

## Programming Crystal Reports with ASP.NET 2.0 - Providing Login information for the CrystalReportViewer control dynamically

(Page 4 of 6 )

The CrystalReportViewer control needs to be provided with database logon information to execute and show the report. At the same time, it is a bit awkward to have the database login page displayed for every Crystal Report. No end user likes it (and of course, it is dangerous too!).

To solve this problem, we can provide database logon information to the CrystalReportViewer control dynamically at runtime. The following is the code to achieve the same:

```
Imports CrystalDecisions.Shared

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Handles form1.Load
            Dim ConnInfo As New ConnectionInfo
            With ConnInfo
                .ServerName = ".sqlexpress"
                .DatabaseName = "Northwind"
                .UserID = "sa"
                .Password = "eXpress2005"
            End With

            For Each cnInfo As TableLogOnInfo In Me.CrystalReportViewer1.LogOnInfo
                cnInfo.ConnectionInfo = ConnInfo
            Next
        End Sub
    End Class
```

You must observe that the CrystalDecisions.Shared namespace is added to the top. In the above code, the database connection information is stored in ConnInfo and is defined as follows:

```
Dim ConnInfo As New ConnectionInfo
With ConnInfo
    .ServerName = ".sqlexpress"
    .DatabaseName = "Northwind"
    .UserID = "sa"
    .Password = "eXpress2005"
End With
```

The above connection information is assigned to CrystalReportViewer control using the following code:

```
For Each cnInfo As TableLogOnInfo In Me.CrystalReportViewer1.LogOnInfo
    cnInfo.ConnectionInfo = ConnInfo
Next
```

## Programming Crystal Reports with ASP.NET 2.0 - Hiding the toolbar and adding First, Last, Next and Previous page buttons to the report

(Page 5 of 6 )

By default, CrystalReportViewer automatically displays toolbar at the top. Not every end user uses all features. At the same time, some end users may not like the toolbar.

To hide the toolbar, modify the CrystalReportViewer code (in Source mode) as follows:

```
<CR:CrystalReportViewer ID="CrystalReportViewer1" runat="server"
AutoDataBind="True" ReuseParameterValuesOnRefresh="True" DisplayToolbar="False"
EnableDatabaseLogonPrompt="False" EnableParameterPrompt="False"
DisplayGroupTree="False"
Height="1064px" ReportSourceID="CrystalReportSource1" Width="928px" />
```

Add First, Last, Next and Previous page buttons to the web page and modify the code behind with new events as follows:

```
Protected Sub btnFirst_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnFirst.Click
    Me.CrystalReportViewer1.ShowFirstPage()
End Sub

Protected Sub btnLast_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnLast.Click
    Me.CrystalReportViewer1.ShowLastPage()
End Sub

Protected Sub btnNext_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnNext.Click
    Me.CrystalReportViewer1.ShowNextPage()
End Sub

Protected Sub btnPrevious_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnPrevious.Click
    Me.CrystalReportViewer1.ShowPreviousPage()
End Sub
```

### **Programming Crystal Reports with ASP.NET 2.0 - Enhancing the report with run-time binding along with session handling**

(Page 6 of 6 )

In all of the above sections, the CrystalReportViewer worked with CrystalReportSource. Now, let us dynamically add a report source to the CrystalReportViewer and bind it at runtime. This gives us the flexibility to use the same viewer for different reports (showing one at a time).

Add a new web page to the project, add the four buttons (First, Previous, Next and Last) and a CrystalReportViewer control. Modify the CrystalReportViewer control, so that it looks like the following:

```
<CR:CrystalReportViewer ID="CrystalReportViewer1" runat="server"
AutoDataBind="True" ReuseParameterValuesOnRefresh="True" DisplayToolbar="False"
```

```
EnableDatabaseLogonPrompt="False" EnableParameterPrompt="False"  
DisplayGroupTree="False"  
Height="1064px" Width="928px" />
```

In the code-behind, add the following at the top:

```
Imports CrystalDecisions.Shared  
Imports CrystalDecisions.CrystalReports.Engine
```

Add a new method as follows:

```
Private Sub BindReport()  
  
    If Session("Rep") Is Nothing Then  
        Dim ConnInfo As New ConnectionInfo  
        With ConnInfo  
            .ServerName = ".sqlexpress"  
            .DatabaseName = "Northwind"  
            .UserID = "sa"  
            .Password = "eXpress2005"  
        End With  
  
        Dim rep As New ReportDocument  
        rep.Load(Server.MapPath("SampleRpt01.rpt"))  
        Me.CrystalReportViewer1.ReportSource = rep  
        Dim RepTbls As Tables = rep.Database.Tables  
        For Each RepTbl As Table In RepTbls  
            Dim RepTblLogonInfo As TableLogOnInfo = RepTbl.LogOnInfo  
            RepTblLogonInfo.ConnectionInfo = ConnInfo  
            RepTbl.ApplyLogOnInfo(RepTblLogonInfo)  
        Next  
        Session("Rep") = rep  
    End If  
  
    Me.CrystalReportViewer1.ReportSource = Session("Rep")  
    Me.CrystalReportViewer1.DataBind()  
End Sub
```

Add the following code to bind the report for every button click and also during the page load event:

```
Protected Sub btnFirst_Click(ByVal sender As Object, ByVal e As  
System.EventArgs) Handles btnFirst.Click  
    BindReport()  
    Me.CrystalReportViewer1.ShowFirstPage()  
End Sub  
  
Protected Sub btnLast_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
Handles btnLast.Click  
    BindReport()  
    Me.CrystalReportViewer1.ShowLastPage()  
End Sub  
  
Protected Sub btnNext_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
Handles btnNext.Click  
    BindReport()  
    Me.CrystalReportViewer1.ShowNextPage()  
End Sub  
  
Protected Sub btnPrev_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
Handles btnPrev.Click  
    BindReport()  
End Sub
```

```
Me.CrystalReportViewer1.ShowPreviousPage()  
End Sub  
  
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)  
Handles Me.Load  
    If Not IsPostBack Then  
        BindReport()  
    End If  
End Sub
```

I hope you enjoyed the article and any suggestions, bugs, errors, enhancements etc. are highly appreciated at <http://jagchat.spaces.live.com>

Crystal Reports is the standard reporting tool for Visual Studio .NET used to display data of presentation quality. You can display multiple-level totals, charts to analyze data, and much more in Crystal Reports. Creating a Crystal Report requires minimal coding since it is created in Designer interface. It is available as an integrated feature of Microsoft Visual Studio .NET, Borland Delphi, and C#Builder.

### ***Advantages of Crystal Reports***

Some of the major advantages of using Crystal Reports are:

1. Rapid report development since the designer interface would ease the coding work for the programmer.
2. Can extend it to complicated reports with interactive charts and enhance the understanding of the business model
3. Exposes a report object model, can interact with other controls on the ASP.NET Web form
4. Can programmatically export the reports into widely used formats like .pdf, .doc, .xls, .html and .rtf

### ***Implementation Models***

Crystal Reports need database drivers to connect to the data source for accessing data. Crystal Reports in .net support two methods to access data from a data source:

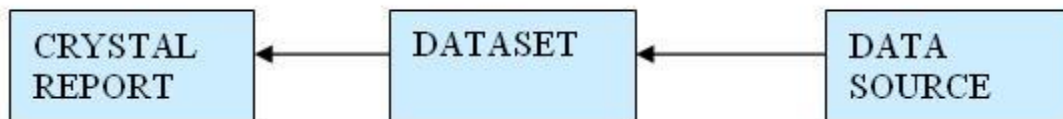
#### ***The Pull Method***

When this model is used to access data from the data source, the database driver directly retrieves the data from the data source. This model does not require the developer to write code for creating a connection and retrieving data from the data source. It is the Crystal report that manages the SQL commands for connecting by using the specified driver.



### ***The Push Method***

When this model is used to access data from data source, the developer writes the code to connect to the data source and retrieve data. The data from the data source is cached in dataset and multiple crystal reports accesses data from the dataset. The performance can be optimized in this manner by using connection sharing and manually limiting the number of records that are passed on to the report.



## **Crystal Reports Types**

Crystal Report Designer can load reports that are included into the project as well as those that are independent of the project.

### ***Strongly-typed Report***

When you add a report file into the project, it becomes a "strongly-typed" report. In this case, you will have the advantage of directly creating an instance of the report object, which could reduce a few lines of code, and cache it to improve performance. The related .vb file, which is hidden, can be viewed using the editor's "show all files" icon in the Solution Explorer.

### ***Un-Typed Report***

Those reports that are not included into the project are "un-typed" reports. In this case, you will have to create an instance of the Crystal Report Engine's "ReportDocument" object and manually load the report into it.

## **Creating Crystal Reports**

You can create a Crystal Report by using three methods:

1. Manually i.e. from a blank document
2. Using Standard Report Expert
3. From an existing report

## ***Using Pull Method***

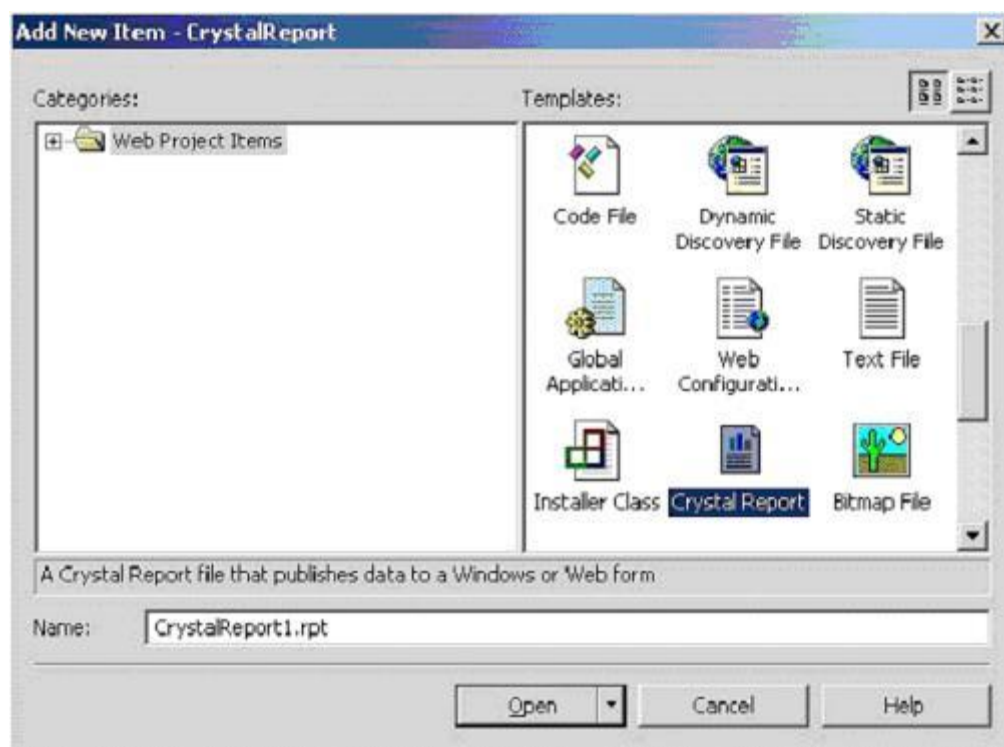
Creating Crystal Reports Manually.

We would use the following steps to implement Crystal Reports using the Pull Model:

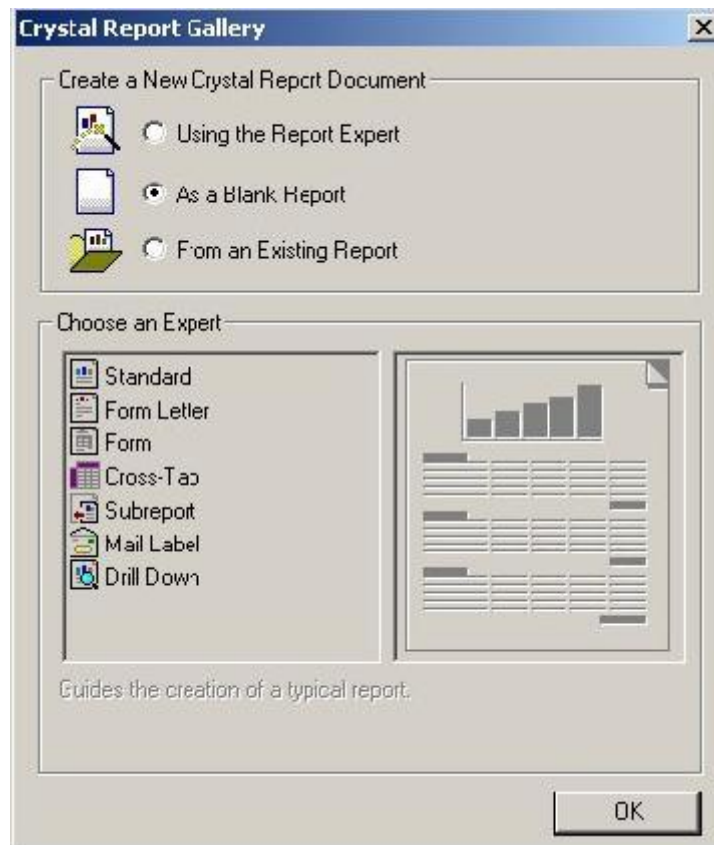
1. **Create the .rpt file** (from scratch) and set the necessary database connections using the Crystal Report Designer interface.
2. **Place a CrystalReportViewer control** from the toolbox on the .aspx page and set its properties to point to the .rpt file that we created in the previous step.
3. Call the databind method from your code behind page.

