

Completing Add Edit and Delete

There are still a number of problems with the program.

When we click an entry in the list we don't want to see a blank page...

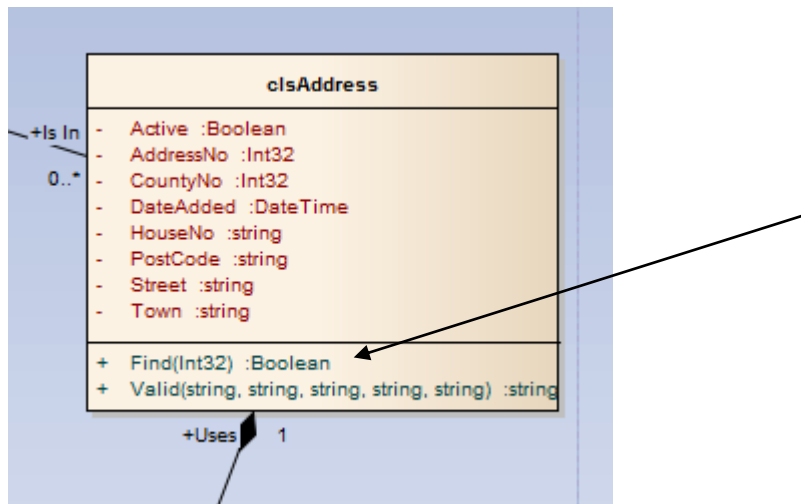
AddressNo	<input type="text" value="3"/>		
House No	<input type="text"/>	County	<input type="button" value="v"/>
Street	<input type="text"/>	Date Added	<input type="text"/>
Town	<input type="text"/>	<input type="checkbox"/>	Active
Post Code	<input type="text"/>		
<div><input type="button" value="OK"/> <input type="button" value="Cancel"/></div>			

We want to see the data we want to edit...

House No	<input type="text" value="33"/>	County	<input type="text" value="Leicestershire"/> v
Street	<input type="text" value="High Street"/>	Date Added	<input type="text" value="07/08/2012"/>
Town	<input type="text" value="Leicester"/>	<input checked="" type="checkbox"/>	Active
Post Code	<input type="text" value="LE1 6FG"/>		
<div><input type="button" value="OK"/> <input type="button" value="Cancel"/></div>			

To make this work we need to make further changes to the middle layer class.

Creating the Find Method



The find method will allow us to pass a primary key value as a parameter. The method will return true or false based on if the record was found or not. We will need two things in place.

1. A suitable stored procedure that looks up records based on the primary key value
2. A middle layer function in clsAddress containing the code for the Find method

Creating the Stored Procedure

In the data layer create the following stored procedure...

```
CREATE PROCEDURE sproc_tblAddress_FilterByAddressNo
    --parameter to identify the record we are looking for
    @AddressNo int
AS
    --select all records with this primary key (there should only be zero or one)
    select * from tblAddress where AddressNo = @AddressNo
RETURN 0
```

Creating the Find Method

In clsAddress add the following function defining the find method...

```

//function for the find public method
public Boolean Find(Int32 AddressNo)
{
    //initialise the DBConnection
    clsDataConnection dBConnection = new clsDataConnection();
    //add the address no parameter
    dBConnection.AddParameter("@AddressNo", AddressNo);
    //execute the query
    dBConnection.Execute("sproc_tblAddress_FilterByAddressNo");
    //if the record was found
    if (dBConnection.Count == 1)
    {
        //get the address no
        mAddressNo = Convert.ToInt32(dBConnection.DataTable.Rows[0]["AddressNo"]);
        //get the house no
        mHouseNo = Convert.ToString(dBConnection.DataTable.Rows[0]["HouseNo"]);
        //get the street
        mStreet = Convert.ToString(dBConnection.DataTable.Rows[0]["Street"]);
        //get the town
        mTown = Convert.ToString(dBConnection.DataTable.Rows[0]["Town"]);
        //get the post code
        mPostCode = Convert.ToString(dBConnection.DataTable.Rows[0]["PostCode"]);
        //get the county code
        mCountyCode = Convert.ToInt32(dBConnection.DataTable.Rows[0]["CountyCode"]);
        //get the date added
        mDateAdded = Convert.ToDateTime(dBConnection.DataTable.Rows[0]["DateAdded"]);
        //get the active state
        try
        {
            mActive = Convert.ToBoolean(dBConnection.DataTable.Rows[0]["Active"]);
        }
        catch
        {
            mActive = true;
        }
        //return success
        return true;
    }
    else
    {
        //return failure
        return false;
    }
}

```

This function will use the stored procedure to find the record specified by the primary key and set the private member variables with the data from the database.

