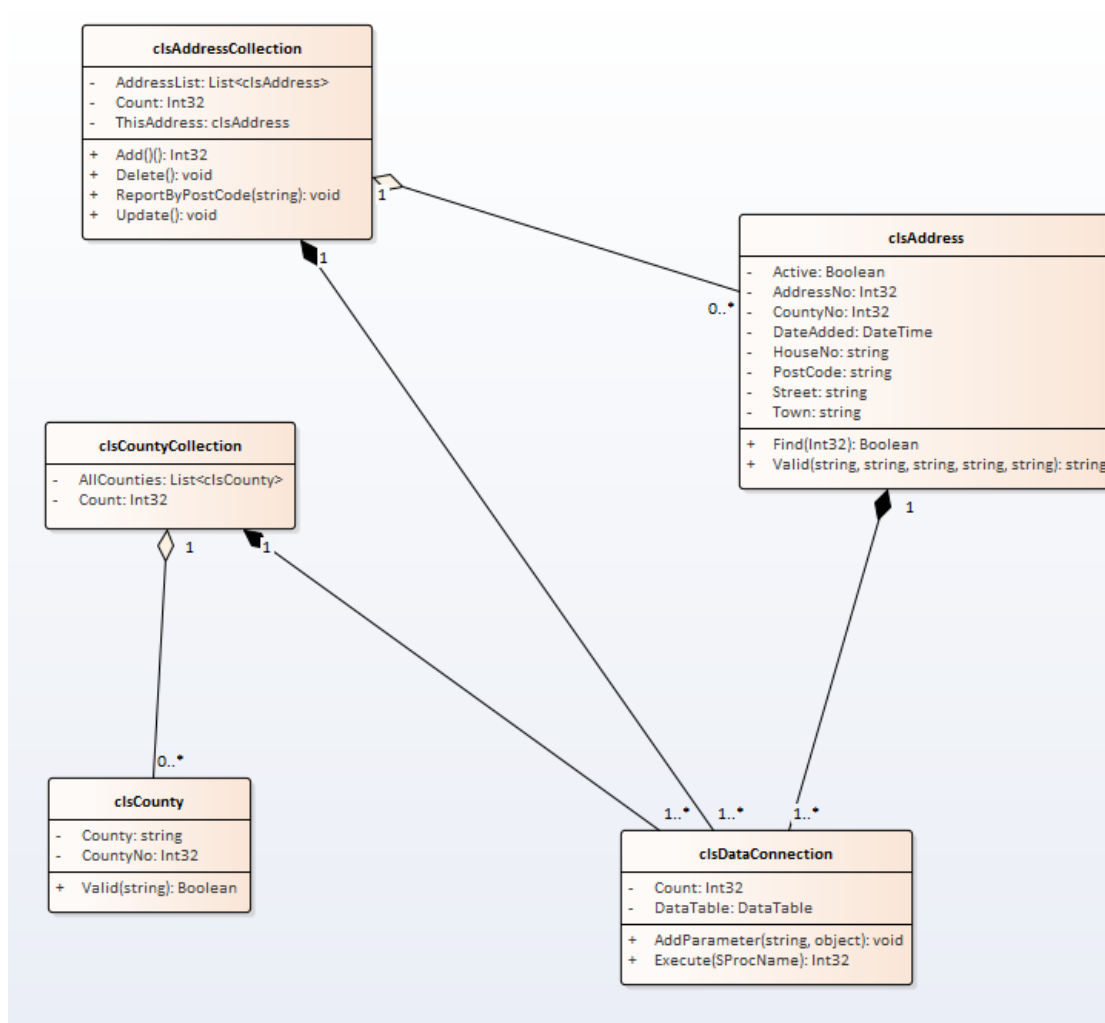


Setting up the Drop-Down List

In creating the system to this point the class diagram looks something like this...



We have a collection class containing functions for managing address. We have a data storage class allowing us to represent single rows in the table. We also have a data connection class allowing us to connect to the database.

There is however a part of the system that remains uncompleted that is the drop down list of counties...

House No	<input type="text"/>	County	<input type="text" value="Unbound"/>
Street	<input type="text"/>	Date Added	<input type="text"/>
Town	<input type="text"/>		<input type="checkbox"/> Active
Post Code	<input type="text"/>		
		OK	Cancel

[!Error]

In order to make this feature work we will need the following items in place.

