2015

C# Database Application

LEARN TO DESIGN & BUILD A WINDOWS C# DATABASE APPLICATION

STEPHEN O'CONNOR

Contact me:
stevo.joc@gmail
github.com/stevo5o
Stackoverflow Stevo5o

Database

What is a database?
A file structured for the repository of data. Organized for easy retrieval, sorting, grouping, relating to other data, used to analysis information in numerous ways.

Databinding

Utilizing Databinding in a C# Win forms App. Data sets working with the System.Data Namespace (aka ADO.NET) Working with the Visual Studio's IDE's tools, windows, etc. Microsoft Visual Studio 2013 makes it easily to create databases for beginners and experts.

Databinding the user interface controls, retrieve and display data from a data source without requiring the programmer to worry about all the programmatic details of this process. Each user interface control has different properties that can be bound to a data source.

ADO = ActiveX Data Objects
User interface controls must be data binding "aware", ADO.NET(System.Data) classes support data binding.

- ADO.NET creates a connection to a data source (database)
- ADO.NET manages the conversation (requests and responses) between your application and the database.
- ADO.NET manages the data that is retrieved forum the response to the database query.
- BindingSource manages the connection between the user interface controls and the underlying data set retrieved from the database. Provides an application interface to reduce learning curve for the end user. Restrict access to the database to maintain security. To control the presentation of the data. Maintain the integrity of the data.Practice

ADO.NET does a lot of the grunt work, it is not necessary to know all about ADO.NET.

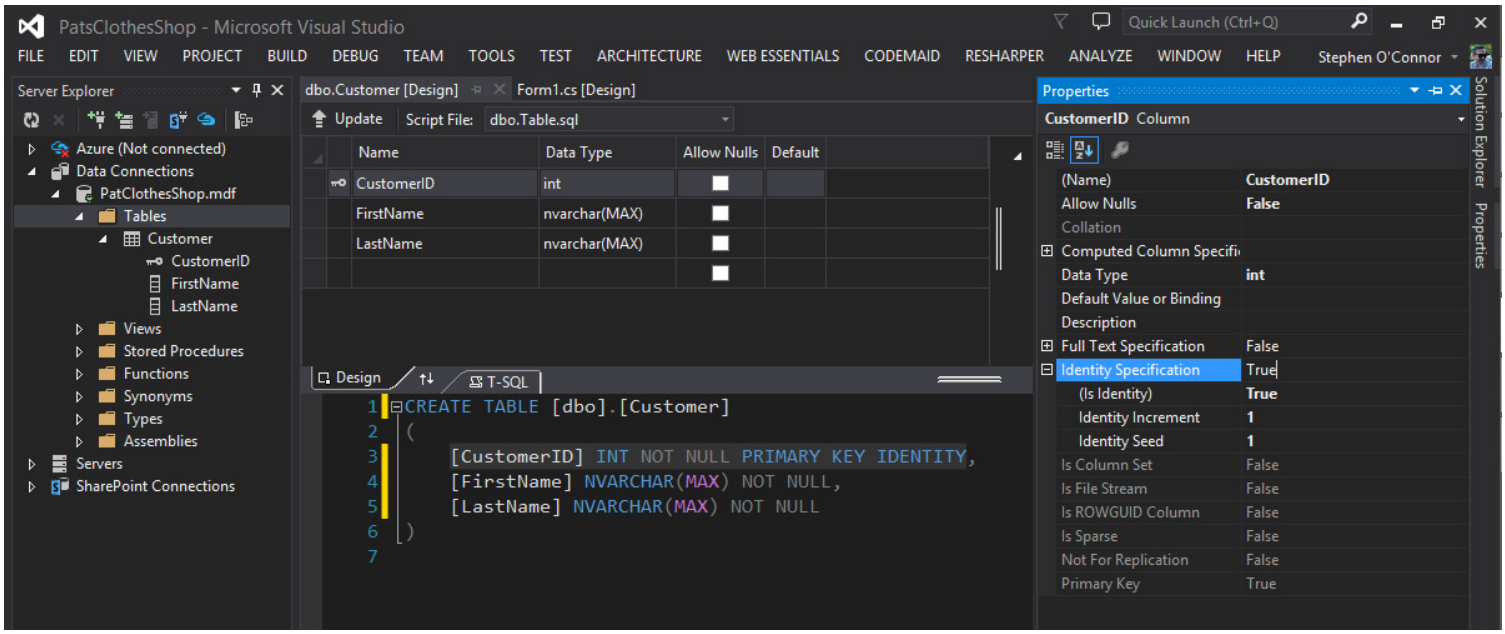
Write application interface

- Reduce the learning curve for the end user
- Restrict access to the database to maintain security
- To control the presentation of the data – website, content management system
- To maintain the integrity of the data

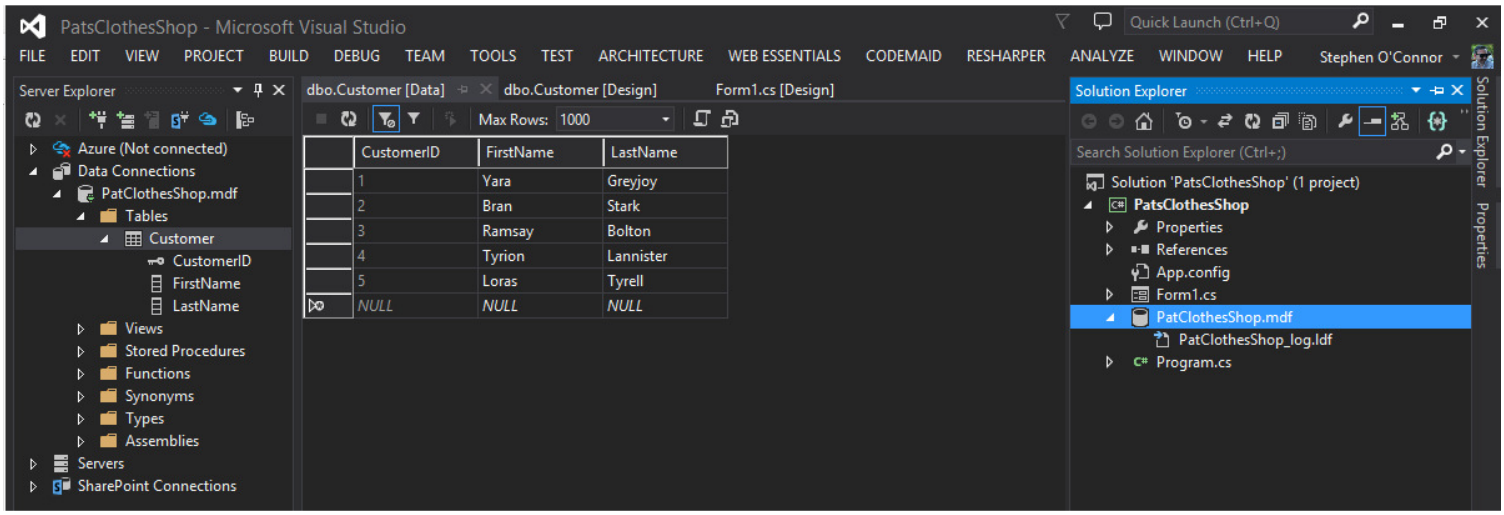
SQL Server
A high end relational database management system.
SQL server 2013 Express Edition, similar power, but intended for smaller projects.
In Visual Studio 2013 download the latest SQL Server Data Tools, if already not installed.

C# Database application

Learn by doing
Create a new project and a database called PatClothesShop
Add a table called customer with the data below.

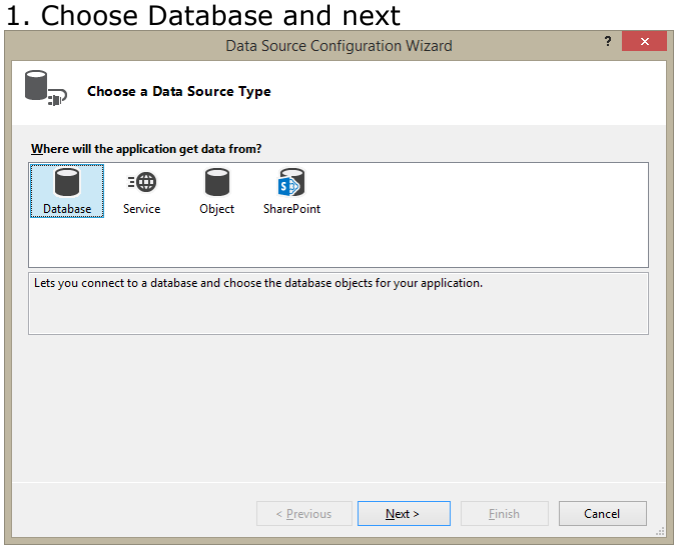


Go to show table data in the Server Explorer add five persons.

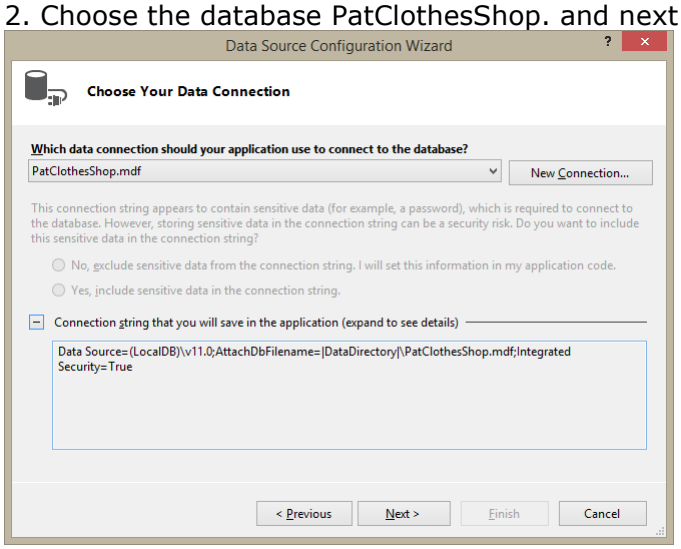


On the menu bar, choose View, Other Windows, Data Sources (or choose the Shift+Alt+D keys). Follow the steps.

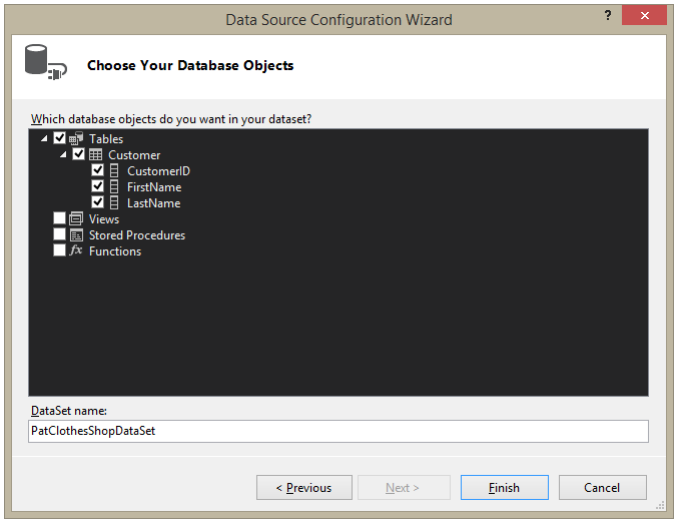
1. Choose Database and next



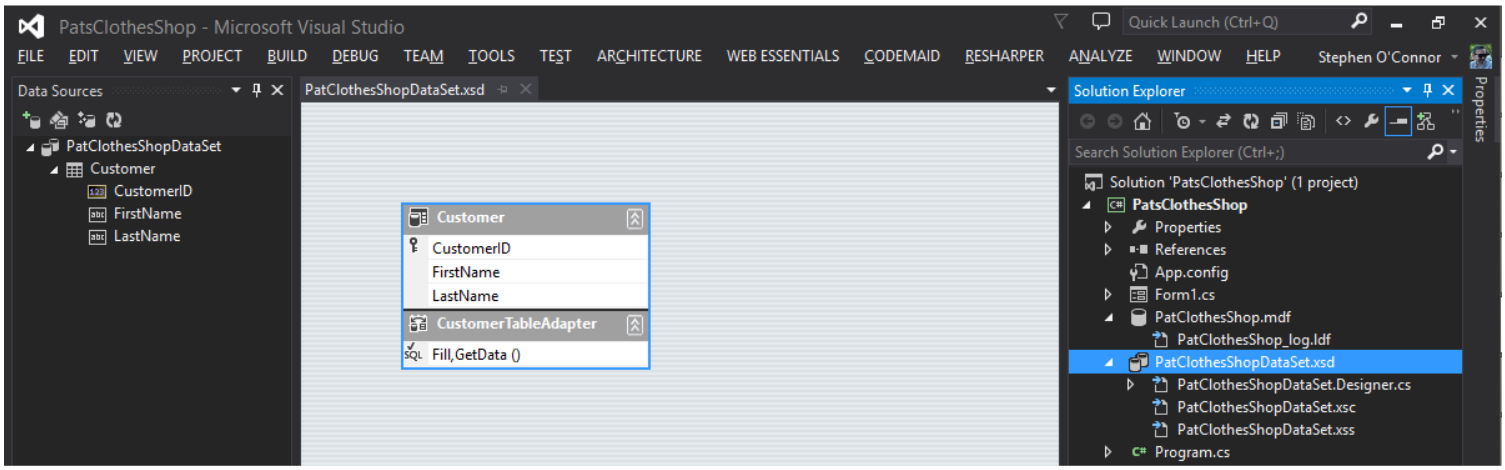
2. Choose the database PatClothesShop. and next



3. select table Customer This creates a .xsd file

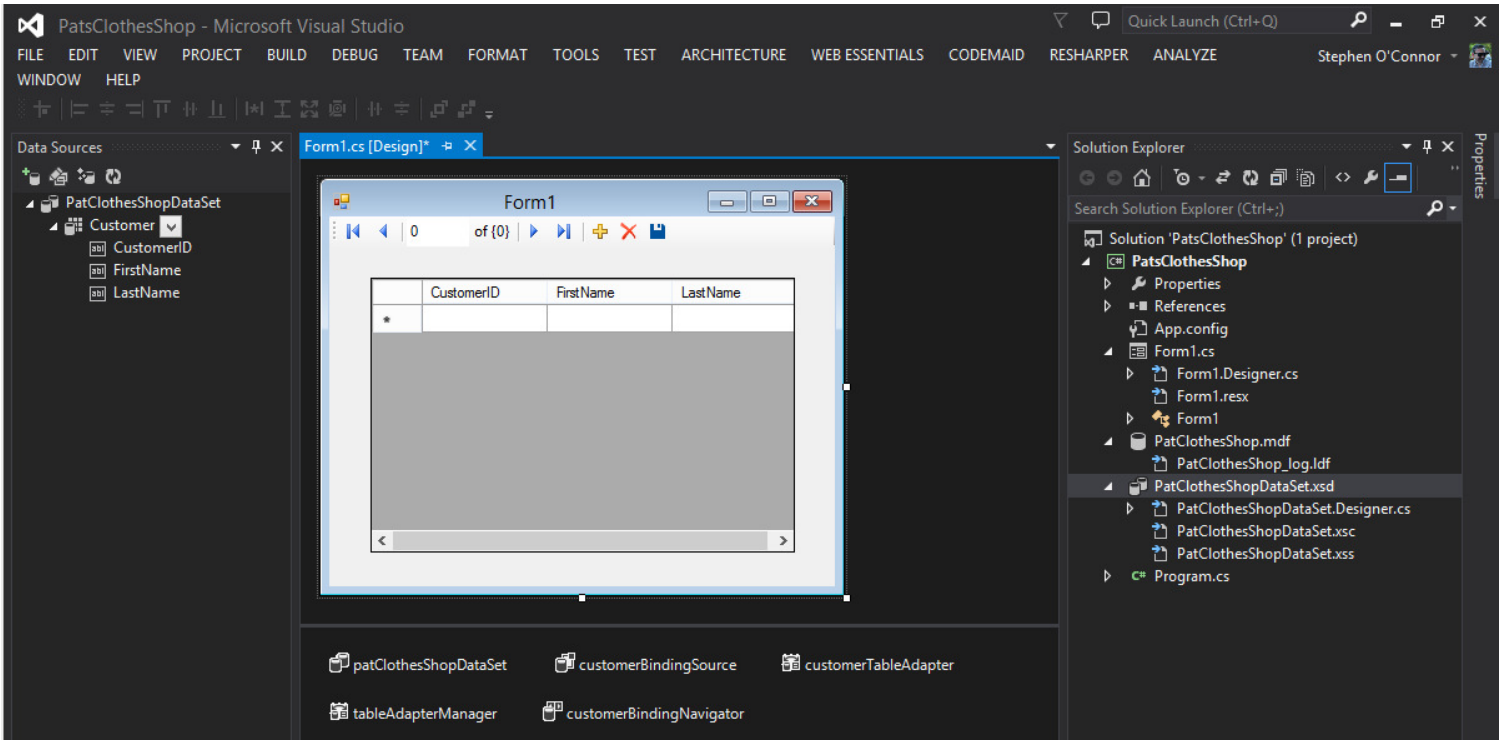


A PatClothesDataSet.xsd file right click on the xsd file and view designer mode. Xsd file is the xml schema document. The xsd file a local copy of database, this file defines the database, temporary stores the data.

