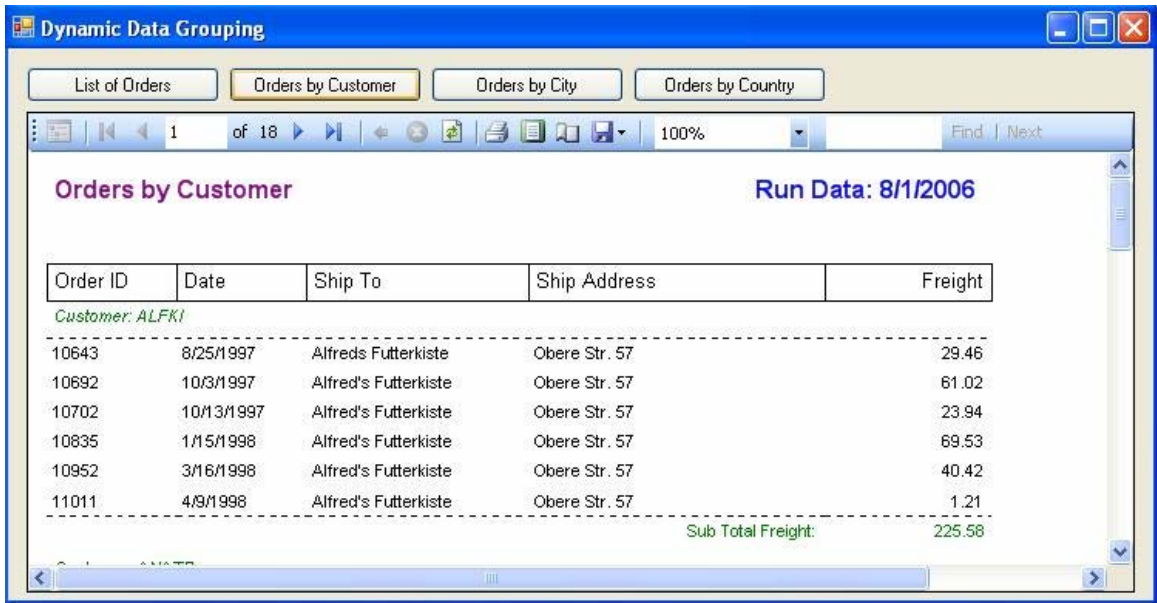


Image: 1.1

Remember the game “spot the difference” from your childhood memories? May I ask you to play the same game with Image 1.0 and 1.1? Sure, they look different; the first image has a title called “Orders List” and the second image has “Orders by Customer”, and so on...

If we pay a close attention then technically the difference is really the output format, the underlying data is same (orders information). By this time, I am sure most of you probably understand my technique and how it will help you generate multiple outputs using dynamic controls to generate reports.

