

# Reporting with Reports Viewer in Visual Studio 2005

Platform Support: .NET 2.0

**Authors:**

**C# Corner Authors Team**

Published on: March 30, 2007

@2007 Mindcracker LLC. All rights reserved. United States and International copyright laws protect this property. Mindcracker LLC or the author of this material has no affiliation with Microsoft or any other companies that develops the software. Some keywords used in this material are registered trademarks of their respective owners.

Reproduction or printing multiple copies of this material is prohibited. If you need multiple copies of this material, please contract [authors@c-sharpcorner.com](mailto:authors@c-sharpcorner.com).

## Introduction

This document discusses the ReportViewer control and how to generate reports using the ReportViewer control. This document covers the following topics:

- Understanding the ReportViewer control and report processing modes
- Creating a simple report
- Generating reports from an object collection
- Generating reports from a DataSet
- Generating reports from an XML document

## Getting Started with ReportViewer Control

If you have written any reporting applications in .NET, you may be familiar with Crystal Reports and/or SQL Server Reporting Services.

The ReportViewer control is a new addition to Visual Studio 2005, which is actually a SQL Server Reporting Services component. This control has two versions – Windows Forms version and Web version. As you may presume, Windows Forms version is used to write reporting applications in Windows Forms applications and Web version is used to write ASP.NET Web applications.

### Report Processing Modes

The ReportViewer control supports two report processing modes – *local* and *remote*.

In local processing mode, the ReportViewer control runs within the client application and the report processing is performed as a local process on the client machine where actual application is running.

In remote processing mode, the ReportViewer control runs on a SQL Server 2005 Reporting Services report server. In this mode, the ReportViewer control is used to view the report that is already generated on the server. All processing from data retrieval to report rendering is performed on the report server. To use remote processing mode, you must have a licensed copy of SQL Server 2005 Reporting Services.

Both Windows and Web controls can be configured to run in local processing mode or remote processing mode.

### Namespace and Assembly

The ReportViewer control assemblies for Windows Forms and Web Forms versions are *Microsoft.ReportViewer.WinForms* and *Microsoft.ReportViewer.WebForms* respectively.

### Create a Simple Report

Now let's create a simple report using the ReportViewer control. In this application, I will load data from a SQL Server database to generate reports.

1. Create a Web Application using Visual Studio 2005
2. Drag and drop a ReportViewer control from Toolbox to the Web page
3. Right click on the Project in Solution Explorer and select Add New Item menu item and select DataSet from the list and keep DataSet1.xsd as name.
4. The next screen is TableAdapter wizard, which asks you question about your database connection and SQL queries where you want data to be selected from. Just follow the wizard steps one by one and you will end up seeing DataSet in designer mode with your table name and columns. Close the designer.
5. Now right click on the Project again and select Add New Item and select Report from the available items and click Add button. See Figure 1.

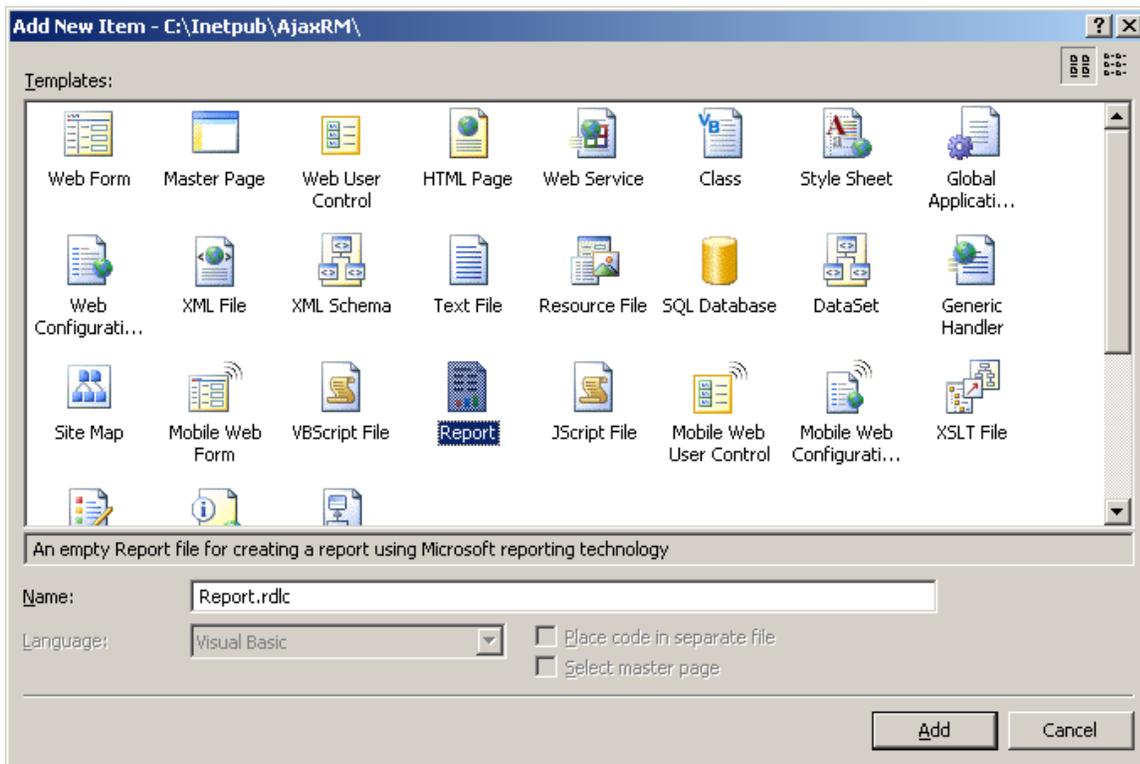


Figure 1.

- It will add Report.rdlc file to your project and will open the report designer, which looks like Figure 2. As you can see in the left side, you will see your DataSet.

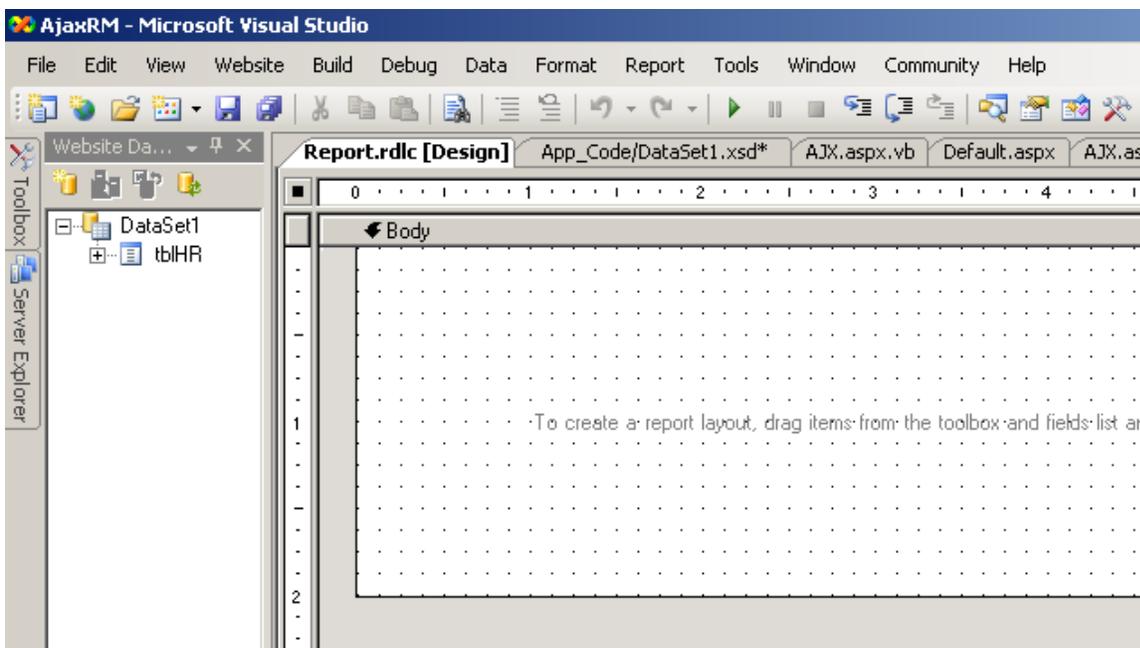


Figure 2.

- Now you can expand your DataSet and drag whatever column you want in the report to the report designer. I select three columns in Figure 3.

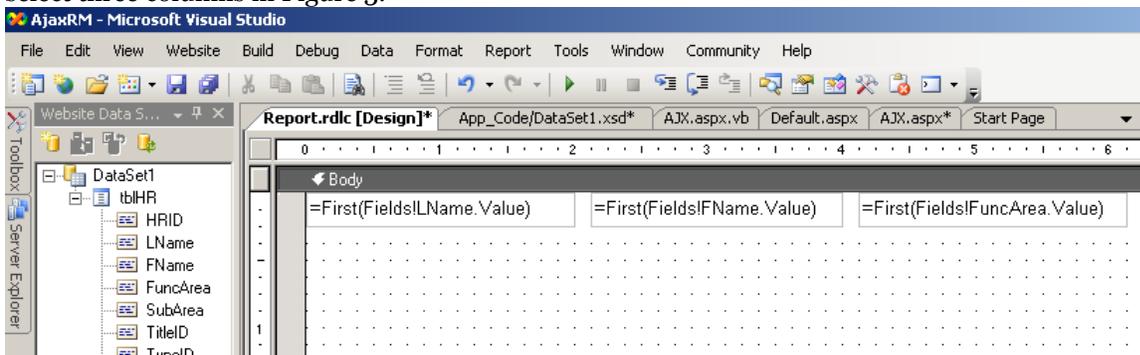


Figure 3.