### ***I. Steps to create the report i.e. the .rpt file***

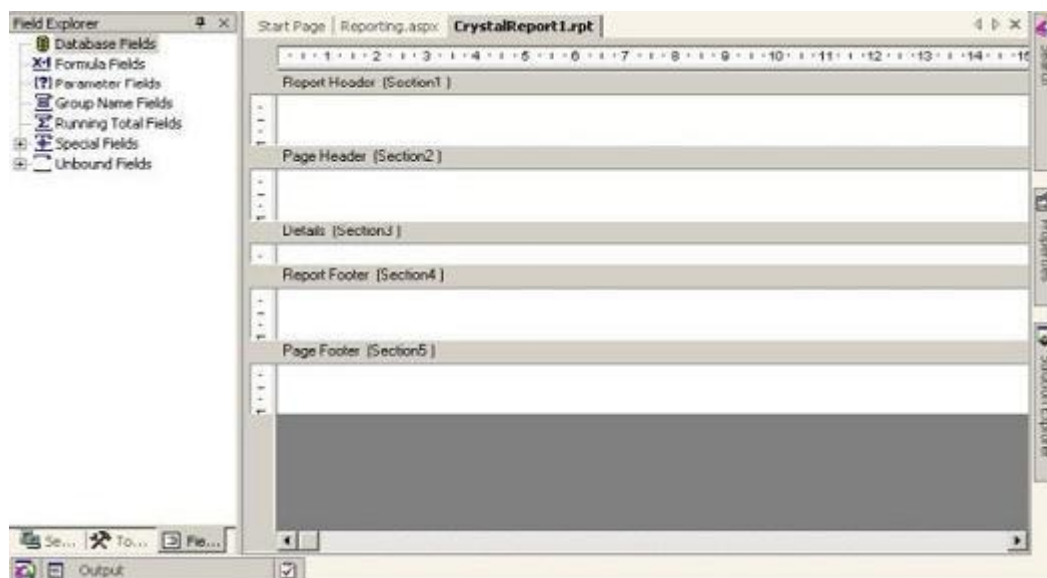
- 1) Add a new Crystal Report to the web form by right clicking on the "Solution Explorer", selecting "Add" --> "Add New Item" --> "Crystal Report".



- 2) On the "Crystal Report Gallery" pop up, select the "As a Blank Report" radio button and click "ok".



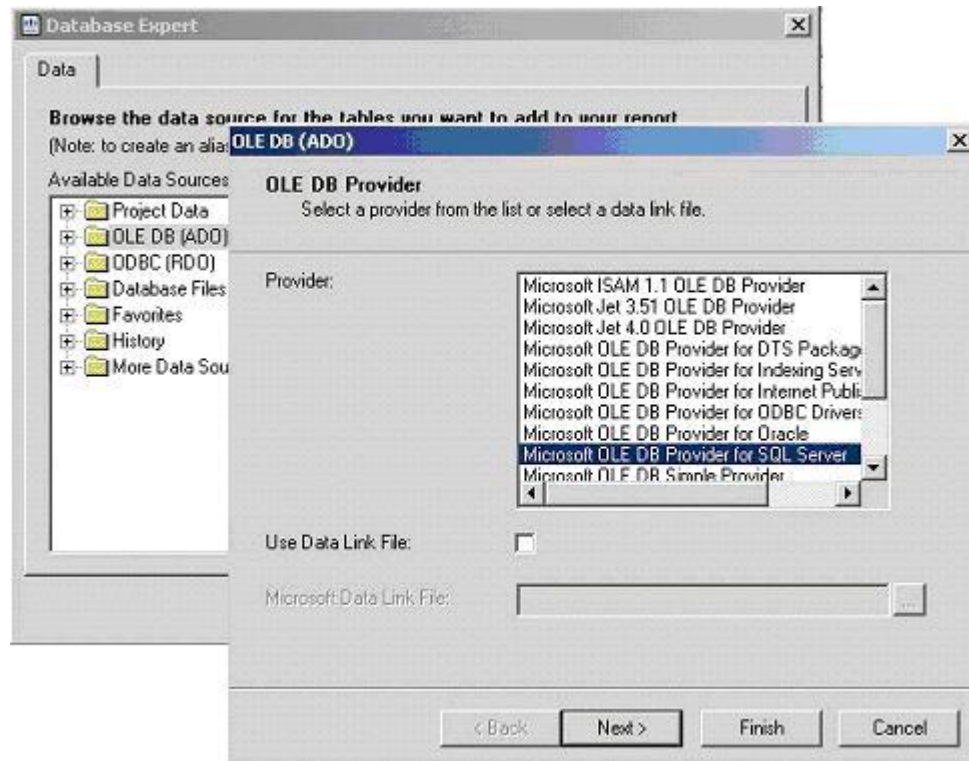
3) This should open up the Report File in the Crystal Report Designer.



4) Right click on the "Details Section" of the report, and select "Database" -> "Add/Remove Database".

5) In the "Database Expert" pop up window, expand the "OLE DB (ADO)" option by clicking the "+" sign, which should bring up another "OLE DB (ADO)" pop up.

6) In the "OLE DB (ADO)" pop up, Select "Microsoft OLE DB Provider for SQL Server" and click Next.



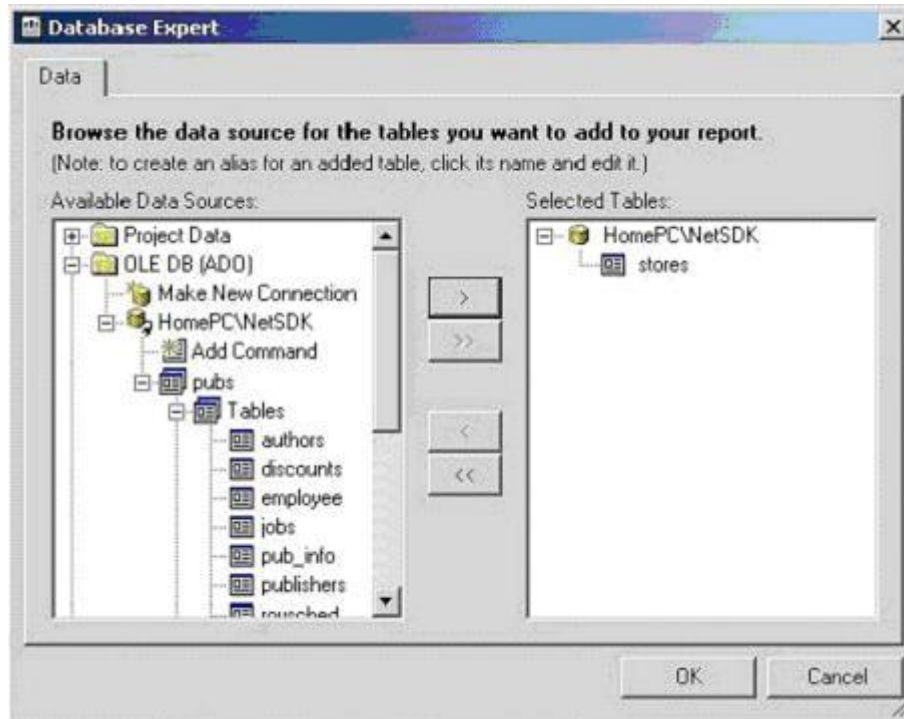
7) Specify the connection information.

8) Click "Next" and then click "Finish"

9) Now you should be able to see the Database Expert showing the table that have been selected

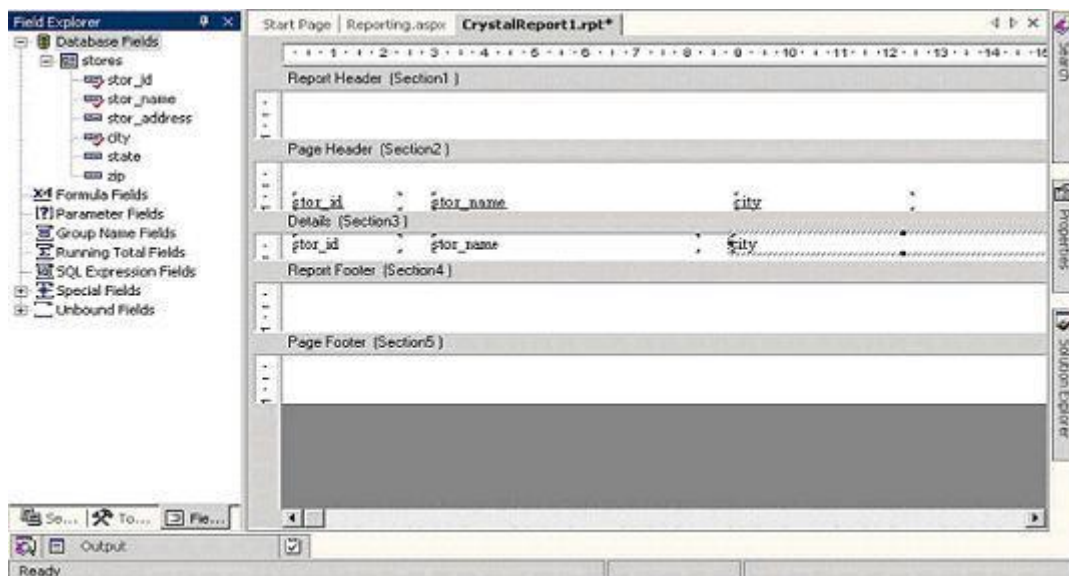
10) Expand the "Pubs" database, expand the "Tables", select the "Stores" table and click on ">" to include it into the "Selected Tables" section.

Note: If you add more than one table in the database Expert and the added tables have matching fields, when you click the OK button after adding the tables, the links between the added tables is displayed under the Links tab. You can remove the link by clicking the Clear Links button.



11) Now the Field Explorer should show you the selected table and its fields under the "Database Fields" section, in the left window.

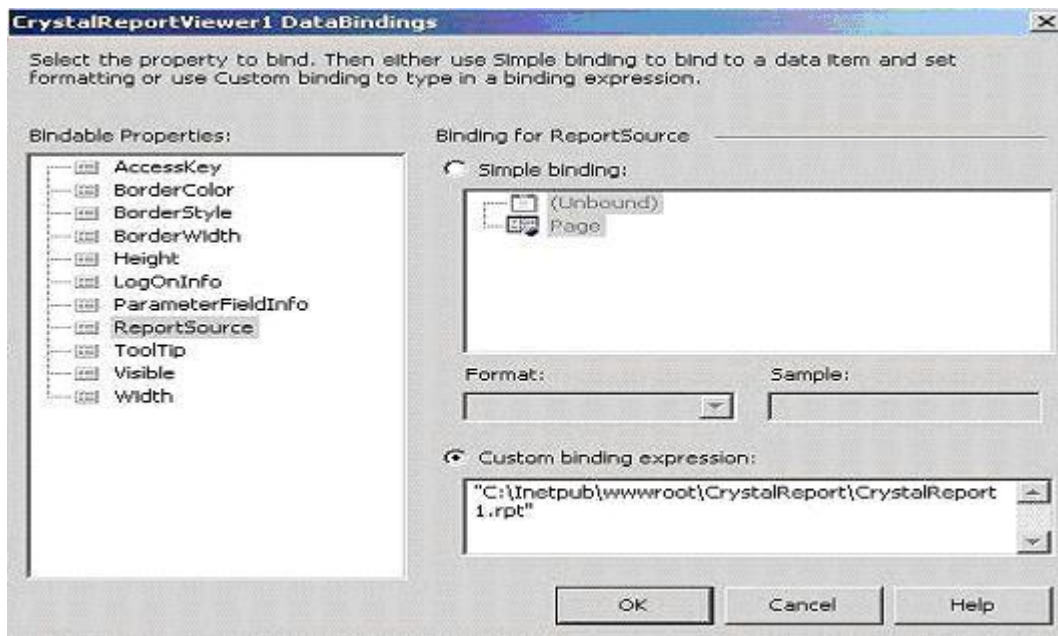
12) Drag and drop the required fields into the "Details" section of the report. The field names would automatically appear in the "Page Header" section of the report. If you want to modify the header text then right click on the text of the "Page Header" section, select "Edit Text Object" option and edit it.



13) Save it and we are through.

## ***II. Creating a Crystal Report Viewer Control***

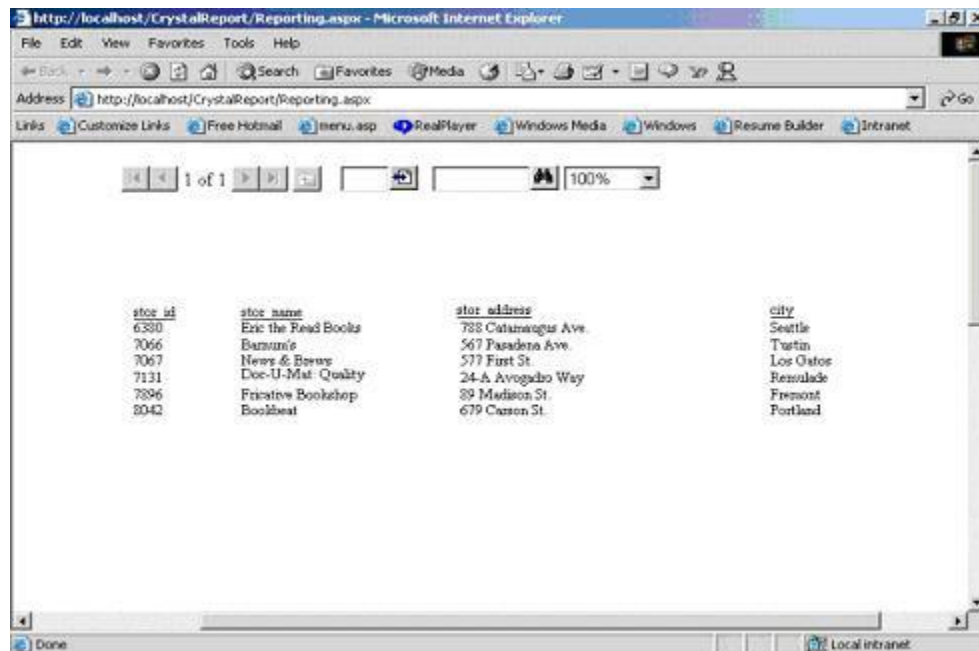
- 1) Drag and drop the "Crystal Report Viewer>" from the web forms tool box on to the .aspx page
- 2) Open the properties window for the Crystal Report Viewer control.
- 3) Click on the [...] next to the "Data Binding" Property and bring up the data binding pop-up window
- 4) Select "Report Source".
- 5) Select the "Custom Binding Expression" radio button, on the right side bottom of the window and specify the sample .rpt filename and path as shown in the fig.



- 6) You should be able to see the Crystal Report Viewer showing you a preview of actual report file using some dummy data and this completes the inserting of the Crystal Report Viewer controls and setting its properties.

Note: In the previous example, the CrystalReportViewer control was able to directly load the actual data during design time itself as the report was saved with the data. In this case, it will not display the data during design time as it not saved with the data - instead it will show up with dummy data during design time and will fetch the proper data only at run time.

- 7) Call the Databind method on the Page Load Event of the Code Behind file (.aspx.vb). Build and run your .aspx page. The output would look like this.



