FACTFILE:
GCE SOFTWARE SYSTEMS DEVELOPMENT

Appendix 2

Rewarding Learning

Display Customer details from a table in the database to the CustomerDisplay form

Tab Page	Controls
Customers By No	ListView
Individual Details	GroupBox with 26 Buttons ListBox for Customer Names Panel with read-only TextBoxes for details

Table 2.1: Set up form with a TabControl for two views

Three tables are copied to the DataSet from the database:

Customers -all details ordered by CustomerNo;
-uses FillSchema to allow search of the table – Find;

CustomerNames -customer names ordered alphabetically
-uses SqlCommand for parameter setting
-uses the parameter to allow filter by first letter of surname;

Letters -holds distinct surnames of customers for use with alpha buttons

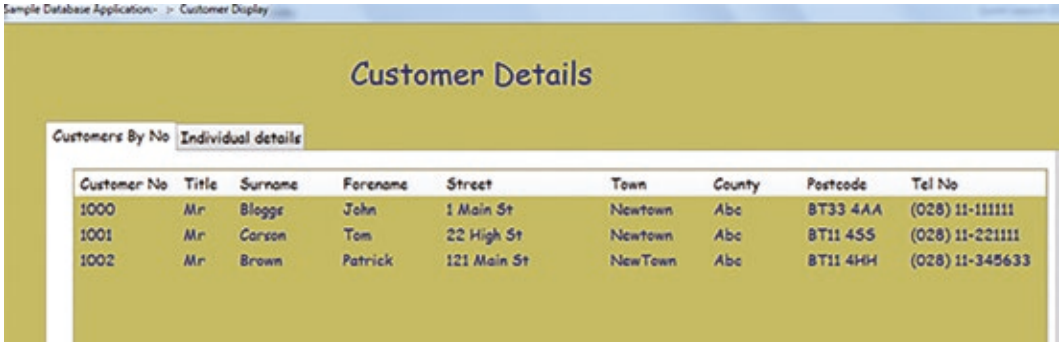

Figure 2.1: Tab1 shows the customers by CustomerNo in a ListView

Coding

Aside from usual libraries you must include the SQL library

```
using System.Data.SqlClient;

namespace ProjSampleDatabase
{
    public partial class FrmCustomerDisplay : Form
    {
        SqlDataAdapter daLetters, daCustomers, daCustomerNames;
        DataSet dsSampleDatabase = new DataSet();
        SqlCommand cmdCustomerNames;
        DataRow drCustomer;
        String  cnstr = Properties.Resources.connectStr;
        String sqlLetters, sqlCustomers, sqlCustomerNames;

        Button[] btns = new Button[26];      // used to reference alpha buttons on form

        public FrmCustomerDisplay()
               {
                      InitializeComponent();       // builds the form design

               //hold application name once in Properties and use on all forms
                      this.Text = Properties.Resources.appStr + " :-  Customer Display";

        }
```

Load event

Code to populate ListView

```
// connect to database and create dataset and table to hold customer details by No
// Use Properties to hold value for maintenance efficiency

Try
{

    // cnstr = Properties.Resources.connectStr;    set globally above

    // Set up table customers for use in Display Customers by CustomerNo - Tab1

    sqlCustomers = @"select * from Customers order by CustomerNo";
    SqlConnection conn = new SqlConnection(cnstr);
    daCustomers = new SqlDataAdapter(sqlCustomers, conn);
    daCustomers.FillSchema(dsSampleDatabase, SchemaType.Source, "Customers");

    // allows Find action on selection
    daCustomers.Fill(dsSampleDatabase, "Customers");

    // set width of a column in ListView
    lsvCustomer.Columns[4].Width = 175;
    lsvCustomer.Columns[7].Width = 110;
    lsvCustomer.Columns[8].Width = 150;

    // fill ListView with contents of dataset table customers
    ListViewItem item = new ListViewItem();
    foreach (DataRow dr in dsSampleDatabase.Tables["customers"].Rows)
    {
        item = lsvCustomer.Items.Add(dr["customerNo"].ToString());
        item.SubItems.Add(dr["title"].ToString());
        item.SubItems.Add(dr["Customersurname"].ToString());
        item.SubItems.Add(dr["Customerforename"].ToString());
        item.SubItems.Add(dr["address1"].ToString());
        item.SubItems.Add(dr["address2"].ToString());
        item.SubItems.Add(dr["address3"].ToString());
        item.SubItems.Add(dr["postcode"].ToString());
        item.SubItems.Add(dr["telNo"].ToString());
    }// end foreach
...
...  // code for Tab2 continues
     // It is advisable to code this and test before continuing with Tab2 code
```