- Now if you right click and select Properties on a TextBox, you will see Figure 4, where you can apply settings like visibility, navigation, format, font, and sorting. For now, you can say OK on this dialog. I will discuss these properties in more details in my following articles.

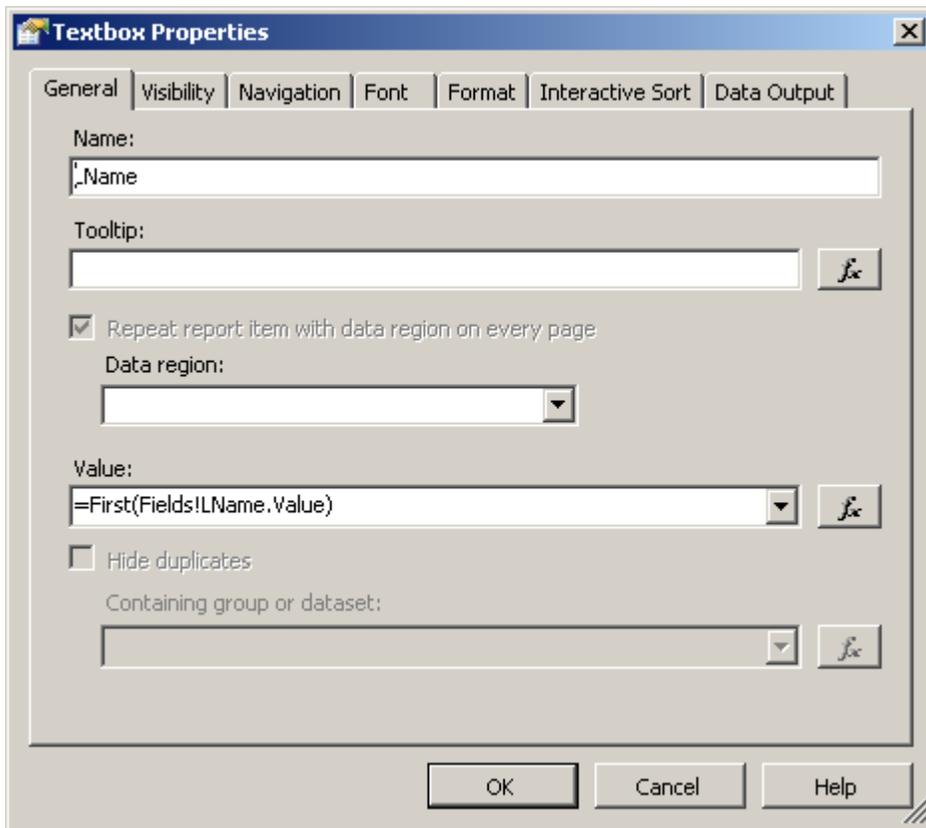


Figure 4.

9. Now go to the Web page and click the smart tag on ReportViewer and select Report.rdlc from the list. See Figure 5.

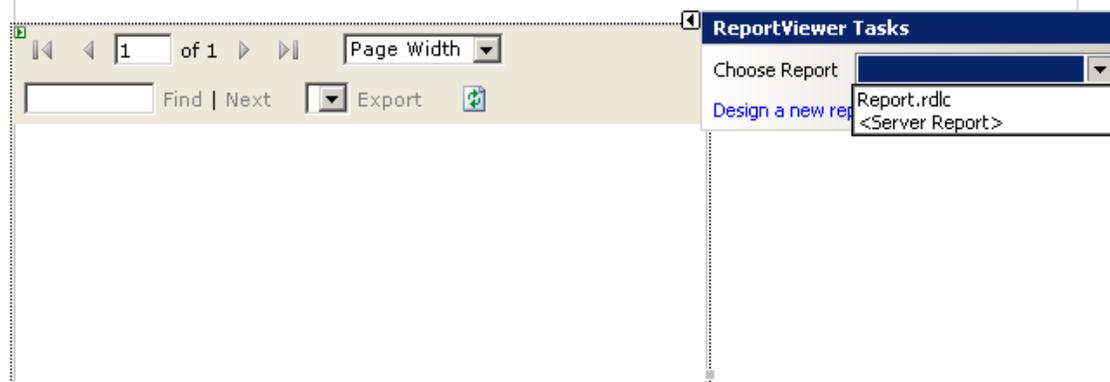


Figure 5.

10. That's it. Now if you run the application, you will see report. You can navigate through records, and export it to Excel or PDF using the Export option.

## Building Reports from an Object Collection

An object collection is In this article, I will discuss how to create reports from a business objects collection. I will be creating a Windows Forms application with reporting being processed as local processing mode. See my article, [Getting started with ReportViewer control](#) to understand local and remote processing modes.

### Step 1. Create an Objects Collection

I have a class that stores an employee's information. The class looks like Listing 1.

```
class Employee
{
    private string name;

    public string Name
    {
        get { return name; }
        set { name = value; }
    }
    private string address;

    public string Address
    {
        get { return address; }
        set { address = value; }
    }
    private string ssn;

    public string Ssn
    {
        get { return ssn; }
        set { ssn = value; }
    }
    private Int16 age;

    public Int16 Age
    {
        get { return age; }
        set { age = value; }
    }

    public Employee(string EmpName, string EmpAddress, string EmpSsn, Int16 EmpAge)
    {
        this.name = EmpName;
        this.address = EmpAddress;
        this.ssn = EmpSsn;
        this.age = EmpAge;
    }
}
```

Listing 1.

Now I build a Company class, which is a generic collection of Employee with three records in it. The Company class is listed in Listing 2.

```
class Company
{
    private List<Employee> m_employees;

    public Company()
    {
        m_employees = new List<Employee>();
        m_employees.Add(new Employee("Mahesh Chand", "112 New Road, Chadds Ford, PA", "123-21-1212", 30));
        m_employees.Add(new Employee("Jack Mohita", "Pear Lane, New York 23231", "878-12-2334", 23));
        m_employees.Add(new Employee("Renee Singer", "Near medow, Philadelphia, PA", "980-00-2320", 20));
    }
    public List<Employee> GetEmployees()
    {
        return m_employees;
    }
}
```

Listing 2.

In Listing 2, the GetEmployees method returns all the employees in the company.

### Step 2. Add a Report to the Project

Now add a report to the project by Right click on the project > Add > New Item and select Report from the items listing. It will add Report1.rdlc file to the project.

### Step 3. Add a Data Source

Now I am going to add a data source to the project. Double click on your Form so Form Designer is open. Now from the Data menu of Visual Studio, select Add New Data Source item. See Figure 1.

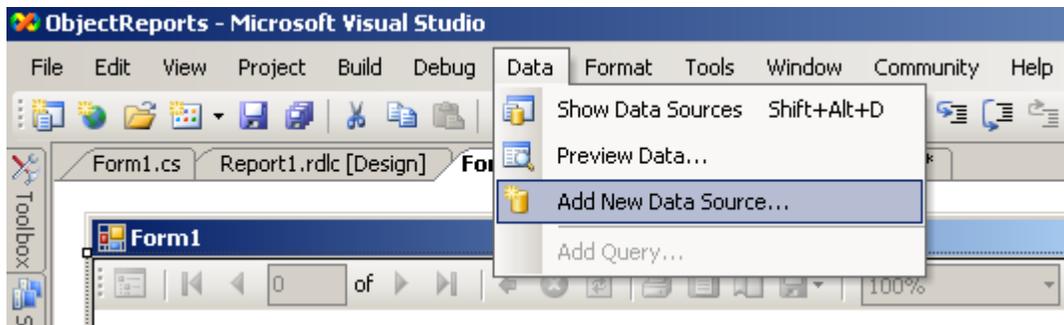


Figure 1.

It will launch the Data Source Configuration Wizard. Select Object item on the first page. See Figure 2.