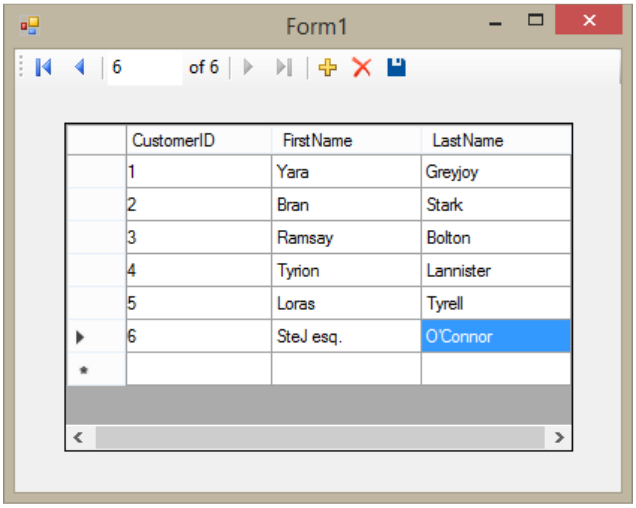


C# Database application

Drag and drop customer table from the Data Sources toolbar.

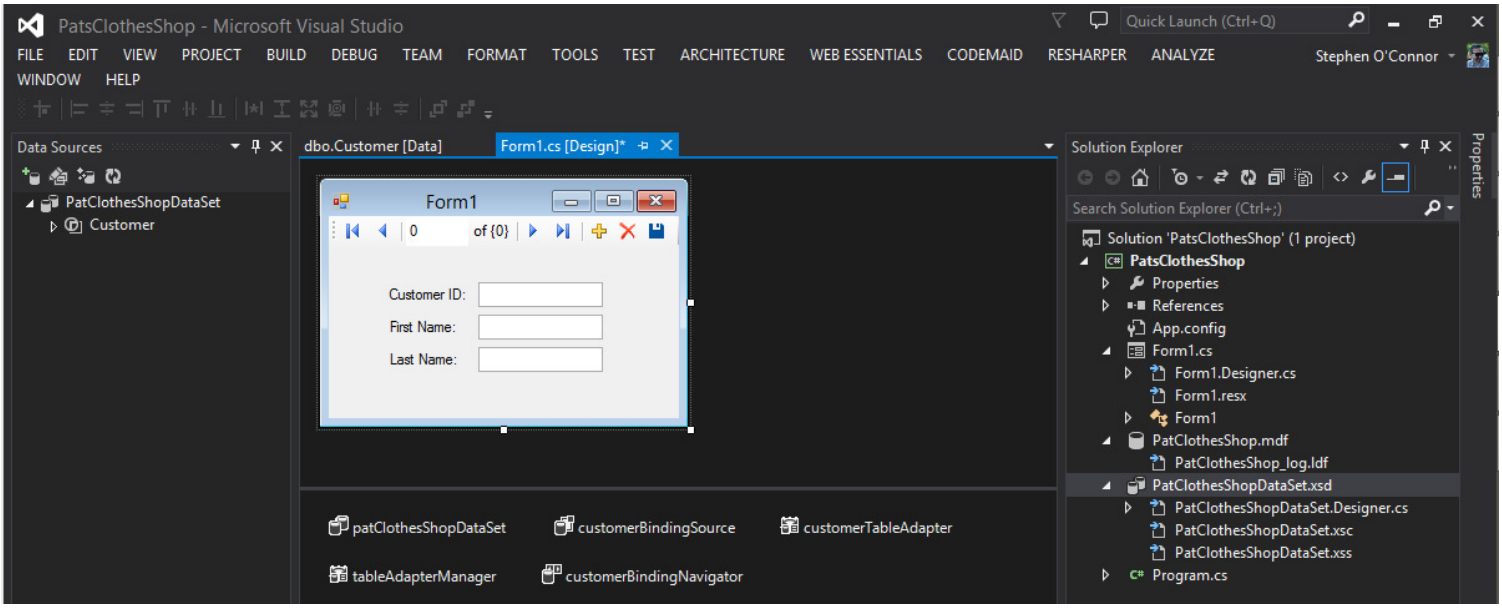


Run the project. A grid of the database navigate through the grid and add an extra row.

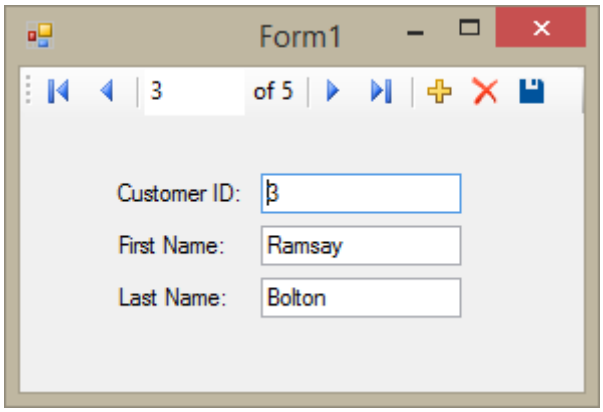


C# Database application

To create a Form view; select details from the drop down onto Form1



Run the application and navigate through the details view.



Designer tray.
patClothesShopDataSet
The local temporary storage container for the data within the application. Once the form is opened the dataset gets populated from the data from the database. Uses the TableAdapter to retrieve and connect to the database.

patClothesShopTableAdapter
Retrives data from the database, it contains a connection to the database. Contains an object that connects to the PatClothesShop.mdf to retrieve and resolve the information back into the database. Delete, update, add.

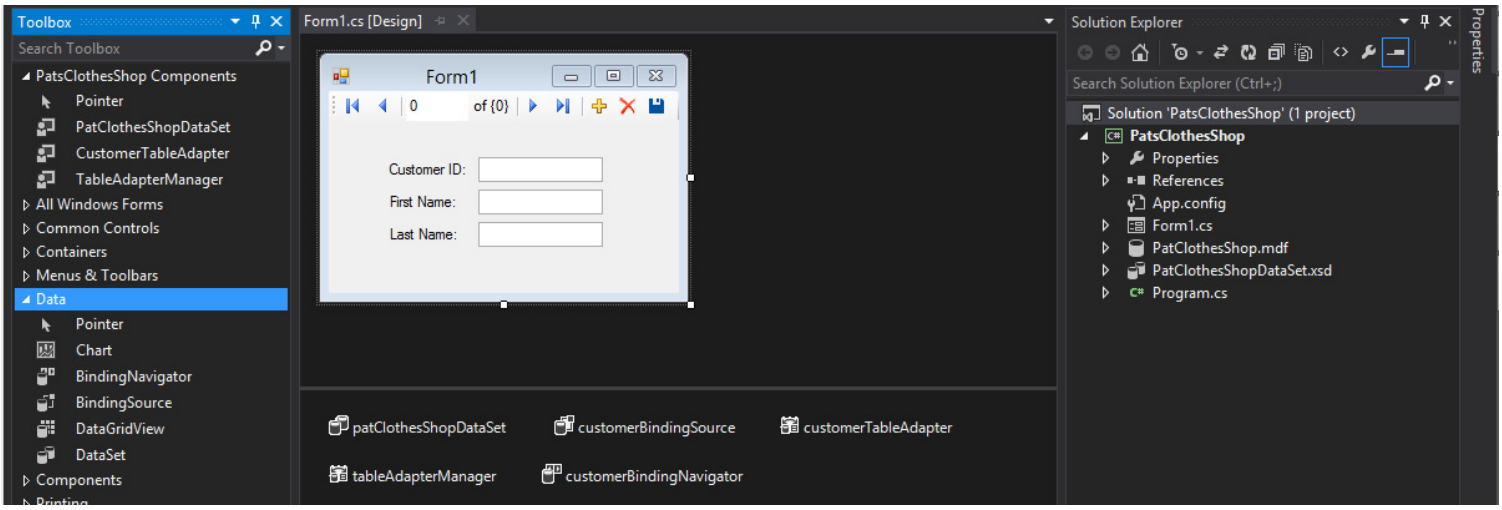
patClothesShopBindingSource
Object /bridge between the information in the dataset and the current row that's being displayed on the form. Keep all of the controls on the form bound to row of data in the DataSet. Co-ordinates what row of data (from the dataset) should be currently displayed. The user indicated they wanted to go to the first row or the next row or the last row in the textbox.

patClothesShopAdapterManager
Service interface

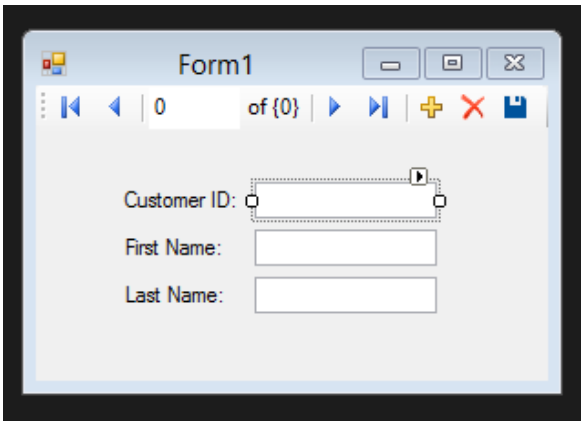
patClothesShopBindingNavigator
Toolbar at the top of the form.

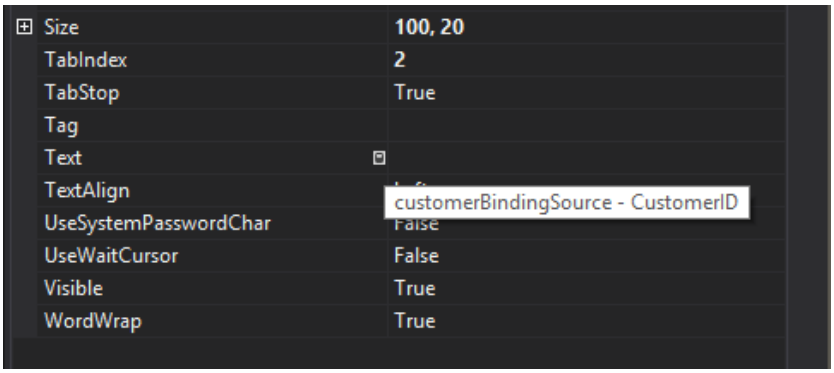
C# Database application

Go to toolbox and select data to show data tools that can be added manually to the form.

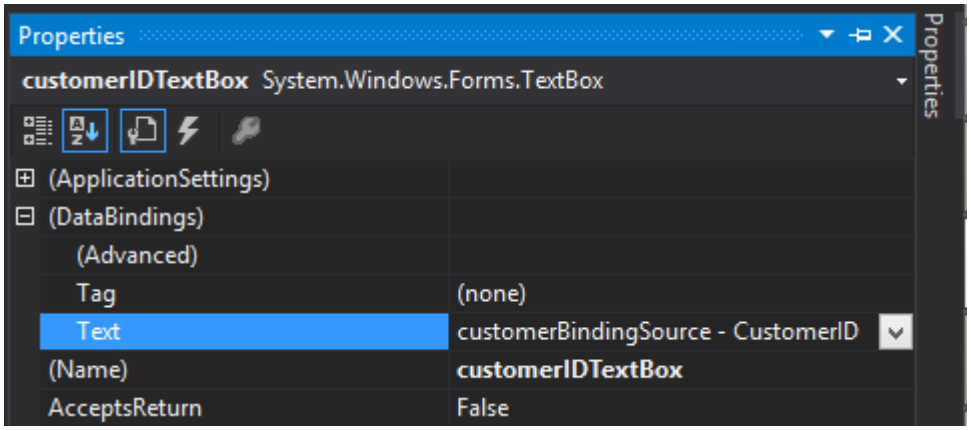


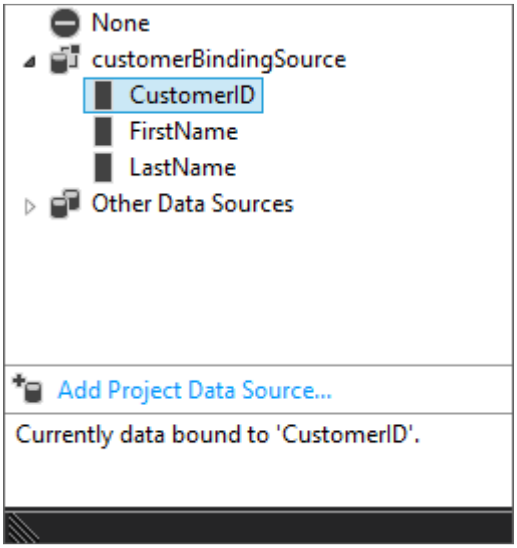
Properties of the first textbox. In the Text a Database icon is displayed.



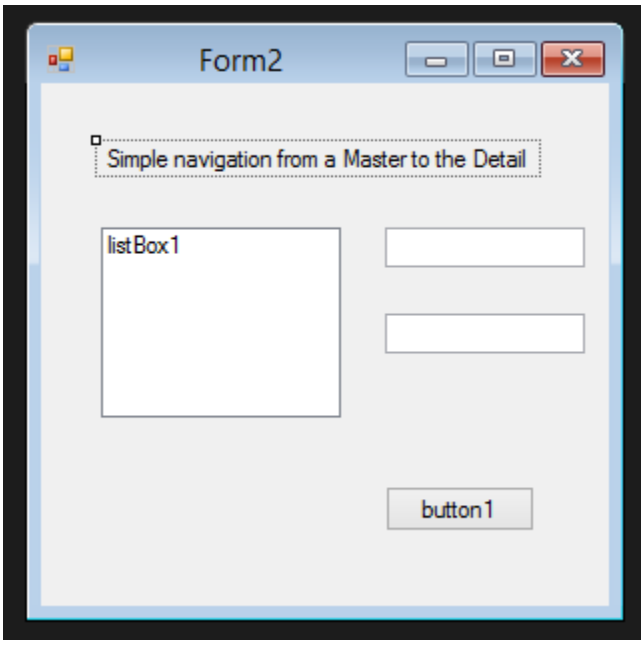


Select and right click to display the properties of the textbox. CustomerID is binded to the first textbox. Binding source schema document. By using the data sources toolbar the database can be dragged and dropped onto the form, sets the textboxs automatically.