Now modify the load event of the form AnAddress.aspx to make use of the new method.

It is best to create this code in its own function in this case DisplayAddress which accepts a single parameter but doesn't return a value.

```

protected void Page_Load(object sender, EventArgs e)
{
    //copy the data from the query string to the text box
    txtAddressNo.Text = Convert.ToString(Request.QueryString["AddressNo"]);
    //if this is not an instruction to add a new record
    if (txtAddressNo.Text != "-1")
    {
        //display the existing data
        DisplayAddress(Convert.ToInt32(txtAddressNo.Text));
    }
}

//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);
    //display the active state
    chkActive.Checked = MyAddressBook.Active;
}

```

Try editing a record to see if it works.

What you should find is that it appears to work but the data is not being updated in the table, why?

The problem is again with the load event.

REMEMBER the load event runs every time an event is triggered.

- We load the form
- The data is displayed (the load event is triggered)
- We edit the data
- We press save
- The data is displayed (the load event is triggered) *
- The data is saved

As above, we only want the data to be displayed the first time the form is initialised.

We do not want the step marked * to trigger again!

To fix this we need to make use of IsPostBack like so...

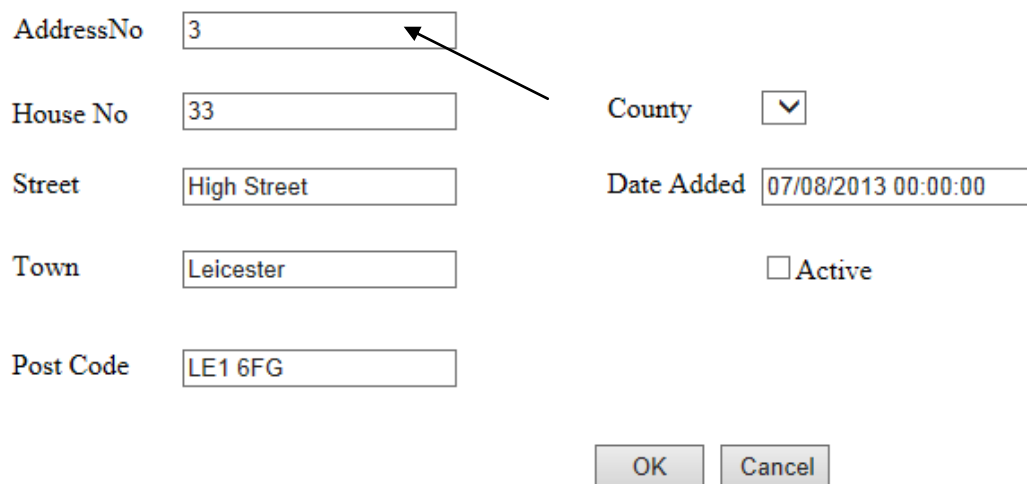
```
protected void Page_Load(object sender, EventArgs e)
{
    //copy the data from the query string to the variable
    AddressNo = Convert.ToInt32(Request.QueryString["AddressNo"]);
    //if the AddressNo is not -1 then display the data from the record
    if (AddressNo != -1)
    {