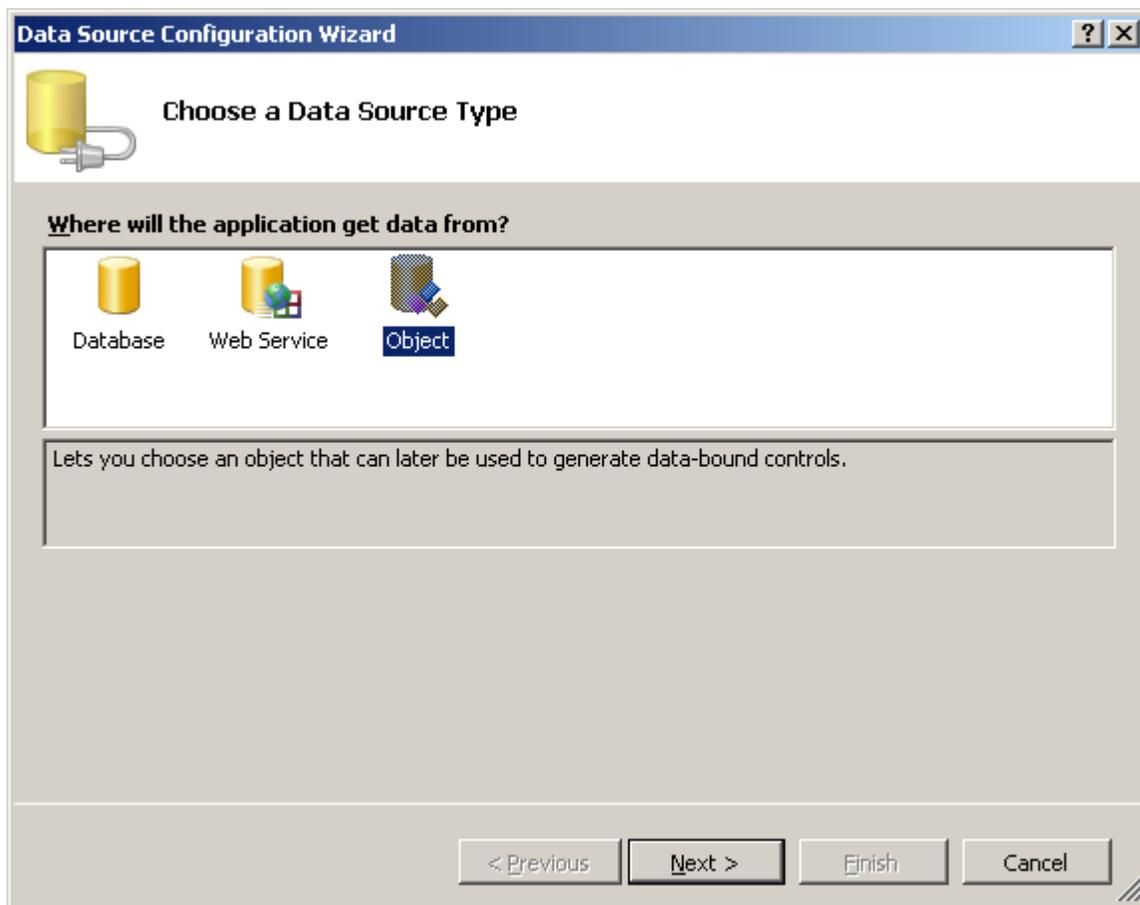


Figure 2.

Now on next screen you should see your application namespace. If you don't see the namespace, make sure to Rebuild the application.

If you expand the namespace, you should see Employee class. See Figure 3. Select Employee and click Next and click Finish on the next screen.

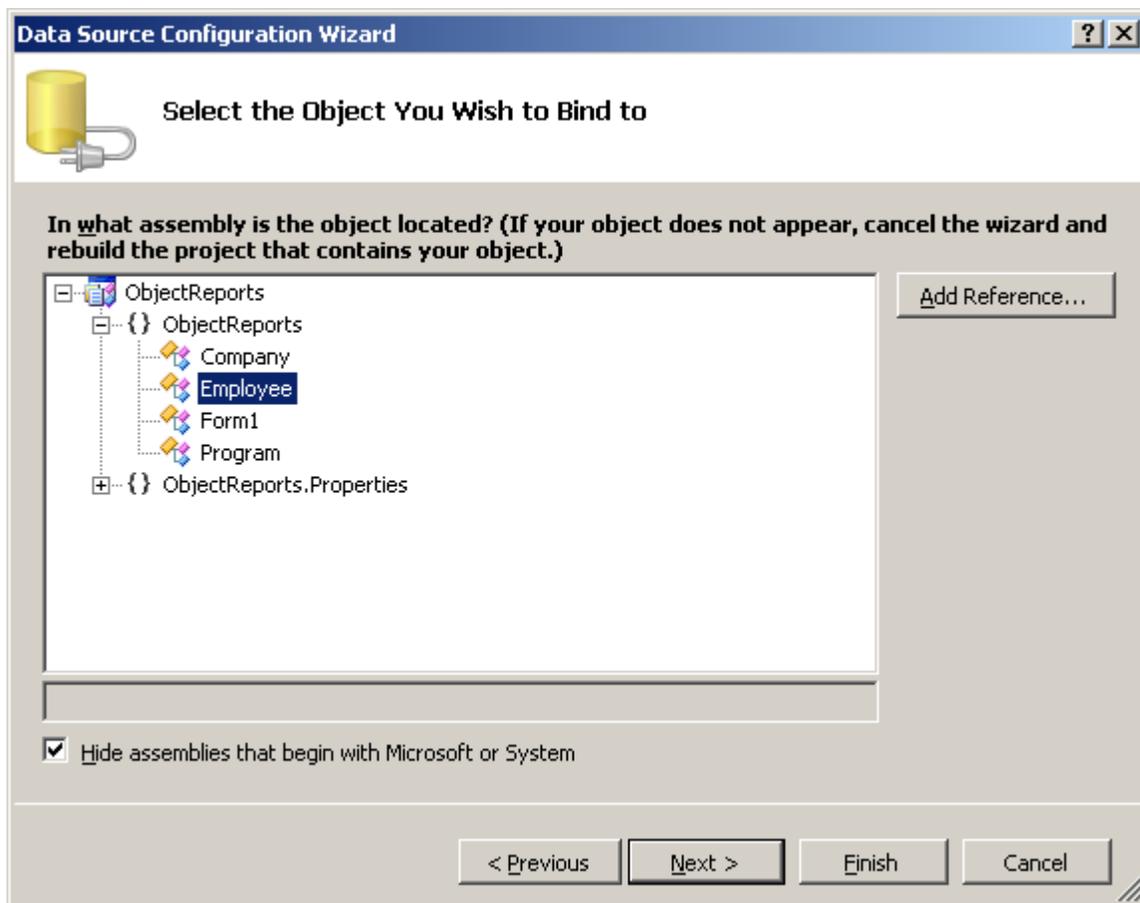


Figure 3.

Now you should see Employees in Data Sources window. See Figure 4.

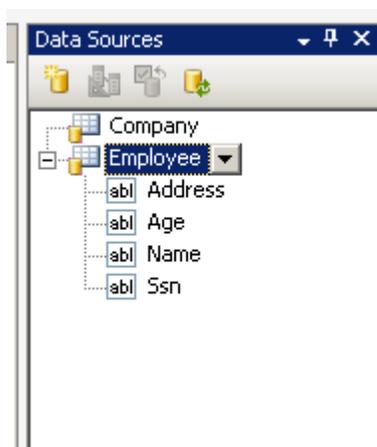


Figure 4.

Note: When you add a DataSource to the project, the designer adds the EmployeeBindingSource to the designer, which will play a major role later.

### Step 4. Bind Data Source with Report

Now open Report1.rdlc file and drag and drop a Table from the Toolbox to the report designer. Add Name, Address, Age, and Ssn items from the DataSource to the second row of the table. See Figure 5.

Note: First row in the table is Header and third row in the table is Footer. The second row is the Details row. Make sure you do not drop the items on the First and Third rows of the table, otherwise you will see only one record in your report.

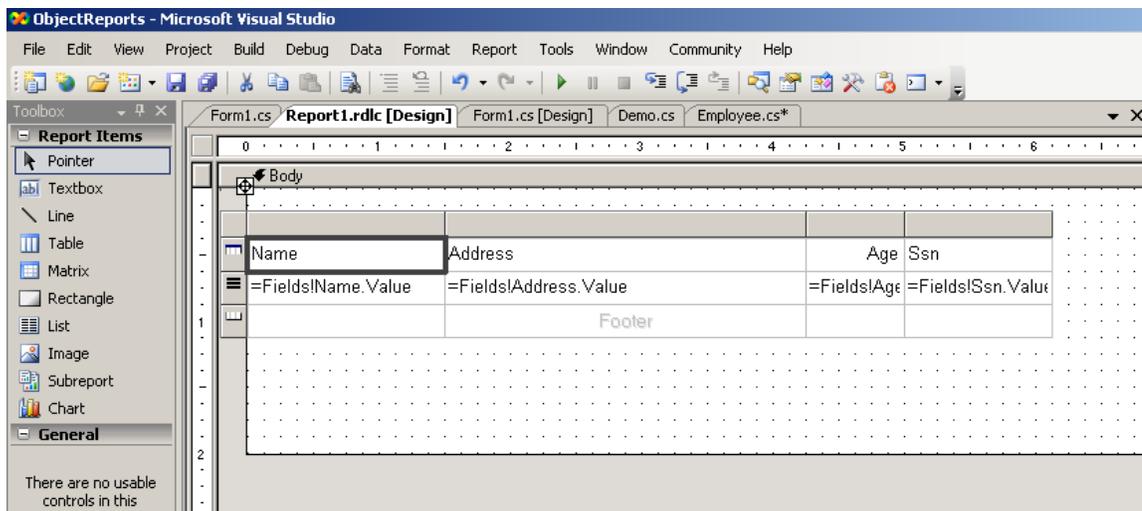


Figure 5.