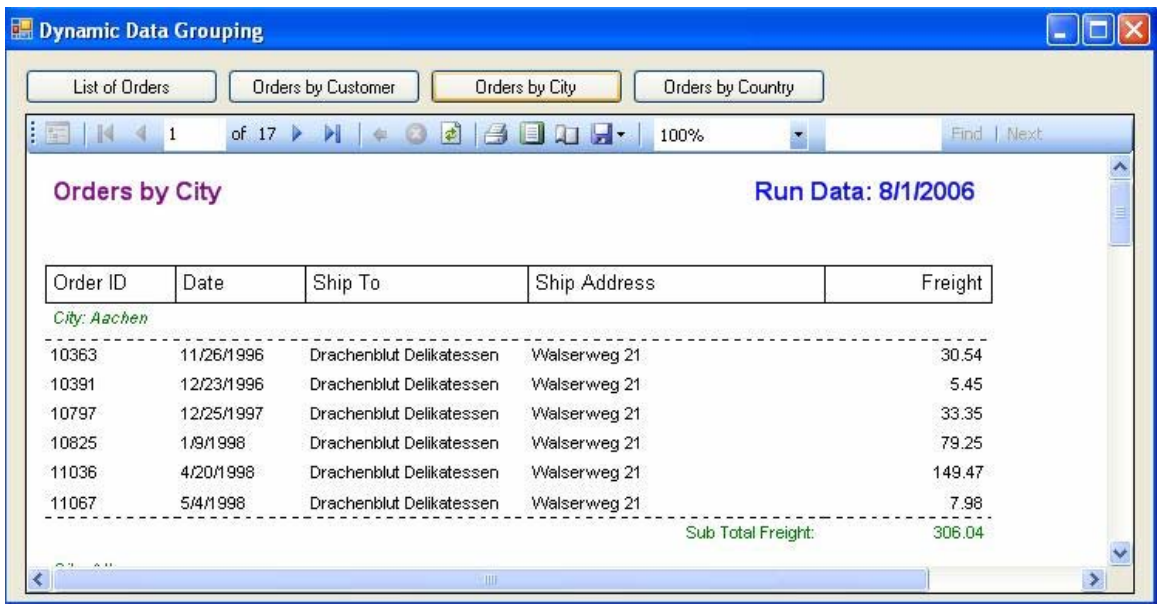
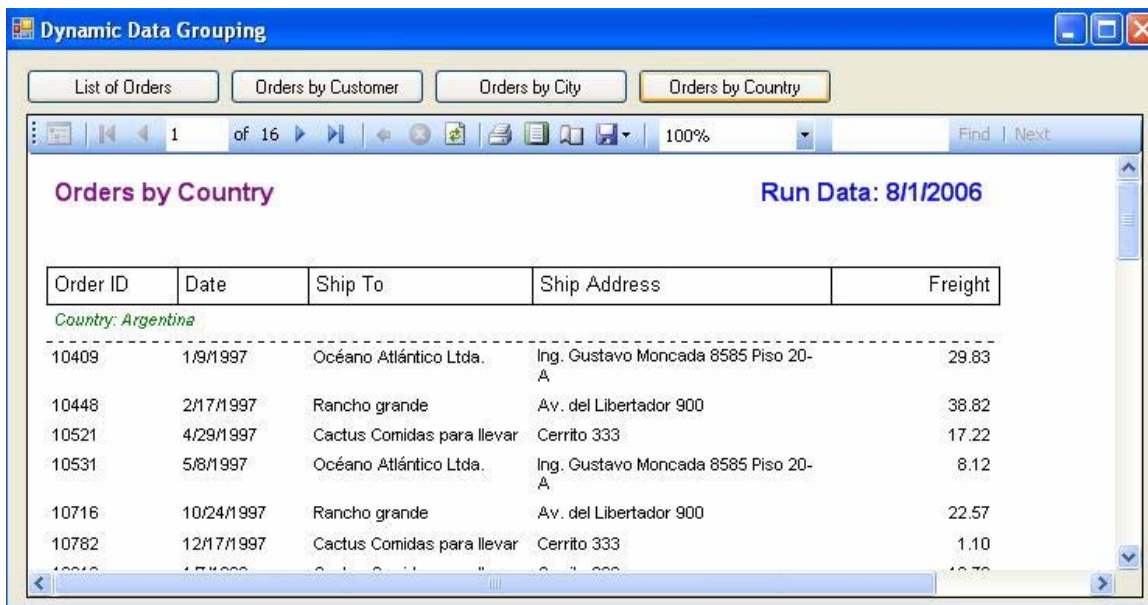


Image: 1.2



Order ID	Date	Ship To	Ship Address	Freight
<i>Country: Argentina</i>				
10409	1/9/1997	Océano Atlántico Ltda.	Ing. Gustavo Moncada 8585 Piso 20-A	29.83
10448	2/17/1997	Rancho grande	Av. del Libertador 900	38.82
10521	4/29/1997	Cactus Comidas para llevar	Cerrito 333	17.22
10531	5/8/1997	Océano Atlántico Ltda.	Ing. Gustavo Moncada 8585 Piso 20-A	8.12
10716	10/24/1997	Rancho grande	Av. del Libertador 900	22.57
10782	12/17/1997	Cactus Comidas para llevar	Cerrito 333	1.10
10812	1/7/1998	Cactus Comidas para llevar	Cerrito 333	10.70

Image: 1.3

How can one report produce four different outputs?

Imagine you're asked to develop a Sales Order System; one of the reporting requirements involved is to produce four different reports to calculate freight paid for all shipped orders.

In a typical scenario, you will create four individual reports. Well, nothing wrong with this approach, we have done this in past. However, since we do lot of code reusing, why not try to reuse a report by creating a template that we can use to generate different outputs.

This brought Microsoft Reporting Service to my attention. I am having a fun time doing this report reuse business. I though, let me share this with my friends here with a hope that it helps you the way it helped me.

The key to make your report generate more then one output is to create a well designed template. Here is what I did to generate four different outputs from this one report.