- A table to store the county data
- A stored procedure selecting all of the records from this table
- A collection class allowing us to manage the data in the table
- A suitable class allowing us to model the data in the table
- An array list to create a public collection of items
- Suitable presentation layer code to populate the drop down list

Creating the Table

The table definition for the table tblCounty is as follows...

```
CREATE TABLE [dbo].[tblCounty]
(
    [CountyNo] INT NOT NULL PRIMARY KEY IDENTITY,
    [County] VARCHAR(30) NULL
)
```

You will also need to add some data to this table.

Creating the Stored Procedure

As with the address book we need a stored procedure allowing us access to the data. A simple select query with no parameters should do the trick...

```

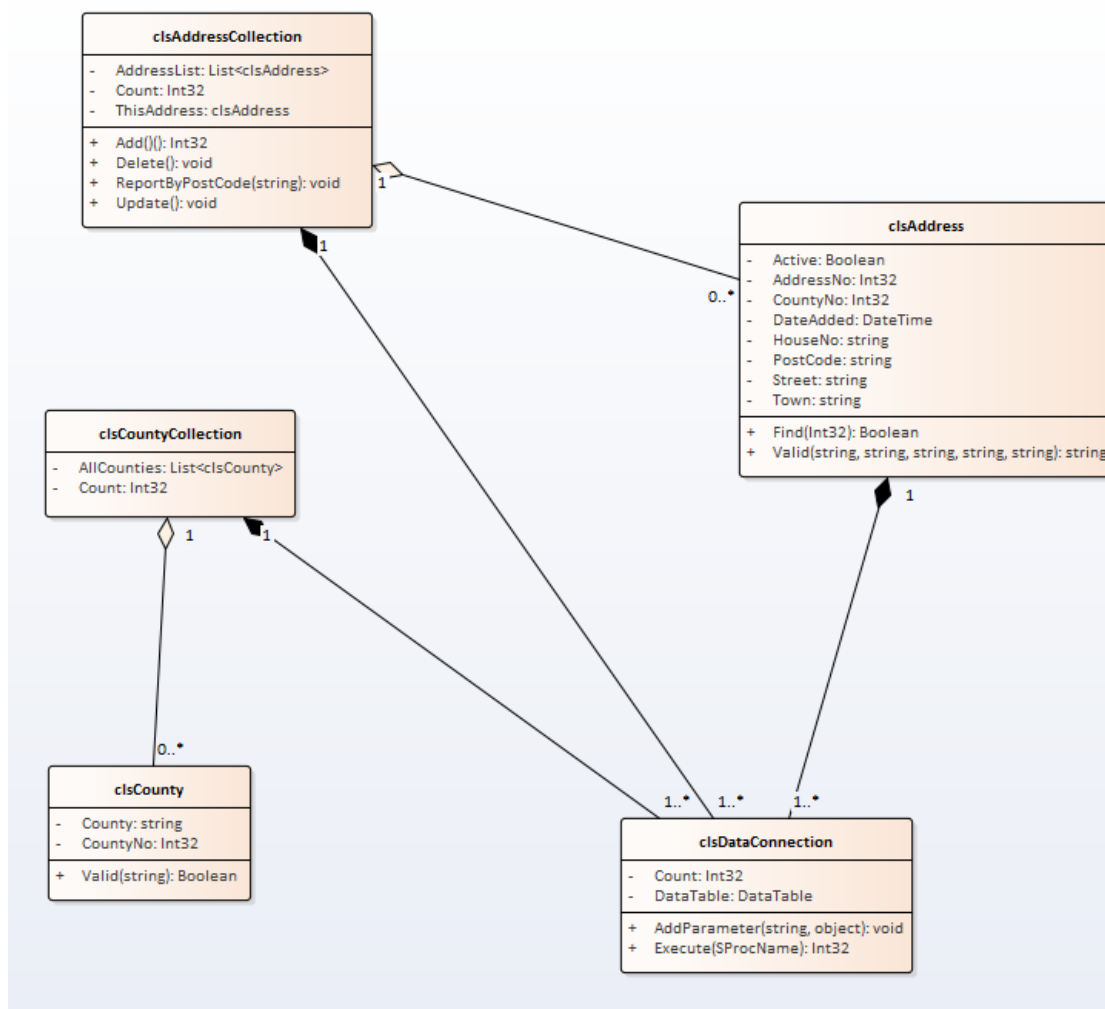
CREATE PROCEDURE sproc_tblCounty_SelectAll
AS
-- select all records from the county table
select * from tblCounty

