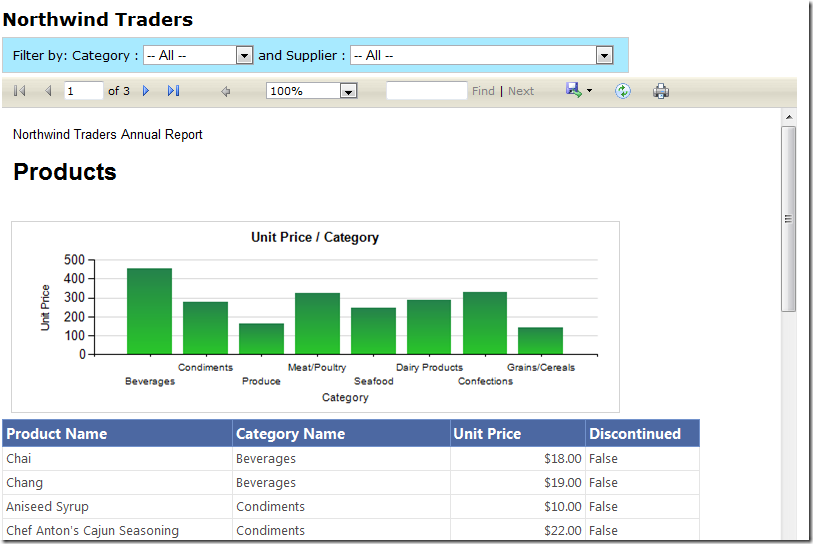
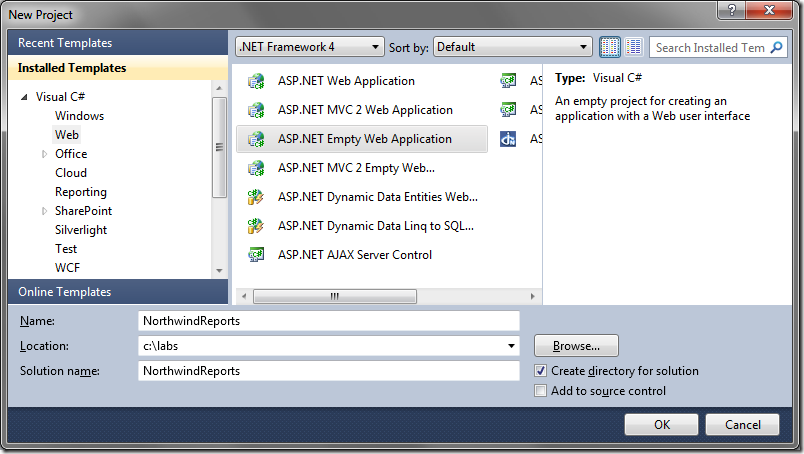
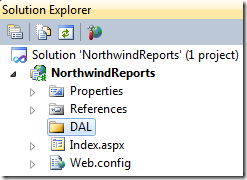
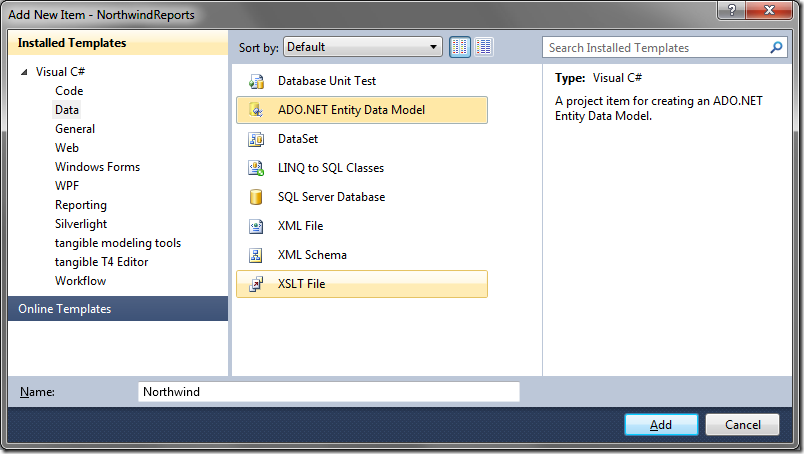
**Creating an ASP.NET report using Visual Studio 2010**

This tutorial walks you through creating an ASP.NET report based on the Northwind sample database. It shows how to add a client report definition file (RDLC), create a dataset for the RDLC, define queries using LINQ to Entities, design the report and add a ReportViewer web control to render the report in a ASP.NET web page. The report will have a chart control. The result can be filtered by two drop downs at the top..   
  