Report designs considerations

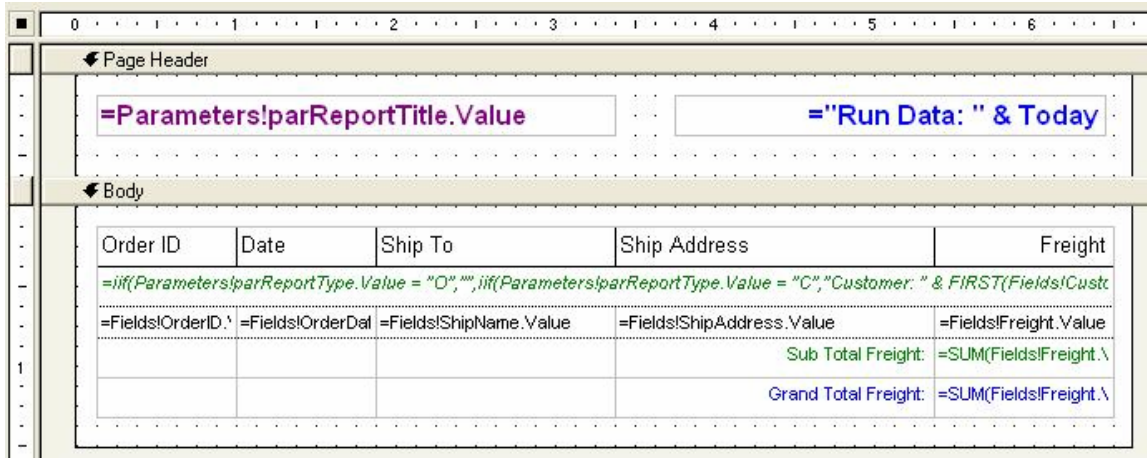


Image: 1.4

A carefully designed template is required to create a multiple reports with different outputs. We will start by making use of *Table Control*; first, we have to identify and lay down all the details columns that are common to all outputs. Please check the report in attached code for details of formatting etc.

Dynamic data grouping

The way information is grouped together will affect the output of the report (apart from the detail section). Now, you would start to wonder how I could change data grouping during runtime to create a different output.

Well, the solution to this problem is introducing some intelligence into our report, using a rendering engine that can leverage on and produce the desired output. I am making use of following two parameters to pass onto report so it can act differently.

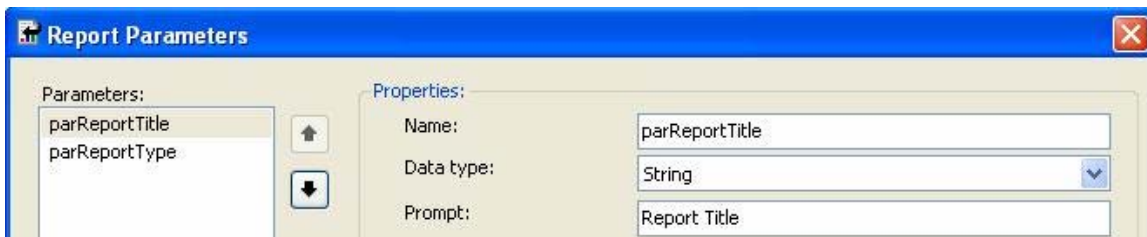


Image: 1.5

I will make use of *parReportType* parameter to pass following four values: *O-Orders*, *C-Customer*, *S-City* and *T-Country*. This one letter type (*O,C,S,T*) provided as dynamic value to group the data before producing the output.