Figure 2: Tab2 shows a GroupBox with buttons coded to hold alphabet, a ListBox of customers in order of name and a panel holding read-only TextBoxes to hold details of a selected customer.

Take great care building the buttons. They must be pasted one by one to ensure that the sequence of letters is correct when text property is set in coding later. These 26 buttons are referenced by an array for processing - Button[] btns = new Button[26];

NB: Do not set any properties when building the buttons aside from the first button's FontSize.

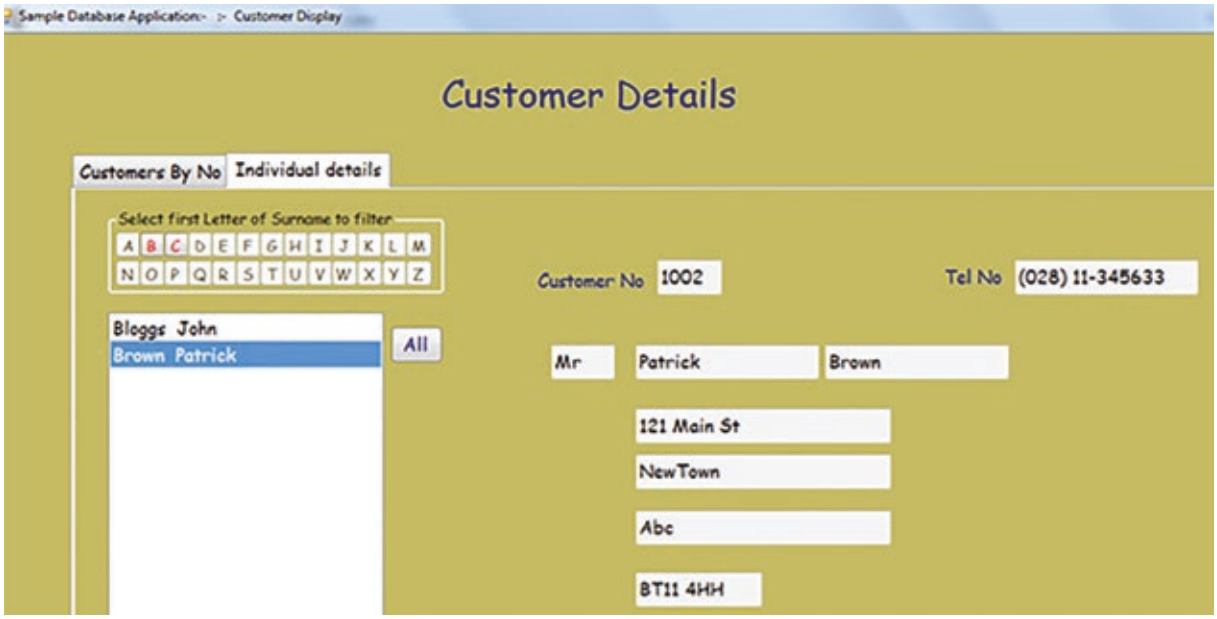

Figure 2.2: Shows a GroupBox, ListBox and TextBoxes

Load event code continued....

```
connectAlphaButtons();    // method to reference 26 Buttons on form to button array

//Set up table Names to hold customerNames in alphabetic order    -Tab2
sqlCustomerNames = @"select customerSurname +' '+customerForename as Name,
                        customerNo from Customers
                        where customerSurname like @letter
                        order by customerSurname,customerForename";

conn = new SqlConnection(cnstr);
cmdCustomerNames = new SqlCommand(sqlCustomerNames, conn);
cmdCustomerNames.Parameters.Add("@letter", SqlDbType.VarChar);
daCustomerNames = new SqlDataAdapter(cmdCustomerNames);

// fill listbox with names - use wildcard this first time
// method used - required by load event, Alpha btns and btnAll
String str = "";
fillListboxNames(str);
}

catch (Exception ex)
{
    MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!",   MessageBoxButtons.OK,
MessageBoxIcon.Error);
}// end catch
```

```
public void connectAlphaButtons()
{
  // dblClk on the first button in groupBox to set up the button1_Click(....) event

  // insert the code to reference the buttons in the groupBox to the array btns
  // set the text of each button to the relevant alphabetic letter     65 - A
  // attach the same event handler to all 26 buttons of the array

  for (int x = 0; x < 26; x++)
  {
btns[x] = (Button)grpAlpha.Controls[25-x];
btns[x].Text = "" + (char)(65 + x);
btns[x].Enabled = false;
btns[x].BackColor = Color.Black;
btns[x].Click += new EventHandler(button1_Click);
  }// end for

  //get surnames and enable relevant alphabet butttons
    sqlLetters = @"Select distinct customerSurname from customers
                   order by customerSurname";