Now you can format the table the way you want by right clicking on the table, column, or a cell and select Properties item.

### Step 5. Binding EmployeeBindingSource and Data

Now last step is to bind EmployeeBindingSource with the Employee data. If you remember, we added a GetEmployees method to the Company class that returns all the Employees in that company.

Now we simply call GetEmployees, which returns a collection and set DataSource property of EmployeeBindingSource as seen in Listing 3.

```
private void Form1_Load(object sender, EventArgs e)
{
    Company company = new Company();
    EmployeeBindingSource.DataSource = company.GetEmployees();
    this.reportViewer1.RefreshReport();
}
```

Listing 3.

### Step 6. Build and Run

Now build and run the application. The output looks like Figure 6.

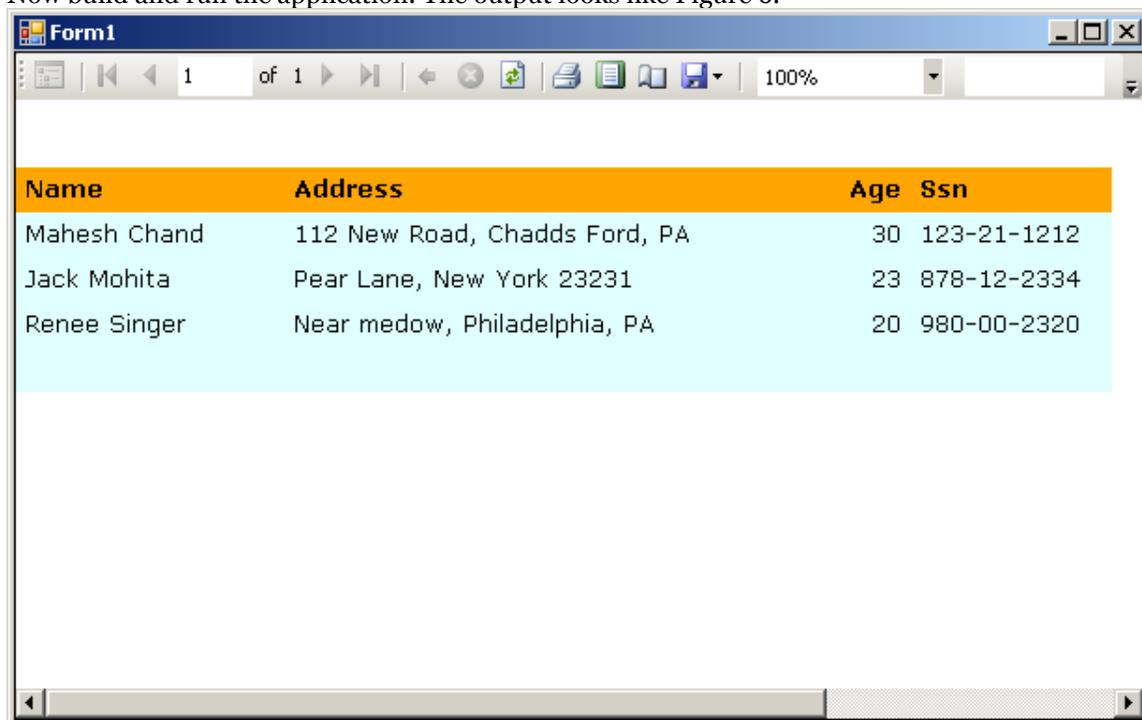


Figure 6.

## Building Reports from XML Documents

In this step-by-step tutorial, you will learn how to create reports from an XML document using the ReportViewer control and Visual Studio 2005.

### The Data

I have an XML file Data.xml, which looks like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<Company>
  <Employees>
    <Employee Name="Mahesh Chand" Age="30" Phone="6101233333" />
    <Employee Name="Rose Garner" Age="56" Phone="2133428778" />
    <Employee Name="Amger Jap" Age="22" Phone="9092349800" />
    <Employee Name="Mike Gold" Age="35" Phone="9908088823" />
    <Employee Name="Renee Flower" Age="19" Phone="4848901003" />
  </Employees>
</Company>
```

### Step 1. Generate and Add the Schema File

My first goal is to generate schema file, which will represent the data. I take help of the DataSet class and its method WriteXmlSchema. The code listed in Listing 1 reads the Data.xml file and generates a schema file called Data.xsd. This file is created in the *Debug* folder of your application.