## ***Using a PUSH model***

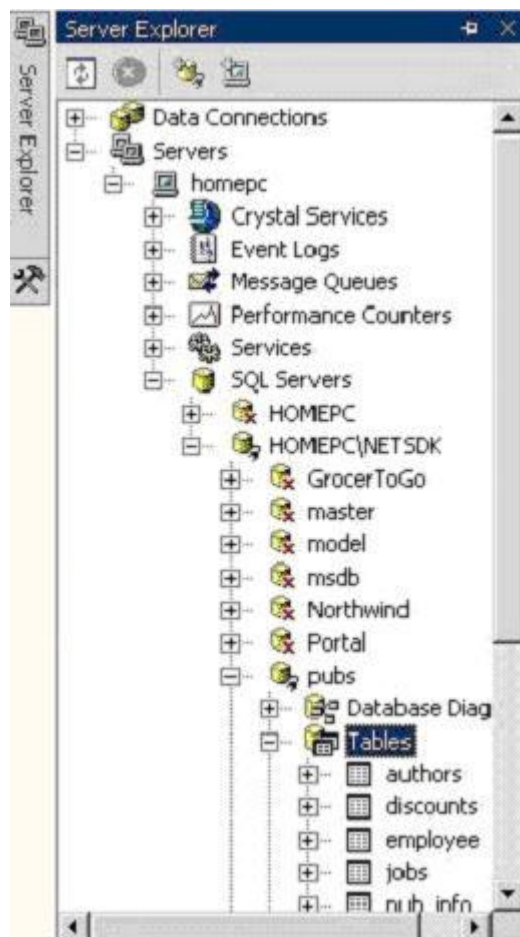
1. Create a Dataset during design time.
2. Create the .rpt file (from scratch) and make it point to the Dataset that we created in the previous step.
3. Place a CrystalReportViewer control on the .aspx page and set its properties to point to the .rpt file that we created in the previous step.
4. In your code behind page, write the subroutine to make the connections to the database and populate the dataset that we created previously in step one.
5. Call the Databind method from your code behind page.

### ***1. Creating a Dataset during Design Time to Define the Fields of the Reports***

- 1) Right click on "Solution Explorer", select "Add" --> select "Add New Item" --> Select "DataSet"



2) Drag and drop the "Stores" table (within the PUBS database) from the "SQL Server" Item under "Server Explorer".



3) This should create a definition of the "Stores" table within the Dataset



The .xsd file created this way contains only the field definitions without any data in it. It is up to the developer to create the connection to the database, populate the dataset and feed it to the Crystal Report.

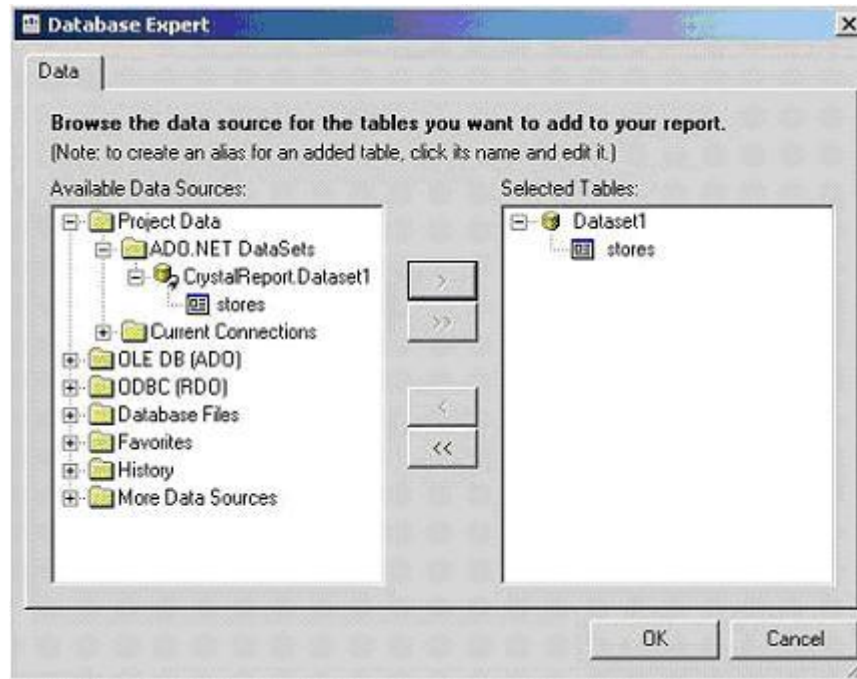
## ***II. Creating the .rpt File***

4) Create the report file using the steps mentioned previously. The only difference here is that instead of connecting to the Database thru Crystal Report to get to the Table, we would be using our DataSet that we just created.

5) After creating the .rpt file, right click on the "Details" section of the Report file, select "Add/Remove Database"

6) In the "Database Expert" window, expand "Project Data" (instead of "OLE DB" that was selected in the case of the PULL Model), expand "ADO.NET DataSet", "DataSet1", and select the "Stores" table.

7) Include the "Stores" table into the "Selected Tables" section by clicking on ">" and then Click "ok"



8) Follow the remaining steps to create the report layout as mentioned previously in the PULL Model to complete the .rpt Report file creation

### ***III. Creating a CrystalReportViewer Control***

9) Follow the steps mentioned previously in the PULL Model to create a Crystal Report Viewer control and set its properties.

Code Behind Page Modifications: 10) Call this subroutine in your page load event:

```
Sub BindReport()
    Dim myConnection As New SqlConnection()
    myConnection.ConnectionString = "server=
(local)\NetSDK;database=pubs;Trusted_Connection=yes"
    Dim MyCommand As New SqlCommand()
    MyCommand.Connection = myConnection
    MyCommand.CommandText = "Select * from Stores"
    MyCommand.CommandType = CommandType.Text
    Dim MyDA As New SqlDataAdapter()
    MyDA.SelectCommand = MyCommand
    Dim myDS As New Dataset1()
    'This is our DataSet created at Design Time
    MyDA.Fill(myDS, "Stores")
    'You have to use the same name as that of your Dataset that you created
    during design time
    Dim oRpt As New CrystalReport1()
    ' This is the Crystal Report file created at Design Time
    oRpt.SetDataSource(myDS)
    ' Set the SetDataSource property of the Report to the Dataset
    CrystalReportViewer1.ReportSource = oRpt
```

```
' Set the Crystal Report Viewer's property to the oRpt Report object that we created  
End Sub
```

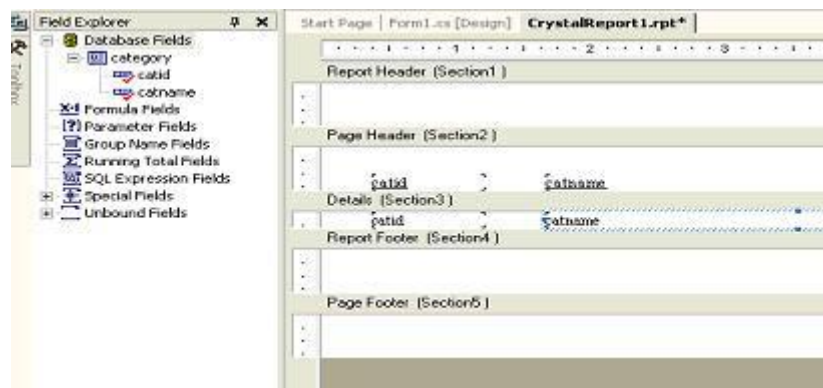
Note: In the above code, you would notice that the object oRpt is an instance of the "Strongly Typed" Report file. If we were to use an "UnTyped" Report then we would have to use a ReportDocument object and manually load the report file into it.

## Enhancing Crystal Reports

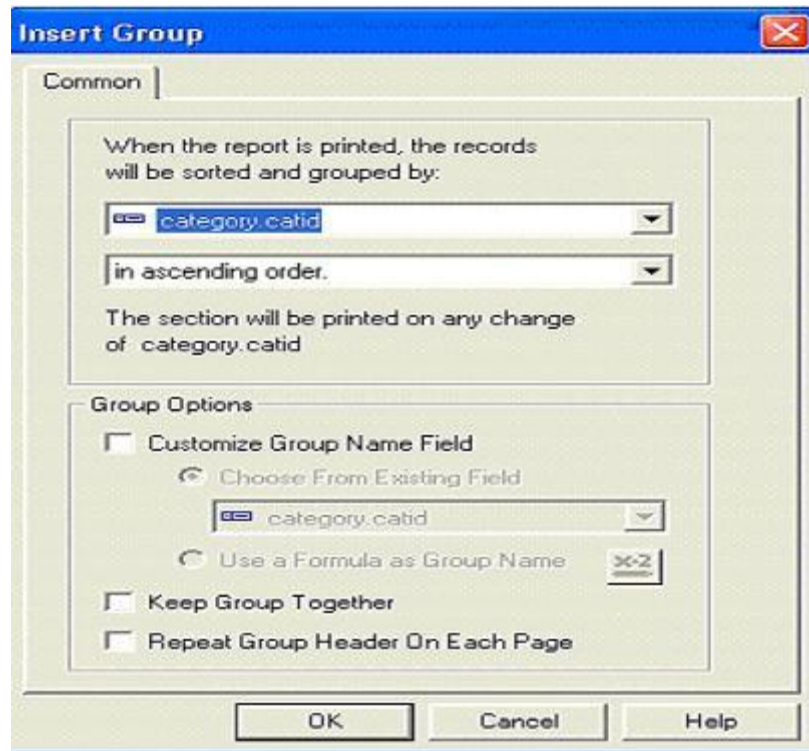
### *Accessing filtered data through Crystal reports*

Perform the following steps for the same.

1. Generate a dataset that contains data according to your selection criteria, say "where (cost>1000)".
2. Create the Crystal Report manually. It would look like this.

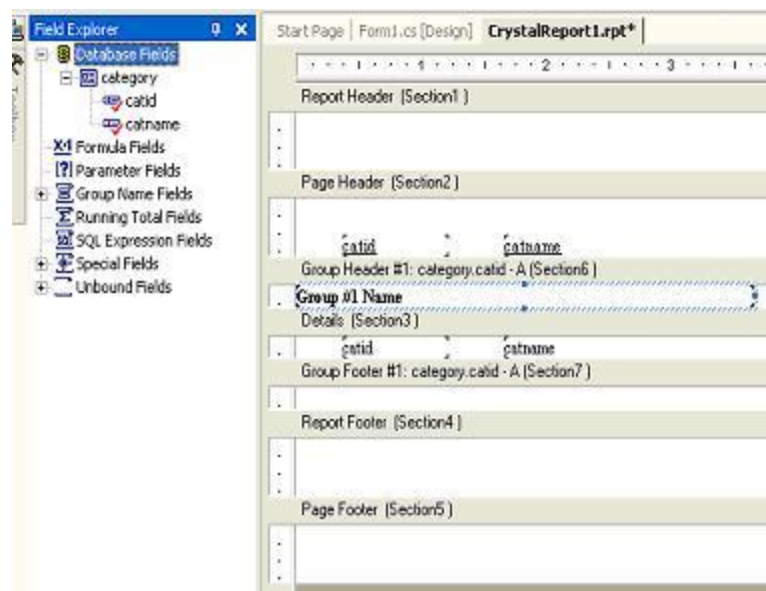


3. Right Click Group Name fields in the field Explorer window and select insert Group from the shortcut menu. Select the relevant field name from the first list box as shown.



The group name field is created, since the data needs to be grouped on the basis of the cat id say.

4. A plus sign is added in front of Group Name Filed in the field explorer window. The Group Name Field needs to be added to the Group Header section of the Crystal Report. Notice this is done automatically.



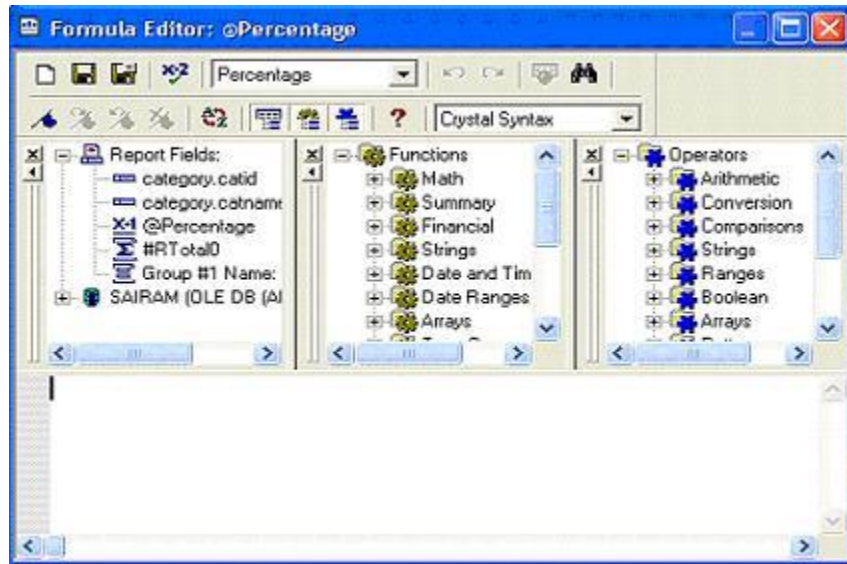
5. Right Click the running total field and select new. Fill the required values through > and the drop down list.

6. Since the count of number of categories is to be displayed for the total categories, drag RTotal0 to the footer of the report.

### **Create a formula**

Suppose if the report required some Calculations too. Perform the following steps:

1. Right Click the formula Fields in the field explorer window and select new. Enter any relevant name, say percentage.



2. A formula can be created by using the three panes in the dialog box. The first pane contains all the crystal report fields, the second contains all the functions, such as Avg, Sin, Sum etc and the third contains operators such as arithmetic, conversion and comparison operators.

3. Double click any relevant field name from the first pane, say there's some field like advance from some CustOrder table. Then expand Arithmetic from the third pane and double click Divide operator.

4. Double click another field name from the first which you want to use as divisor of the first field name already selected say it is CustOrder.Cost.

5. Double Click the Multiply from third pane and the type 100.

6. The formula would appear as  $\{CustOrder.Advance\} / \{CustOrder.Cost\} * 100$ .

7. Save the formula and close Formula Editor:@Percentage dialog box.

8. Insert the percentage formula field in the details pane.

9. Host the Crystal report.

## ***Exporting Crystal reports***

When using Crystal Reports in a Web Form, the CrystalReportViewer control does not have the export or the print buttons unlike the one in Windows Form. Although, we can achieve export and print functionality through coding. If we export to PDF format, Acrobat can handle the printing for us, as well as saving a copy of the report.

You can opt to export your report file into one of the following formats:

- PDF (Portable Document Format)
- DOC (MS Word Document)
- XLS (MS Excel Spreadsheet)
- HTML (Hyper Text Markup Language - 3.2 or 4.0 compliant)
- RTF (Rich Text Format)

To accomplish this you could place a button on your page to trigger the export functionality.