At the end of the walkthrough, you should have a UI like the following.  As shown, there is a product list report with an embedded chart. The chart shows the sum of Unit price for a given category. It is possible to filter the results by Category and Supplier. The drop downs auto post back when the selection is changed.  This demo uses **Visual Studio 2010 RTM.**   
  
This post is split into three parts. **The last part has the sample code attached.**   
    
  
Lets start by creating a new ASP.NET empty web application called “NorthwindReports”   
   
**Creating the Data Access Layer (DAL)**

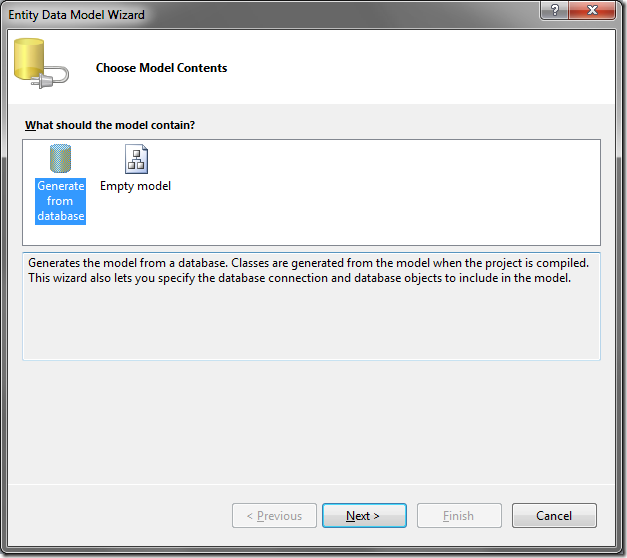
Add a web form called index.aspx to the root directory. You do this by right clicking on the NorthwindReports web project and selecting “Add item..”   
.   
Create a folder called “DAL”. We will store all our data access methods and any data transfer objects in here.

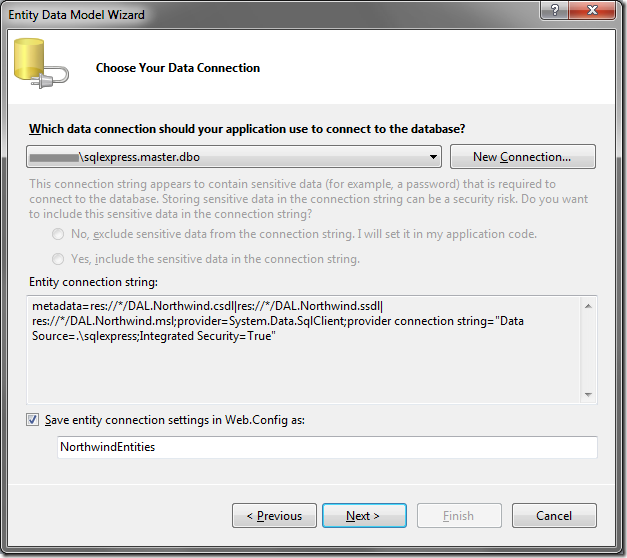
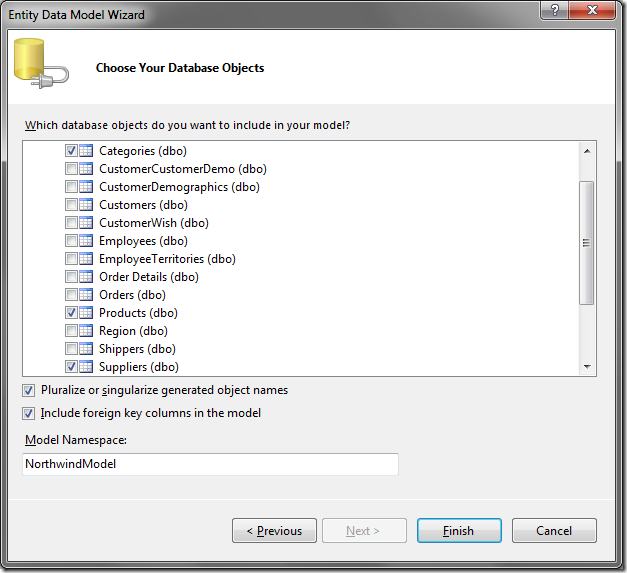
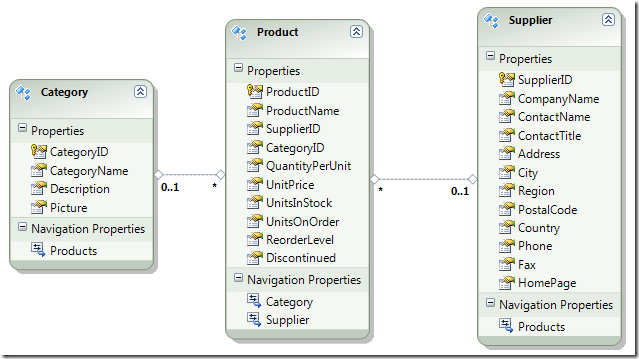


Right click on the DAL folder and add a ADO.NET Entity data model called Northwind.



Select “Generate from database” and click Next.

   
Create a connection to your database containing the Northwind sample database and click Next.

    
From the table list, select Categories, Products and Suppliers and click next.   
  