```
DataSet ds = new DataSet();
ds.ReadXml("Data.xml");
ds.WriteXmlSchema("Data.xsd");
```

Listing 1.

Let's add Data.xsd file to your project. Right click on the project in Solution Explorer, select Add >> Existing Item and browse for Data.xsd file and add it to the project.

Now let's Rebuild the project.

### Step 2. Create the Report

Now we will add a new report file to the project. Right click on the project in Solution Explorer and select Add >> New Item and select Report from the Items list. It will add Report1.rdlc file to the project.

Once the report is added, our next step is to add a data source. First double click on the Form and select Data Menu item from the Main Menu. Click on Data Menu item and select Add New Data Source item.

It will launch Data Source Configuration Wizard. Select Object from the list and click the Next button on the Wizard. See Figure 2.

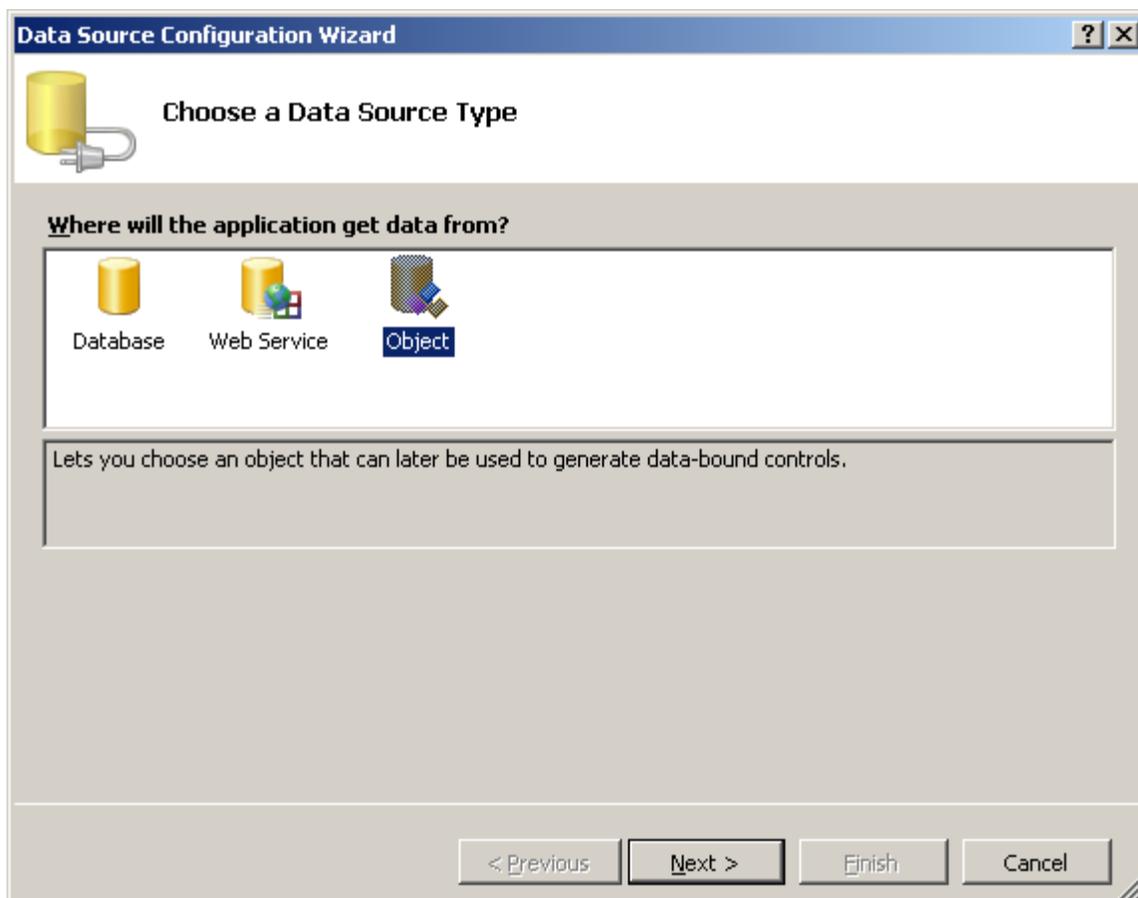


Figure 2.

On next dialog, you should see all namespaces and classes in your project. Expand your namespace and you will see class Company. See Figure 3.

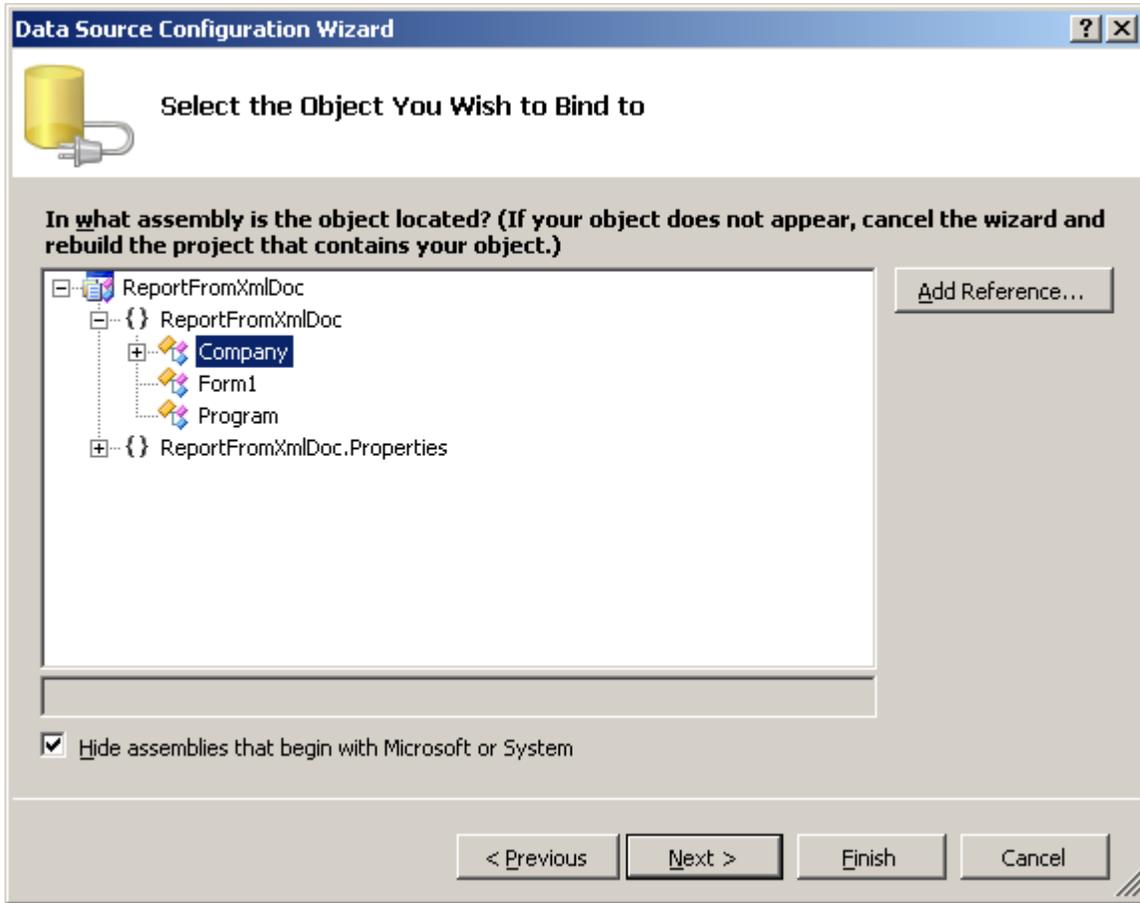


Figure 3.

Select Company class and click the Next button. On next dialog, you will see a confirmation message. Select Finish there and get out of the wizard.

Now double click on the Report1.rdlc file and you should see Figure 4 in your Data Sources window.

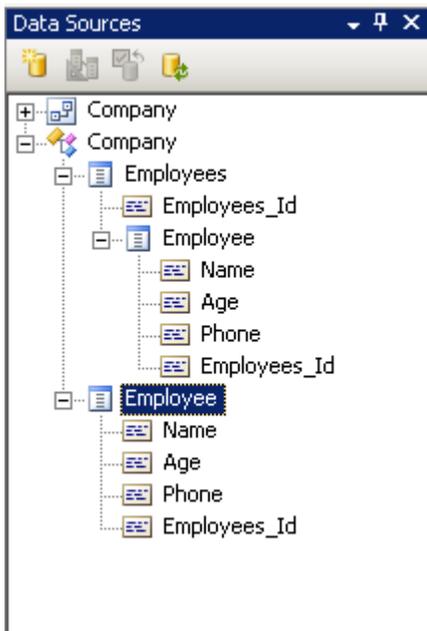


Figure 4.

Now let's create and format the report.

Drag a Table from the Toolbox and drag Name, Age, and Phone columns from the Data Sources to the report's middle row. As you can see from Figure 5, the name of the column is automatically added to the header (first) row of the report.

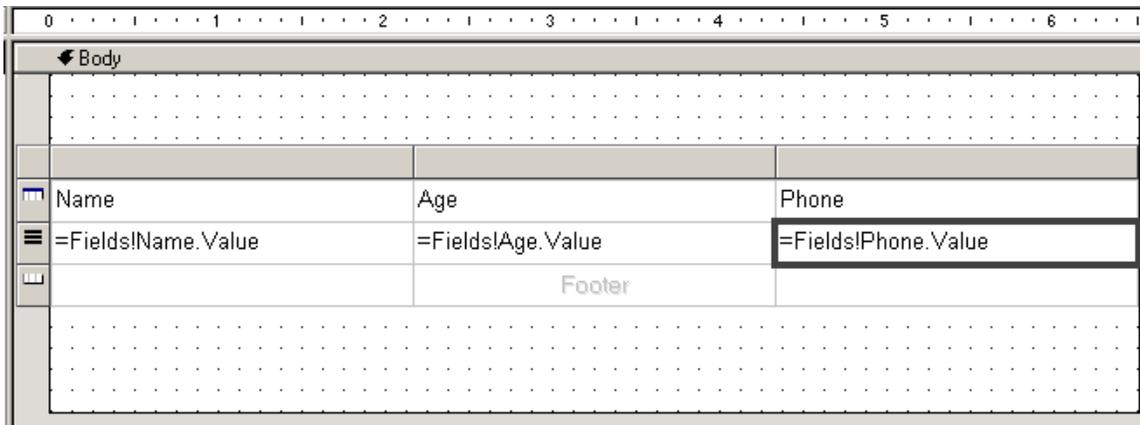


Figure 5.

### Step 3. Create a Report Viewer and Bind the Report

Now open the Form1 again and drag a ReportViewer control from the Toolbox to the Form. Click on the smart tag and select Report1.rdlc from the Choose Report drop down list. See Figure 6.

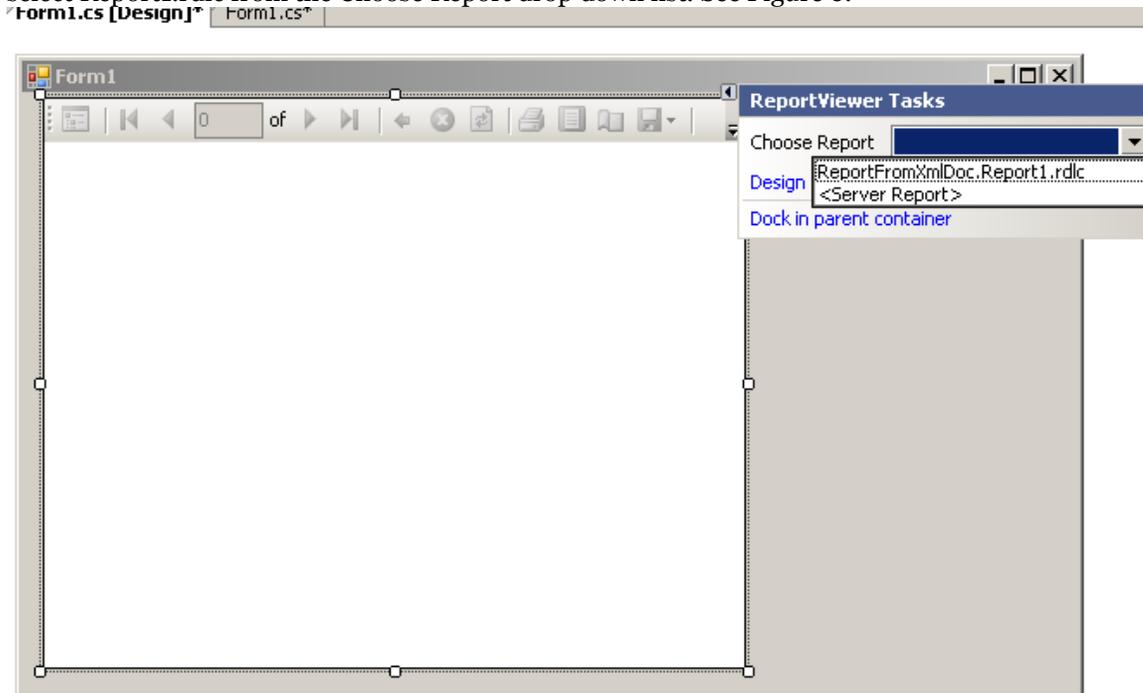


Figure 6.

By doing so, you will see an EmployeeBindingSource control is added at the bottom of the Form. See Figure 7. The BindingSource control provides connection between the data and the ReportViewer control.



Figure 7.

### Step 4. Fill the Data

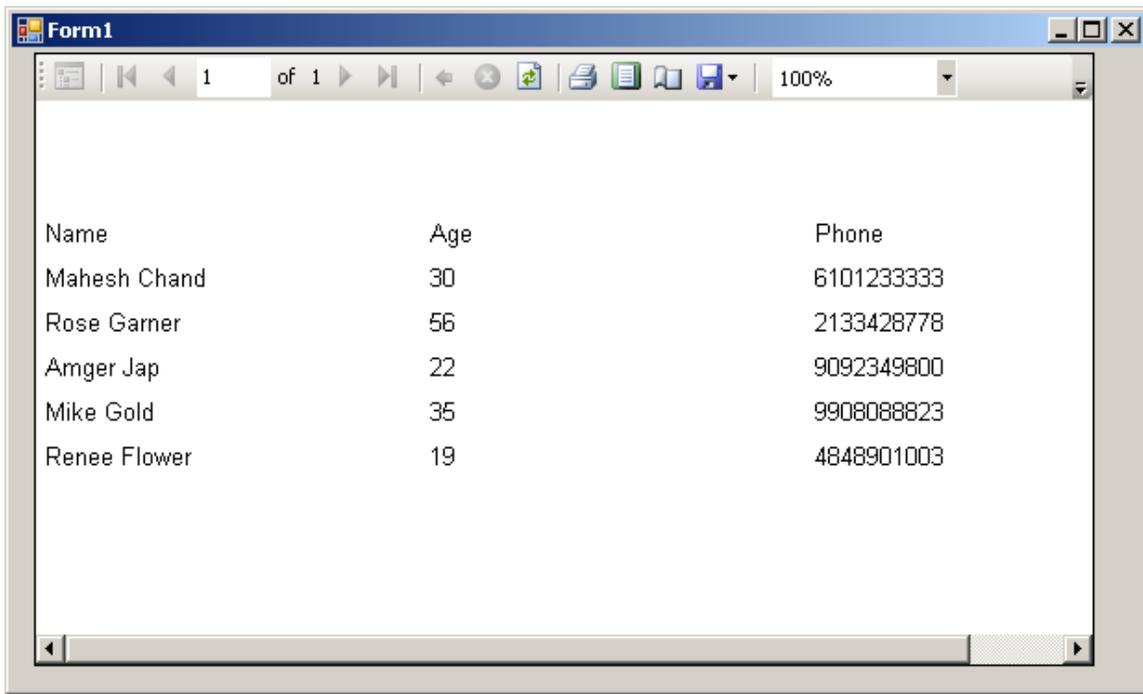
Now write the code listed in Listing 2 on the Form's load event handler. This code creates a DataSet, loads the data from Data.xml file and sets EmployeeBindingSource.DataSource as DataSet. The last line is added by you by the designer.