```

Updating the Class Diagram

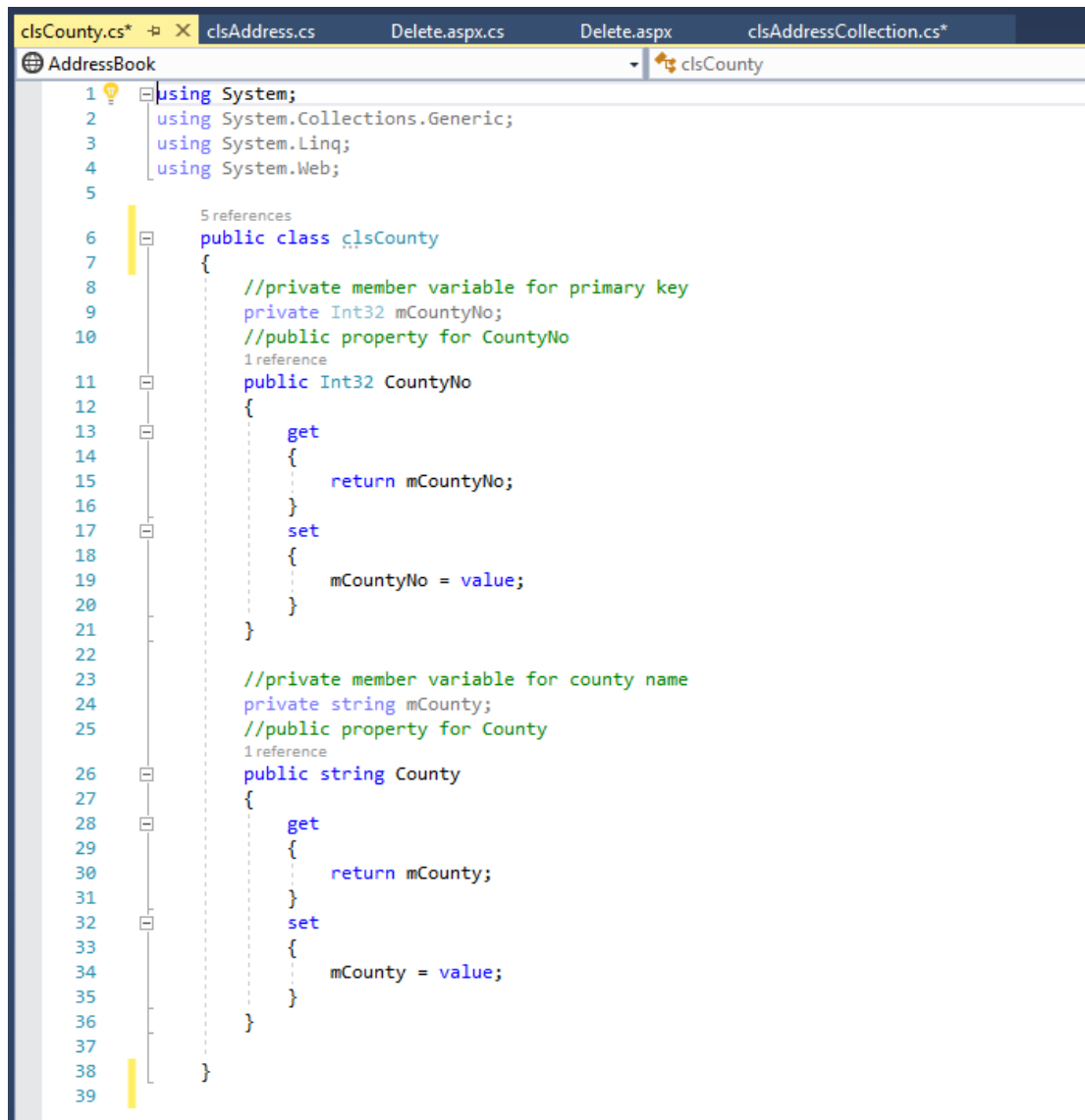
Now that we have the table and stored procedure in place we need to think about how the classes will work.