   
Our Entity data model gets created and looks like this:   
  
     
**Adding data transfer objects**  
Right click on the DAL folder and add a ProductViewModel. Add the following code. This class contains properties we need to render our report.

public class ProductViewModel

{

public int? ProductID { get; set; }

public string ProductName { get; set; }

public System.Nullable<decimal> UnitPrice { get; set; }

public string CategoryName { get; set; }

public int? CategoryID { get; set; }

public int? SupplierID { get; set; }

public bool Discontinued { get; set; }

}

Add a SupplierViewModel class. This will be used to render the supplier DropDownlist.

public class SupplierViewModel

{

public string CompanyName { get; set; }

public int SupplierID { get; set; }

}

Add a CategoryViewModel class.

public class CategoryViewModel

{

public string CategoryName { get; set; }

public int CategoryID { get; set; }

}

Create an IProductRepository interface. This will contain the signatures of all the methods we need when accessing the entity model.  This step is not needed but follows the [**repository pattern**](http://msdn.microsoft.com/en-us/library/ff649690.aspx).

interface IProductRepository

{

IQueryable<Product> GetProducts();

IQueryable<ProductViewModel> GetProductsProjected(int? supplierID, int? categoryID);

IQueryable<SupplierViewModel> GetSuppliers();

IQueryable<CategoryViewModel> GetCategories();

}

Create a ProductRepository class that implements the IProductReposity above. The methods available in this class are as follows:

* GetProducts – returns an IQueryable of all products.
* GetProductsProjected – returns an IQueryable of ProductViewModel. The method filters all the products based on SupplierId and CategoryId if any. It then projects the result into the ProductViewModel.
* GetSuppliers() – returns an IQueryable of all suppliers projected into a SupplierViewModel
* GetCategories() – returns an IQueryable of all categories projected into a CategoryViewModel

public class ProductRepository : IProductRepository

{

/// <summary>

/// IQueryable of all Products

/// </summary>

/// <returns></returns>

public IQueryable<Product> GetProducts()

{

var dataContext = new NorthwindEntities();

var products = from p in dataContext.Products

select p;

return products;

}

/// <summary>

/// IQueryable of Projects projected

/// into the ProductViewModel class

/// </summary>

/// <returns></returns>

public IQueryable<ProductViewModel> GetProductsProjected(int? supplierID, int? categoryID)

{

var projectedProducts = from p in GetProducts()

select new ProductViewModel

{

ProductID = p.ProductID,

ProductName = p.ProductName,

UnitPrice = p.UnitPrice,

CategoryName = p.Category.CategoryName,

CategoryID = p.CategoryID,

SupplierID = p.SupplierID,

Discontinued = p.Discontinued

};

// Filter on SupplierID

if (supplierID.HasValue)

{

projectedProducts = projectedProducts.Where(a => a.SupplierID == supplierID);

}

// Filter on CategoryID

if (categoryID.HasValue)

{

projectedProducts = projectedProducts.Where(a => a.CategoryID == categoryID);

}

return projectedProducts;

}

public IQueryable<SupplierViewModel> GetSuppliers()

{

var dataContext = new NorthwindEntities();

var suppliers = from s in dataContext.Suppliers

select new SupplierViewModel

{

SupplierID = s.SupplierID,

CompanyName = s.CompanyName

};

return suppliers;

}

public IQueryable<CategoryViewModel> GetCategories()

{

var dataContext = new NorthwindEntities();

var categories = from c in dataContext.Categories

select new CategoryViewModel

{

CategoryID = c.CategoryID,

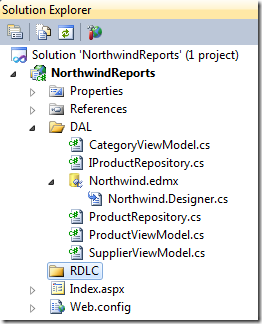
CategoryName = c.CategoryName

};

return categories;

}

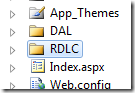
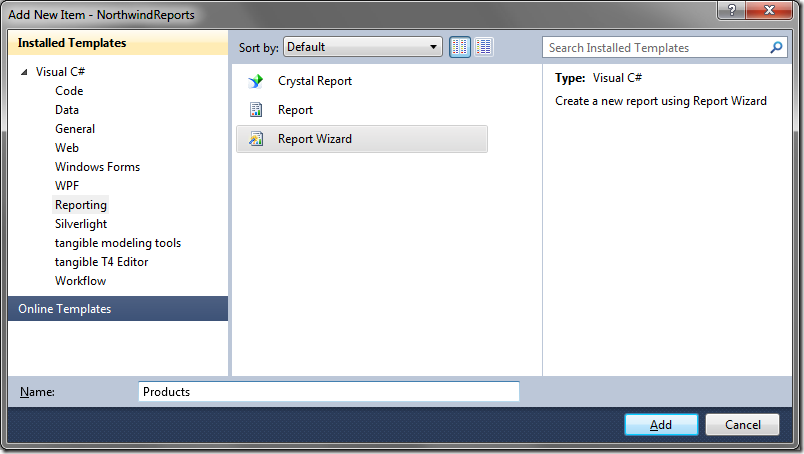
}

Your solution explorer should look like the following.   