```
DataSet ds = new DataSet ();
ds.ReadXml ("Data.xml");
EmployeeBindingSource.DataSource = ds;
this.reportViewer1.RefreshReport ();
```

Listing 2.

### Step 5. Build and Run

That's all. Build and run the application. The output should look like Figure 8.



The screenshot shows a report viewer window titled "Form1". The window has a toolbar at the top with navigation and action icons, and a status bar showing "1 of 1" and "100%". The main content area displays a table with the following data:

Name	Age	Phone
Mahesh Chand	30	6101233333
Rose Garner	56	2133428778
Amger Jap	22	9092349800
Mike Gold	35	9908088823
Renee Flower	19	4848901003

Figure 8.

## Building Reports from a DataSet

In this step-by-step tutorial, you will learn how to create reports from a Database using a DataSet and the ReportViewer control and Visual Studio 2005.

### Step 1. Generate a Typed DataSet

First of all, we need a typed DataSet that will represent the data types. You can generate a typed DataSet from a database table, stored procedure, and a view or from a SQL query. After generating the typed DataSet, report generation is same.

Right click on the project in Solution Explorer, select Add >> New Item and select DataSet from the list.

Change DataSet1.xsd to Company.xsd. After adding the DataSet, you will see the designer. On the designer, click Server Explorer link, which will let you create a Database connection and can view database objects including tables, views, and stored procedures.

Drag a database table, or view, or stored procedure where you want to display the data from and it will create a DataSet schema for you. Once you have a typed DataSet, just follow these steps.

Before that, make sure you Rebuild the project.

### Step 2. Create the Report

Now we will add a new report file to the project. Right click on the project in Solution Explorer and select Add >> New Item and select Report from the Items list. It will add Report1.rdlc file to the project.

Once the report is added, our next step is to add a data source. First double click on the Form and select Data Menu item from the Main Menu. Click on Data Menu item and select Add New Data Source item.

It will launch Data Source Configuration Wizard. Select Object from the list and click the Next button on the Wizard. See Figure 2.

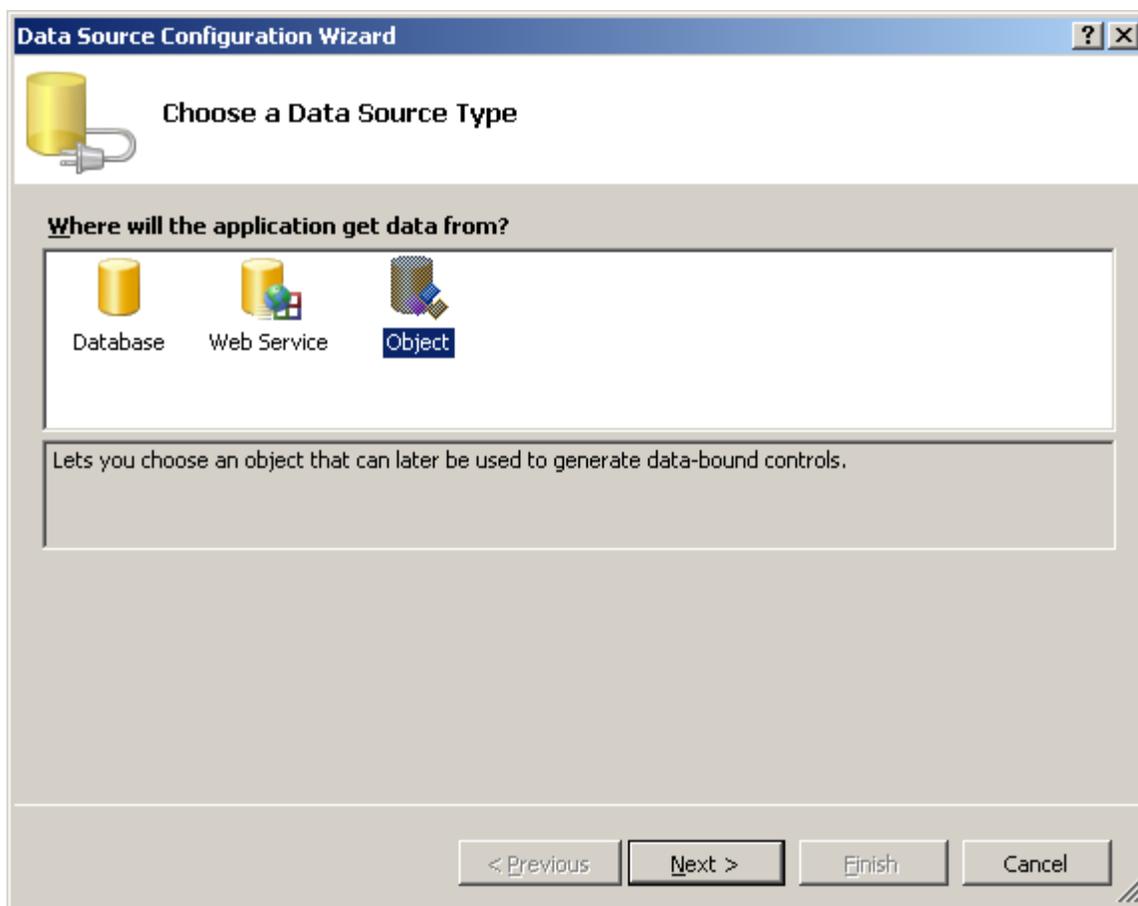


Figure 2.

On next dialog, you should see all namespaces and classes in your project. Expand your namespace and you will see class Company. See Figure 3.

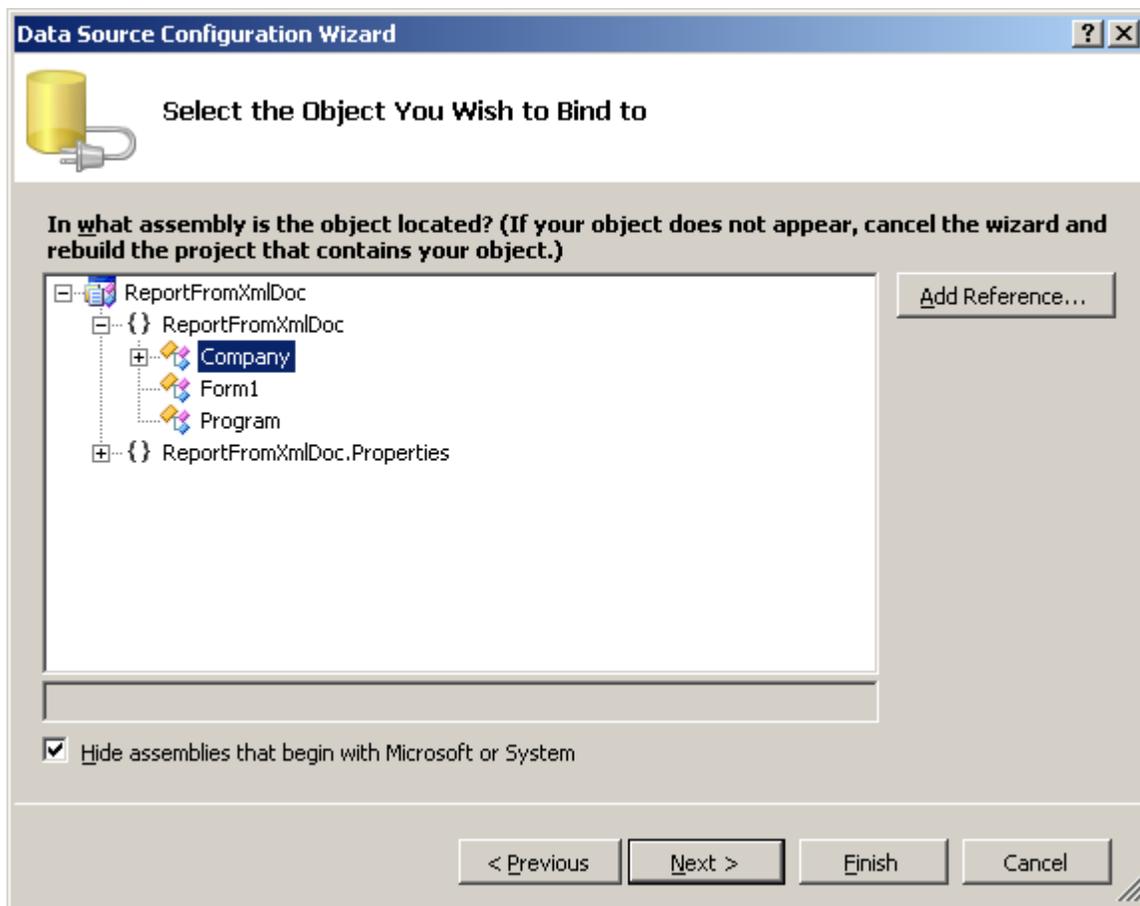


Figure 3.

Select Company class and click the Next button. On next dialog, you will see a confirmation message. Select Finish there and get out of the wizard.

Now double click on the Report1.rdlc file and you should see Figure 4 in your Data Sources window.

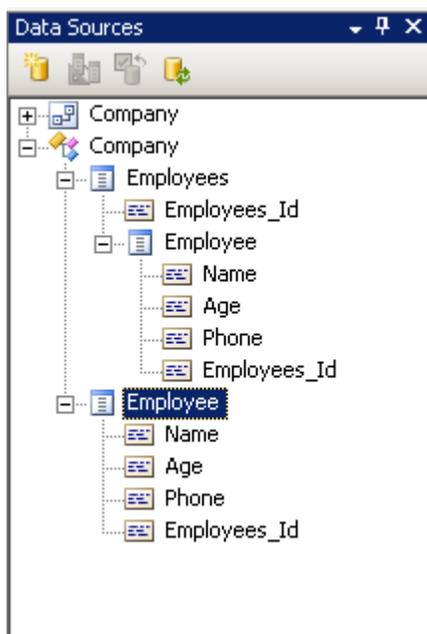


Figure 4.

Now let's create and format the report.

Drag a Table from the Toolbox and drag Name, Age, and Phone columns from the Data Sources to the report's middle row. As you can see from Figure 5, the name of the column is automatically added to the header (first) row of the report.

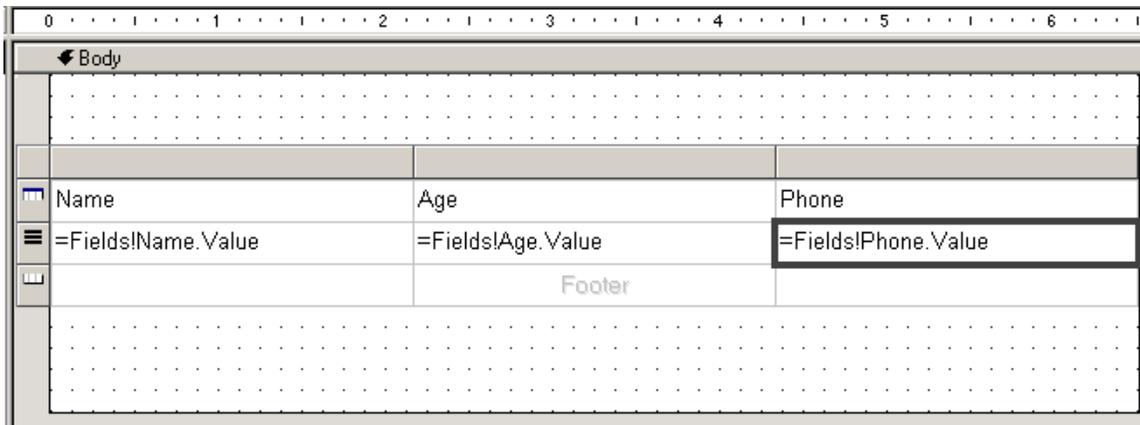


Figure 5.

### Step 3. Create a Report Viewer and Bind the Report

Now open the Form1 again and drag a ReportViewer control from the Toolbox to the Form. Click on the smart tag and select Report1.rdlc from the Choose Report drop down list. See Figure 6.

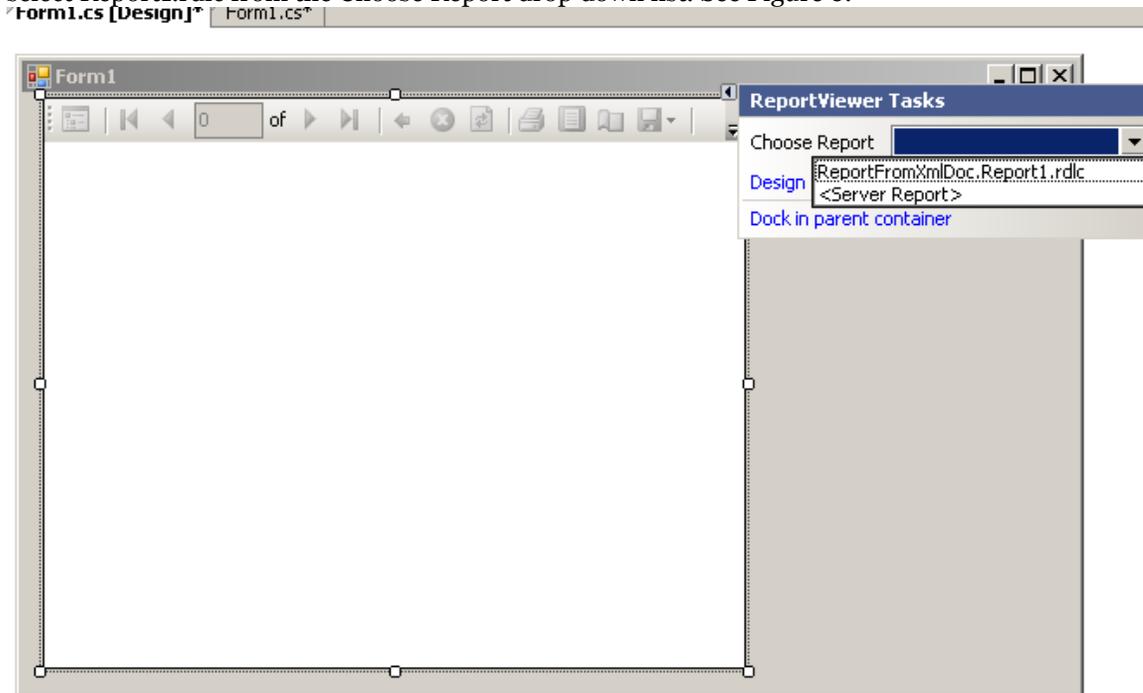


Figure 6.

By doing so, you will see an EmployeeBindingSource control is added at the bottom of the Form. See Figure 7. The BindingSource control provides connection between the data and the ReportViewer control.



Figure 7.

### Step 4. Fill the Data

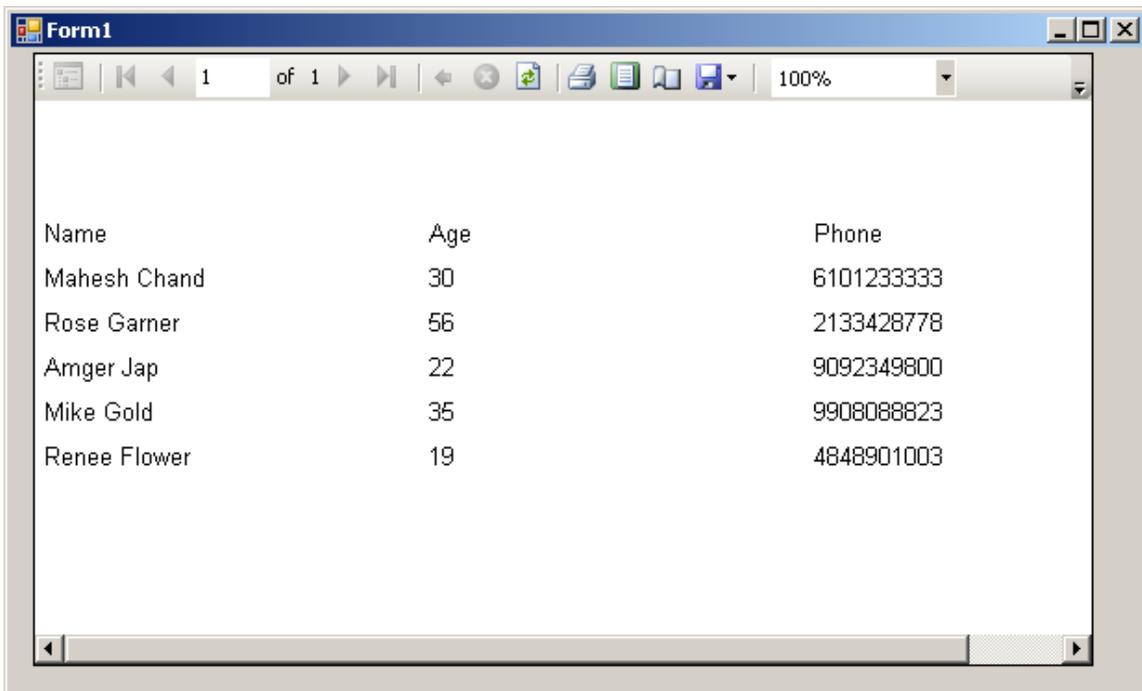
Now write the code listed in Listing 2 on the Form's load event handler. This code creates a DataSet, loads the data from Data.xml file and sets EmployeeBindingSource.DataSource as DataSet. The last line is added by you by the designer.