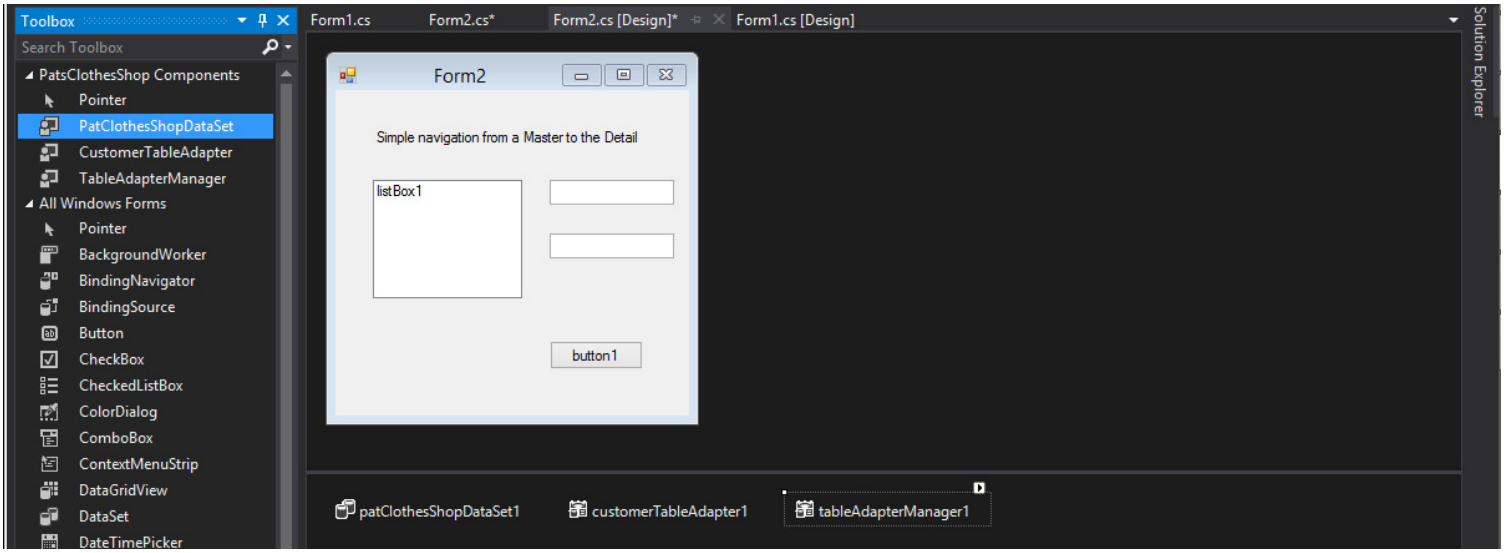


Add the data items manually.
Create a new form form2 add the items displayed in the below image. Label, Listbox, 2 textboxs and a button. Strongly typed components or preconfigured.



Add a button to form1 to open form2. Type the code below to create an instance of the class Form2

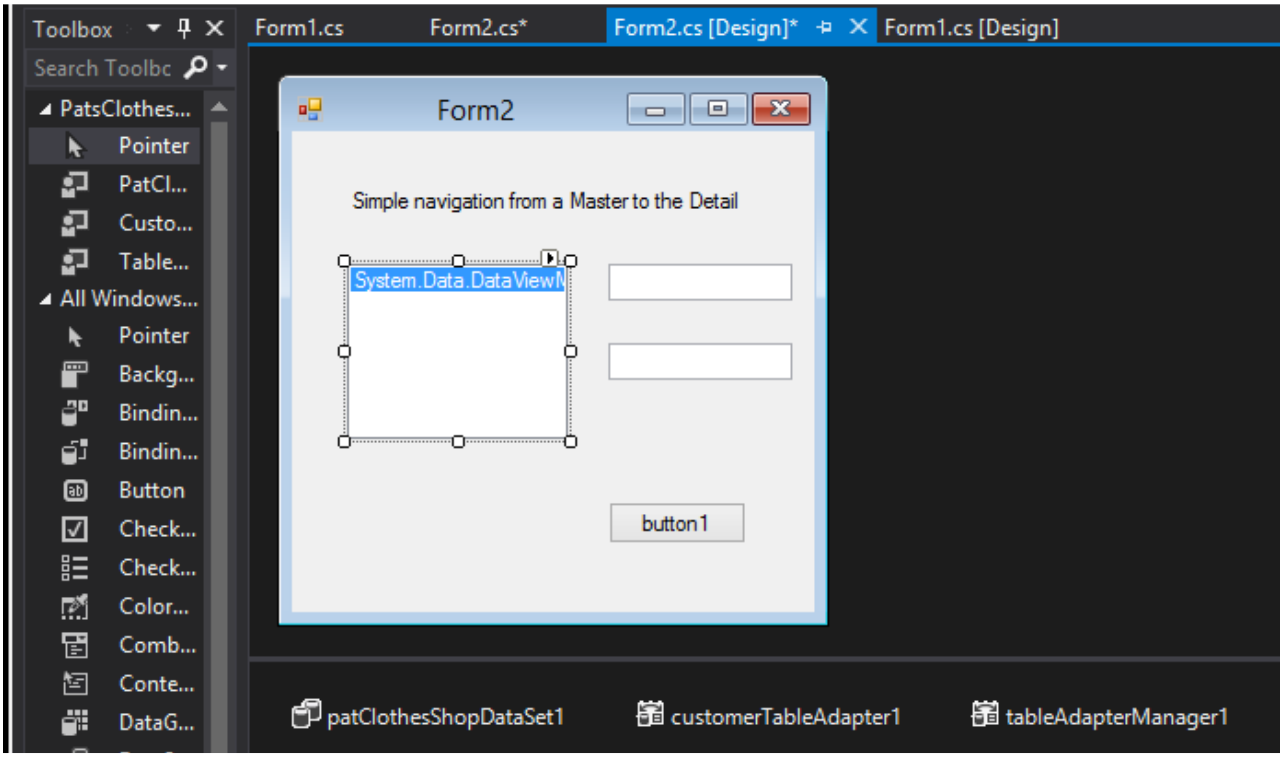
```
Form2 myForm = new Form2();
            myForm.Show();
```

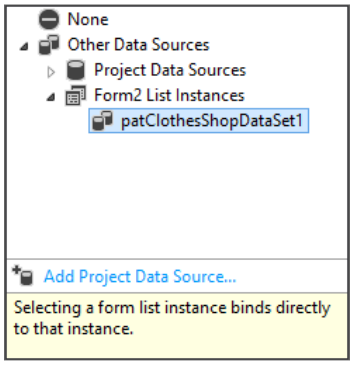


Double click form2 type the code below. Fill method passes in `patClothesShopDataSet1.Customer`
Fill method takes action to grab the data from the database and populate the customer table of the database with the data it retrieves.

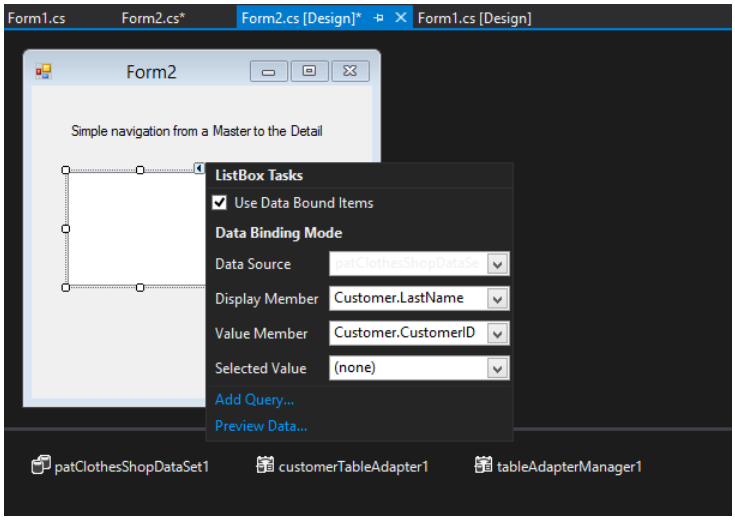
```
private void Form2_Load(object sender, EventArgs e)
        {
            customerTableAdapter1.Fill(patClothesShopDataSet1.Customer);
        }
```

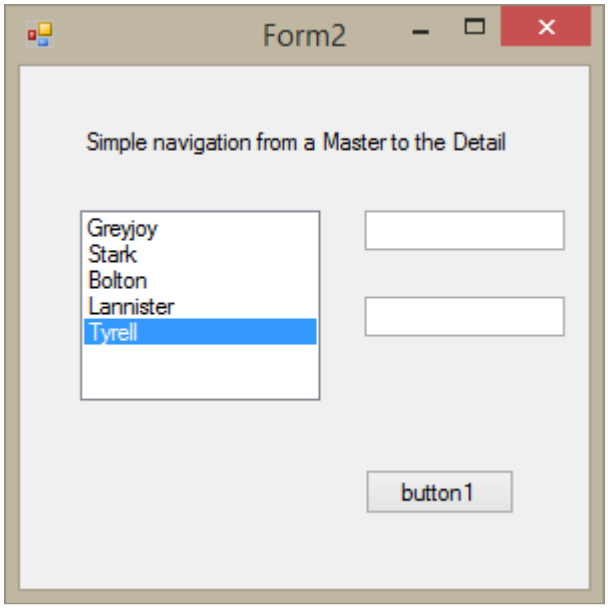
Select the listbox click the arrow and select patClothesShopDataSet1, from the pop-up box.





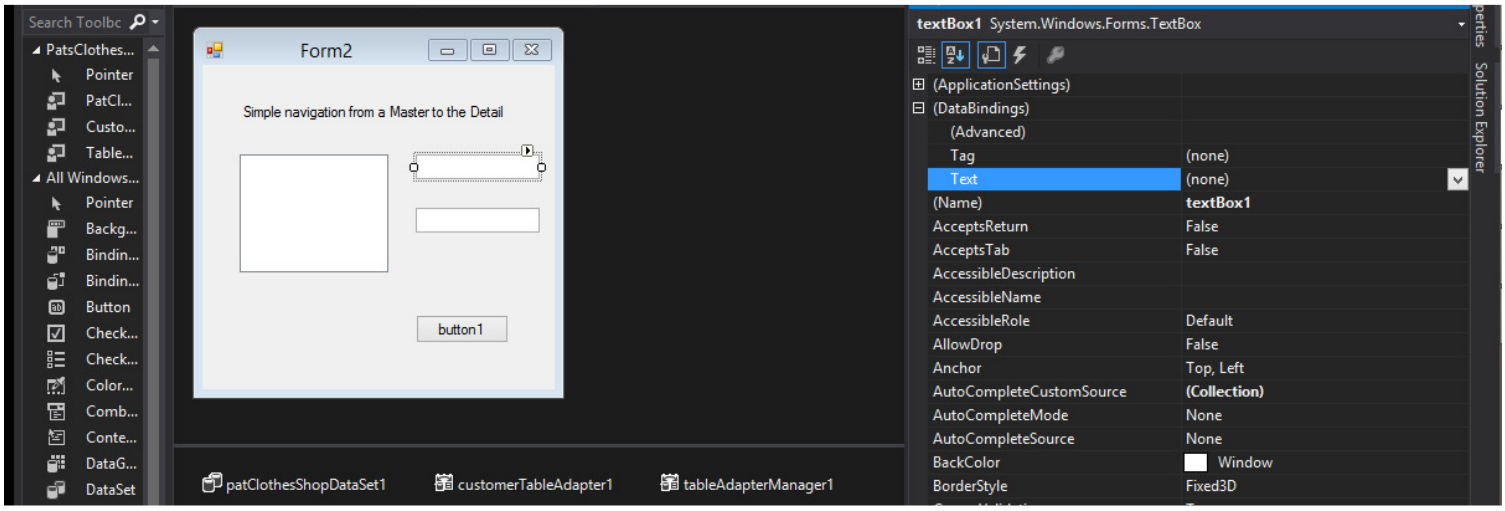
Select the list box and click the arrow button to data bound the listbox to the dataabse. Check the checkbox Use data bound items. From the pop-up select the data source patclothesshop1. In Display member select the last name and Customer last name this will displayed. In value member select the CustomerID, which is always unique.



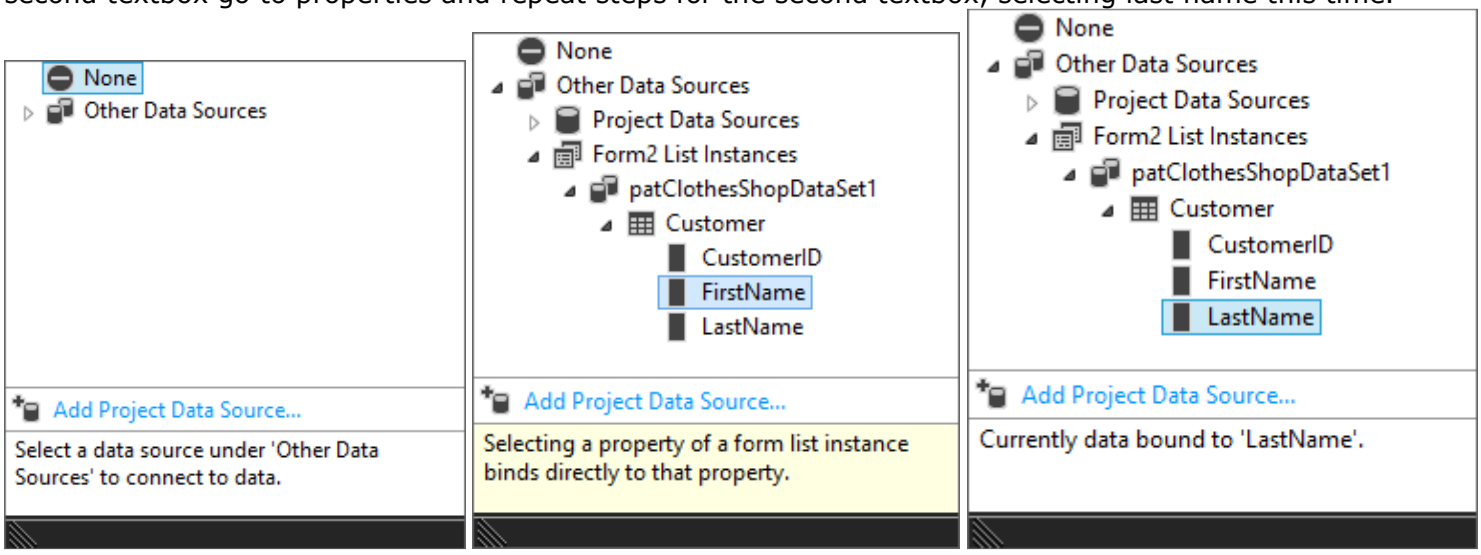


C# Database application

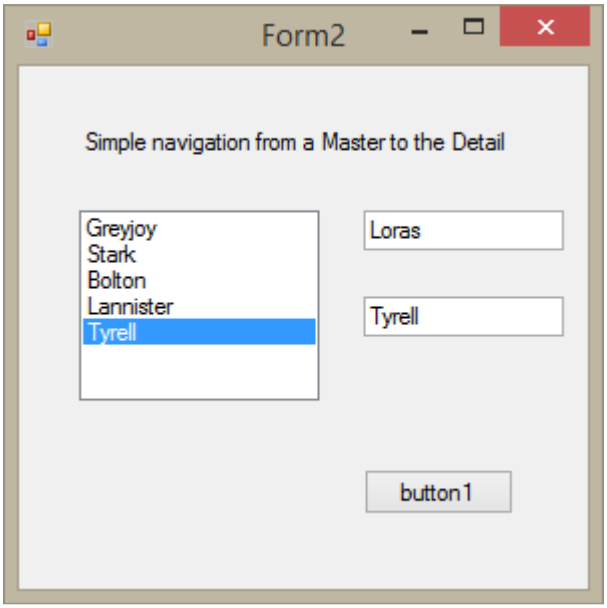
Bind the textboxs on the right with the listbox on the left. Select the first textbox and right click properties go to databindings and select text click the drop down arrow.