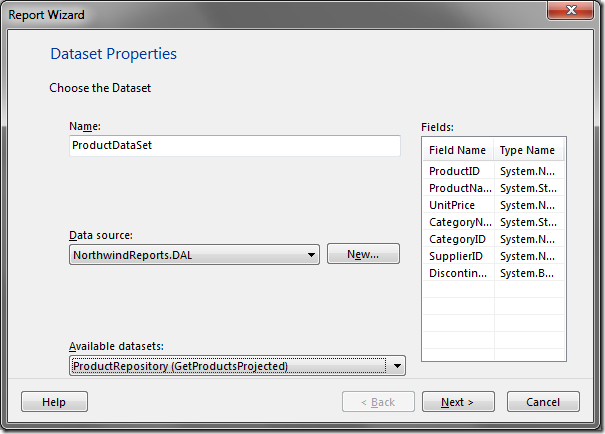
**Build your project and make sure you don’t get any errors.**

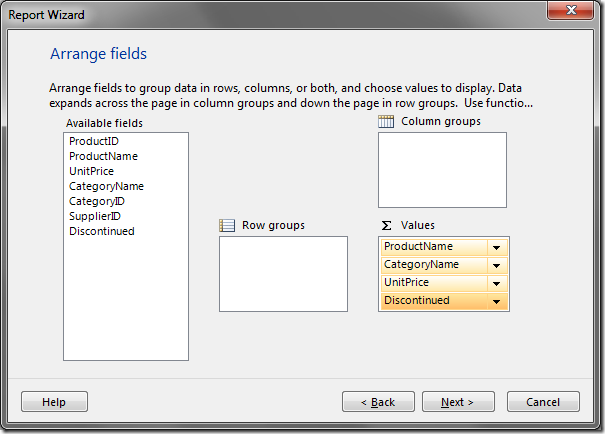
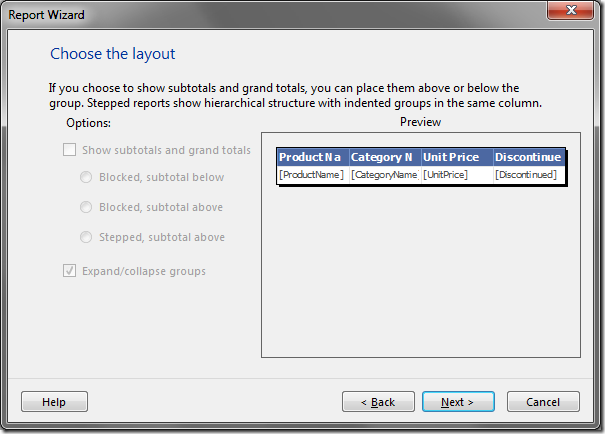
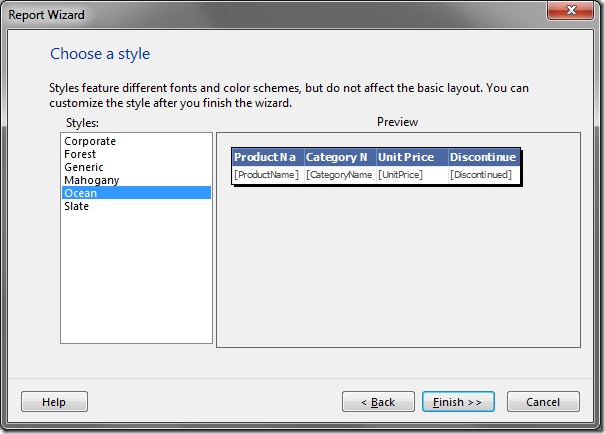
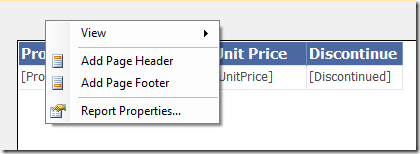
In the next part, we will see how to create the client report definition file using the Report Wizard.