```
DataSet ds = new DataSet();
ds.ReadXml("Data.xml");
EmployeeBindingSource.DataSource = ds;
this.reportViewer1.RefreshReport();
```

Listing 2.

### Step 5. Build and Run

That's all. Build and run the application. The output should look like Figure 8.



The screenshot shows a report viewer window titled "Form1". The window has a toolbar at the top with navigation icons and a page indicator showing "1 of 1". The main content area displays a table with three columns: "Name", "Age", and "Phone". The table contains the following data:

Name	Age	Phone
Mahesh Chand	30	6101233333
Rose Garner	56	2133428778
Amger Jap	22	9092349800
Mike Gold	35	9908088823
Renee Flower	19	4848901003

Figure 8.

## Implementing Search in Reports

The ReportViewer control provides Find and FindNext methods to find a text in the report. The Find method takes two parameters – the text and the starting page number.

I have Form with a TextBox and a Button control. See Figure 1.

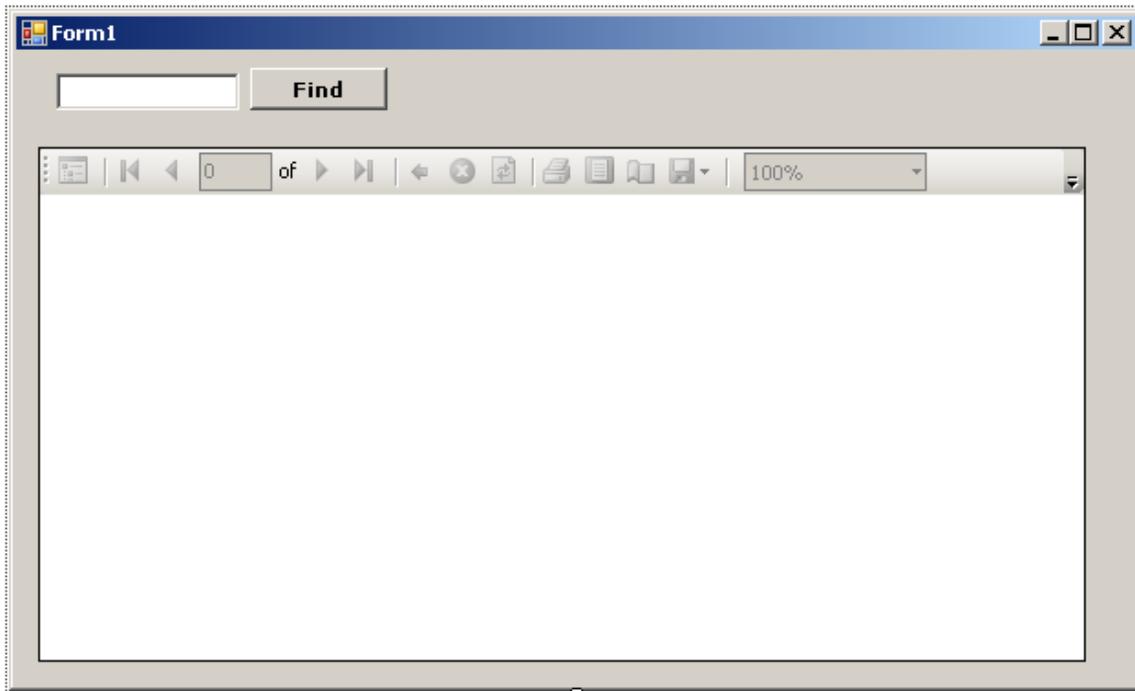


Figure 1.

I define the following variable in the class:

```
private bool firstFound = false;
```

The following code written on the Find Button click event handler.

```
private void FindButton_Click(object sender, EventArgs e)
{
    if (firstFound)
        this.reportViewer1.FindNext();
    else if (this.reportViewer1.Find(FindTextBox.Text, 1) >= 0)
        firstFound = true;
}
```

Now if I type “Chand” in the TextBox and click the Find button, the first row is selected. Clicking Find again select the next row that has text Chand in it. See Figure 2.

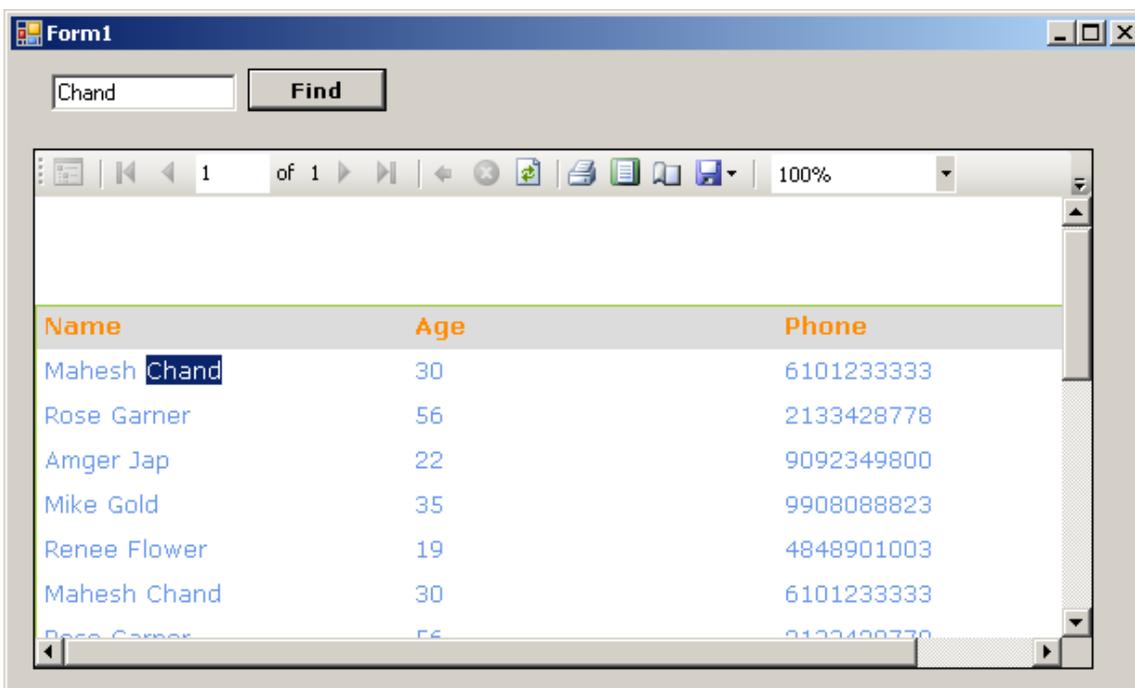


Figure 2.