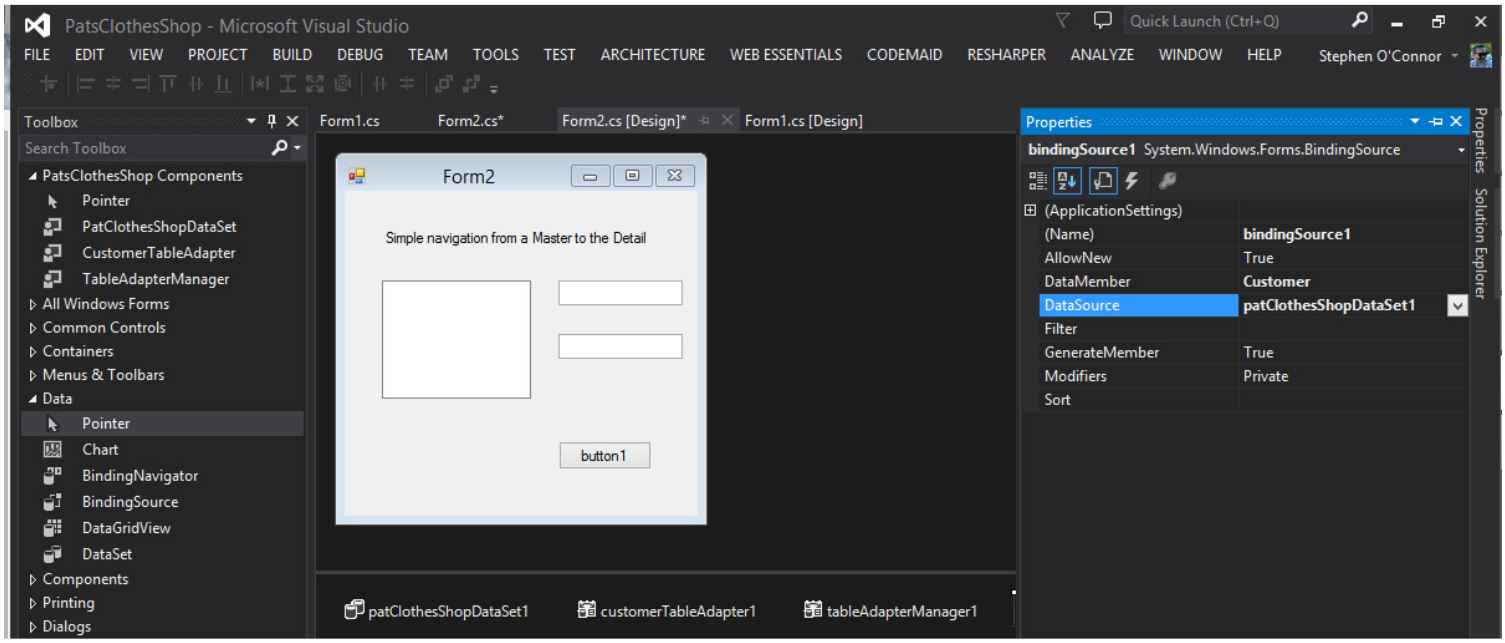
From the pop-up select first name for the first textbox. Go back to the form and select and right click the second textbox go to properties and repeat steps for the second textbox, selecting last name this time.



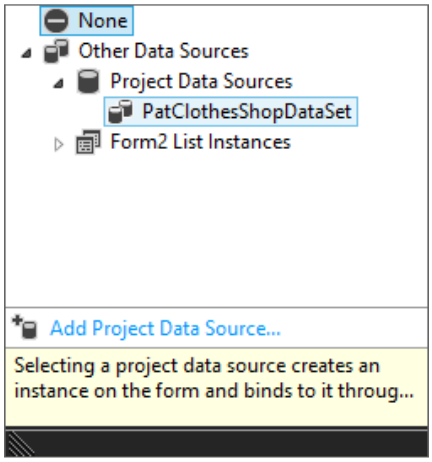
Run the application, open second form. Select from the listbox to display the first name and last name of the person in the textboxs.



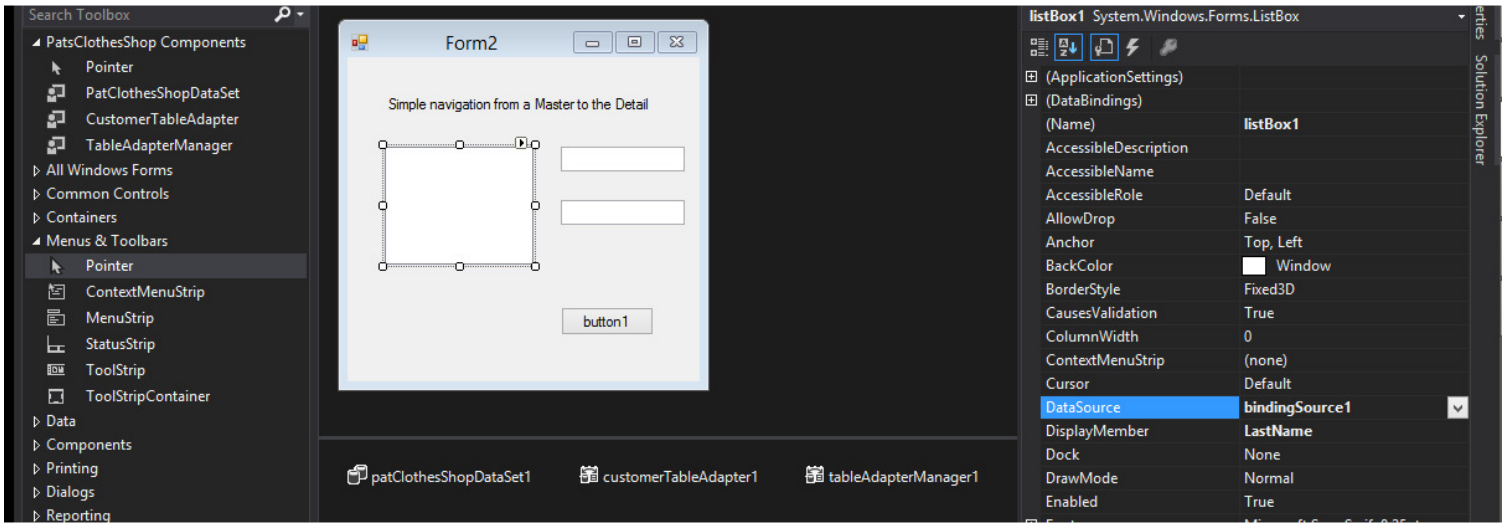
Updating capabilities. Drag and drop the BindingSource onto form2. To update the data. Hover over and read the description of the BindingSource tool. The bindingSource1 has been added to the designer tray.

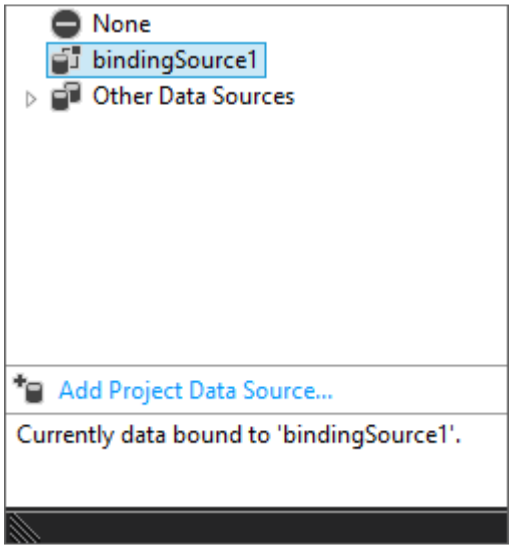


Click bindingSource1 and select PatClothesShopDataSet from the properties DataSource pop-up. Select Customer as the DataMember.

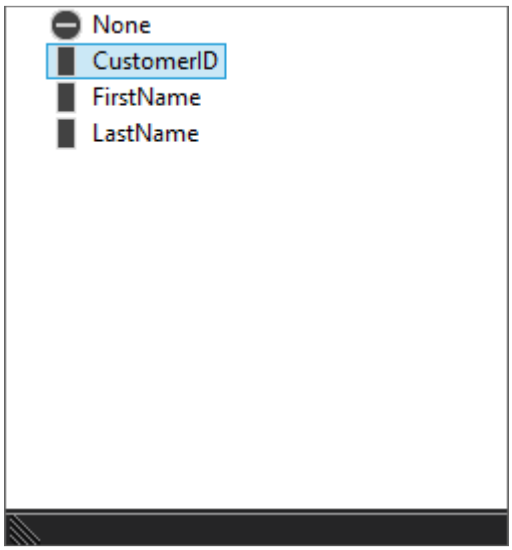


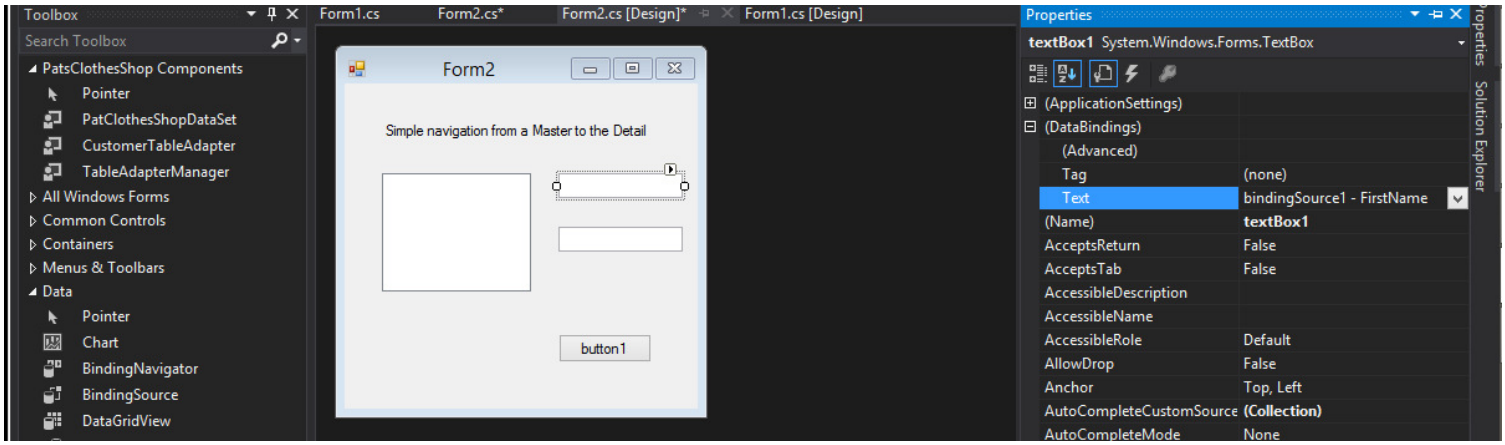
Select listbox go to data source select bindingSource1. Change the DisplayMember to LastName.



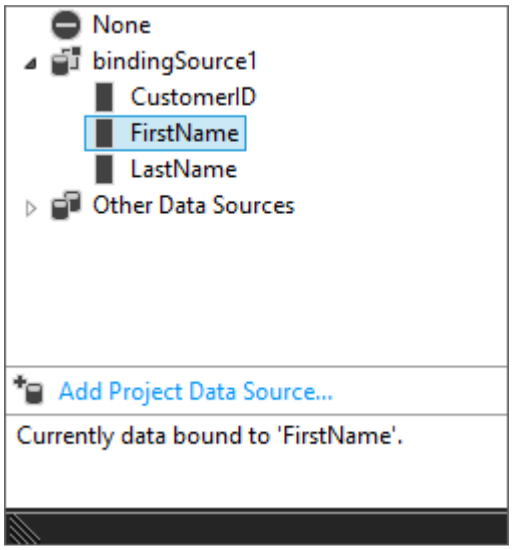


Select Value member in the listbox from properties and change to CustomerID.

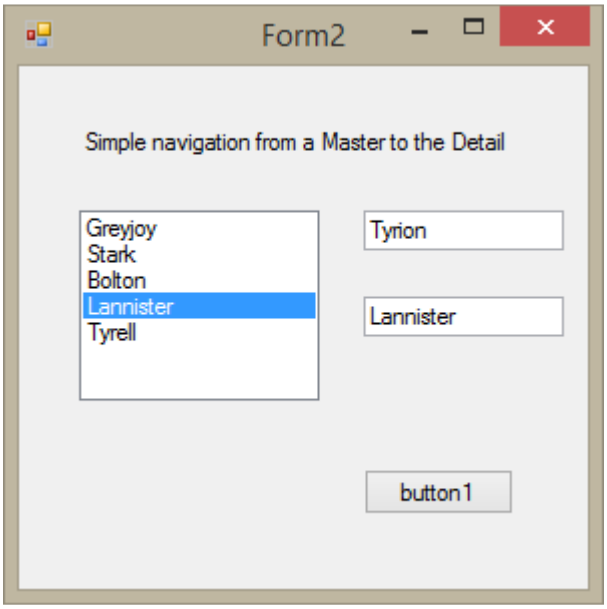




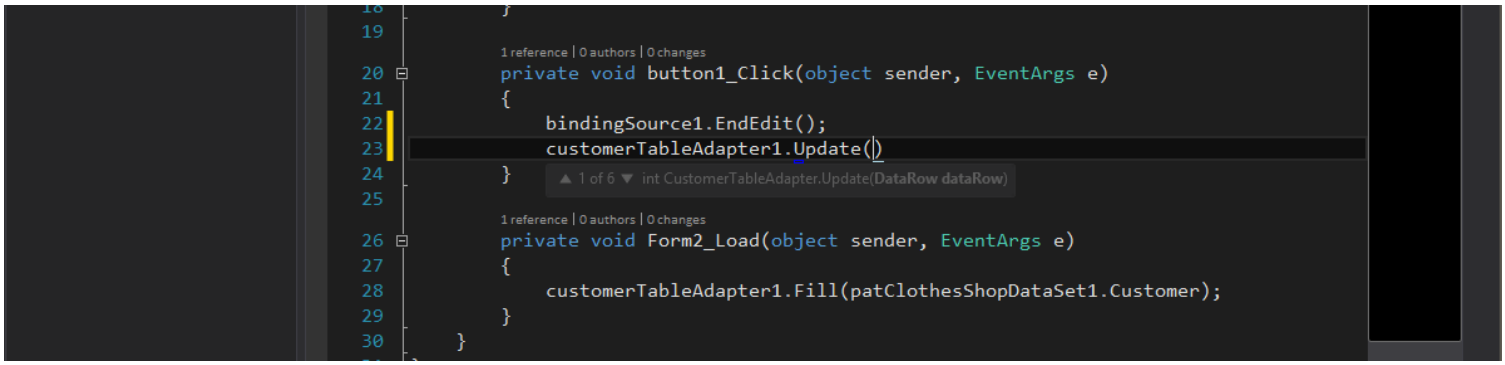
Select the properties of the first and second textboxs and select first and last name respectively.



Run application to check if it still works.



Double click the button on form2

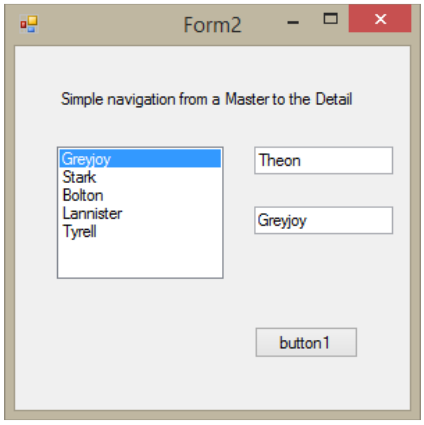



```
private void button1_Click(object sender, EventArgs e)
        {
            bindingSource1.EndEdit();
            customerTableAdapter1.Update(patClothesShopDataSet1.Customer);
        }
```

```
            // save changes to the dataset
            bindingSource1.EndEdit();

            // select TableAdapter and return number of items updated
            customerTableAdapter1.Update(patClothesShopDataSet1.Customer);
```

```
            // save changes to the dataset
            bindingSource1.EndEdit();

            int result = 0;

            // return number of items updated
            result = customerTableAdapter1.Update(patClothesShopDataSet1.Customer);

            // display the row has been updated
            MessageBox.Show(result.ToString());
```

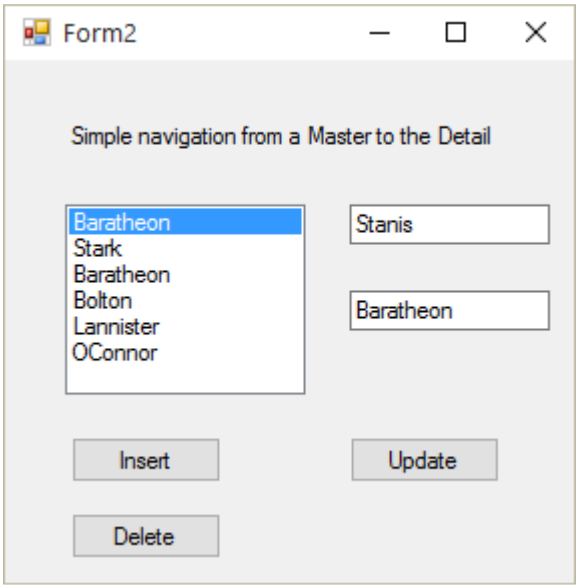
Open the .exe file in the debug /bin folder. Change Yara (first name to Theon) close and reopen .exe file and see the change made to the database.



stackoverflow
Actually this is not a issue with update command. Visual Studio keeps two databases. One in project folder and one in bin/debug folder. Database in bin/debug folder always update with Database in project folder. If you view Database through Visual Studio, it always shows the Database inside the project folder not other one inside bin/debug folder.