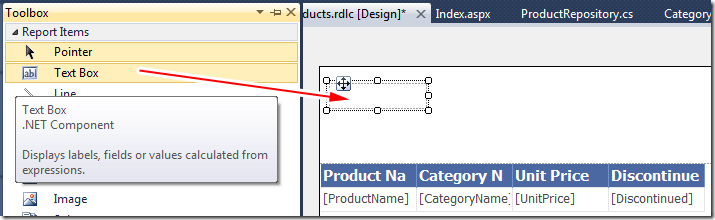
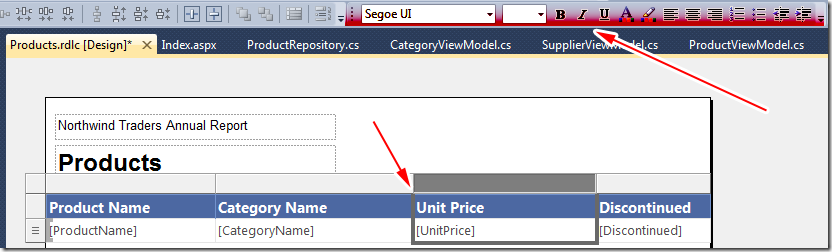
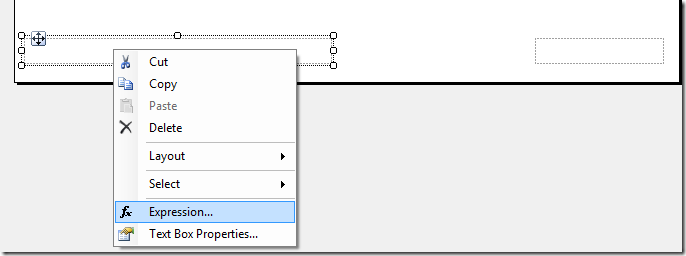
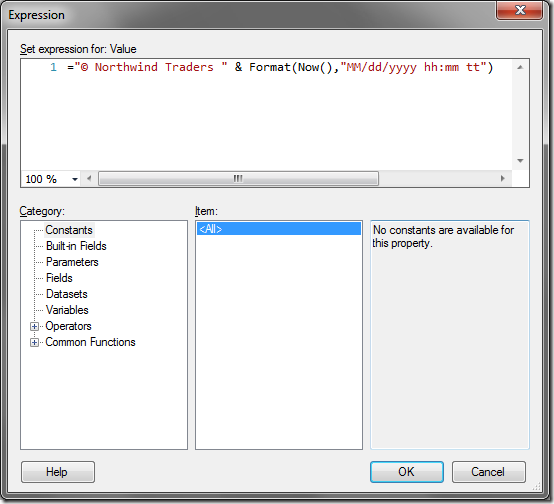
**Creating an ASP.NET report using Visual Studio 2010**

We continue building our report in this three part series.   
**Creating the Client Report Definition file (RDLC)**Add a folder called “RDLC”. This will hold our RDLC report.   
  
****    
Right click on the RDLC folder, select “Add new item..” and add an “RDLC” name of “Products”. We will use the “Report Wizard” to walk us through the steps of creating the RDLC.   
**[](http://weblogs.asp.net/blogs/rajbk/image_5F58A3EC.png)**    
In the next dialog, give the dataset a name called “ProductDataSet”. Change the data source to “NorthwindReports.DAL” and select “ProductRepository(GetProductsProjected)”.

The “Data Source” may show up empty. To get it populated, make sure your project is compiled and there is an index.aspx file in the root folder. This may be a bug.

The fields that are returned from the method are shown on the right. Click next.   
**[](http://weblogs.asp.net/blogs/rajbk/image_5139B4EF.png)**    
Drag and drop the ProductName, CategoryName, UnitPrice and Discontinued into the Values container. Note that you can create much more complex grouping using this UI. Click Next.

    
Most of the selections on this screen are grayed out because we did not choose a grouping in the previous screen. Click next.   
   
Choose a style for your report. Click next.   
   
The report graphic design surface is now visible. Right click on the report and add a page header and page footer.   
   
With the report design surface active, drag and drop a TextBox from the tool box to the page header. Drag one more textbox to the page header. We will use the text boxes to add some header text as shown in the next figure.

   
You can change the font size and other properties of the textboxes using the formatting tool bar (marked in red). You can also resize the columns by moving your cursor in between columns and dragging.   
   
 **Adding Expressions**  
  
Add two more text boxes to the page footer. We will use these to add the time the report was generated and page numbers. Right click on the first textbox in the page footer and select “Expression”.   
   
Add the following expression for the print date (note the = sign at the left of the expression in the dialog below)   


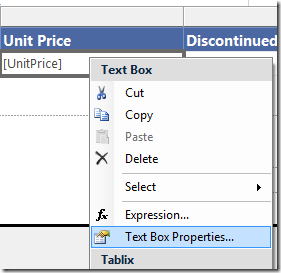
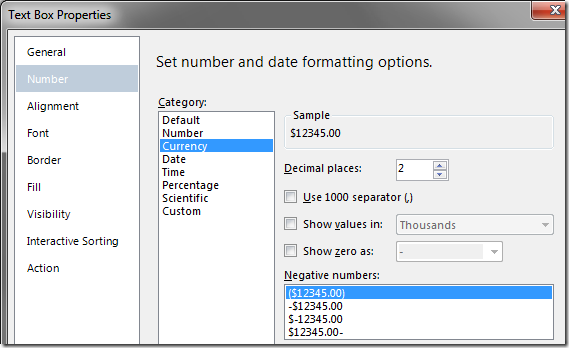
"© Northwind Traders " & Format(Now(),"MM/dd/yyyy hh:mm tt")

Right click on the second text box and add the following for the page count.