When using Crystal Reports in ASP.NET in a Web Form, the CrystalReportViewer control does not have the export or the print buttons like the one in Windows Form. We can still achieve some degree of export and print functionality by writing our own code to handle exporting. If we export to PDF format, Acrobat can be used to handle the printing for us and saving a copy of the report.

### ***Exporting a Report File Created using the PULL Model***

Here the Crystal Report takes care of connecting to the database and fetching the required records, so you would only have to use the below given code in the Event of the button.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Dim myReport As CrystalReport1 = New CrystalReport1()
    'Note : we are creating an instance of the strongly-typed Crystal Report file
    here.

    Dim DiskOpts As CrystalDecisions.Shared.DiskFileDestinationOptions = New
CrystalDecisions.Shared.DiskFileDestinationOptions
    myReport.ExportOptions.ExportDestinationType =
CrystalDecisions.[Shared].ExportDestinationType.DiskFile
    ' You also have the option to export the report to other sources
    ' like Microsoft Exchange, MAPI, etc.

    myReport.ExportOptions.ExportFormatType =
CrystalDecisions.[Shared].ExportFormatType.PortableDocFormat
    'Here we are exporting the report to a .pdf format. You can
    ' also choose any of the other formats specified above.

    DiskOpts.DiskFileName = "c:\Output.pdf"
    'If you do not specify the exact path here (i.e. including
```

```
' the drive and Directory),  
'then you would find your output file landing up in the  
'c:\WinNT\System32 directory - atleast in case of a  
' Windows 2000 System  
myReport.ExportOptions.DestinationOptions = DiskOpts  
'The Reports Export Options does not have a filename property  
'that can be directly set. Instead, you will have to use  
'the DiskFileDestinationOptions object and set its DiskFileName  
'property to the file name (including the path) of your choice.  
'Then you would set the Report Export Options  
'DestinationOptions property to point to the  
'DiskFileDestinationOption object.  
  
myReport.Export()  
'This statement exports the report based on the previously set properties.
```

End Sub

## ***Exporting a Report File Created Using the PUSH Model***

Using Push Model, the first step would be to manually create connections and populate the Dataset, and set the Report's "SetDataSource" property to the required Dataset (in the same manner as shown in the Push Model example before). The next step would be to call the above given Export.

## ***Crystal Reports alternatives***

Many people found Crystal Reports hard to learn and use and they are seeking for simpler alternatives. I suggest you to try [Stimulsoft reports](#) especially if you work with complex reports. Stimulsoft is much easier to use but also easier to deploy with your application since you deploy only one more .NET assembly, written in C#. In some areas (designer, working with different data sources, subreports) it is even better than Crystal.

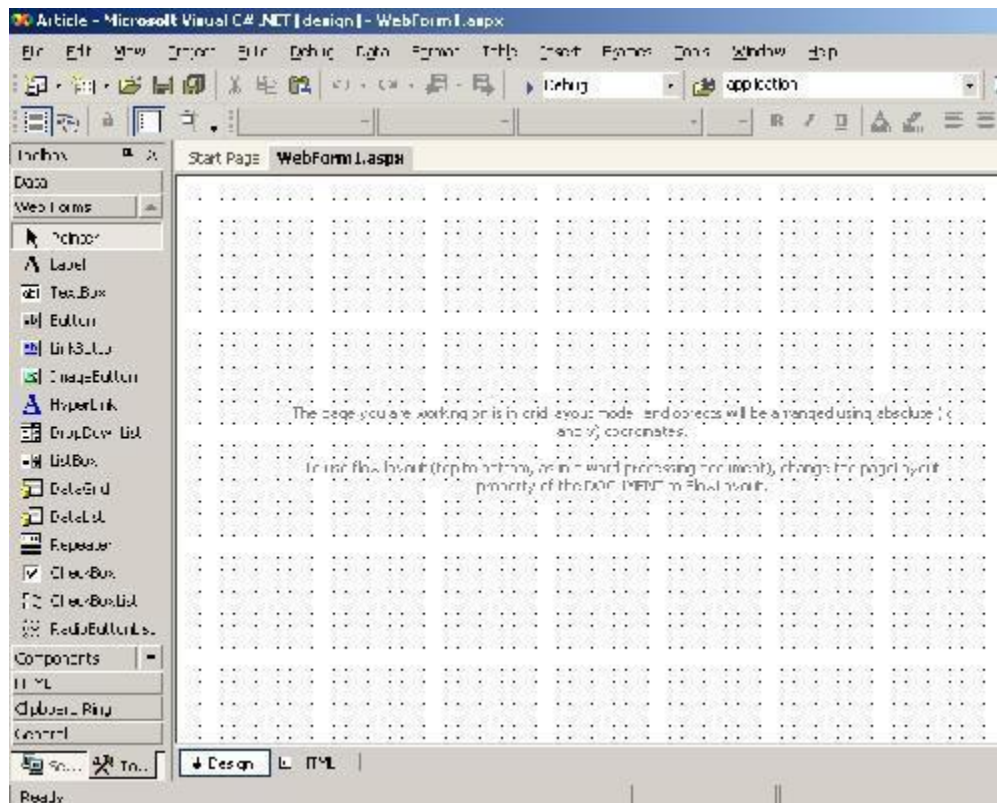
## ***Introduction***

This article explains how to use **PUSH** method for drawing reports. It will also explain how to use user **DataSets** in an ASP.NET page for reports. There are two types of methods for drawing the reports:

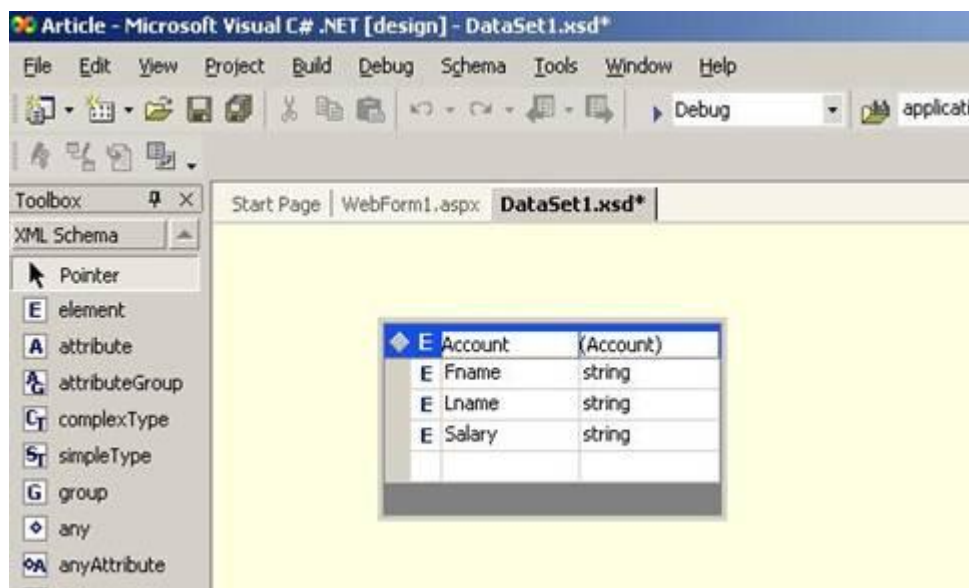
1. **PULL** method- the crystal report makes connection with the database, brings the fields data and draws the report.
2. **PUSH** method- we create the **DataSet**, choose fields of the **DataSet** as report fields and then push it to the crystal report. Here I am going to explain the **PUSH** method only.

## ***Steps***

1. Create a new ASP.NET project.



2. Insert new item as **DataSet**.



3. Add elements to the **DataSet** which you want on the report. **Save All**, then right click on the **DataSet** filename in the solution explorer and select command "Build and browse".
4. Then add a new item to the project as **Crystal report** and insert a blank report.

5. In the server explorer, right click database field and select database expert and expand the project data and select **DataSet** in the table selector and press OK.



6. Then drag the fields from the database fields of server explorer in the detail section of the report. Arrange the fields as you want.

From the toolbox, add crystal report viewer control on to the page. That will add this variable:

  [Copy Code](#)

```
protected CrystalDecisions.Web.CrystalReportViewer CrystalReportViewer1;
```

7. Populate the **DataSet**. Set the report **DataSource** and **CrystalReportViewer** report source.

  [Copy Code](#)

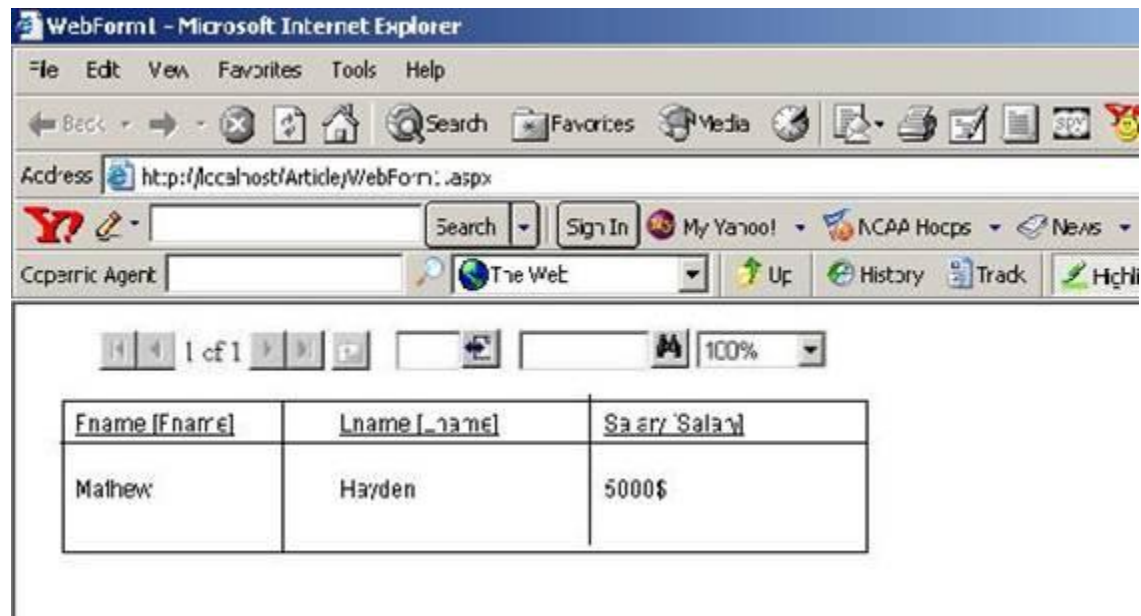
```
private void Page_Load(object sender, System.EventArgs e)
{
    CrystalReport1 report=new CrystalReport1();
    CrystalReportViewer1.Visible=true;
    DataSet ds=new DataSet("Account");//give same name as on
    //dataset1 table header
    DataTable table=new DataTable("Account");//give same name as on
    //dataset1 table header
```

```

table.Columns.Add("Fname",typeof(System.String));
table.Columns.Add("Lname",typeof(System.String));
table.Columns.Add("Salary",typeof(System.String));
DataRow row=table.NewRow();
row["Fname"]="Mathew";
row["Lname"]="Hayden";
row["Salary"]="5000$";
// add to table
table.Rows.Add(row);
ds.Tables.Add(table);
// set report's dataset
report.SetDataSource(ds);
// set report source
CrystalReportViewer1.ReportSource =report;
}

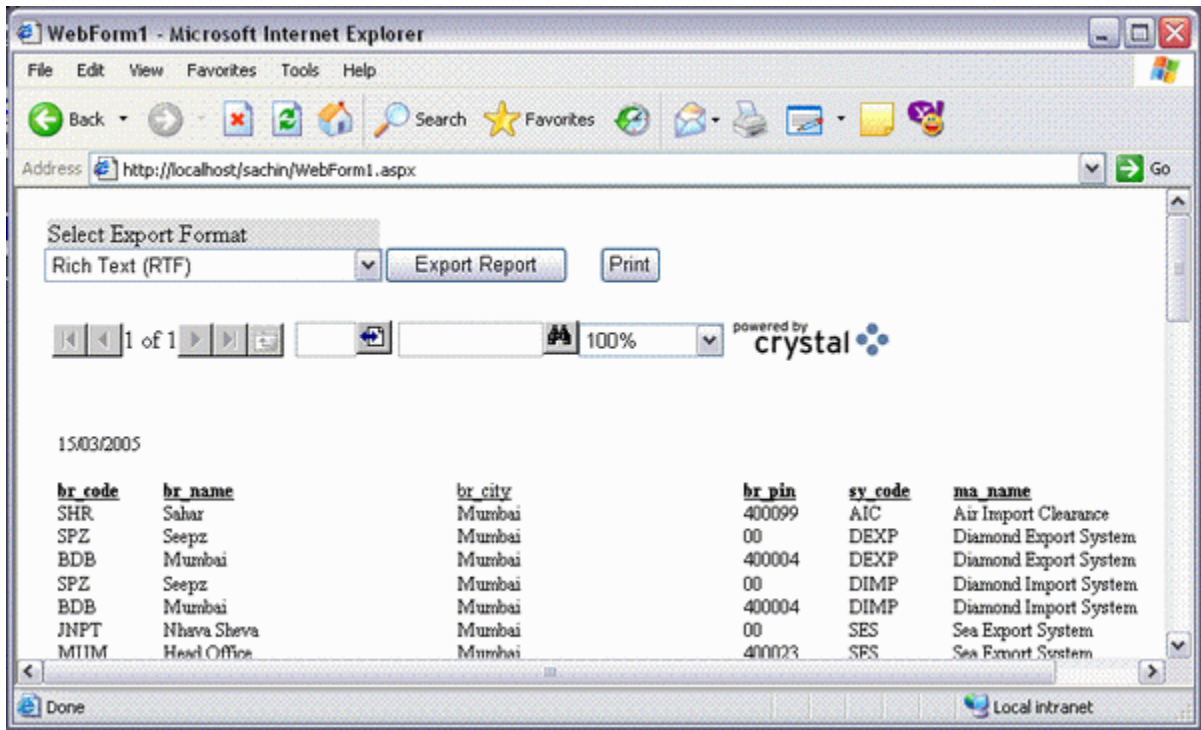
```

The output will be as follows...



### ***Points to be careful of***

1. Give same names on **DataSet** and the table element name of inserted **DataSet**.
2. As and when you modify **DataSet**, build it again and log off the current connections in report. Set the **DataSource** location again pointing to new **DataSet**, otherwise database fields of the report will not take the change.
3. Setting up **DataBind** properties of the report viewer can be avoided. It can be done at runtime



## Introduction

The main purpose of this document is to display the report without any error. I was bugged by the "Logon Failed Error" for several days, and now I finally have a code that displays report without any error. The code also Exports the report into .pdf, .xls, .rtf and .doc formats. It also prints the report directly to the printer.

## Using the code

Unzip the *crCodes.Zip* and then run the *crCode.vbproj* project file.

--- OR ---

Just insert the *Webform1.aspx* file into your existing project, then copy the *crystalreport2.rpt* file into your project, and start using the code.

The entire source code of *Webform1.aspx.vb* is as follows. Simply design the form as shown in the image and place the Crystal Report Viewer control, and leave the name of controls to default:

[Copy Code](#)