    SqlConnection conn = new SqlConnection(cnstr);
    daLetters = new SqlDataAdapter(sqlLetters, conn);
    daLetters.Fill(dsSampleDatabase, "Letters");

    // enable relevant alpha buttons
    int no;
    foreach (DataRow dr in dsSampleDatabase.Tables["Letters"].Rows)
    {
      // convert first letter to ascii code
      no = (int)dr["customerSurname"].ToString()[0] - 65;
      btns[no].Enabled = true;
      btns[no].ForeColor = Color.Red;
    }// end for

}// end connectAlphaButtons

private void button1_Click(object sender, EventArgs e)
{
   Button b = (Button)sender;
   // get  customer details for listbox   - use selected button letter for parameter
   String str = b.Text;

   //empty current data held in dataset table CustomerNames
   dsSampleDatabase.Tables["Names"].Clear();

   fillListboxNames(str);

}// end button1_Click


private void btnAll_Click(object sender, EventArgs e)
{
   //empty dataset table customerNames
   dsSampleDatabase.Tables["Names"].Clear();

   // fill listbox of customers with all names
   String str = "";
   fillListboxNames(str);
}// end btnAll_Click
```

```
private void lstNames_Click(object sender, EventArgs e)
{
   try
   {
     //Find customerNo in the table Customers – fill DataRow with customer details
     //set textbox Text values
     drCustomer = dsSampleDatabase.Tables["customers"].Rows.Find(lstNames.SelectedValue.ToString());

     //set read property to read only
     txtCustomerNo.Text = drCustomer["CustomerNo"].ToString();
     txtTitle.Text = drCustomer["Title"].ToString();
     txtSurname.Text = drCustomer["customerSurname"].ToString();
     txtForename.Text = drCustomer["customerForename"].ToString();
     txtStreet.Text = drCustomer["Address1"].ToString();
     txtTown.Text = drCustomer["Address2"].ToString();
     txtCounty.Text = drCustomer["Address3"].ToString();
     txtPostcode.Text = drCustomer["Postcode"].ToString();
     txtTelNo.Text = drCustomer["TelNo"].ToString();
  }// end try

  catch (Exception ex)
  {
    MessageBox.Show("" + ex.TargetSite + "" + ex.Message, "Error!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
  }// end catch

} // end lstNames_Click
```

An advanced ADD would use the class Customer to validate the fields and throw a customised Exception which would be caught in the ADD. The **catch** would set the error provider with a specific error message.

