Add and Update Methods

In an earlier session you should have created two stored procedures in your data layer like so...

```
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
```

and...

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

AS

--update the fields with the data in the parameters
UPDATE tblAddress
    SET     HouseNo = @HouseNo,
            Street = @Street,
            Town = @Town,
            PostCode = @PostCode,
            CountyCode = @CountyCode,
            DateAdded = @DateAdded,
            Active = @Active
--where the AddressNo matches the value of @AddressNo passed as the parameter
WHERE AddressNo=@AddressNo;



RETURN 0
```

We shall now add the code allowing us to perform both of these functions from our web application.

Much of the code should be familiar by now but there are a few things to watch out for as we add these functions.

We shall start by creating the Add method.

Creating the Add Function

The first step is to create the stub for the method in the collection class...

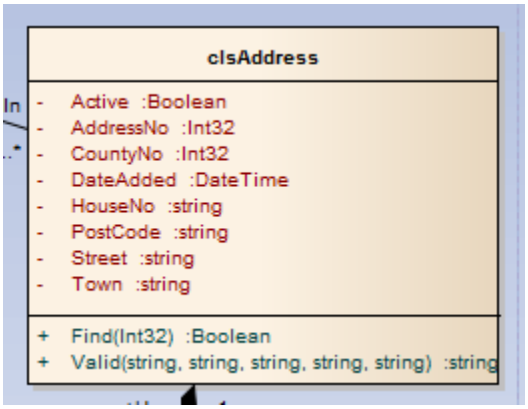
```
public Int32 Add(clsAddress NewAddress)
    ///this function will add a new address to the database
    ///it accepts a single parameter an object of type clsAddress
    ///once the record is added the function returns the primary key value of the new record
 {

 }
```

As usual the function name is underlined in red since we are yet to specify a return value.

Notice also how we are using the class clsAddress as a parameter.

We could create a parameter for each field but since the class bundles up everything in a convenient way like so.

The class looks like this...



The Add function's return data type is Integer. It will return the primary key value of any new record added.

Now that we have our function we may make use of the data connection to connect to the database.

```
public Int32 Add(clsAddress NewAddress)
    ///this function will add a new address to the database
    ///it accepts a single parameter an object of type clsAddress
    ///once the record is added the function returns the primary key value of the new record
{
    //connect to the database
    clsDataConnection NewDBAddress = new clsDataConnection();
}
```

The next step is to pass the parameters to the query. (It is also a good idea to copy and paste the SQL into the function as a comment to help remind you.)

```
public Int32 Add(clsAddress NewAddress)
    ///this function will add a new address to the database
    ///it accepts a single parameter an object of type clsAddress
    ///once the record is added the function returns the primary key value of the new record
    ///        ///
    //INSERT INTO tblAddress
    //    ( HouseNo, Street, Town, PostCode, CountyCode, DateAdded, Active )
    //SELECT
    //    @HouseNo, @Street, @Town, @PostCode, @CountyCode, @DateAdded, @Active;
{
    //connect to the database
    clsDataConnection NewDBAddress = new clsDataConnection();
    //add the parameters
    NewDBAddress.AddParameter("@HouseNo", NewAddress.HouseNo);
    NewDBAddress.AddParameter("@Street", NewAddress.Street);
    NewDBAddress.AddParameter("@Town", NewAddress.Town);
    NewDBAddress.AddParameter("@PostCode", NewAddress.PostCode);
    NewDBAddress.AddParameter("@CountyCode", NewAddress.CountyCode);
    NewDBAddress.AddParameter("@DateAdded", NewAddress.DateAdded);
    NewDBAddress.AddParameter("@Active", NewAddress.Active);
}
```

Lastly we execute the query. Since the execute method returns the primary key value of any new records we may do the following...

```
public Int32 Add(clsAddress NewAddress)
    ///this function will add a new address to the database
    ///it accepts a single parameter an object of type clsAddress
    ///once the record is added the function returns the primary key value of the new record
    ///        ///
    //INSERT INTO tblAddress
    //    ( HouseNo, Street, Town, PostCode, CountyCode, DateAdded, Active )
    //SELECT
    //    @HouseNo, @Street, @Town, @PostCode, @CountyCode, @DateAdded, @Active;
{
     //connect to the database
     clsDataConnection NewDBAddress = new clsDataConnection();
     //add the parameters
     NewDBAddress.AddParameter("@HouseNo", NewAddress.HouseNo);
     NewDBAddress.AddParameter("@Street", NewAddress.Street);
     NewDBAddress.AddParameter("@Town", NewAddress.Town);
     NewDBAddress.AddParameter("@PostCode", NewAddress.PostCode);
     NewDBAddress.AddParameter("@CountyCode", NewAddress.CountyCode);
     NewDBAddress.AddParameter("@DateAdded", NewAddress.DateAdded);
     NewDBAddress.AddParameter("@Active", NewAddress.Active);
     //execute the stored procedure returning the primary key value of the new record
     return NewDBAddress.Execute("sproc_tblAddress_Insert");
}
```

There is one last change we need to make we need to tell the stored procedure to return the primary key value by modifying it like so...

```
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 @@Identity
```

Having created the middle layer functionality the next step is to link it to the presentation layer.

Try the following code in the presentation layer...

```
//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 (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);
    }
    //redirect back to the main page
    Response.Redirect("Default.aspx");
}
else
{
    //display the error message
    lblError.Text = ErrorMessage;
}
```

Test the code to make sure that a new record is added to the table.

Creating the Update Method

The update method is very similar to the insert method however the database needs one extra parameter. We need to pass the value of the AddressNo so that we know which record we are to update.

```
public void Update(clsAddress ExistingAddress)
{
    ///this function will update an existing address in the database
    ///it accepts a single parameter an object of type clsAddressPage
    ///the AddressNo property must have a valid value for this to work
    //SET   HouseNo = @HouseNo,
    //        Street = @Street,
    //        Town = @Town,
    //        PostCode = @PostCode,
    //        CountyCode = @CountyCode,
    //        DateAdded = @DateAdded,
    //        Active = @Active
    //--where the AddressNo matches the value of @AddressNo passed as the parameter
    //WHERE AddressNo=@AddressNo;    {
    //connect to the database
    clsDataConnection ExistingDBAddress = new clsDataConnection();
    //add the parameters
    ExistingDBAddress.AddParameter("@AddressNo", ExistingAddress.AddressNo);
    ExistingDBAddress.AddParameter("@HouseNo", ExistingAddress.HouseNo);
    ExistingDBAddress.AddParameter("@Street", ExistingAddress.Street);
    ExistingDBAddress.AddParameter("@Town", ExistingAddress.Town);
    ExistingDBAddress.AddParameter("@PostCode", ExistingAddress.PostCode);
    ExistingDBAddress.AddParameter("@CountyCode", ExistingAddress.CountyCode);
    ExistingDBAddress.AddParameter("@DateAdded", ExistingAddress.DateAdded);
    ExistingDBAddress.AddParameter("@Active", ExistingAddress.Active);
    //execute the query
    ExistingDBAddress.Execute("sproc_tblAddress_Update");
}
```

Completing the Event Handler

Now that the two functions are set up we need to build them both into the event handler for the save button.

In the case of insert if the user types an AddressNo of -1 we want the record to be added.

In the case of update however if the user types a valid integer value then we will assume it is the primary key value of an existing record and update it.

Like so...

```
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 (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");
}
else
{
    //display the error message
    lblError.Text = ErrorMessage;
}
```

Test the program to see if you are able to insert and update records in the table.