```
Imports CrystalDecisions.Shared
Imports System.IO

Public Class WebForm1
    Inherits System.Web.UI.Page

    Dim crReportDocument As CrystalReport2 = New CrystalReport2
```

```

    Protected WithEvents DropDownList1 As
System.Web.UI.WebControls.DropDownList
    Protected WithEvents Button1 As System.Web.UI.WebControls.Button
    Protected WithEvents Label1 As System.Web.UI.WebControls.Label
    Protected WithEvents CrystalReportViewer1 As _
        CrystalDecisions.Web.CrystalReportViewer
    Protected WithEvents Button2 As System.Web.UI.WebControls.Button

#Region " Web Form Designer Generated Code "
    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()>
    Private Sub InitializeComponent()

    End Sub

    'NOTE: The following placeholder declaration is required
    'by the Web Form Designer.
    'Do not delete or move it.
    Private designerPlaceholderDeclaration As System.Object

    Private Sub Page_Init(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
        InitializeComponent()

        ' this is the most IMPORTANT line of code
        ' if this line is not written the
        ' " LOGON FAILED" error starts displaying
        crReportDocument.SetDatabaseLogon("username", _
            "password", "sql-server", "database")

        ' the Above line works even if only username
        ' and password is supplied as below

        'crReportDocument.SetDatabaseLogon("username", _
            "password") ', "sql-server", "database")

        ' this will hide the group tree
        CrystalReportViewer1.DisplayGroupTree = False

        CrystalReportViewer1.ReportSource = crReportDocument

```

```

' IF REPORT Uses Parameter's
' Pass Paramaters As Follows
crReportDocument.SetParameterValue("city", "Mumbai")

' city = Parameter Name
' Mumbai = Parameter Value

' :-) thats ALL your Report Will Be displayed
' now Without Logon Failed Error

With DropDownList1.Items
    .Add("Rich Text (RTF)")
    .Add("Portable Document (PDF)")
    .Add("MS Word (DOC)")
    .Add("MS Excel (XLS)")
End With
End Sub

#End Region

Private Sub Page_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
End Sub

Sub ExportReport()

    Dim oStream As New MemoryStream ' // using System.IO

    'this contains the value of the selected export format.
    Select Case DropDownList1.SelectedItem.Text

        Case "Rich Text (RTF)"
            '-----

            oStream = crReportDocument.ExportToStream(_
                CrystalDecisions.Shared.ExportFormatType.WordForWindows)
            Response.Clear()
            Response.Buffer = True
            Response.ContentType = "application/rtf"
            '-----

            '-----

        Case "Portable Document (PDF)"

```

```

        oStream = crReportDocument.ExportToStream(_
            CrystalDecisions.Shared.ExportFormatType.PortableDocFormat)
        Response.Clear()
        Response.Buffer = True
        Response.ContentType = "application/pdf"
        '-----

        '-----
Case "MS Word (DOC)"

        oStream = crReportDocument.ExportToStream(_
            CrystalDecisions.Shared.ExportFormatType.WordForWindows)
        Response.Clear()
        Response.Buffer = True
        Response.ContentType = "application/doc"
        '-----

-
        '-----
-

Case "MS Excel (XLS)"

        oStream = crReportDocument.ExportToStream(_
            CrystalDecisions.Shared.ExportFormatType.Excel)
        Response.Clear()
        Response.Buffer = True
        Response.ContentType = "application/vnd.ms-excel"
        '-----

-

End Select 'export format
Try
    Response.BinaryWrite(oStream.ToArray())
    Response.End()
Catch err As Exception
    Response.Write("< BR >")
    Response.Write(err.Message.ToString)
End Try
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button2.Click
    ExportReport()
End Sub

```

```
Private Sub Button1_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles Button1.Click  
    crReportDocument.SetDatabaseLogon("USER", _  
        "PASSWORD", "SQL-SERVER", "DATABASE")  
    crReportDocument.PrintToPrinter(1, False, 0, 0)  
End Sub  
End Class
```

## Why is the "Logon Failed Error" generated

The "Logon Failed" error is generated basically because, when the report is being displayed it tries to log on to the database server. Even though you have selected the server while designing, the report still needs the server name while displaying or exporting or printing.

The code line that removes the error is:

  [Copy Code](#)

```
crReportDocument.SetDatabaseLogon("USER", "PASSWORD", "SQL-SERVER", "DATABASE")
```

Or can also be used as:

  [Copy Code](#)

```
crReportDocument.SetDatabaseLogon("USER", "PASSWORD")
```

### Abstract

One of the most common questions regarding Crystal Reports in ASP.NET is how to automatically print a Crystal Report. Because the Internet is a stateless, disconnected medium, automatic printing of a Crystal Report using Web Forms cannot be accomplished as easily as when using Windows Forms. This article examines two options for automatic printing of a Crystal Report using ASP.NET and compares one alternative method where the user must initiate printing.

### Article Contents:

- [Introduction](#)
- [Using Server-Side ReportDocument.PrintToPrinter Method](#)
- [Client Side JavaScript: window.Print](#)
- [Export to PDF](#)
- [Conclusion and References](#)

### Introduction

[ [Back To Top](#) ]

Unlike the Windows Forms CrystalReportViewer toolbar, the web-based

CrystalReportViewer toolbar does not include a print button. For reports that are directly rendered as HTML, the user can simply use the browser's print button. For reports rendered as PDF or Excel, the user can use the print functionality in either the Acrobat Reader or Excel to print the report.

Most users can figure out how to print the report, but some still need a little help, and in some cases, clients request this as part of their application specs. In this article, we'll review two methods for automatically printing a Crystal Report using the web-based CrystalReportViewer: the server side ReportDocument.PrintToPrinter method, and a small client-side JavaScript that uses the window.print method. We'll also discuss rendering a report in PDF and allowing the user to print that document as an alternative to the two other methods. Because of the disconnected nature of web client applications, and security considerations, there isn't a perfect way to make every web-based Crystal Report automatically print. But, under certain conditions, we can provide this functionality in user-friendly ways.

#### **Using Server-Side ReportDocument.PrintToPrinter Method**

[ [Back To Top](#) ]

This server-based method is documented in the Visual Studio help files. Open the Help Index, and enter PrintToPrinter in the "Look for:" box. The syntax for this method is:

```
Report.PrintToPrinter(<copies as int>, <collated as True/False>, <startpage as int>, <endpage as int>)
```

'Collated' in this context has nothing to do with database collation. Some advanced printers (like copier/printers) will sort each document into its own shelf. Not every printer supports this functionality, so check the printer dialog before setting to true. To print the entire report, set startpage and endpage each to 0.

An example in use might look like

```
MyReport.PrintToPrinter(1,False,0,0)
```

One limitation of this method is that a printer name must be specified. You can set the default printer at design time in the report, and you can change the printer name at run time by setting the ReportDocument.PrintOptions.PrinterName property (the PrintOptions are also where you can assign page margins, portrait/landscape, etc.). Keep in mind that this method prints from the server itself, not from the client machine. This means that any printer you wish to use must be accessible from the server. You cannot print to a client's desktop printer using this method unless that printer is shared on the network and mapped to the server.

If your LAN has networked printers, you can make some guesses as to which printer to assign as the default in each report. For instance, if the Accounting Department has a departmental printer, that would be a good choice to use for the default in an accounting report. You can provide users the option to choose a networked printer by enumerating the printers mapped to the server and populating a drop-down list. The PrinterSettings.InstalledPrinters property returns a collection of installed printers,

and can be bound to a DropDownList as below.

```
DropDownList1.DataSource = System.Drawing.Printing.PrinterSettings.InstalledPrinters  
DropDownList1.DataBind()
```

Again, since this code is executing on the server, this will only enumerate the printers on the server mapped to the System user. If a printer does not appear in the drop-down list, you need to ensure that it is properly mapped to the System user (see below).

All printers and the .NET Framework need to be mapped to the System user in order for this to work. This is not a trivial task to configure. Printers can be mapped to the system account only by editing the registry, and since the Framework is mapped to the System user, you may need to reassign security permissions on files and folders. This process is outlined in the "Printing Web-Based Reports From the Server" article in the VS .NET help files (look up "Crystal Reports, printing" in the VS help index to link to the articles.) and also in Reference 3 (at the end of this article). The process is a little too intricate to cover here (leave comments if help is needed, and I can amend the article later).

**Client Side JavaScript: window.Print**

[ [Back To Top](#) ]

A second option you have uses the JavaScript window.Print method. Adding this method to the onLoad event of the BODY tag will trigger the browser to print the current page. Adding this method to a button.onClick event can also be done, just make sure you are using a standard HTML button and not an ASP.NET button control (if you do use an ASP.NET button control, you'll cause a postback, not execute the JavaScript). In a future article, we'll look at adding a print button to a custom CrystalReportViewer toolbar that uses this method to print a report.

The window.print method does cause a printer selection window to open. The user will have a chance to select the printer they want to use and must click the "Print" button to start printing. There is no way to disable the prompt with JavaScript, so for Netscape and other browsers, users will have to see the dialog. However, in some versions of Internet Explorer, a call to the ExecWB OLE method from a second browser object can be used to circumvent the print dialog (note: spaces have been added to OBJECT declarations to display the code properly in this article--you should remove the spaces to use the code):

```
function PrintWindow(){  
if (navigator.appName == "Microsoft Internet Explorer") {  
    var PrintCommand = '< O B J E C T ID="PrintCommandObject" WIDTH=0 HEIGHT=0 '  
    PrintCommand += 'CLASSID="CLSID:8856F961-340A-11D0-A96B-00C04FD705A2"></O B J E C  
    T>';  
    document.body.insertAdjacentHTML('beforeEnd', PrintCommand);  
    PrintCommandObject.ExecWB(6, -1); PrintCommandObject.outerHTML = ""; }  
else { window.print(); } }
```

The ExecWB method requires IE 5.5 or higher to function. This script essentially creates another browser object of size 0 at the end of the current page. This new browser object then issues a print command without prompting the user.

A limitation of the `window.Print` method is that only the current window will print. In order to print all pages, you must set the `SeparatePages` property of your `CrystalReportViewer` to `False` when the report is rendered. For example:

```
CrystalReportViewer1.SeparatePages=False  
CrystalReportViewer1.ReportSource=MyReport
```

Setting `SeparatePages` to `False` will cause the entire report to be shown in the current window. The user will not have to page through the report, but a lot of scrolling may be required. Page breaks will appear wherever the browser puts them--there is no way to control where page breaks occur--and the browser may wrap the layout in order for it to fit the printer page dimensions. Your formatting may be completely lost using the `window.Print` method if your report is wider than the browser window.

The basic `window.Print` method also prints everything in the window, including whatever form inputs you have added, and the `CrystalReportViewer` toolbar. If you have set `SeparatePages=False`, you probably don't need the toolbar displayed.

If you do not have any other inputs or buttons on your report page, turning off the toolbar may be all you need to do. If you did add inputs or other buttons, having them show on the printout will not look very professional. To make the report appear a little more professional, we can enclose the report output in a `DIV` (by enclosing the `CrystalReportViewer` in a `DIV` or `Panel`) and then print just that `DIV` using a JavaScript function that opens a new window and includes only the information inside the `DIV`. In a future article, we'll look at "skinning" the toolbar so you can have a toolbar and this printing functionality at the same time. The basic code is shown below (source: Reference 2; note that spaces have been added in some tag names for them to display in this article - you should remove the spaces to use the code):

```
var gAutoPrint = true; // Flag for whether or not to automatically call the print  
function  
function printSpecial() {  
    if (document.getElementById != null) {  
        var html = '\n\n';  
        if (document.getElementsByTagName != null) {  
            var headTags = document.getElementsByTagName("head");  
            if (headTags.length > 0)  
                html += headTags[0].innerHTML;  
        }  
        html += '\n< / H E A D >\n< B O D Y>\n';  
        var printReadyElem = document.getElementById("printReady");  
        if (printReadyElem != null) {  
            html += printReadyElem.innerHTML; }  
        else {  
            alert("Could not find the printReady section in the HTML"); return; }  
        html += '\n</ B O D Y >\n</ H T M L >';  
        var printWin = window.open("", "printSpecial");  
        printWin.document.open();  
        printWin.document.write(html);  
        printWin.document.close();  
        if (gAutoPrint) printWin.print();  
    }  
    else {  
        alert("Sorry, the print ready feature is only available in modern browsers.");  
    }  
}
```

If you want to get really fancy, you can combine the two methods so that a report window will automatically open and print in IE browsers or open and prompt for printing in non-IE browsers. The combined code is shown below and has been tested in both IE 6.0 and FireFox 0.9 (note: spaces have been inserted into tag names so code will display properly--you should remove the spaces to use the code):