Customised Exception

```
class CustomerException : Exception
  {

    public CustomerException(string message)
      : base(message)
    { }

  }
```

Class Customer

```
class Customer
  {
    private int customerNo;
    private String title;
    private String surname;
    private String forename;
    private String street;
    private String town;
    private String county;
    private String postcode;
    private String telNo;

    public Customer()
{
      customerNo=0;
      title=null;
      surname=null;
      forename=null;
      street=null;
      town=null;
      county=null;
      postcode=null;
      telNo=null;
   }//end Customer

public Customer(int customerNo, String title, String surname, String forename,
                String street, String town, String county, String postcode, String telNo)
{
                CustomerNo=customerNo;
                Title= title;
                Surname= surname;
                Forename= forename;
                Street= street;
                Town= town;
                County= county;
                Postcode= postcode;
                TelNo= telNo;
        }
```

```
public int CustomerNo
{
   get { return customerNo; }
   set { customerNo = value; }
}

public String Title
{
   get { return title; }
   set { title = value; }
}

public String Surname
{
   get { return surname; }
   set { //implement validation / throw new exception
      String str = value;
      if (validString(str, 3, 15).CompareTo ("ok")!=0)
         throw new CustomerException("Invalid Surname - Please check");

      else
         surname  = value;
      } //end set
}

public String Forename
{
   get { return forename ; }
   set { forename = value; }
}

public String Street
{
   get { return street ; }
   set { street  = value; }
}

public String Town
{
   get { return town ; }
   set { town  = value; }
}

public String County
{
   get { return county ; }
   set { county  = value; }
}

public String Postcode
{
   get { return postcode ; }
   set { postcode  = value; }
}
```

```
public String TelNo
{
   get { return telNo ; }
   set { telNo = value; }
}

public String toString()
{   //basic - extend for all fields
   return String.Format("\n {0:d4}  {1} {2} {3} {4} {5}", customerNo,
                surname, forename, street, postcode, telNo);
}

private String validString(String str, int min, int max)
{
   String message = "ok";

   if (String.IsNullOrEmpty(str))
   {
      message = "Required field - must enter data";
   } //end if

   else if (str.Length < min || str.Length > max)
   {
      message= " must have minimum of " + min + " chars and a maximum of " + max;
   } //end else if

   return message;
} //end validString

} //end class Customer
} // looks like an extra bracket!!!
```

Revised Click event for ADD button

```
private void btnAdd_Click(object sender, EventArgs e)
  {
  // add using customer class validation for the surname - implement for other fields
  // try-catch used for each individual field
  // could use around all fields but only first error found would be shown by error
  // provider

            errP.Clear();
            Boolean ok = true;
            Customer c = new Customer();

            if(lstTitle.SelectedIndex == -1)
            {
               ok = false;
               errP.SetError(lstTitle, "Must select Title");
            }

            if (!validString(txtForename, 3, 15))
               ok = false;
            else
               if (!validLetters(txtForename))
                  ok = false;
            try
            {
               c.Surname = txtSurname.Text.Trim();
            }

            catch (CustomerException ex)
            {
               ok = false;
               errP.SetError(txtSurname, ex.Message);
            }

            if (!validString(txtStreet, 5, 20))
               ok = false;

             // if valid move details to new row in dataset table
            if (ok)
            {
               drCustomer = dsSampleDatabase.Tables["Customer"].NewRow();

               drCustomer["CustomerNo"] = int.Parse(txtCustomerNo.Text);
               drCustomer["Title"] = lstTitle.SelectedItem.ToString();
               drCustomer["Forename"] = txtForename.Text.Trim();
               drCustomer["Surname"] = c.Surname;
               drCustomer["Address1"] = txtStreet.Text.Trim();
               drCustomer["Address2"] = txtTown.Text.Trim();
               drCustomer["Address3"] = txtCounty.Text.Trim();
               drCustomer["Postcode"] = txtPostcode.Text.Trim();
               drCustomer["TelNo"] = txtTelNo.Text.Trim();

               dsSampleDatabase.Tables["Customer"].Rows.Add(drCustomer);
               daCustomer.Update(dsSampleDatabase, "Customer");

               MessageBox.Show("Customer Added");
            } //end if

      } //end btnAdd_Click
```