As with the address book we will need a collection class and a class to store each item.



We will keep things simple and set up the county collection with no methods and only the ability to generate an unfiltered array list of `clsCounty`.

To create the class `clsCounty` we need to use the following code...



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5
6 public class clsCounty
7 {
8     //private member variable for primary key
9     private Int32 mCountyNo;
10    //public property for CountyNo
11    public Int32 CountyNo
12    {
13        get
14        {
15            return mCountyNo;
16        }
17        set
18        {
19            mCountyNo = value;
20        }
21    }
22
23    //private member variable for county name
24    private string mCounty;
25    //public property for County
26    public string County
27    {
28        get
29        {
30            return mCounty;
31        }
32        set
33        {
34            mCounty = value;
35        }
36    }
37 }
38
39
```

To create the collection class we need to use the following code in clsCountyCollection...

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

1 reference
public class clsCountyCollection
{
    //create a connection to the database with class level scope
    clsDataConnection Counties = new clsDataConnection();

    //constructor
    0 references
    public clsCountyCollection()
    {
        //execute the select all query
        Counties.Execute("sproc_tblCounty_SelectAll");
    }

    //this read only function gives us the count property
    0 references
    public Int32 Count
    {
        get
        {
            //return the count of the data from the database
            return Counties.Count;
        }
    }

    //this readonly function exposes the query results collection
    0 references
    public List<clsCounty> AllCounties
    {
        get
        {
            //create a instance of a list called mAllCounties
            List<clsCounty> mAllCounties = new List<clsCounty>();
            //var to store the index for the loop
            Int32 Index = 0;
            //while the index is less that the number of records to process
            while (Index < Counties.Count)
            {
                //set up the new entry to be added to the list
                clsCounty NewCounty = new clsCounty();
                //get the county number from the database
                NewCounty.CountyNo = Convert.ToInt32(Counties.DataTable.Rows[Index]["CountyNo"]);
                //get teh county name from the database
                NewCounty.County = Convert.ToString(Counties.DataTable.Rows[Index]["County"]);
                //add the new entry to the list
                mAllCounties.Add(NewCounty);
                //increment the index to the next record
                Index++;
            }
            //return the query results from the database
            return mAllCounties;
        }
    }
}

```

Constructors

Notice in the above code that we are using a constructor for the class.

```

//constructor
0 references
public clsCountyCollection()
{
    //execute the select all query
    Counties.Execute("sproc_tblCounty_SelectAll");
}

```

A constructor is a special kind of function that always runs when we create an instance of the class. We use the constructor to initialise the class.

By giving the declaration of the object Counties class level scope i.e. creating it at the top of the class...

```

//constructor
public clsCountyCollection()
{
    //execute the select all query
    Counties.Execute("sproc_tblCounty_SelectAll");
}

```

It makes this object available to all functions in the class.

By placing the initialisation in the constructor it removes the need to initialise the object ourselves as it is done automatically every time we create an instance of an object.

Creating the presentation layer code

In order to display the data from the array list in the drop down list we need to add another function to the code for AnAddress.aspx...

```

//this function is used to populate the counties drop down list
Int32 DisplayCounties()
{
    //create an instance of the county class
    clsCountyCollection Counties = new clsCountyCollection();
    //var to store the county number primary key
    string CountyNo;
    //var to store the name of the county
    string County;
    //var to store the index for the loop
    Int32 Index = 0;
    //while the index is less that the number of records to process
    while (Index < Counties.Count)
    {
        //get the county number from the database
        CountyNo = Convert.ToString(Counties.AllCounties[Index].CountyNo);
        //get teh county name from the database
        County = Counties.AllCounties[Index].County;
        //set up the new row to be added to the list
        ListItem NewCounty = new ListItem(County, CountyNo);
        //add the new row to the list
        ddlCounty.Items.Add(NewCounty);
        //increment the index to the next record
        Index++;
    }
    //return the number of records found
    return Counties.Count;
}

```

We also need to modify the load event so that the above function is called whenever the page is loaded...

```

protected void Page_Load(object sender, EventArgs e)
{
    //copy the data from the query string to the variable
    AddressNo =Convert.ToInt32(Request.QueryString["AddressNo"]);
    if (IsPostBack != true)
    {
        //update the contents of the drop down list
        DisplayCounties();
        //if the AddressNo is not -1 then display the data from the record
        if (AddressNo != -1)
        {
            //display the data for this address
            DisplayAddress(AddressNo);
        }
    }
}

```

The drop down list should now be populated with data.