Globals.PageNumber & " of " & Globals.TotalPages

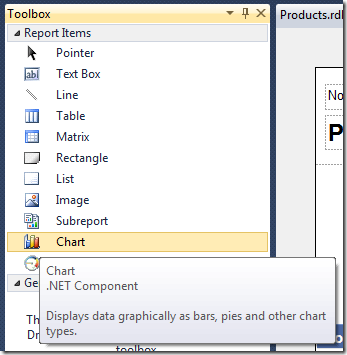
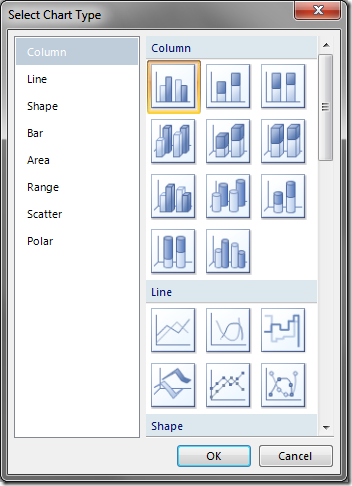
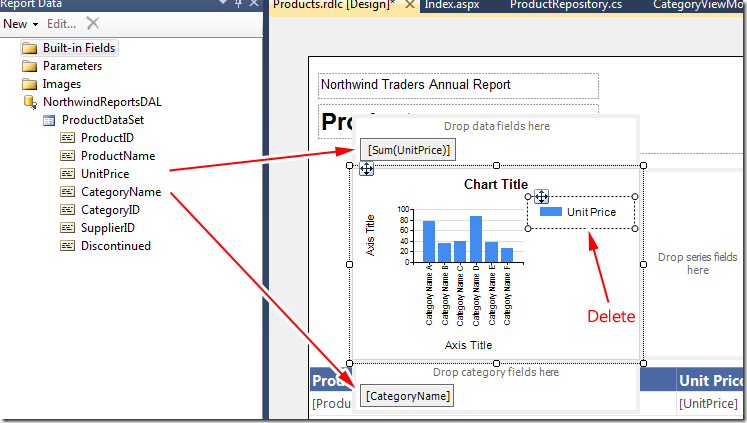
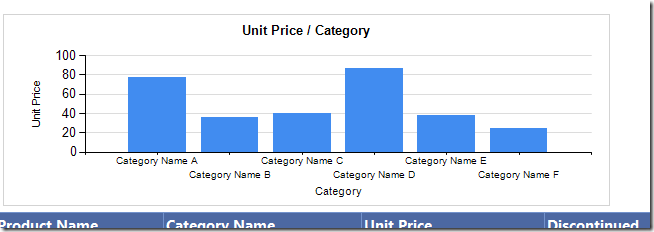
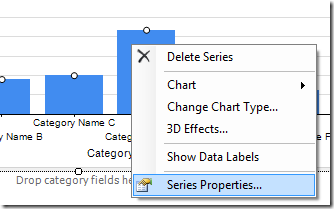
Formatting the page footer is complete.

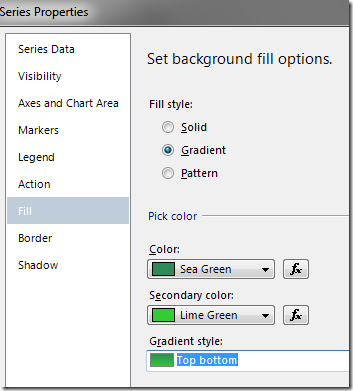
We are now going to format the “Unit Price” column so it displays the number in currency format.  Right click on the [UnitPrice] column (not header) and select “Text Box Properties..”

   
Under “Number”, select “Currency”. Hit OK.   


**Adding a chart**

With the design surface active, go to the toolbox and drag and drop a chart control. You will need to move the product list table down first to make space for the chart contorl. The document can also be resized by dragging on the corner or at the page header/footer separator.

   
In the next dialog, pick the first chart type. This can be changed later if needed. Click OK. The chart gets added to the design surface.   
    
Click on the blue bars in the chart (not legend). This will bring up drop locations for dropping the fields. Drag and drop the UnitPrice and CategoryName into the top (y axis) and bottom (x axis) as shown below. This will give us the total unit prices for a given category. That is the best I could come up with as far as what report to render, sorry :-) Delete the legend area to get more screen estate.   
   
Resize the chart to your liking. Change the header, x axis and y axis text by double clicking on those areas.   
  
   
We made it this far. Let’s impress the client by adding a gradient to the bar graph :-) Right click on the blue bar and select “Series properties”.   


Under “Fill”, add a color and secondary color and select the Gradient style.   
**[](http://weblogs.asp.net/blogs/rajbk/image_100AE9F1.png)**

We are done designing our report. In the next section you will see how to add the report to the report viewer control, bind to the data and make it refresh when the filter criteria are changed.

**Creating an ASP.NET report using Visual Studio 2010**

We continue building our report in this three part series.   
**Adding the ReportViewer control and filter drop downs.**Open the source code for index.aspx and add a ScriptManager control. This control is required for the ReportViewer control. Add a DropDownList for the categories and suppliers. Add the ReportViewer control. The markup after these steps is shown below.