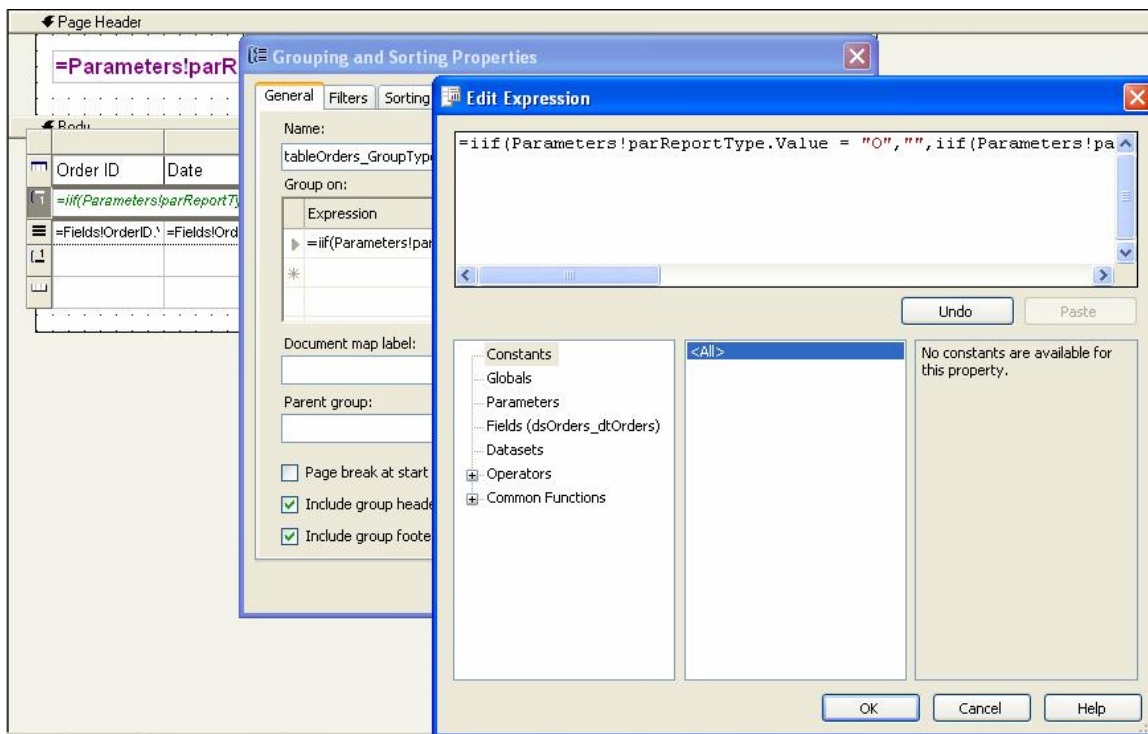


Image: 1.6

Our dose of intelligence to the report designer is nothing but following *Grouping Expression*, which changes data group based on information supplied through *parReportType*:

```
=iif(Parameters!parReportType.Value = "O", "",
iif(Parameters!parReportType.Value = "C", Fields!CustomerID.Value,
iif(Parameters!parReportType.Value = "S",
Fields!ShipCity.Value, Fields!ShipCountry.Value)))
```

If you are not sure what *iif()* is, then not to worry, *MS Reporting* uses *VB.NET* syntax for coding expressions and custom code. If you have done any custom coding for example with *Crystal Reports*, then interacting with *MS Reporting Services* will not be a big deal.

The expression supplies instructions to the rendering engine on how to group and sort the data based on our choice of report selection. It starts with checking, if choice is "O" that means; create a simple output without grouping. Subsequently, check for rest of choices and switch the behavior of report generation.

We need to repeat the same expression in sorting tab of grouping and sorting properties window.

Handling of group header & footer

We are dealing with three different groups in this report and one output has no grouping required. We have to do the following to generate proper group names and handle visibility property of header & footer.

Apply following expression to group header & footer visibility property:

```
=IIF(Parameters!parReportType.Value = "O", True, False)
```

The above-mentioned expression will take care of hiding the group header & footer in case of default report "Order List" selected.

As group changes dynamically, we do have to change the output to reflect the current scenario. If the user selects "Order by Customer" then we have to make sure to change the group header to "Customer: xyz" and so forth.

The following expression entered as group header title takes care of dynamically changing the header based on provided grouping criteria:

```
=iif(Parameters!parReportType.Value = "O", "",  
iif(Parameters!parReportType.Value = "C",  
"Customer: " & FIRST(Fields!CustomerID.Value),  
iif(Parameters!parReportType.Value = "S",  
"City: " & FIRST(Fields!ShipCity.Value),  
"Country: " & FIRST(Fields!ShipCountry.Value))))
```

Coding time

So far so good, we have created an intelligent report template; we made sure all steps taken to achieve the desired result. However, what prompts the report to act in certain way? How does the report know, it should generate a report based on by Customer or City or if it should ignore grouping altogether and produce a plain orders list?

Now, we have done the design part of report. Now we have to provide a mechanism to collect data from *SQL Server* and bind it to reporting engine. Out of many different ways data can be bound to reporting engine, my favorite is using *DataSet*.

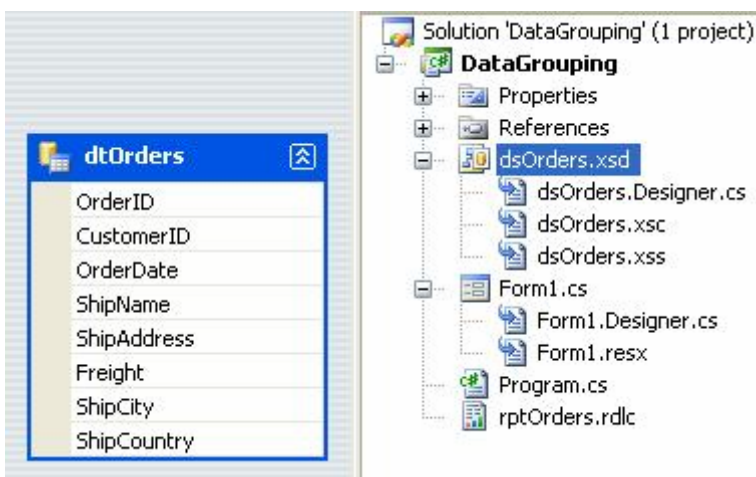


Image: 1.7

Make sure to have *DataSet* ready as per image 1.7.

