Insert and Update and Building the Data Layer

We are now at the stage in developing our web application where we start to allow the user of the system to enter data into the table.

There are two considerations to keep in mind as we reach this stage.

1. Security

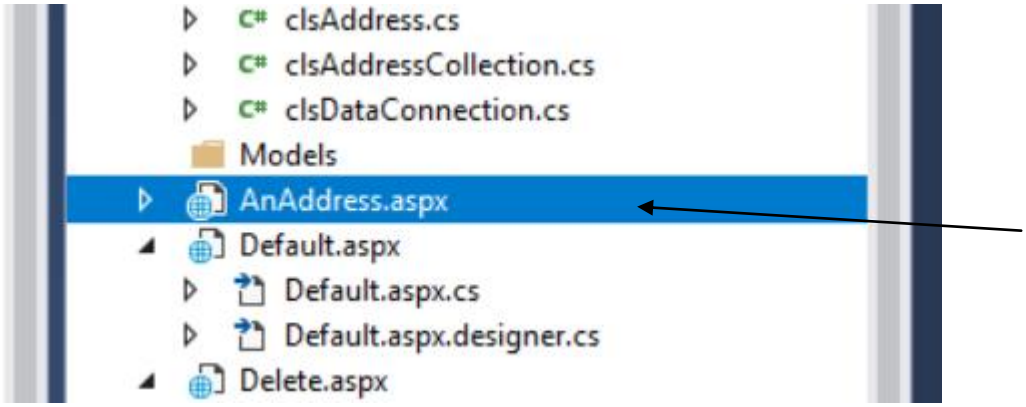
Creating a page which is “internet facing” creates the inevitable problem of unwanted attention. Along with the users of your system there will be plenty of other people out there interested in breaking into your system to obtain login details of users, bank details etc.

2. Validation

We also need to keep in mind that the legitimate users will also make mistakes while entering data into our system. Our system needs to provide useful feedback to them about what was wrong and also not crash when bad data is entered.

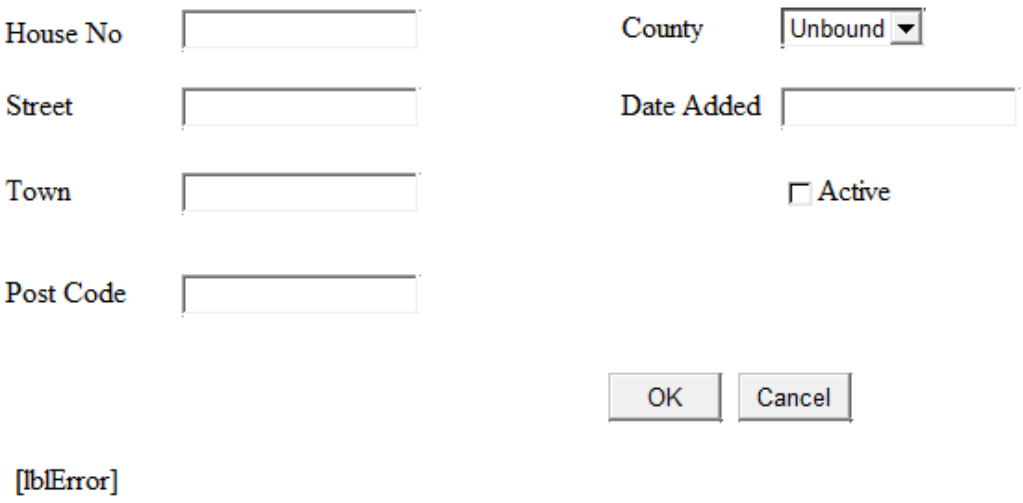
Thinking about the Data Entry Form

You have already created the presentation layer at the start of the module. Notice that you have a single web form to handle insert and update.