C# Database application

Add /Select /Delete from /to the database.



Add two buttons called Insert and Delete. In the code file add the SQL Client namespace.

```
using System.Data.SqlClient;
```

Create an `SqlConnection` using DB connection string (right click .mdf file) from properties copy paste connection string.

```
        private void btn_Insert_Click(object sender, EventArgs e)
        {
            // initialize a new instance of the SqlConnection class, containing the
connection string
            SqlConnection myConnection = new SqlConnection(@"Data
Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\Stephen\Documents\VSProjects\cshar
p-db-app\PatsClothesShop\PatClothesShop.mdf;Integrated Security=True");

            try
            {
                // open connection
                myConnection.Open();
                Console.WriteLine("Connection"); ;
            }
            catch (Exception)
            {
                Console.WriteLine("Error");
            }

            // Initialize a new instance of the SqlCommand, query and SqlConnection
instance
            SqlCommand myCommand = new SqlCommand("INSERT INTO Customer (FirstName,
LastName) " +
                                        "Values ('Stanis', 'Baratheon'), ('Ned',
'Stark'), ('Renly', 'Baratheon'), ('Ramsay', 'Bolton'), ('Cersei', 'Lannister'),
('SteJ', 'OConnor')", myConnection);

            // executes a Transact-SQL statement against the connection and returns
the number of rows affected
            myCommand.ExecuteNonQuery();
```

```
            try
            {
                SqlCommand myCommandSel = new SqlCommand("SELECT * FROM Customer",
                                                         myConnection);
                // Initialize a new instance of SqlReader, sends the CommandText to
the Connection and builds a SqlDataReader
                SqlDataReader myReader = myCommandSel.ExecuteReader();
                while (myReader.Read())
                {
                    Console.WriteLine(myReader["FirstName"].ToString());
                    Console.WriteLine(myReader["LastName"].ToString());
                }
            }
            catch (Exception)
            {
                Console.WriteLine("Broke");
            }
        }

        private void btn_Delete_Click(object sender, EventArgs e)
        {
            SqlConnection myConnection = new SqlConnection(@"Data
Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\Stephen\Documents\VSProjects\cshar
p-db-app\PatsClothesShop\PatClothesShop.mdf;Integrated Security=True");

            try
            {
                myConnection.Open();
                Console.WriteLine("Connection"); ;
            }
            catch (Exception)
            {
                Console.WriteLine("Error");
            }

            SqlCommand myCommandDel = new SqlCommand("DELETE FROM Customer WHERE
CustomerID in(SELECT TOP 3 CustomerID FROM Customer ORDER BY CustomerID DESC)",
myConnection);

            myCommandDel.ExecuteNonQuery();

            try
            {

                SqlCommand myCommandSel = new SqlCommand("SELECT * FROM Customer",
                                                         myConnection);
                SqlDataReader myReader = myCommandSel.ExecuteReader();
                while (myReader.Read())
                {
                    Console.WriteLine(myReader["FirstName"].ToString());
                    Console.WriteLine(myReader["LastName"].ToString());
                }
            }
```

```
            catch (Exception)
            {
                Console.WriteLine("Broke");
            }
        }
```

Delete from the database.

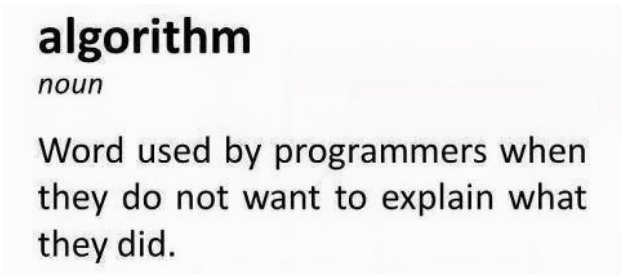
```
SqlCommand myCommand = new SqlCommand("DELETE FROM Customer WHERE CustomerID
in(SELECT TOP 3 CustomerID FROM Customer ORDER BY CustomerID DESC)", myConnection);
```

Adding to the dataset.

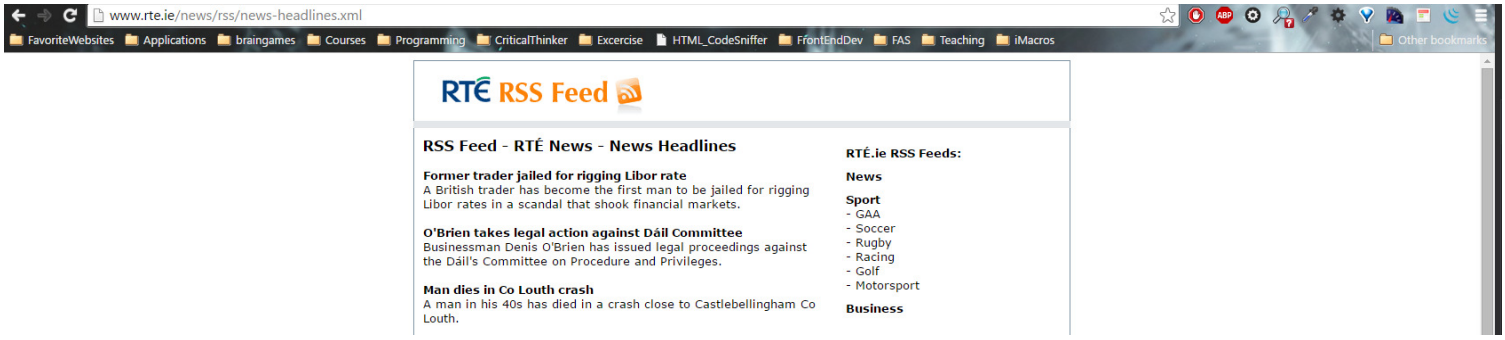
```
        private void btn_Insert_Click(object sender, EventArgs e)
        {
            bindingSource1.EndEdit();
            patClothesShopDataSet1.Customer.AddCustomerRow("Yara", "Grayjoy");
            MessageBox.Show("Name Added");
        }
```

Deleting from the dataset.

```
        private void btn_Delete_Click(object sender, EventArgs e)
        {
            patClothesShopDataSet1.Tables["Customer"].Rows[0].Delete();
            MessageBox.Show("Name Deleted");
        }
```



RSS (Really Simple Syndication) reader
An RSS link is a way of getting all your favorite information collected into one place
Rather than have you search the web every day trying to find it, use it to have everything from sports stats to breaking news to job openings all forwarded to you. We will be creating our own RSS reader



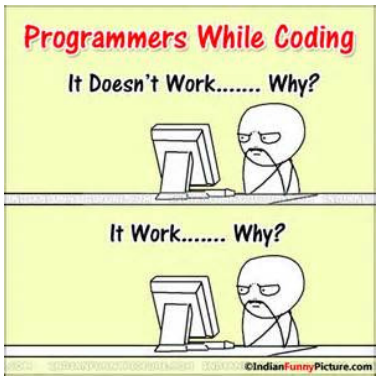
Required channel elements
RSS 2.0 Specification
Here's a list of the required channel elements, each with a brief description, an example, and where available, a pointer to a more complete description.

Element	Description	Example
title	The name of the channel. It's how people refer to your service. If you have an HTML website that contains the same information as your RSS file, the title of your channel should be the same as the title of your website.	GoUpstate.com News Headlines
link	The URL to the HTML website corresponding to the channel.	http://www.goupstate.com/
description	Phrase or sentence describing the channel.	The latest news from GoUpstate.com, a Spartanburg Herald-Journal Web site.

Bringing all of what we have learned together
Building a C# RSS reader. Purpose of this exercise concept to deployment, the steps of building an application from start to finish.

The Process
Most formal software development processes are created for teams of developers to help them communicate better and stay organized.

MSF – Microsoft Solutions Framework
Microsoft Solutions Framework (MSF) Overview
The Microsoft Solutions Framework (MSF) is an adaptable approach for successfully delivering technology solutions faster, with fewer people and less risk, while enabling higher quality results. MSF helps teams directly address the most common causes of technology project failure — improving success rates, solution quality, and business impact.



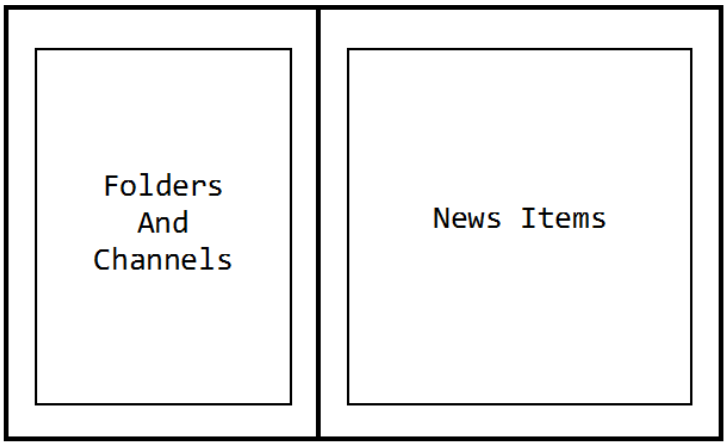
Basic stages of MSF

- Envision the solution
  - Allows users to read news items from RSS feeds
  - Subscribe to channels
  - Organize channels
  - View news items in a web browser
  - Refresh channels
  - Edit and delete channels and folders
  - Unobtrusive feedback /notifications
- Plan the solution
  - Logical design
  - Physical design
  - User Interface Design
    - Using a "low-tech mock up"
  - Database Design
- Develop the solution
  - Draw up the design for the UI and the database
  - The RSS specification helps us to determine most of the data structure we'll need
    - Folder - folderID
      - Channels – title URL
        - News items – title, description, link, date channelID FK relationship

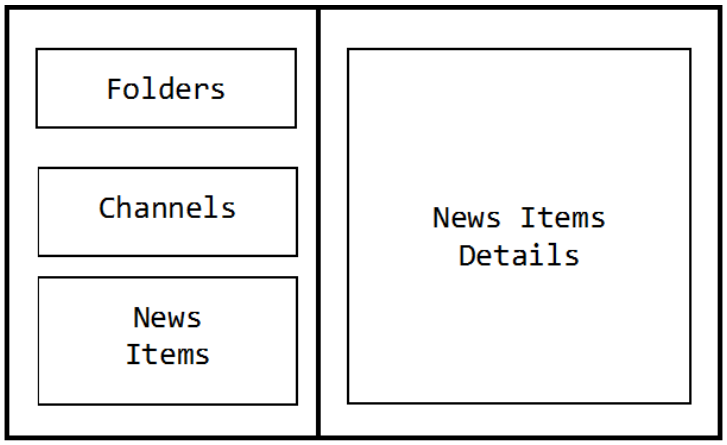
    - User Interface design
- Test the solution
- Deploy the solution

User Interface design
Keep it simple for now, if need be add more features later. Keep application scope small enough so that it actually gets finished. Avoiding scope creep, Scope creep is when a change – an update or addition – to the whole or even part of the project has been requested when the project is already underway. The change may come from your client, co-worker, or customer – normally whoever requested the project. They have decided they want something done differently or would basically like the outcome to be different than originally requested. This can happen once or several times within the life of a project. - See more at: [teamgantt.com scope creep the two dirtiest words](). A quick success before adding too many features, version 1.

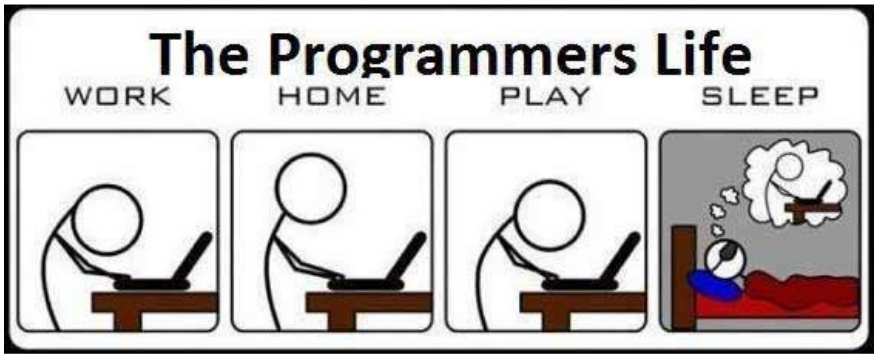
1. Basic mock up windows explorer type view



2. When a folder is select this will be the view



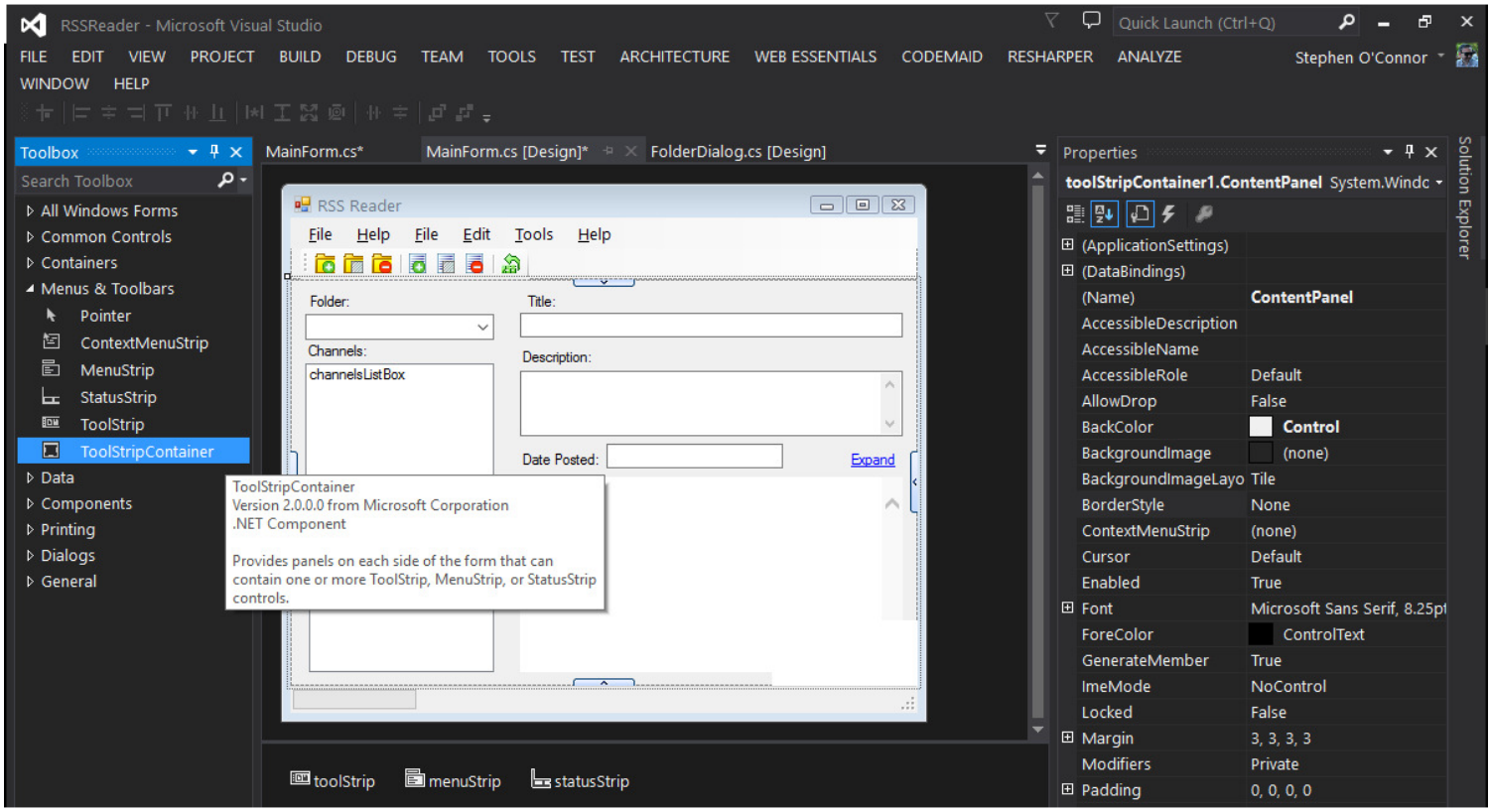
Each "pass" through these steps (to implement additional functionality is called an "iteration".