        if (IsPostBack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

If the code is working you should be able to add and edit records.

Tidying up the Interface

There is another point that is quite clunky.

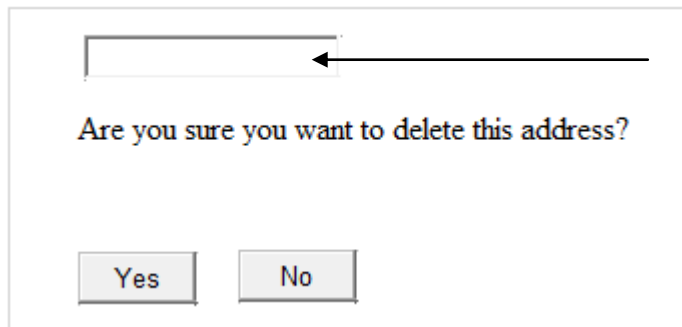
It isn't very good that the user sees the primary key value on the data entry form.



The screenshot shows a data entry form with the following fields and controls:

- AddressNo**: Text input field containing the value "3". An arrow points to this field.
- House No**: Text input field containing the value "33".
- Street**: Text input field containing the value "High Street".
- Town**: Text input field containing the value "Leicester".
- Post Code**: Text input field containing the value "LE1 6FG".
- County**: Dropdown menu with a downward arrow icon.
- Date Added**: Text input field containing the value "07/08/2013 00:00:00".
- Active**: A checkbox followed by the text "Active".
- Buttons**: "OK" and "Cancel" buttons at the bottom right.

Also on the delete form...



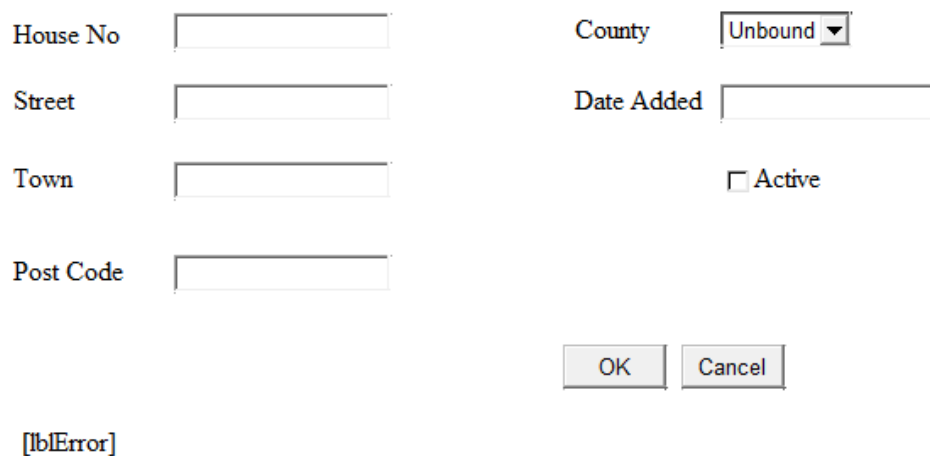
Are you sure you want to delete this address?

Yes No

Apart from looking unprofessional this actually creates a security problem.

There is nothing to stop a user accessing any record they like by typing values into this field. Also they may add records, delete records or make the system crash using invalid primary key values.

To fix this delete the text box and associated label so that the form looks like this...



House No

Street

Town

Post Code

County

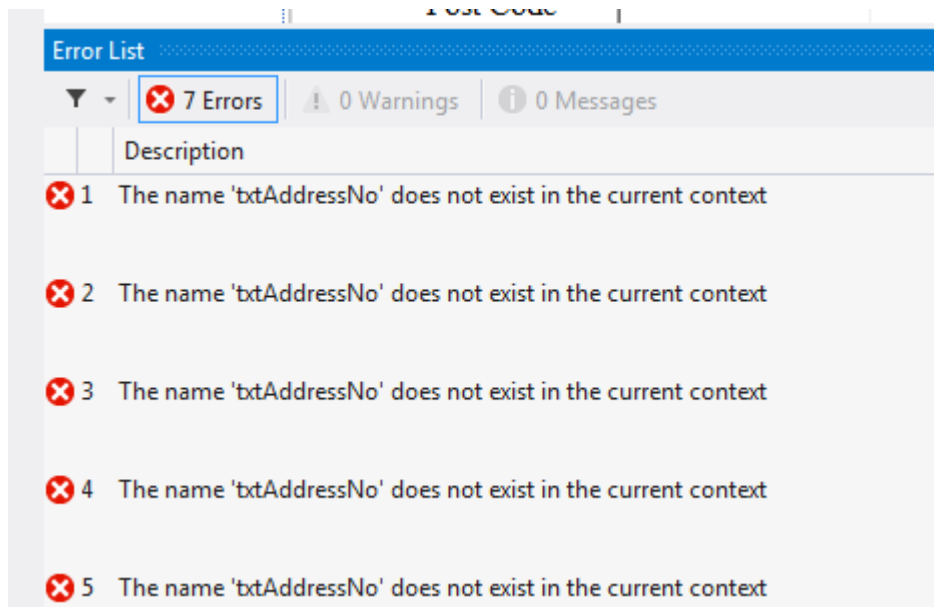
Date Added

☐ Active

OK Cancel

[lblError]

The only issue is that the program won't run. Try it and you should get compile errors.



Double click on one of the errors to see what the problem is...