County

Date Added

OK

C

Avon

Bedfordshire

Berkshire

Buckinghamshire

Cambridgeshire

Cambridgeshire and Isle of Ely

Cheshire

City of Bristol

City of London

Cleveland

Cornwall

Cumberland

Cumbria

Derbyshire

Devon

Dorset

Durham

East Suffolk

East Sussex

Essex

Gloucestershire

Greater London

Greater Manchester

Hereford and Worcester

Herefordshire

Hertfordshire

Humberside

Huntingdon and Peterborough

Huntingdonshire

Isle of Ely

The last thing to do is make sure that when opening an existing record the drop down list points to the right record...

```
//this function displays the data for an address on the web form
void DisplayAddress(Int32 AddressNo)
{
    //create an instance of the addresses class
    clsAddress MyAddressBook = new clsAddress();
    //find the record we want to display
    MyAddressBook.Find(AddressNo);
    //display the house no
    txtHouseNo.Text = MyAddressBook.HouseNo;
    //display the street
    txtStreet.Text = MyAddressBook.Street;
    //display the town
    txtTown.Text = MyAddressBook.Town;
    //display the post code
    txtPostCode.Text = MyAddressBook.PostCode;
    //display the data added
    txtDateAdded.Text = MyAddressBook.DateAdded.ToString("dd/MM/yyyy");
    //set the drop down list to display the county
    ddlCounty.SelectedValue = Convert.ToString(MyAddressBook.CountyCode);
    //set the drop down list to display the county
    ddlCounty.SelectedValue = Convert.ToString(MyAddressBook.CountyCode);
    //display the active state
    chkActive.Checked = MyAddressBook.Active;
}
```


And make sure that upon pressing save the selected value of the drop down list is written to the middle layer from the presentation layer...

```
protected void btnOK_Click(object sender, EventArgs e)
{
    //create an instance of the address page class
    clsAddress ThisAddress = new clsAddress();
    //var to store any error messages
    string ErrorMessage;
    //test the data on the web form
    ErrorMessage = ThisAddress.AddressValid(txtHouseNo.Text,
                                           txtStreet.Text,
                                           txtTown.Text,
                                           txtPostCode.Text,
                                           txtDateAdded.Text);

    //if there is no error message
    if (ErrorMessage == "")
    {
        //create a new instance of the address book class
        clsAddressCollection AddressBook = new clsAddressCollection();
        //do something with the data - insert or update
        //
        //if the Address Number is -1
        if (AddressNo == -1)
        {
            //copy the data from the interface to the object
            ThisAddress.HouseNo = txtHouseNo.Text;
            ThisAddress.Street = txtStreet.Text;
            ThisAddress.Town = txtTown.Text;
            ThisAddress.PostCode = txtPostCode.Text;
            ThisAddress.CountyCode = Convert.ToInt32(ddlCounty.SelectedValue);
            ThisAddress.DateAdded = Convert.ToDateTime(txtDateAdded.Text);
            //add the new record
            AddressBook.Add(ThisAddress);
        }
    }
}
```

```
else
{
    //this is an existing record
    //copy the data from the interface to the object
    ThisAddress.AddressNo = Convert.ToInt32(AddressNo);
    ThisAddress.HouseNo = txtHouseNo.Text;
    ThisAddress.Street = txtStreet.Text;
    ThisAddress.Town = txtTown.Text;
    ThisAddress.PostCode = txtPostCode.Text;
    ThisAddress.CountyCode = Convert.ToInt32(ddlCounty.SelectedValue);
    ThisAddress.DateAdded = Convert.ToDateTime(txtDateAdded.Text);
    //update the existing record
    AddressBook.UpdateAddress(ThisAddress);
}
//redirect back to the main page
Response.Redirect("Default.aspx");
}
else
{
    //display the error message
    lblError.Text = ErrorMessage;
}
```