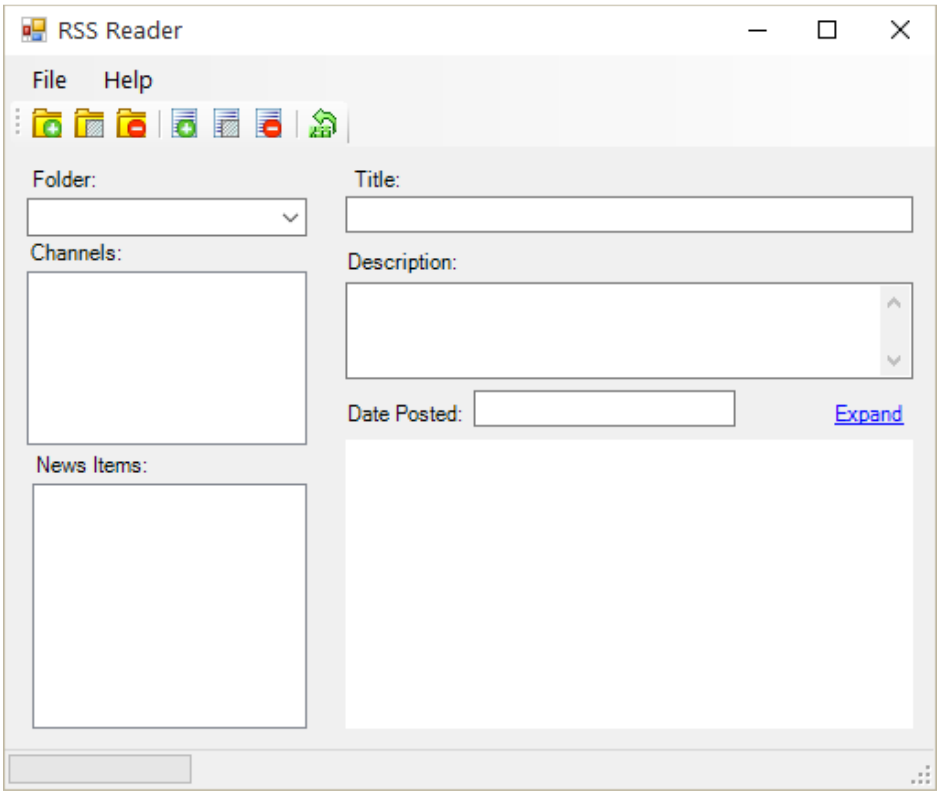
Let's start
Create a new project called RSSReader.
Create three forms

- MainForm – RSS reader
- FolderDialog
- ChannelDialog

MainForm.cs
Drag and drop a toolStripContainer onto the RSS reader form. Run the application toolStripContainer allows you to customize the UI and more things around and docking toolbar controls.



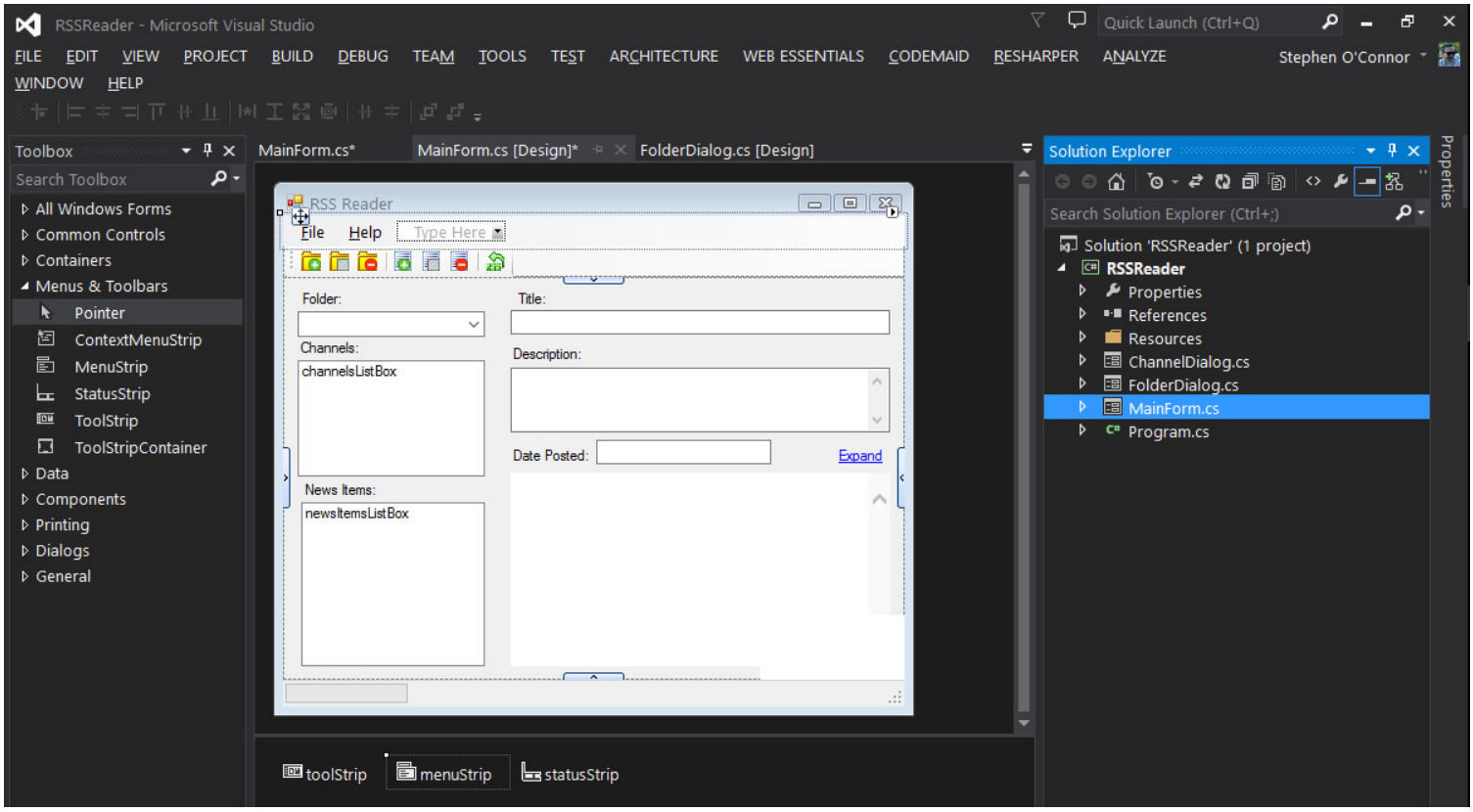
Finished form.



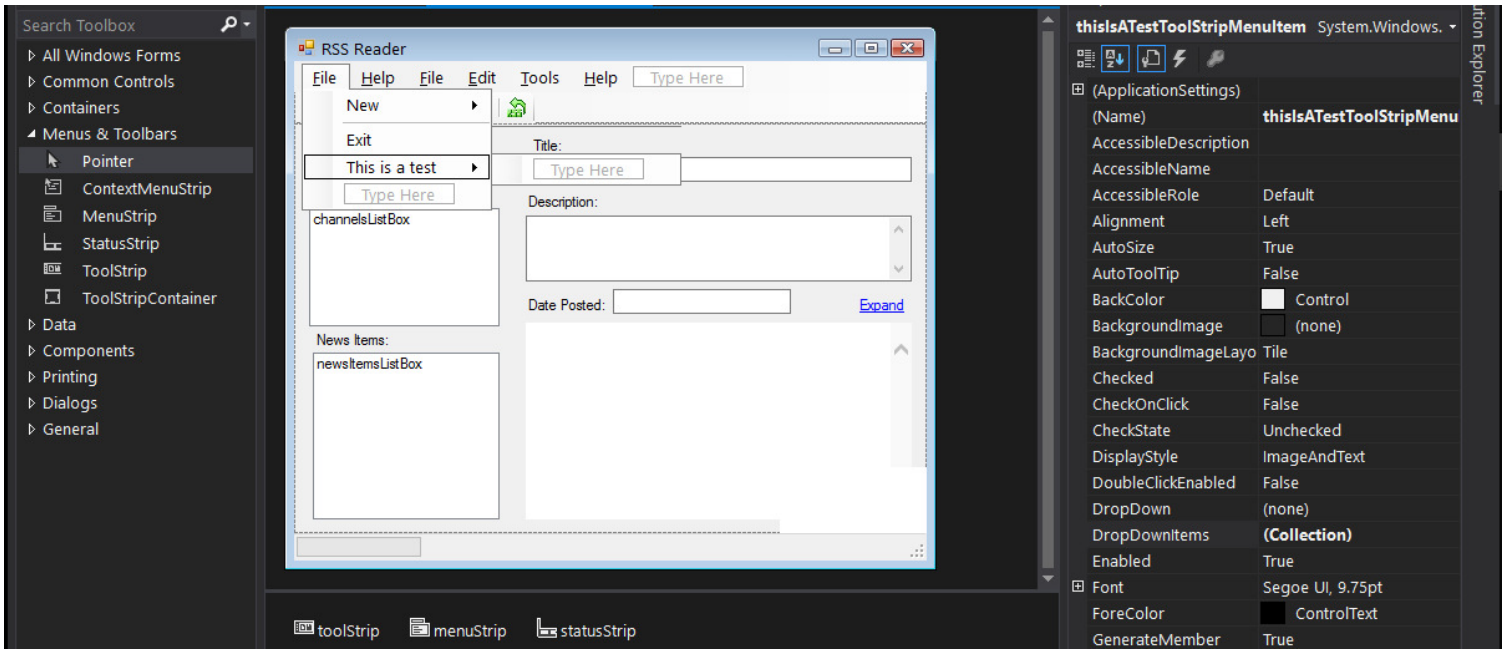
Drag and drop into the toolStripContainer

- Labels named as below
- Drop down menu – to organize channels feds, when select a folder will view all the RSS feeds in the channelsListbox
- channelsListbox – view all the channels feeds
- newsItemslistBox – when a feed is select all news items will be displayed here
- textbox on the right – view the title (details) of the new feed
- textbox on the right – view the description (details) of the new feed – Set multiline to true and set scroll bar to vertical
- textbox on the right – view the date posted (details) of the new feed
- Expand link to view in browser – label control – select underline, forecolor to blue and set both to true

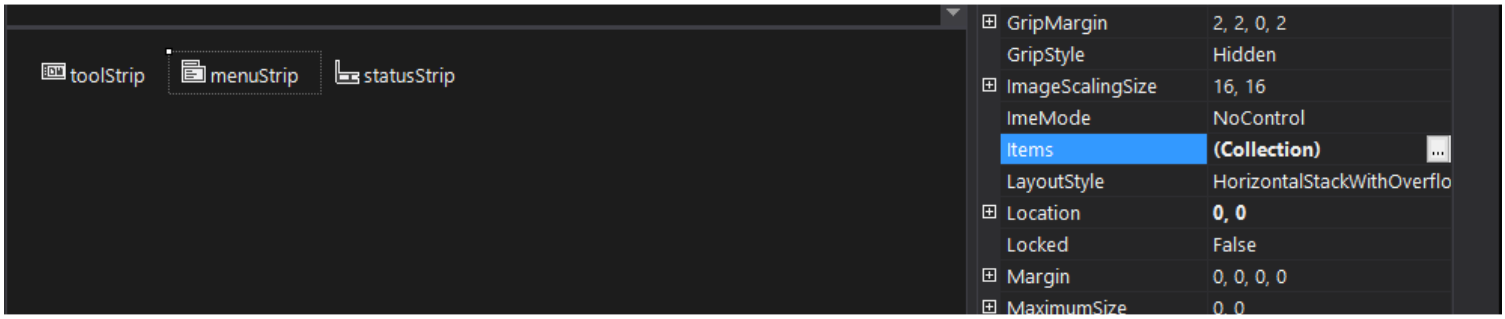
- WebBrowser – browser view, to view web pages
- Create a naming convention to avoid confusion



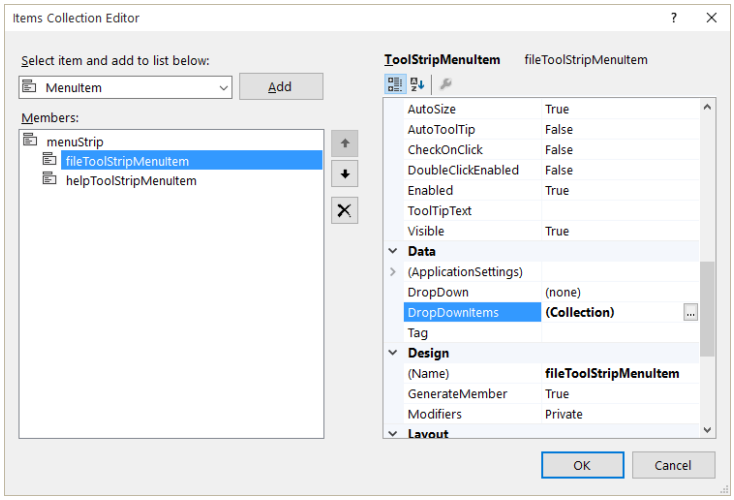
Drag and drop a menuStrip . Add a new menu item and if double clicked an event handler will be created.

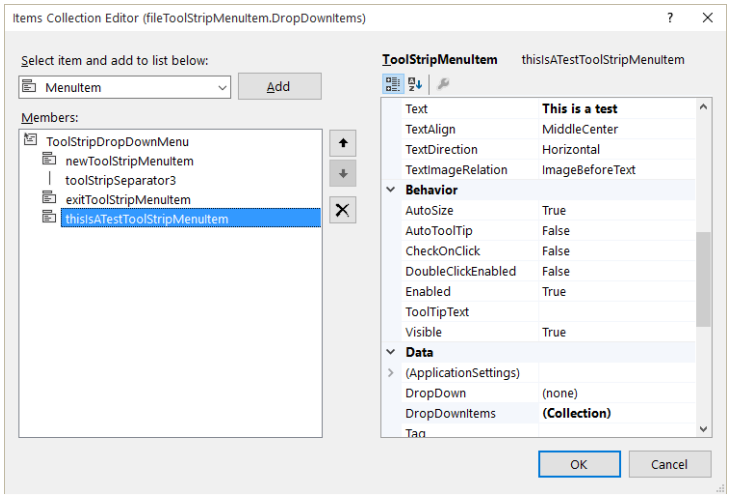


Edit the menuStrips from the properties menu. Select items (Collection) button.

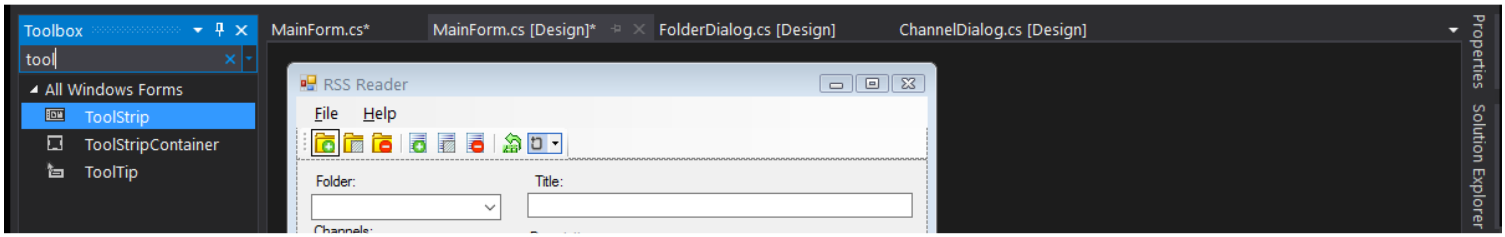


Pop up Items Collection Editor, displaying the menu for editing.