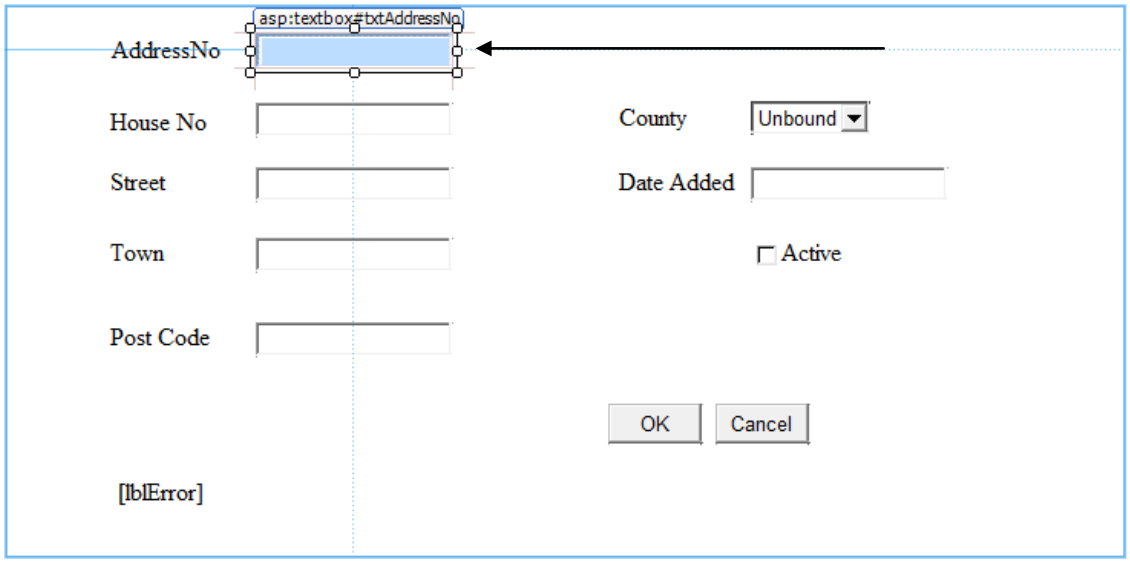
This is a much better policy (normally) as the functions surrounding insert and update tend to be very similar. Creating separate forms for insert and update tends to create a lot of duplicate functionality and produces hard to manage code.

The data entry for should look something like this...



Again as is common in programming we are going to tweak the design purely for testing purposes.

Add a new text box onto the existing form so that we may type the primary key value for the underlying data.



We will set up the data entry form so that it works as follows.

The key part of how this works is based on what value you enter in the Address No text box.

If you enter -1 into the text box the data in the page is inserted as a new record.

If you enter a number which is the primary key value of an existing record then the data is used to update that record.

(The page doesn't need to check if the primary key value is valid or not!)

Last of all make the data entry form the start up page for the web site.

Creating the Insert Stored Procedure

In order to insert a new record into a table using SQL we need a suitable stored procedure.

The query we want to create has the following structure…

```
CREATE PROCEDURE sproc_tblAddress_Insert
    --parameters for the query
    @HouseNo    VARCHAR (6),
    @Street     VARCHAR (50),
    @Town       VARCHAR (50),
    @PostCode   VARCHAR (9),
    @CountyCode INT,
    @DateAdded  DATE,
    @Active     BIT

AS

--insert the values stored in the parameters into the fields of the new record
INSERT INTO tblAddress
           ( HouseNo, Street, Town, PostCode, CountyCode, DateAdded, Active )
SELECT
           @HouseNo, @Street, @Town, @PostCode, @CountyCode, @DateAdded, @Active;

RETURN 0
```

The first section of the stored procedure defines the parameters needed to create the new record.

```
 --parameters for the query
 @HouseNo    VARCHAR (6),
 @Street     VARCHAR (50),
 @Town       VARCHAR (50),
 @PostCode   VARCHAR (9),
 @CountyCode INT,
 @DateAdded  DATE,
 @Active     BIT
```

The first part of the SQL states which table the data is being inserted into

```
INSERT INTO tblAddress
```

The next section lists the specific field that we want to write data to…

```
( HouseNo, Street, Town, PostCode, CountyCode, DateAdded, Active )
```

The next section states the data that is being used as the source of the data, in this case a set of parameters…

```
SELECT
            @HouseNo, @Street, @Town, @PostCode, @CountyCode, @DateAdded, @Active;
```

Notice that the order of parameters must match the order of the destination fields for the data…

HouseNo	Street	Town	PostCode	CountyCode	DateAdded	Active
@HouseNo	@Street	@Town	@PostCode	@CountyCode	@DateAdded	@Active

Save the query as sproc_tblAddress_Insert and test it to make sure that it adds data to the table.

Creating the Update Stored Procedure

The update query has the following structure…

```
CREATE PROCEDURE sproc_tblAddress_Update
    @AddressNo int,
    @HouseNo varchar (6),
    @Street varchar (50),
    @Town varchar (50),
    @PostCode varchar (9),
    @CountyCode int,
    @DateAdded date,
    @Active bit

    AS

    update tblAddress
    set HouseNo=@HouseNo,
        Street=@Street,
        Town=@Town,
        PostCode=@PostCode,
        CountyCode=@CountyCode,
        DateAdded=@DateAdded,
        Active=@Active

    where AddressNo=@AddressNo
```

Again like the insert query we specify the destination table for the update…

```
UPDATE tblAddress
```

And then we state which field we are updating and the data to update with…

```
SET     HouseNo = @HouseNo,
        Street = @Street,
        Town = @Town,
        PostCode = @PostCode,
        CountyCode = @CountyCode,
        DateAdded = @DateAdded,
        Active = @Active
```

Lastly we need to add a where clause to make sure that we are updating the right record…

```
WHERE AddressNo=@AddressNo;
```

Save your query as sproc_tblAddress_Update and test it to make sure that it updates an existing record.