<div>

<asp:ScriptManager ID="smScriptManager" runat="server">

</asp:ScriptManager>

<div id="searchFilter">

Filter by: Category :

<asp:DropDownList ID="ddlCategories" runat="server" />

and Supplier :

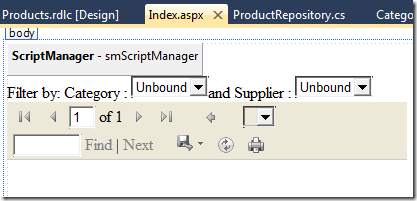
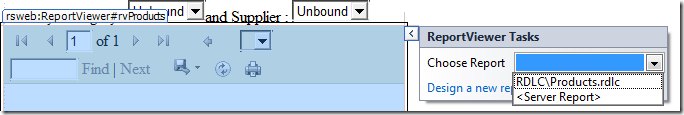
<asp:DropDownList ID="ddlSuppliers" runat="server" />

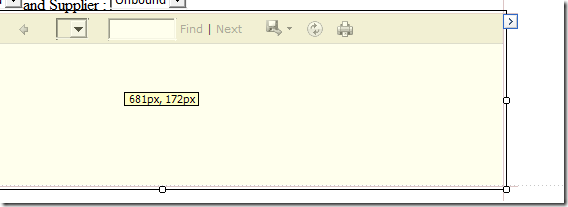
</div>

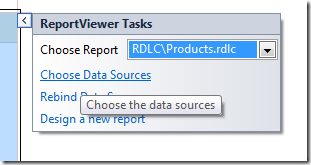
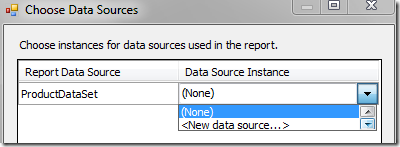
<rsweb:ReportViewer ID="rvProducts" runat="server">

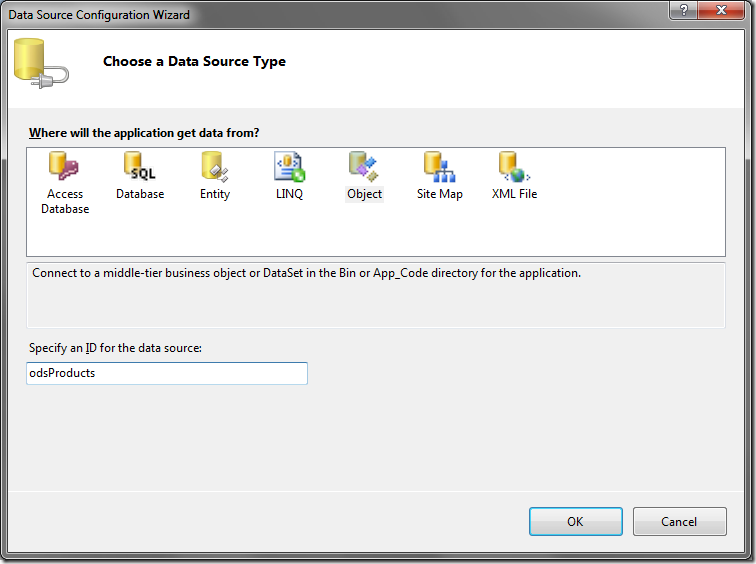
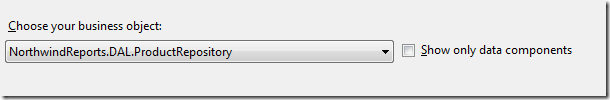
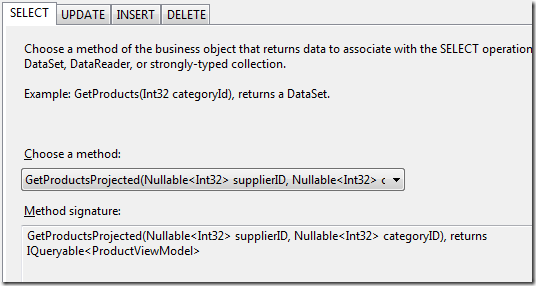
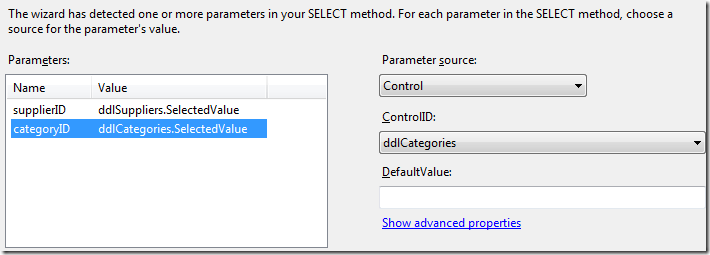
</rsweb:ReportViewer>