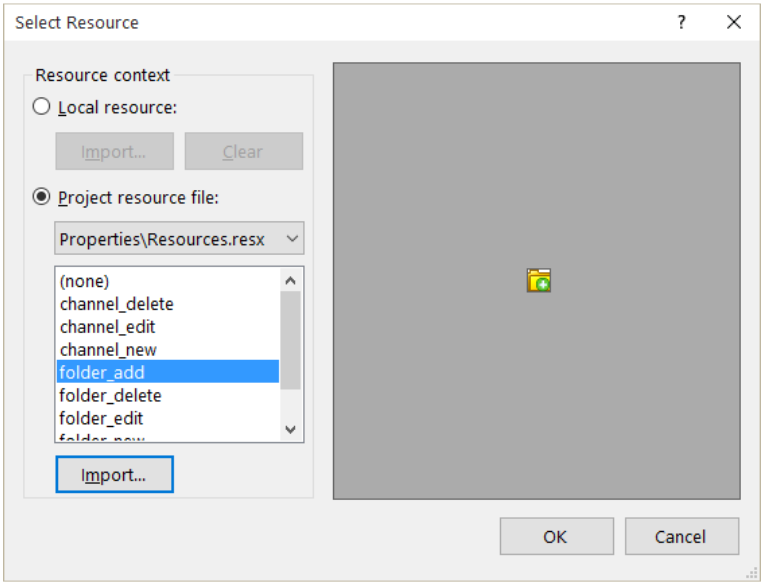




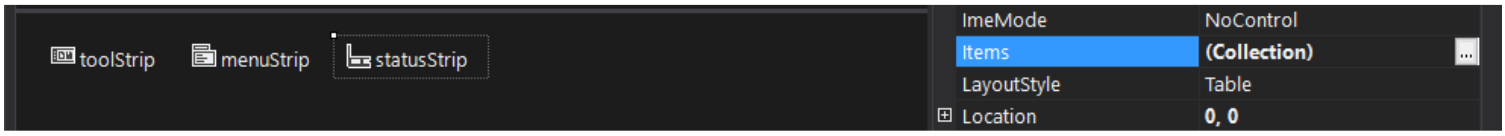
Drag and drop a toolStrip and add seven buttons, folder plus, edit and delete. Channel plus edit and delete and the seventh button RSS feed refresh.

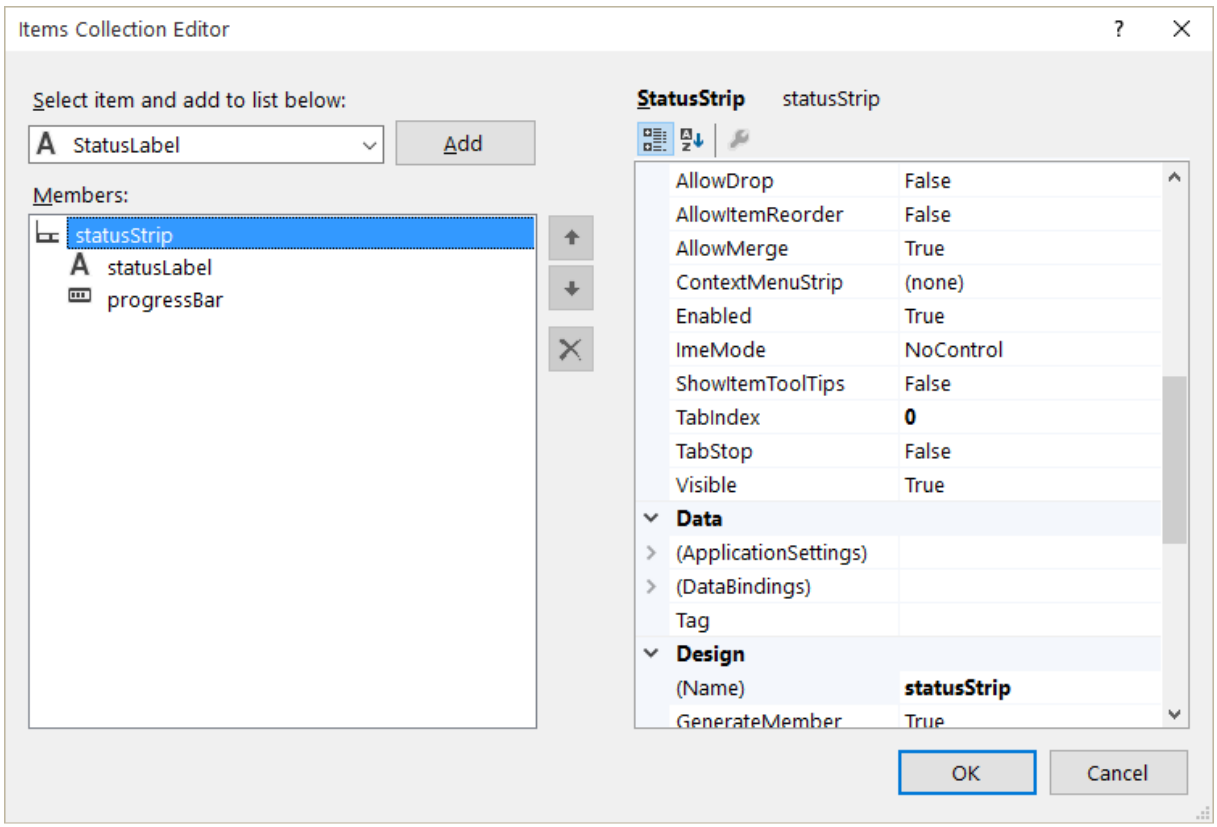


Right click buttons in the toolStrip menu and set image from the pop up select RSS icon image

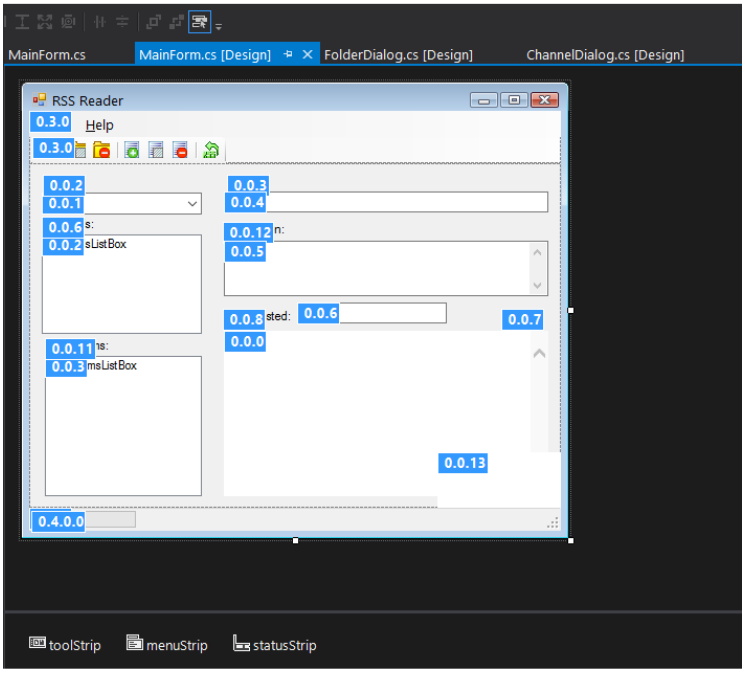


At the very bottom is a status strip to give the user feedback on what is being downloaded. Add status label and progress bar the status label will be used to give the user a message on where the feed has been downloaded or not, and the progress bar will show that the feed is being downloaded.

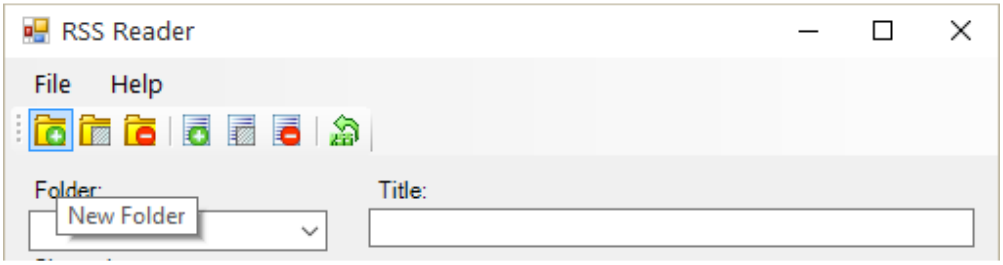


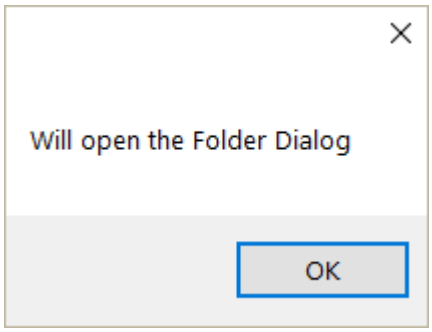


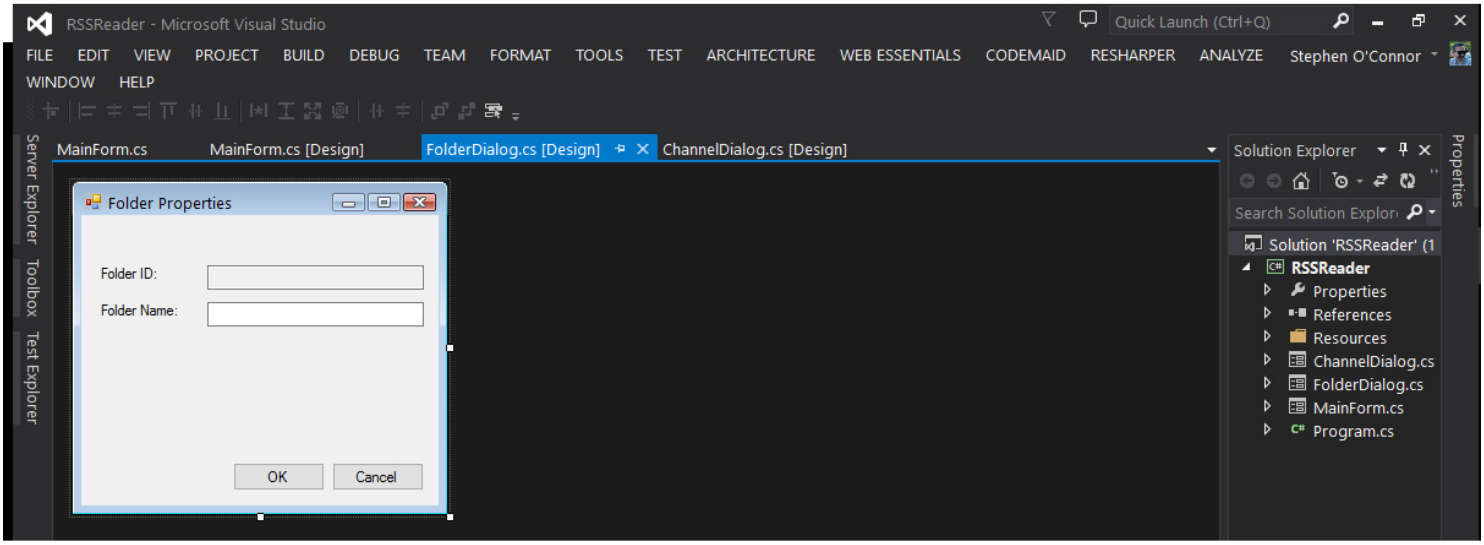
Tab order sets the order of the tabs.



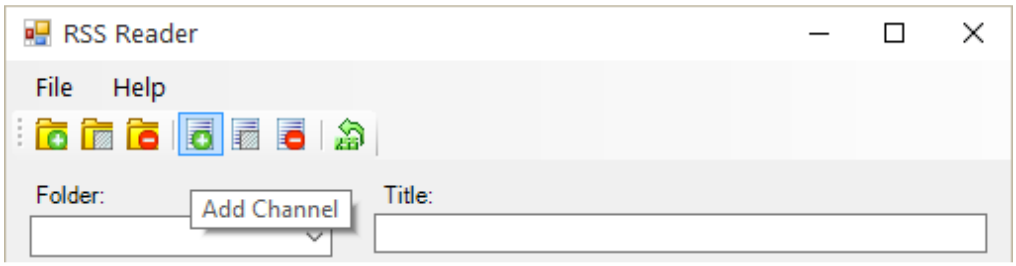
FolderDialog.cs
Click new folder icon will open a pop-up, however it will open FolderDialog.cs

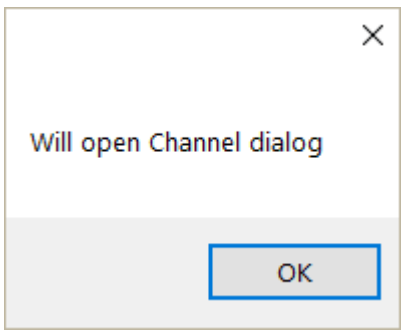




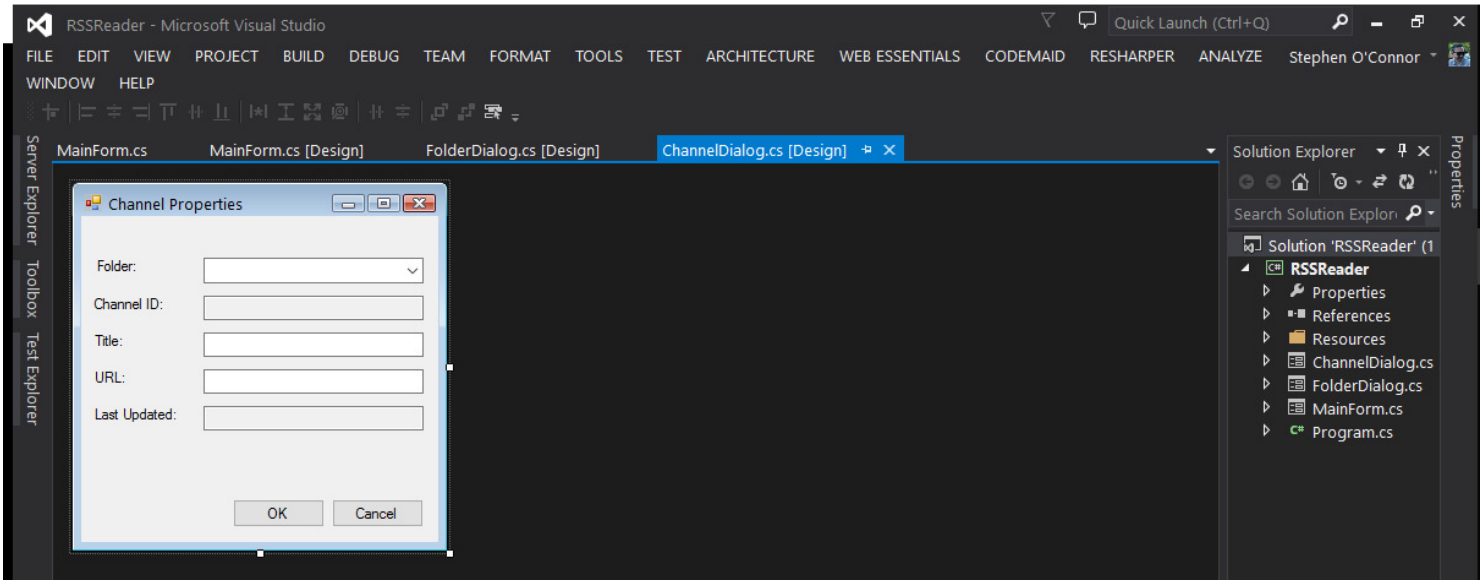


ChannelDialog.cs
Click new folder icon will open a pop-up, however it will open ChannelDialog.cs





C# Database application



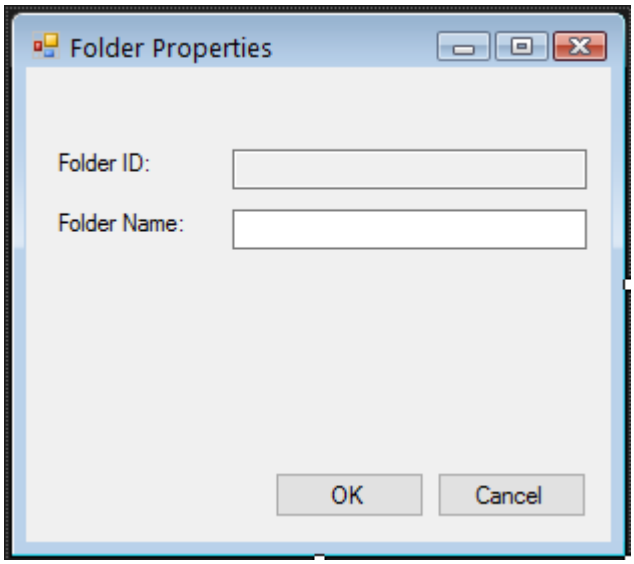
Add the form code to MainForm.cs
Do not copy paste, please type code.