```
var gAutoPrint = true; // Flag for whether or not to automatically call the print
function
function printSpecial() {
if (document.getElementById != null) {
var html = '< H T M L >\n< H E A D >\n';
if (document.getElementsByTagName != null) {
var headTags = document.getElementsByTagName("head");
if (headTags.length > 0) html += headTags[0].innerHTML;
}
if (gAutoPrint) {
if (navigator.appName == "Microsoft Internet Explorer") {
html += '\n</ H E A D >\n<'
html += 'B O D Y onLoad="PrintCommandObject.ExecWB(6, -1);">\n';
}
else {
html += '\n</ H E A D >\n< B O D Y >\n';
} }
else {
html += '\n</ H E A D >\n< B O D Y >\n';
}
}

var printReadyElem = document.getElementById("printReady");
if (printReadyElem != null) {
html += printReadyElem.innerHTML;
}
else {
alert("Could not find the printReady section in the HTML");
return;
}
}
if (gAutoPrint) {
if (navigator.appName == "Microsoft Internet Explorer") {
html += '< O B J E C T ID="PrintCommandObject" WIDTH=0 HEIGHT=0 '
html += 'CLASSID="CLSID:8856F961-340A-11D0-A96B-00C04FD705A2"></ O B J E C T >\n</ B
O D Y >\n</ H T M L >'; }
else {
html += '\n</ B O D Y >\n</ H T M L >';
} }
else {
html += '\n</ B O D Y >\n</ H T M L >';
}
}
var printWin = window.open("", "printSpecial");
printWin.document.open();
printWin.document.write(html);
printWin.document.close();
if (gAutoPrint) {
if (navigator.appName != "Microsoft Internet Explorer") {
printWin.print();
}}}
else {
alert("Sorry, the print ready feature is only available in modern browsers.");
}}
```

**Export to PDF**

[ [Back To Top](#) ]

As you may have learned from experience, the output from a CrystalReportViewer does not always look like you had intended. That is because the output is rendered as HTML and sent to the browser that then generates the web page. In order to get the exact formatting you want, you should export your reports as PDF documents. One advantage of doing so is that most users know how to print from the Acrobat Reader. Also, users can save reports and send them as e-mail attachments.

One downside to this option is that users must start the printing process themselves, which does not solve the problem of minimizing user interaction. Unfortunately, since a PDF is not a page rendered by the browser, none of the JavaScript tricks above will work for us. There is no simple way to automatically print an exported Crystal Report in PDF format.

### **Conclusion and References**

[ [Back To Top](#) ]

If your report is fairly simple, you can use the client-side JavaScript `window.Print` command and a little scripting magic to come up with a satisfactory solution. Complex reports generated using the JavaScript `window.Print` may lose the formatting you desired, and, in non-IE browsers, some user intervention is required to answer the Printer Setup dialog.

In a controlled networked environment, the `PrintToPrinter()` method may provide all the functionality you need. Networked printers can be enumerated in a drop-down list, and the user can choose which printer to use. Most of the limitations have to do with the Internet being a disconnected medium and browser security, which also limits intranet functionality. Using the `PrintToPrinter` method requires editing the registry and remapping the .NET Framework user context and limits printing to networked printers.

Printing Crystal Reports is not a process that can be easily automated in a very satisfactory manner. There is no single magic solution to fit every need; you will need to see which option works best in each situation. Future versions of Crystal Reports .NET may contain additional functionality, and there are additional tricks we can use in Crystal Reports 10 that will be looked at in a future article.

A web developer may or may not be aware of crystal report in depth as compared to desktop application developer.

Couple of months back I was suppose to access crystal reports in asp.net, but I found many problems while working with crystal report using crystalreportviewer in asp.net. I am writing this application for web-developers who want to access crystal reports in asp.net by using multiple tables or views etc. This application might help them and save their development time.

List of problems which I came across while accessing crystal reports in asp.net

- Logon failure  
Refer to following URL  
<http://support.businessobjects.com/library/kbase/articles/c2010371.asp>
- DLL not found  
On a development machine if crystal report is not installed then you might get above error
- Fail to render page  
Basically there could be two possible ways for the solution
  1. This error can occur due to the unsuppressed report header, removing space between the header might solve the problem.
  2. This error can be solved by assigning permission to ASPNET user by default. ASPNET user does not have permission on windows XP and XP professional machine. To explain in detail it creates crystal report image in the local machine's temp directory or in the same directory where application is residing, due to security issue or permission problem. ASP.net unable to render image created in the memory and hence error occur.

Refer to following URL for the detail information

<http://support.businessobjects.com/library/kbase/articles/c2013414.asp>  
<http://support.businessobjects.com/library/kbase/articles/c2014843.asp>  
<http://support.businessobjects.com/library/kbase/articles/c2014618.asp>

## Merge Modules

Once build is ready and wanted to deployed it on the development server. You should add merge module in the build. This is important because you never know that crystal report is installed on the development machine and it is not possible for you to provide crystal report to the client. You can include following modules in the build and it will take care of all the further issues of deployment on the server.

- Crystal\_managed2003.msm
- Crystal\_database\_Access2003.msm
- Crystal\_datavase\_Access2003.msm
- Crystal\_regwiz2003.msm

-- I have used Northwind [database](#) of SQL-SERVER  
 --View Categorywise\_products on product and categories table

```
create view ProductsList AS
SELECT Categories.CategoryName, Products.ProductName, Products.QuantityPerUnit,
Products.UnitsInStock, Products.Discontinued
FROM Categories INNER JOIN Products ON Categories.CategoryID =
Products.CategoryID
WHERE Products.Discontinued <> 1
```

To create a dataset object from a database

Create a new schema [file](#) in the project:

1. In the Solution Explorer, right-click the project name, point to Add, and click Add New Item.
2. In the Categories area of the Add New Item dialog box, expand the folder and select Data.
3. In the Templates area, select Dataset.
4. Accept the default name Dataset1.xsd.

This creates a new schema file (Dataset1.xsd) that will be used to generate a strongly-typed dataset. The schema file will be displayed in the ADO.NET Dataset Designer.

Specify where the database is located:

1. In the Server Explorer, right-click Data Connections and select Add Connection.
2. In the Data Link Properties dialog box, click the Provider tab and select a provider (for example, Microsoft OLE DB Provider for [SQL Server](#)).
3. Click the Connection tab and specify the location of your database. Enter [server](#) and logon information where necessary.
4. Click OK.

Your database, its tables, and its fields now appear in the Server Explorer under the Data Connections node.

In the Solution Explorer, double-click Dataset1.xsd, if it is not already the active view. Dataset1.xsd should now be displayed in the Dataset tab.

To build a schema for your dataset, drag the desired tables from the Server Explorer to the Dataset tab of Dataset1.xsd. Click Save Dataset1.xsd to save the Dataset1.xsd file. On the Build menu, click Build to generate the dataset object for the project.

There is another reason why I have created xsd (schema file) for report. Let us consider a situation where you want to deploy your application on client's server. Now situation is you have set location of crystal report to your development server and though you set logon information programmatically logon failure occurs. To overcome this problem, I have a view associated in schema file. I can set location of ADO.NET Dataset to the xsd view and this will take care of all the problems for setting location while runtime, since it is independent of any of the server and to access dataset we are applying separate connection string from the web.config file. This way you can get rid of all the problems to access crystal reports in asp.net.

Actual code as below

```
Imports System
Imports System.Collections
Imports System.ComponentModel
Imports System.Data
Imports System.Data.SqlClient
Imports System.Drawing
Imports System.Web
```

```

Imports System.Web.SessionState
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Web.UI.HtmlControls
Imports CrystalDecisions.Shared
Imports CrystalDecisions.CrystalReports.Engine
Imports System.Configuration
Namespace CrystalReports
    ''' <summary>
    ''' Summary description for WebForm1.
    ''' </summary>
Public Class WebForm1
    Inherits System.Web.UI.Page
    Protected CrystalReportViewer1 As CrystalDecisions.Web.CrystalReportViewer
    Private Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        ' Put user code to initialize the page here
    End Sub 'Page_Load

Protected Overrides Sub OnInit(ByVal e As EventArgs)
    '
    ' CODEGEN: This call is required by the ASP.NET Web Form Designer.
    '
    InitializeComponent()
    MyBase.OnInit(e)
    'Database connectivity
    Dim mycon As New
SqlConnection(ConfigurationSettings.AppSettings("connectionString"))
    Dim dtbl As New DataTable
    Dim dtst As New DataSet
    Dim sqdt As SqlDataAdapter
    Try
        'Database activity
        sqdt = New SqlDataAdapter("SELECT * FROM ProductsList", mycon)
        sqdt.Fill(dtbl)
        dtst.Tables.Add()
        Dim crptProList As New productList
        crptProList.SetDataSource(dtst)
        CrystalReportViewer1.DisplayGroupTree = False
        CrystalReportViewer1.ReportSource = crptProList
        'Passing parameter to the crystal report
        'let us assume that i have added a parameter to the crystal report which
accepts
        'categoryname and based on this generate report this parameter you can pass
        'from any web form as a request parameter
        'string catName = Request.QueryString["catageoryName"];
        'you can set parameter after setting report source as below

'crptProList.SetParameterAValue("CatagoryName",catName) similarly you can set
'multiple parameters to the crystal report
'Overcomming problem of setting location at runtime -----
-----

```

```

                                'Setting Logon Information-----
                                Catch ex As Exception
                                Response.Write(ex.Message)
                                End Try
                                End Sub

                                ''' <summary>
                                ''' Required method for Designer support - do not modify
                                ''' the contents of this method with the code editor.
                                ''' </summary>

                                PrivateSub InitializeComponent()
                                EndSub'InitializeComponent

                                EndClass'WebForm1
                                EndNamespace'CrystalReports

```

## Working with Parameters with Crystal Reports and ASP.NET 2.0

(Page 1 of 6 )

This is the second article in a series that covers programming with Crystal Reports with ASP.NET 2.0. In this article, we will focus on working with parameters using Crystal Reports and passing the parameter values from an ASP.NET 2.0 web site.

---

A [downloadable zip file](#) is available for this article.

---

If you are new to Crystal report programming, I strongly suggest that you read the first article in this series, [Programming Crystal Reports with ASP.NET 2.0](#). This article uses and enhances the solution (source code) provided at the above link.

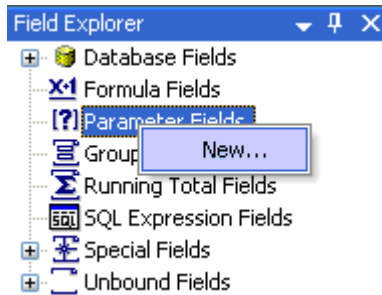
The entire solution (source code) for this article is available as a free download in the form of a zip. The source code in this article has been developed using Microsoft [Visual Studio 2005](#) Professional Edition on Microsoft Windows XP Professional Edition with Microsoft SQL Server 2005 Express Edition. I used the same version of Crystal Reports which comes with Visual Studio 2005 Professional Edition. I didn't test the code in any other tools/IDEs/servers/editions/versions. If you have any problems, please feel free to post in the discussion area.

### Adding a single parameter to the Crystal Report at design time

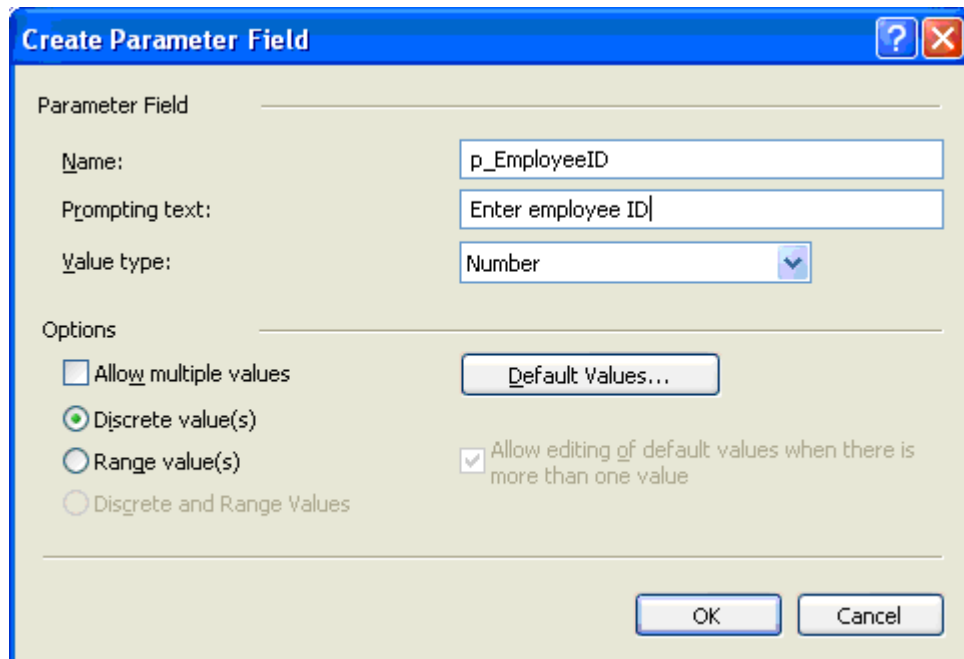
Before proceeding further, open the "SampleWebSite01" website (available as a download with this article) and add a new Crystal Report (named "SampleParam1.rpt") focusing on the Orders table of the Northwind database. Make sure you add OrderID, CustomerID, EmployeeID and OrderDate to the report. All of this is explained in the first article of this series.

The following are the steps required to add a parameter to the Crystal report:

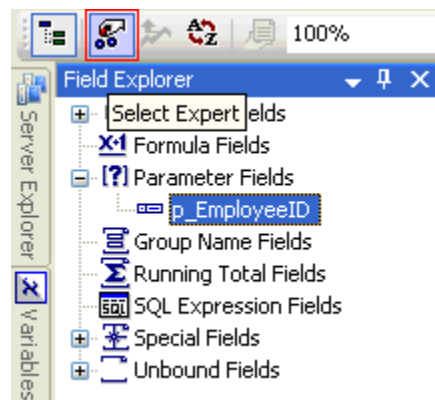
Using the "Field Explorer," right click on "Parameter Fields" and click "New" as shown in the following figure (01).



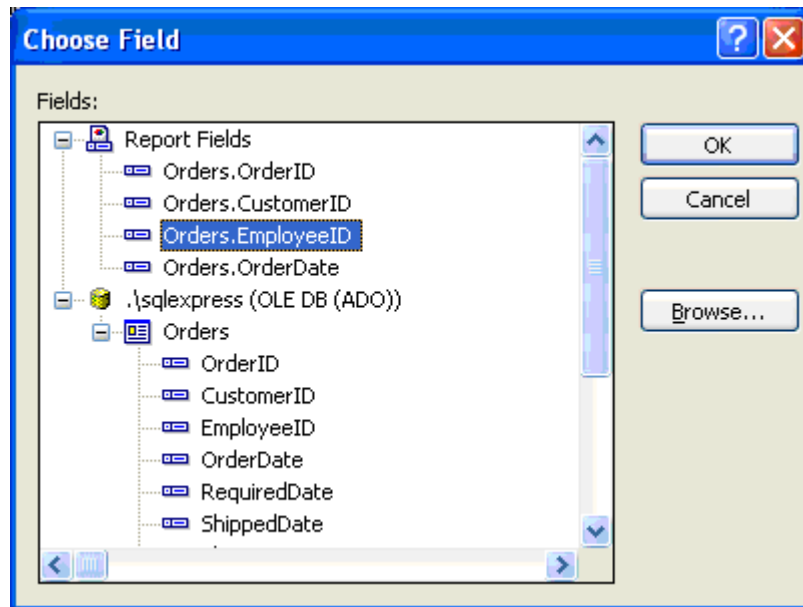
Enter "Name" as "p\_EmployeeID," "Prompting Text" as "Enter Employee ID," specify "Value Type" as "Number" and click on "OK" as shown below in figure 02.



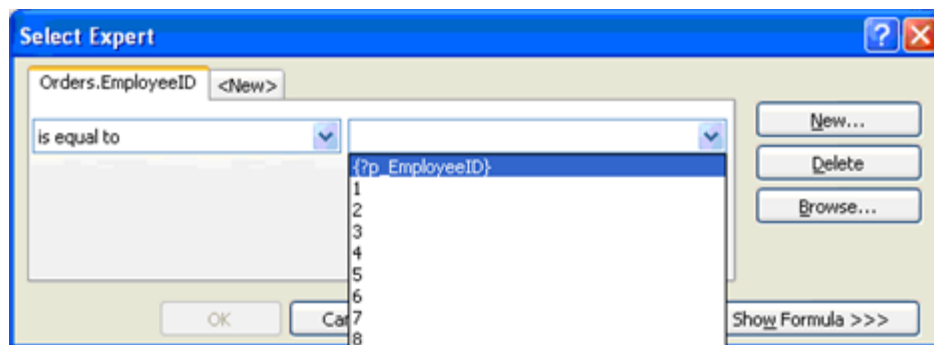
Click on "Select Expert" on the tool bar, as highlighted in red in figure 03.



In the "Choose Field" dialog box, select "Orders.EmployeeID" and click on OK as shown in figure 04.



In the "Select Expert" dialog box, select the operator as "is equal to," select "{?p\_EmployeeID}" as the value (shown in Figure 05) and click OK.

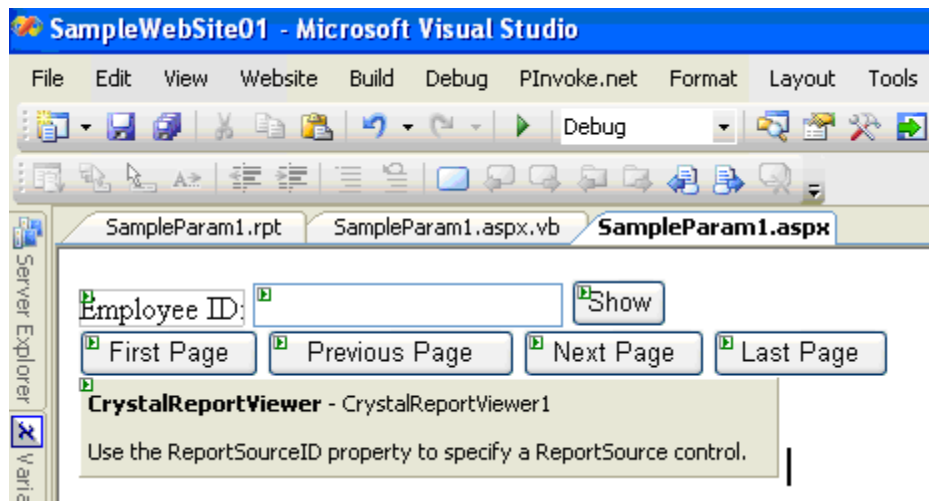


### **Working with Parameters with Crystal Reports and ASP.NET 2.0 - Passing the value to Crystal Report Parameter dynamically: source code**

(Page 2 of 6 )

In the previous section, we added a parameter named "p\_EmployeeID" to the report. Now it is time to access the report (say, get a list of orders) based on the user specified value of Employee ID.

Let us modify the web page so that it looks like the following (Fig 06):



The Source for the above page design is as follows:

```
<form id="form1" runat="server">
  <div>
    <asp:Label ID="lblEmployeeID" runat="server" Text="Employee
ID:"></asp:Label>
    <asp:TextBox ID="txtEmployeeID" runat="server"></asp:TextBox>
    <asp:Button ID="btnShow" runat="server" Text="Show" /><br />
    <asp:Button ID="btnFirst" runat="server" Text="First Page" />
    <asp:Button ID="btnPrevious" runat="server" Text="Previous Page" />
    <asp:Button ID="btnNext" runat="server" Text="Next Page" />
    <asp:Button ID="btnLast" runat="server" Text="Last Page" /><br />
    <CR:CrystalReportViewer ID="CrystalReportViewer1" runat="server"
AutoDataBind="true" DisplayGroupTree="False" DisplayToolbar="False"
EnableDatabaseLogonPrompt="False" EnableParameterPrompt="False" Height="1055px"
ReuseParameterValuesOnRefresh="True" Width="789px" />
  </div>
</form>
```

The following is the code for the "Show" button:

```
Protected Sub btnShow_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnShow.Click
  Dim ConnInfo As New ConnectionInfo
  With ConnInfo
    .ServerName = ".sqlexpress"
    .DatabaseName = "Northwind"
    .UserID = "sa"
    .Password = "eXpress2005"
  End With

  Me.CrystalReportViewer1.ParameterFieldInfo.Clear()
  If Me.txtEmployeeID.Text.Trim.Length > 0 Then
    Me.CrystalReportViewer1.ReportSource = Server.MapPath("SampleParam1.rpt")
    Dim ParamFields As ParameterFields =
Me.CrystalReportViewer1.ParameterFieldInfo
    Dim p_EmpID As New ParameterField
    p_EmpID.Name = "p_EmployeeID"
    Dim p_EmpID_Value As New ParameterDiscreteValue
    p_EmpID_Value.Value = Me.txtEmployeeID.Text
    p_EmpID.CurrentValues.Add(p_EmpID_Value)
    ParamFields.Add(p_EmpID)
```

```

Else
    Me.CrystalReportViewer1.ReportSource = Server.MapPath("SampleRpt01.rpt")
End If

For Each cnInfo As TableLogOnInfo In Me.CrystalReportViewer1.LogOnInfo
    cnInfo.ConnectionInfo = ConnInfo
Next
Me.CrystalReportViewer1.RefreshReport()
End Sub

```

The explanation for the above code is provided in the next section.

### **Working with Parameters with Crystal Reports and ASP.NET 2.0 - Passing the value to Crystal Report Parameter dynamically at run-time: explanation**

(Page 3 of 6 )

First of all, you must observe that the CrystalDecisions.Shared namespace is added to work with Crystal Reports-related classes and objects. In the previous section's code, the database connection information is stored in ConnInfo and is defined as follows:

```

Dim ConnInfo As New ConnectionInfo
With ConnInfo
    .ServerName = ".sqlexpress"
    .DatabaseName = "Northwind"
    .UserID = "sa"
    .Password = "eXpress2005"
End With

```

The above connection information is assigned to the CrystalReportViewer control using the following code:

```

For Each cnInfo As TableLogOnInfo In Me.CrystalReportViewer1.LogOnInfo
    cnInfo.ConnectionInfo = ConnInfo
Next

```

Finally, the ReportSource and Parameter are added as following:

```

Me.CrystalReportViewer1.ParameterFieldInfo.Clear()
Me.CrystalReportViewer1.ReportSource = Server.MapPath("SampleParam1.rpt")
Dim ParamFields As ParameterFields =
Me.CrystalReportViewer1.ParameterFieldInfo
Dim p_EmpID As New ParameterField
p_EmpID.Name = "p_EmployeeID"
Dim p_EmpID_Value As New ParameterDiscreteValue
p_EmpID_Value.Value = Me.txtEmployeeID.Text
p_EmpID.CurrentValues.Add(p_EmpID_Value)
ParamFields.Add(p_EmpID)

```

We can add as many numbers of parameters as we want to the "ParameterFields" collection to provide values at run-time. The following is the code which deals with paging:

```

Protected Sub btnFirst_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnFirst.Click
    Me.CrystalReportViewer1.ShowFirstPage()
End Sub

Protected Sub btnPrevious_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnPrevious.Click
    Me.CrystalReportViewer1.ShowPreviousPage()
End Sub

Protected Sub btnNext_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnNext.Click
    Me.CrystalReportViewer1.ShowNextPage()
End Sub

Protected Sub btnLast_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnLast.Click
    Me.CrystalReportViewer1.ShowLastPage()
End Sub

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
    If Not IsPostBack Then
        btnShow_Click(Nothing, Nothing)
    End If

```

End Sub

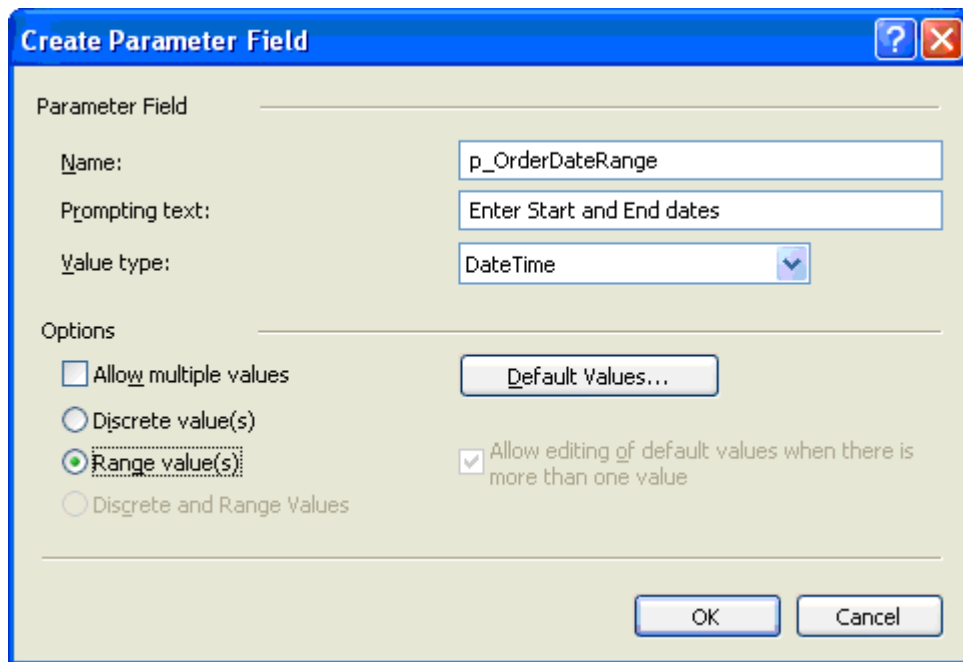
### **Adding a Range Value parameter to Crystal Report**

Add a new report named "SampleParam2.rpt" with the same fields, in a way that is similar to the previous report. Now, we shall add a range value parameter to the Crystal Report.

The following are the steps to take to add a range value parameter:

In the "FieldExplorer," right click on "Parameter Fields" and click "Add."

In the "Create Parameter Field" dialog box, provide "Name" as "p\_OrderDateRange," "Prompting Text" as "Enter Start and End Dates," "Value Type" as "DateTime," select "Range Values" in the "Options" and click OK (Fig 07).



**Create Parameter Field**

Parameter Field

Name: p\_OrderDateRange

Prompting text: Enter Start and End dates

Value type: DateTime

Options

☐ Allow multiple values

☐ Discrete value(s)

☒ Range value(s)

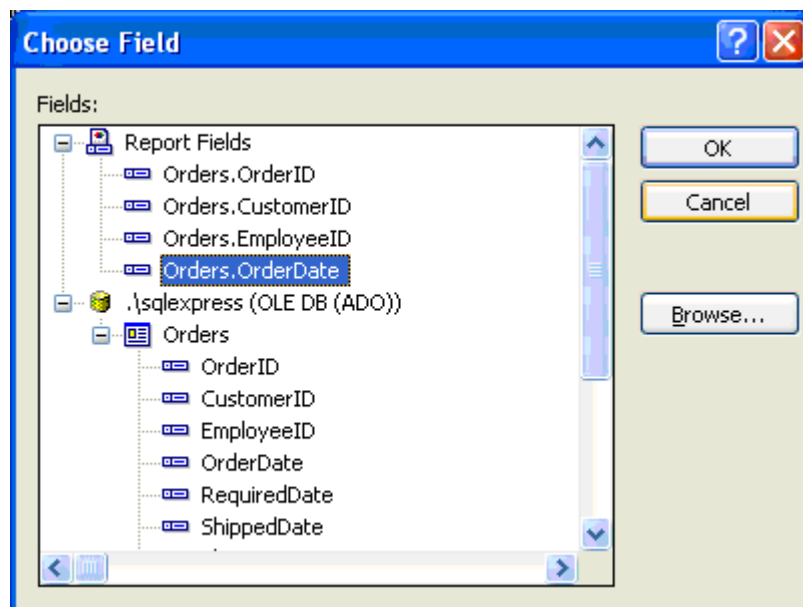
☐ Discrete and Range Values

Default Values...

☒ Allow editing of default values when there is more than one value

OK Cancel

In the "Choose Field" dialog box, select "Orders.OrderDate" and click on OK (as shown in figure 08).



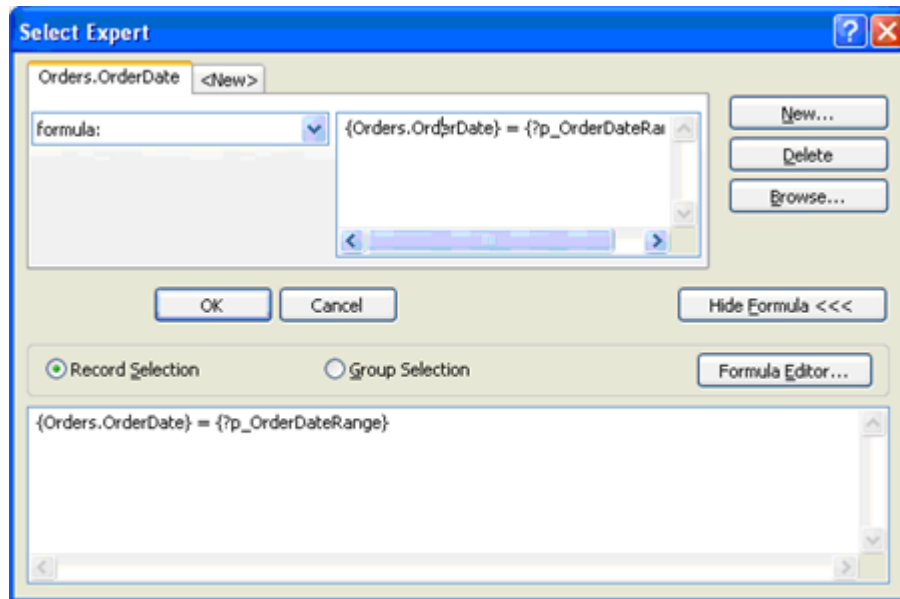
**Choose Field**

Fields:

- Report Fields
  - Orders.OrderID
  - Orders.CustomerID
  - Orders.EmployeeID
  - Orders.OrderDate
- .\sqlservr (OLE DB (ADO))
  - Orders
    - OrderID
    - CustomerID
    - EmployeeID
    - OrderDate
    - RequiredDate
    - ShippedDate

OK Cancel Browse...

Using the "Select Expert" provide information as follows (Fig 09) and click OK.

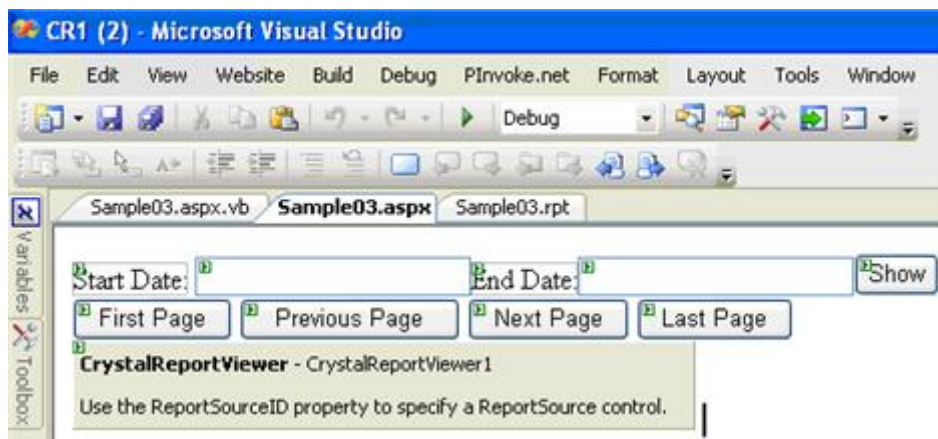


### Working with Parameters with Crystal Reports and ASP.NET 2.0 - Passing the value to the Crystal Report Range Parameter dynamically: source code

(Page 4 of 6 )

In the previous section, we added a parameter named "p\_OrderDateRange" to the report. Now, it is time to access the report (say, get a list of orders) based on the user-specified range of dates.

Let us modify the web page so that it looks like the following (Fig 10):



The source for the above page design is as follows:

```
<form id="form1" runat="server">
  <div>
    <asp:Label ID="lblStartDate" runat="server" Text="Start Date:"></asp:Label>
    <asp:TextBox ID="txtStartDate" runat="server"></asp:TextBox><asp:Label
```

```

        ID="lblEndDate" runat="server" Text="End Date:"></asp:Label><asp:TextBox
ID="txtEndDate"
        runat="server"></asp:TextBox><asp:Button ID="btnShow" runat="server"
Text="Show" /><br />
        <asp:Button ID="btnFirst" runat="server" Text="First Page" />
        <asp:Button ID="btnPrevious" runat="server" Text="Previous Page" />
        <asp:Button ID="btnNext" runat="server" Text="Next Page" />
        <asp:Button ID="btnLast" runat="server" Text="Last Page" /><br />
        <CR:CrystalReportViewer ID="CrystalReportViewer1" runat="server"
AutoDataBind="true" DisplayGroupTree="False" DisplayToolbar="False"
EnableDatabaseLogonPrompt="False" EnableParameterPrompt="False" Height="1055px"
ReuseParameterValuesOnRefresh="True" Width="789px" />
    </div>
</form>

```

Following is the code for the "Show" button:

```

Protected Sub btnShow_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnShow.Click
    Dim ConnInfo As New ConnectionInfo
    With ConnInfo
        .ServerName = ".sqlexpress"
        .DatabaseName = "Northwind"
        .UserID = "sa"
        .Password = "eXpress2005"
    End With

    Me.CrystalReportViewer1.ParameterFieldInfo.Clear()
    If Me.txtStartDate.Text.Trim.Length > 0 And
Me.txtStartDate.Text.Trim.Length > 0 Then
        Me.CrystalReportViewer1.ReportSource = Server.MapPath("SampleParam2.rpt")
        Dim ParamFields As ParameterFields =
Me.CrystalReportViewer1.ParameterFieldInfo
        Dim p_OrderDateRange As New ParameterField
        p_OrderDateRange.Name = "p_OrderDateRange"
        Dim p_OrderDateRange_Value As New ParameterRangeValue
        p_OrderDateRange_Value.StartValue = Me.txtStartDate.Text
        p_OrderDateRange_Value.EndValue = Me.txtEndDate.Text
        p_OrderDateRange.CurrentValues.Add(p_OrderDateRange_Value)
        ParamFields.Add(p_OrderDateRange)
    Else
        Me.CrystalReportViewer1.ReportSource = Server.MapPath("SampleRpt01.rpt")
    End If

    For Each cnInfo As TableLogOnInfo In Me.CrystalReportViewer1.LogOnInfo
        cnInfo.ConnectionInfo = ConnInfo
    Next
    Me.CrystalReportViewer1.RefreshReport()
End Sub

```

### Working with Parameters with Crystal Reports and ASP.NET 2.0 - Adding Multiple Value (Discrete and Range) parameters to Crystal Report

(Page 5 of 6 )

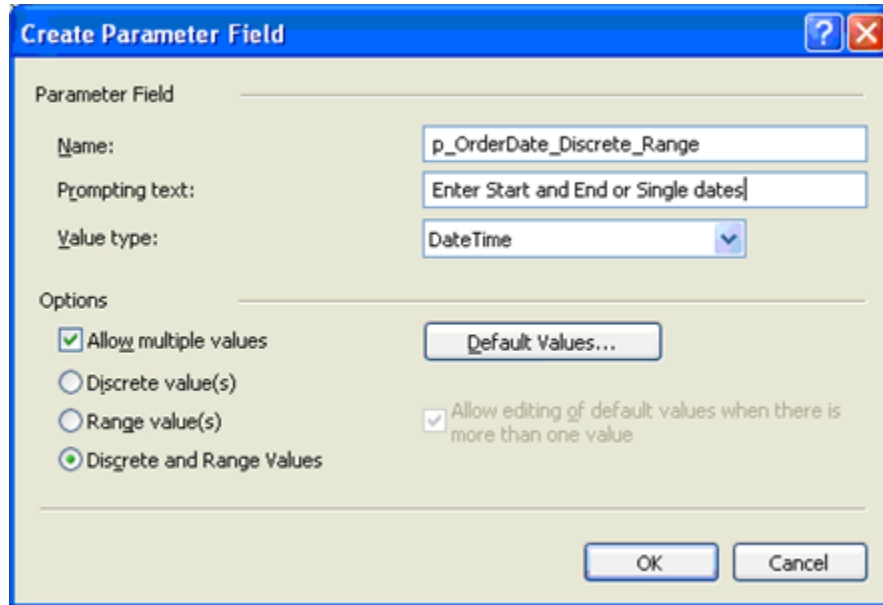
This is very similar to the previous report except that this accepts multiple values (both discrete and range parameters).

In a way similar to the previous reports, add a new report named "SampleParam3.rpt" with the same fields. Now we shall add a new parameter to the Crystal Report.

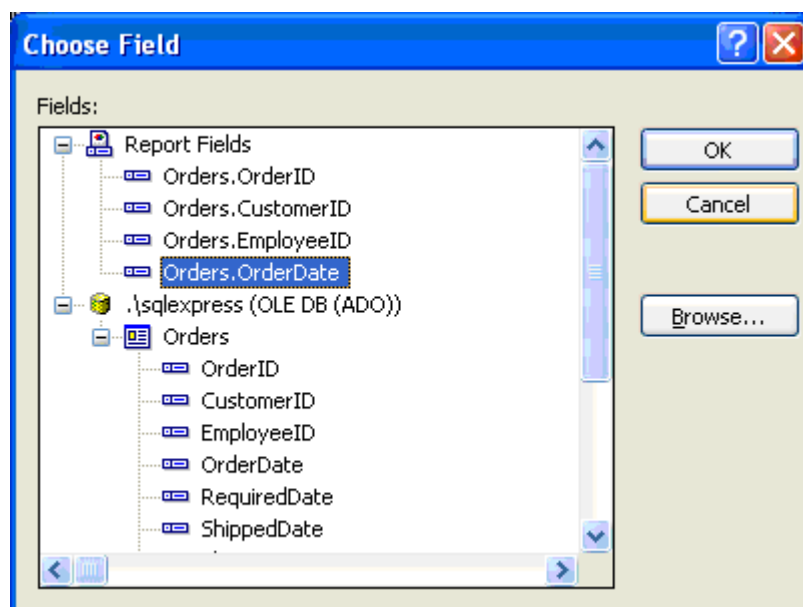
The following are the steps required to add multiple value (Discrete and Range) parameters:

In the "FieldExplorer," right click on "Parameter Fields" and click "Add."

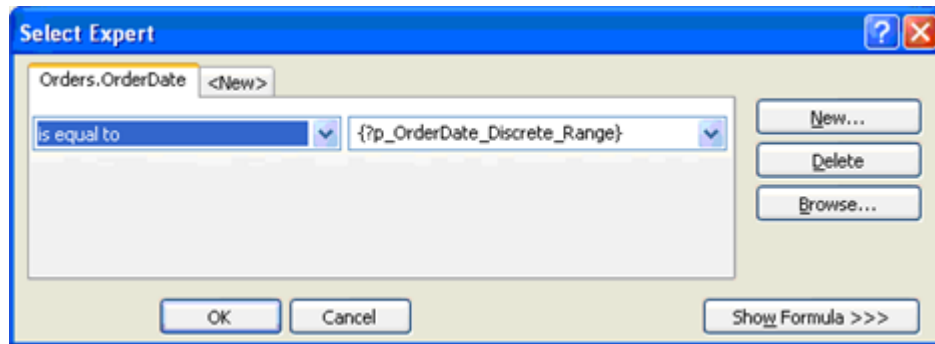
In the "Create Parameter Field" dialog box, provide "Name" as "p\_OrderDate\_Discrete\_Range," "Prompting Text" as "Enter Start and End Dates or single date," "Value Type" as "DateTime," switch "Allow multiple values" to on in the "Options," select "Discrete and Range Values" from the radio button list and click OK (Fig 11).



Using the "Select Expert," provide the information for the "Choose Field" dialog box as being from "Orders.OrderDate" and click on OK (as shown in figure 12).



Using the "Select Expert" provide information as follows (Fig 13) and click OK.

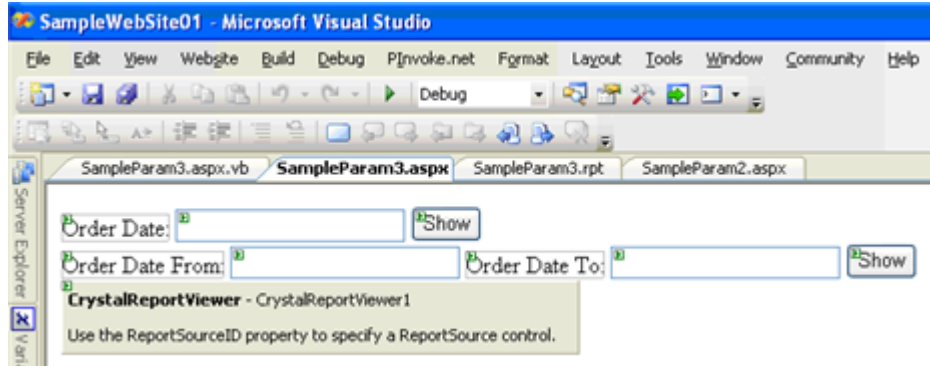


### Working with Parameters with Crystal Reports and ASP.NET 2.0 - Passing the value to Multiple Crystal Report Parameters dynamically: source code

(Page 6 of 6 )

In the previous section, we added a parameter named "p\_OrderDate\_Discrete\_Range" to the report. Now it is time to access the report (say, get a list of orders) based on the user-specified range of dates.

Let us modify the web page so that it looks like the following (Fig 14):



The source for the above page design is as follows:

```
<form id="form1" runat="server">
  <div>
    <asp:Label ID="lbl" runat="server" Text="Order Date:"></asp:Label>
    <asp:TextBox ID="txtParticularOrderDate" runat="server"></asp:TextBox>
    <asp:Button ID="btnShow" runat="server" Text="Show" /><br />
    <asp:Label ID="Label1" runat="server" Text="Order Date
From:"></asp:Label>
    <asp:TextBox ID="txtOrderDateFrom" runat="server"></asp:TextBox>
    <asp:Label ID="Label2" runat="server" Text="Order Date To:"></asp:Label>
    <asp:TextBox ID="txtOrderDateTo" runat="server"></asp:TextBox>
    <asp:Button ID="btnRangeShow" runat="server" Text="Show" />
    <CR:CrystalReportViewer ID="CrystalReportViewer1" runat="server"
AutoDataBind="true" DisplayGroupTree="False" DisplayToolbar="False"
```

```

EnableDatabaseLogonPrompt="False" EnableParameterPrompt="False"
Height="1055px" ReuseParameterValuesOnRefresh="True" Width="789px" />

</div>
</form>

```

The following is the code for the "Show" button:

```

Protected Sub btnShow_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnShow.Click
.
.
.
Dim ParamCurrentValues As New ParameterValues
Dim p_OrderDate_Value As New ParameterDiscreteValue
p_OrderDate_Value.Value = Me.txtParticularOrderDate.Text
ParamCurrentValues.Add(p_OrderDate_Value)
Dim ParamFields As ParameterFieldDefinitions =
rep.DataDefinition.ParameterFields
Dim p_OrderDate As ParameterFieldDefinition = ParamFields
("p_OrderDate_Discrete_Range")
p_OrderDate.ApplyCurrentValues(ParamCurrentValues)
End Sub

```

Following is the code for the "Range Show" button:

```

Protected Sub btnRangeShow_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnRangeShow.Click
.
.
.
Dim ParamCurrentValues As New ParameterValues
Dim p_OrderDate_Value As New ParameterRangeValue
p_OrderDate_Value.StartValue = Me.txtOrderDateFrom.Text
p_OrderDate_Value.EndValue = Me.txtOrderDateTo.Text
ParamCurrentValues.Add(p_OrderDate_Value)
rep.DataDefinition.ParameterFields
("p_OrderDate_Discrete_Range").ApplyCurrentValues(ParamCurrentValues)

Me.CrystalReportViewer1.ReportSource = rep
Me.CrystalReportViewer1.DataBind()
End Sub

```

I hope you enjoyed the article and any suggestions, bugs, errors, enhancements etc. are highly appreciated at <http://jagchat.spaces.live.com>