I have written a method called *loadReport* and passing a single parameter to it as *reportType*. I am calling this method from all four buttons, every time passing a different argument.

Following is the code for the method:

```
private void loadReport(String reportType)
{
    //declare connection string
    string cnString = @"Data Source=(local);
Initial Catalog=northwind;" +
    "User Id=northwind;Password=northwind";

    //use following if you use standard security
    //string cnString = @"Data Source=(local);Initial
    Catalog=northwind; Integrated Security=SSPI";

    //declare Connection, command and other related objects
    SqlConnection conReport = new SqlConnection(cnString);
    SqlCommand cmdReport = new SqlCommand();
    SqlDataReader drReport;
    DataSet dsReport = new dsOrders();

    try
    {
        //open connection
        conReport.Open();

        //prepare connection object to get the data through reader and
        populate into dataset
        cmdReport.CommandType = CommandType.Text;
        cmdReport.Connection = conReport;
        cmdReport.CommandText = "Select * FROM Orders
Order By OrderID";

        //read data from command object
        drReport = cmdReport.ExecuteReader();

        //new cool thing with ADO.NET... load data directly from reader
        to dataset
        dsReport.Tables[0].Load(drReport);

        //close reader and connection
        drReport.Close();
        conReport.Close();

        //provide local report information to viewer
        reportViewer.LocalReport.ReportEmbeddedResource =
            "DataGrouping.rptOrders.rdlc";

        //prepare report data source
        ReportDataSource rds = new ReportDataSource();
    }
}
```

```
rds.Name = "dsOrders_dtOrders";
rds.Value = dsReport.Tables[0];
reportViewer.LocalReport.DataSources.Add(rds);

//add report parameters
ReportParameter[] Param = new ReportParameter[2];

//set dynamic properties based on report selection
//O-order, C-Customer, S-City, T-Country
switch (reportType)
{
    case "O":
        Param[0] = new ReportParameter("parReportTitle",
"Orders List");
        Param[1] = new ReportParameter("parReportType", "O");
        break;
    case "C":
        Param[0] = new ReportParameter("parReportTitle",
"Orders by Customer");
        Param[1] = new ReportParameter("parReportType", "C");
        break;
    case "S":
        Param[0] = new ReportParameter("parReportTitle",
"Orders by City");
        Param[1] = new ReportParameter("parReportType", "S");
        break;
    case "T":
        Param[0] = new ReportParameter("parReportTitle",
"Orders by Country");
        Param[1] = new ReportParameter("parReportType", "T");
        break;
}

reportViewer.LocalReport.SetParameters(Param);

//load report viewer
reportViewer.RefreshReport();
}
catch (Exception ex)
{
    //display generic error message back to user
    MessageBox.Show(ex.Message);
}
finally
{
    //check if connection is still open then attempt to close it
    if (conReport.State == ConnectionState.Open)
    {
```



```
        conReport.Close();
    }
}
```

Code behind each button is as follows:

```
private void btnOrders_Click(object sender, EventArgs e)
{
    //orders list
    loadReport("O");
}

private void btnByCustomer_Click(object sender, EventArgs e)
{
    //orders by customer
    loadReport("C");
}

private void btnByCity_Click(object sender, EventArgs e)
{
    //orders by city
    loadReport("S");
}

private void btnByCountry_Click(object sender, EventArgs e)
{
    //orders by country
    loadReport("T");
}
```

Conclusion

I would love to participate in any discussion about pros and cons of this approach. To me, template based approach and reusability makes more sense then anything else. Any constructive criticism is always welcome.

I would leave you with this final thought here, if you think that I was wrong in saying; "Four birds with one stone" and I should have said; "Seven birds with one stone" then you are 100% correct my friend. I would leave it that up to you to figure it out. This is bonus exercise for you, if you end up playing with attached code.

Ok, here is the hint: Imagine if the end user asks you I need another three reports, but this time I do not care about details. Just give me *Customer...Freight Total*, same for *City* and *Country*. Then the solution is; HIDE THE DETAIL!