```
protected void Page_Load(object sender, EventArgs e)
{
    //copy the data from the query string to the text box txtAddressNo
    txtAddressNo.Text = Request.QueryString["AddressNo"];
    //if the AddressNo is not -1 then display the data from the record
    if (txtAddressNo.Text != "-1")
    {
        if (IsPostBack != true)
        {
            //display the data for this address
            DisplayAddress(Convert.ToInt32(txtAddressNo.Text));
        }
    }
}
```

The problem is that we have deleted the text box which is used in the code. Since the text box no longer the code doesn't know what to do.

We may fix the problem with a variable.

Like so...

```
protected void Page_Load(object sender, EventArgs e)
{
    //var to store the AddressNo
    Int32 AddressNo;
    //copy the data from the query string to the variable
    AddressNo =Convert.ToInt32(Request.QueryString["AddressNo"]);
    //if the AddressNo is not -1 then display the data from the record
    if (AddressNo != -1)
    {
        if (IsPostBack != true)
        {
            //display the data for this address
            DisplayAddress(AddressNo);
        }
    }
}
```

And then we hit two problems...

Problem 1

The validation for ValidAddress wants to make use of address no. We need to remove that from the middle layer class.

Primary key values in a database system are usually read only so there is no point trying to write to them.

```
public string AddressValid(    string HouseNo,
                             string Street,
                             string Town,
                             string PostCode,
                             string DateAdded)
    ///this function is used to validate the data in a new address
```

You will need to modify both your code in the validation function and the call to the validation function in the presentation layer.

Problem 2

What do we replace these bits of code with?


```

//if the Address Number is -1
if (txtAddressNo.Text == "-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.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(txtAddressNo.Text);
    ThisAddress.HouseNo = txtHouseNo.Text;
    ThisAddress.Street = txtStreet.Text;
    ThisAddress.Town = txtTown.Text;
    ThisAddress.PostCode = txtPostCode.Text;
    ThisAddress.DateAdded = Convert.ToDateTime(txtDateAdded.Text);
    //update the existing record
    AddressBook.Update(ThisAddress);
}
//redirect back to the main page
Response.Redirect("Default.aspx");

```

Scope

What we really need to use is the variable AddressNo declared in the load event of the form.

```

protected void Page_Load(object sender, EventArgs e)
{
    //var to store the AddressNo
    Int32 AddressNo;
    //copy the data from the query string to the variable
    AddressNo = Convert.ToInt32(Request.QueryString["AddressNo"]);
    //if the AddressNo is not -1 then display the data from the record
    if (AddressNo != -1)
    {
        if (IsPostBack != true)
        {
            //display the data for this address
            DisplayAddress(AddressNo);
        }
    }
}

```

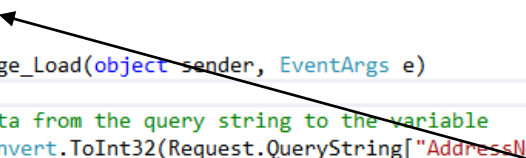
The problem is that the scope of this variable makes it available only to code in the load event.

To make it available to the entire form all we need to do is change its scope so that it is declared at the top of the form like so...

```
//var to store the AddressNo
Int32 AddressNo;

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 the AddressNo is not -1 then display the data from the record
    if (AddressNo != -1)
    {

```



It is now available to all functions in the form.

Using the example of the edit add form tidy up the interface for delete so that there is no text box for the address number.