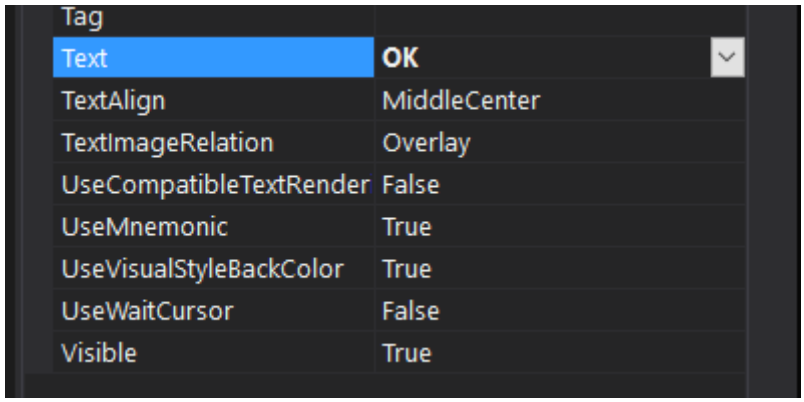
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace RSSReader
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private void newFolderToolStripMenuItem_Click(object sender, EventArgs e)
        {
            // TODO: Will call the 'New Folder' helper method
            MessageBox.Show("Will create a New Folder");
        }

        private void newChannelToolStripMenuItem_Click(object sender, EventArgs e)
        {
            // TODO: Will call the 'New Channel' helper method
            MessageBox.Show("Will create a New Channel");
        }

        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void contentsToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            // TODO: Will open the Help Contents
            MessageBox.Show("Will open the Help Contents");
        }

        private void indexToolStripMenuItem_Click(object sender, EventArgs e)
        {
            // TODO: Will open the Help Index
            MessageBox.Show("Will open the Help Index");
        }

        private void searchToolStripMenuItem_Click(object sender, EventArgs e)
        {
            // TODO: Will open the Help Search
            MessageBox.Show("Will open the Help Search");
        }

        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            // TODO: Will open the About Dialog
            MessageBox.Show("Will open the About Dialog");
        }

        private void editFolderToolstripButton_Click(object sender, EventArgs e)
        {
            // TODO: Will open the Folder Dialog
            MessageBox.Show("Will open the Folder Dialog");
        }

        private void addFolderToolstripButton_Click(object sender, EventArgs e)
        {
            // TODO: Will open the Folder Dialog
            MessageBox.Show("Will open the Folder Dialog");
        }

        private void deleteFolderToolStripButton_Click(object sender, EventArgs e)
        {
            // TODO: Will delete current folder.
            MessageBox.Show("Will Delete current Folder");
        }

        private void addChannelToolstripButton_Click(object sender, EventArgs e)
        {
            // TODO: Will open folder dialog.
            MessageBox.Show("Will open Channel dialog");
        }

        private void editChannelToolstripButton_Click(object sender, EventArgs e)
        {
            // TODO: Will open folder dialog.
            MessageBox.Show("Will open Channel dialog");
        }

        private void deleteChannelToolstripButton_Click(object sender, EventArgs e)
        {
```

```
            // TODO: Delete selected Channel
            MessageBox.Show("Will delete channel");
        }

        private void refreshRSSFeed_Click(object sender, EventArgs e)
        {
            // TODO: Refresh RSS Feed
            MessageBox.Show("Will referesh RSS feed");
        }
    }
}
```

Setting up the folders, get some of the easy things out of the way. Get the flow right. Open the Folder form and right click on both OK and Cancel buttons go to properties and in Text select OK for the OK button. For the cancel button select Cancel. Do the same for the Channel form.



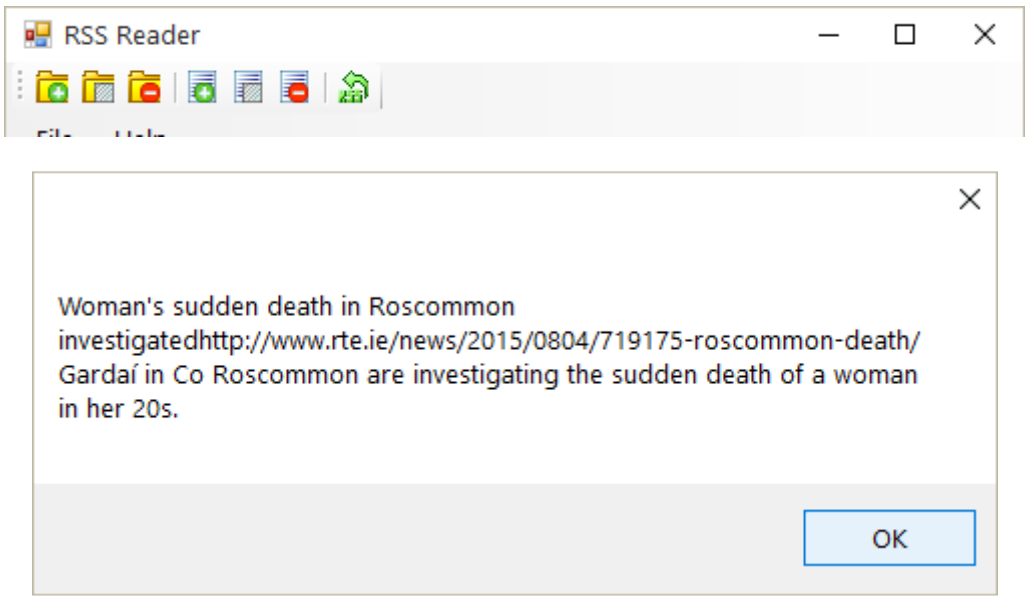


For the add new folder, edit folder, add new channel and edit channel add the follow code.

```
        private void editFolderToolstripButton_Click(object sender, EventArgs e)
        {
            // create an instance of FolderDialog
            FolderDialog folderDia = new FolderDialog();

            // has to work with the current form
            DialogResult result = folderDia.ShowDialog();

            // did the user click the OK button
            if (result == DialogResult.OK)
            {
                MessageBox.Show("User selected OK");
            }
            else
            {
                MessageBox.Show("User selected Cancel");
            }
        }

        private void addFolderToolstripButton_Click(object sender, EventArgs e)
        {
            // create an instance of FolderDialog
```

```
            FolderDialog folderDia = new FolderDialog();

            DialogResult result = folderDia.ShowDialog();

            if (result == DialogResult.OK)
            {
                MessageBox.Show("User selected OK");
            }
            else
            {
                MessageBox.Show("User selected Cancel");
            }
        }

        private void deleteFolderToolStripButton_Click(object sender, EventArgs e)
        {
            // TODO: Will delete current folder.
            MessageBox.Show("Will Delete current Folder");
        }

        private void addChannelToolstripButton_Click(object sender, EventArgs e)
        {
            // create an instance of ChannelDialog
            ChannelDialog channelDia = new ChannelDialog();

            DialogResult result = channelDia.ShowDialog();

            if (result == DialogResult.OK)
            {
                MessageBox.Show("User clicked OK");
            }
            else
            {
                MessageBox.Show("User clicked Cancel");
            }
        }

        private void editChannelToolstripButton_Click(object sender, EventArgs e)
        {
            // create an instance of ChannelDialog
            ChannelDialog channelDia = new ChannelDialog();

            DialogResult result = channelDia.ShowDialog();

            if (result == DialogResult.OK)
            {
                MessageBox.Show("User clicked OK");
            }
            else
            {
                MessageBox.Show("User clicked Cancel");
            }
        }
```

RSS

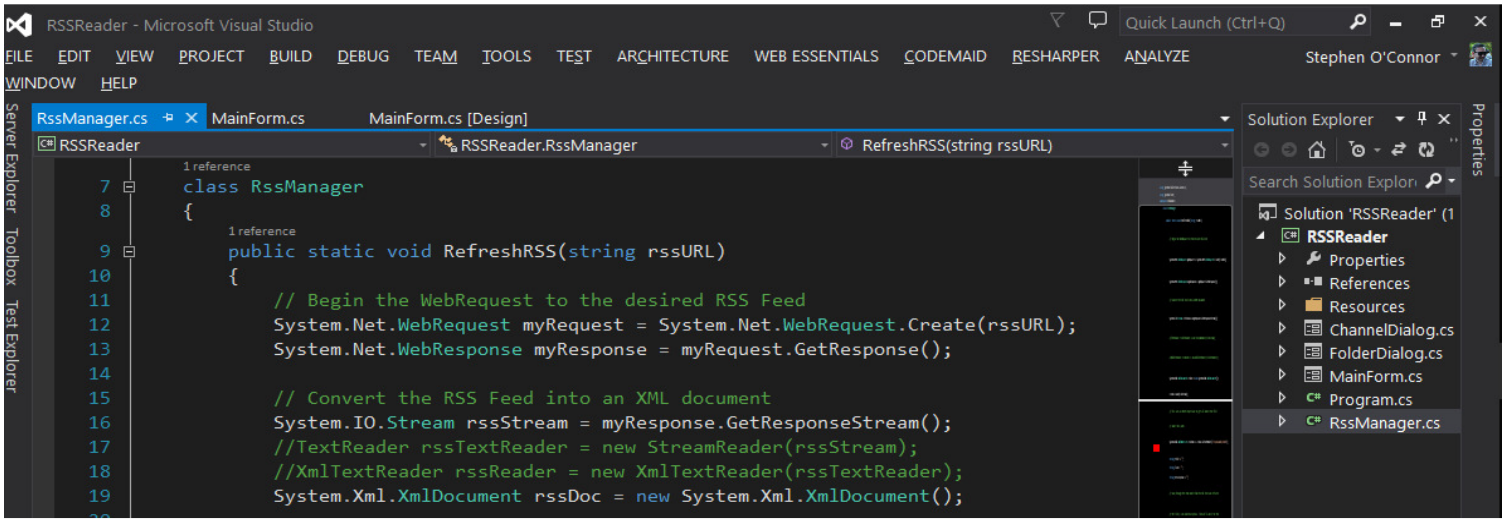



```
        private void refreshRSSFeed_Click(object sender, EventArgs e)
        {
            string rssURL = "http://www.rte.ie/news/rss/news-headlines.xml";

            // Begin the WebRequest to the desired RSS Feed
            System.Net.WebRequest myRequest = System.Net.WebRequest.Create(rssURL);
            System.Net.WebResponse myResponse = myRequest.GetResponse();

            // Convert the RSS Feed into an XML document
            System.IO.Stream rssStream = myResponse.GetResponseStream();
            //TextReader rssTextReader = new StreamReader(rssStream);
            //XmlTextReader rssReader = new XmlTextReader(rssTextReader);
            System.Xml.XmlDocument rssDoc = new System.Xml.XmlDocument();

            rssDoc.Load(rssStream);


            // This uses an XPath expression to get all nodes that fall
            // under this path.
            System.Xml.XmlNodeList rssItems =
rssDoc.SelectNodes("rss/channel/item");

            string title = "";
            string link = "";
            string description = "";

            // Loop through the Item nodes from the RSS Feed and retrieve
            // the Title, Link and Description.  Then we'll add it to the
            // database (if not already present.)

            for (int i = 0; i < rssItems.Count; i++)
            {
                System.Xml.XmlNode rssDetail;

                rssDetail = rssItems.Item(i).SelectSingleNode("title");
                if (rssDetail != null)
                {
```

```
                    title = rssDetail.InnerText;
                }
                else
                {
                    title = "";
                }

                rssDetail = rssItems.Item(i).SelectSingleNode("link");
                if (rssDetail != null)
                {
                    link = rssDetail.InnerText;
                }
                else
                {
                    link = "";
                }

                rssDetail = rssItems.Item(i).SelectSingleNode("description");
                if (rssDetail != null)
                {
                    description = rssDetail.InnerText;
                }
                else
                {
                    description = "";
                }

                // TODO: Check to make sure this news Item isn't already in the
database

                // TODO: Add this newsItem to the database

                // TODO: Remove this test

                MessageBox.Show(title + link + description);
            }
        }
```

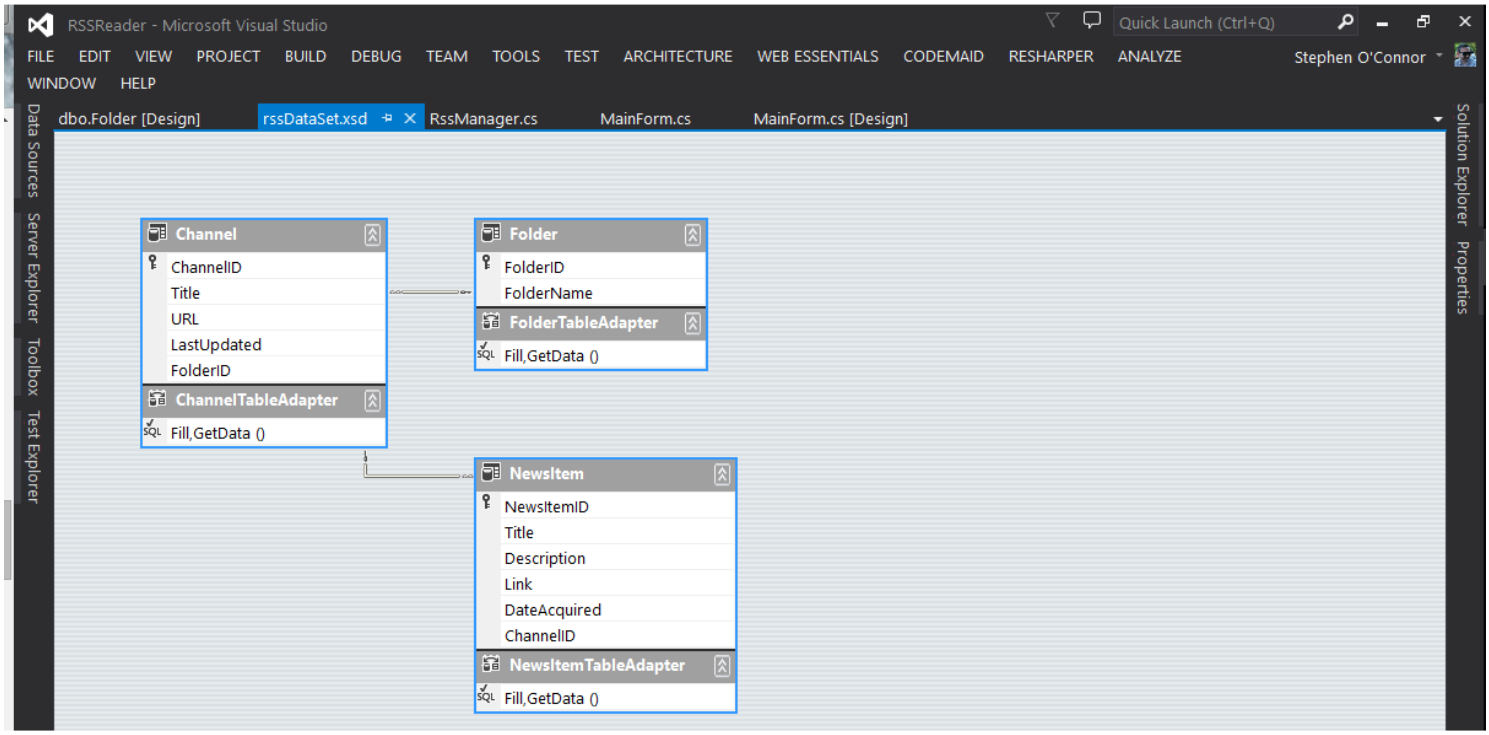
C# Database application

Create a class RssManager.cs copy paste all. Creating a loosely coupled application. Create a method `RefreshRSS` passing in a string variable rssURL. Create `RefreshRSS` as a static method, so that an instance does not have to be created.




```
private void refreshRSSFeed_Click(object sender, EventArgs e)
      {
          string rssURL = "http://www.rte.ie/news/rss/news-headlines.xml";

          RssManager.RefreshRSS(rssURL);
      }
```

Setting up the database.



My Set-up

C# Database application