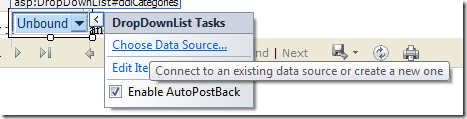
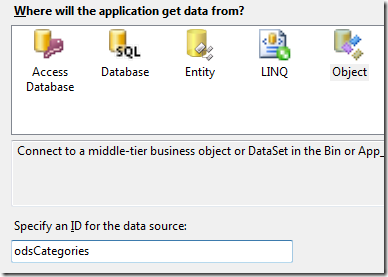
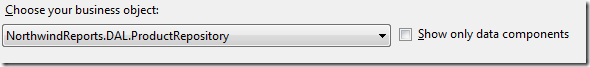
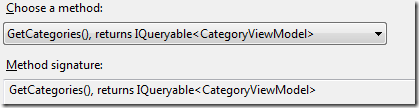
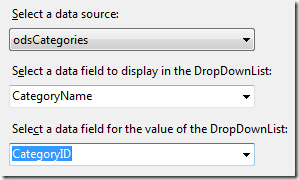
</div>

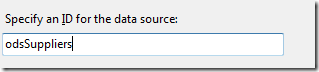
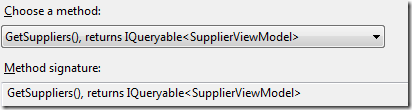
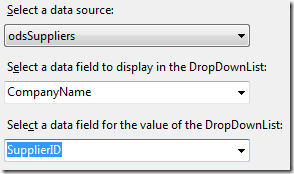
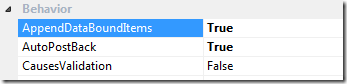
The design view for index.aspx is shown below. The dropdowns will display the categories and suppliers in the database. The report will be filtered by the selections in the dropdowns. You will see how to do this in the next steps.   
**[](http://weblogs.asp.net/blogs/rajbk/image_618D04AA.png)**    
Attaching the RDLC to the ReportViewer control by clicking on the top right of the control, going to Report Viewer tasks and selecting Products.rdlc.     
**[](http://weblogs.asp.net/blogs/rajbk/image_6767A843.png)**   
Resize the ReportViewer control by dragging at the bottom right corner. I set mine to 800px x 500px. You can set this value in source view also.

   
  
**Defining the data sources.**

We will now define the Data Source used to populate the report. Go back to the “ReportViewer Tasks” and select “Choose Data Sources”   
   
Select a “New data source..”   


Select “Object” and name your Data Source ID “odsProducts”   
  
    
In the next screen, choose “ProductRepository” as your business object.   
   
Choose “GetProductsProjected” in the next screen.   
  
    
The method requires a SupplierID and CategoryID. We will have the data source use the selected values of the drop down lists we defined earlier. Set the parameter source to be of type “Control” and set the ControlIDs to be ddlSuppliers and ddlCategories respectively. Your screen will look like this:   
   
We are now going to define the data source for our drop downs. Select the ddlCategory drop down and pick “Choose Data Source”.

   
Pick “Object” and give it an id “odsCategories”   
    
In the next screen, choose “ProductRepository”   
   
Select the GetCategories() method in the next screen.   
**[](http://weblogs.asp.net/blogs/rajbk/image158_6A1D3769.png)**    
Select “CategoryName” and “CategoryID” in the next screen. We are done defining the data source for the Category drop down.   
   
Perform the same steps for the Suppliers drop down.

   
   
    
Select each dropdown and set the AppendDataBoundItems to true and AutoPostback to true.   
      
The AppendDataBoundItems is needed because we are going to insert an “All“ list item with a value of empty. This will be the first item in each drop down list. Go to each drop down and add this list item markup as shown below.   
**[](http://weblogs.asp.net/blogs/rajbk/image_6008963E.png)**

Double click on each drop down in the designer and add the following code in the code behind. This along with the “Autopostback= true” attribute refreshes the report anytime the selection in the drop down is changed.

protected void ddlCategories\_SelectedIndexChanged(object sender, EventArgs e)

{

rvProducts.LocalReport.Refresh();

}

protected void ddlSuppliers\_SelectedIndexChanged(object sender, EventArgs e)

{

rvProducts.LocalReport.Refresh();

}

Compile your report and run the page. You should see the report rendered. Note that the tool bar in the ReportViewer control gives you a couple of options including the ability to export the data to Excel, PDF or word.



**Conclusion**

Through this three part series, we did the following:

* Created a data layer for use by our RDLC.
* Created an RDLC using the report wizard and define a dataset for the report.
* Used the report design surface to design our report including adding a chart.
* Used the ReportViewer control to attach the RDLC.
* Connected our ReportWiewer to a data source and take parameter values from the drop downlists.
* Used AutoPostBack to refresh the reports when the dropdown selection was changed.

RDLCs allow you to create interactive reports including drill downs and grouping. For even more advanced reports you can use [**Microsoft® SQL Server™ Reporting Services**](http://www.microsoft.com/sqlserver/2008/en/us/reporting.aspx) with RDLs. With RDLs, the report is rendered on the report server instead of the web server. Another nice thing about RDLs is that you can define a parameter list for the report and it gets rendered automatically for you. RDLCs and RDLs both have their advantages and its best to compare them and choose the right one for your requirements.