FACTFILE:
GCE SOFTWARE SYSTEMS DEVELOPMENT

Appendix 4

Rewarding Learning

ListBox, ListView, DataGridView Controls with reference to Database Tables

Manipulating ListBoxes.

The ListBox allows selection of one or more items from a list. The list can be ordered for ease of selection. A more advanced use for ListBoxes is for selection of items originating from a database. It can be populated from a table using the following properties

dataSource:	entity name
displayMember:	e.g. category Name displayed in listBox - user friendly
valueMember:	e.g. categoryNo tracked in background - use for database access

- The listBox property *selectedValue* is used to access the categoryNo selected.
- The listBox property *selectedText* is used to access the selected display text.

To fill a listbox with categories of stock for selection in Add new Stock

```
//list box for Category of stock

// get copy of category table in dataset
sqlCategory = @"Select  CategoryNo,  CategoryName
from Category
order by CategoryName";
conn = new SqlConnection(cnstr);

daCategory= new SqlDataAdapter(sqlCategory,  conn);
daCategory.Fill(dsSampleDatabase,  "Category");

// fill listbox with contents of dataset table category

lstCategory.DataSource = dsSampleDatabase.Tables["Category"];
lstCategory.DisplayMember = "CategoryName";
lstCategory.ValueMember = "CategoryNo";
```

Reference the selected category code

```
lstCategory.SelectedValue.ToString()        // accesses value

lstcategory.ClearSelected()                 // clears selection
```

See Appendix 2 – Display Code for example with a parameter which requires the use of SqlCommand

Manipulating ListView

The ListView control is very useful for displaying multiple fields of records and is more versatile than the ListBox or ComboBox .

The row can be selected or a field within the row and is editable.
Item is the first field on a row.
SubItems are subsequent fields on a row.

Minimum setup for ListView control:-

- Set the **Columns** property
  - Add columns for the items to be viewed
  - Set the text property of each appropriately - (heading)
  - Set the width relevant to size of field
- Set the **View** property to 'details'

Display several fields from table(s)

Example: code to add items to a ListView component named 'lsvStock', from a DataSet table, StockDisplay, holding details from Stock, Category and Supplier Tables in the Database.

```
lsVCust.Items.Clear();

ListViewItem item = new ListViewItem();
foreach (DataRow dr in dsSampleDatabase.Tables["StockDisplay"].Rows)
{
  item = lsvStock.Items.Add(dr["stockNo"].ToString());
  item.SubItems.Add(dr["description"].ToString();
  item.SubItems.Add(dr["categoryName"].ToString());
  item.SubItems.Add(dr["supplierName"].ToString());
  item.SubItems.Add(dr["reorderLevel"].ToString());
  item.SubItems.Add(dr["reorderQty"].ToString());
  item.SubItems.Add(dr["price"].ToString());
  item.SubItems.Add(dr["QtyInStock"].ToString());

}// end foreach
```

An example of a ListView component used to display a list of treatments for selection in making an appointment for a customer is shown below. A second listView displaying the selected treatments and the total cost of the treatments for the appointment would be user friendly.

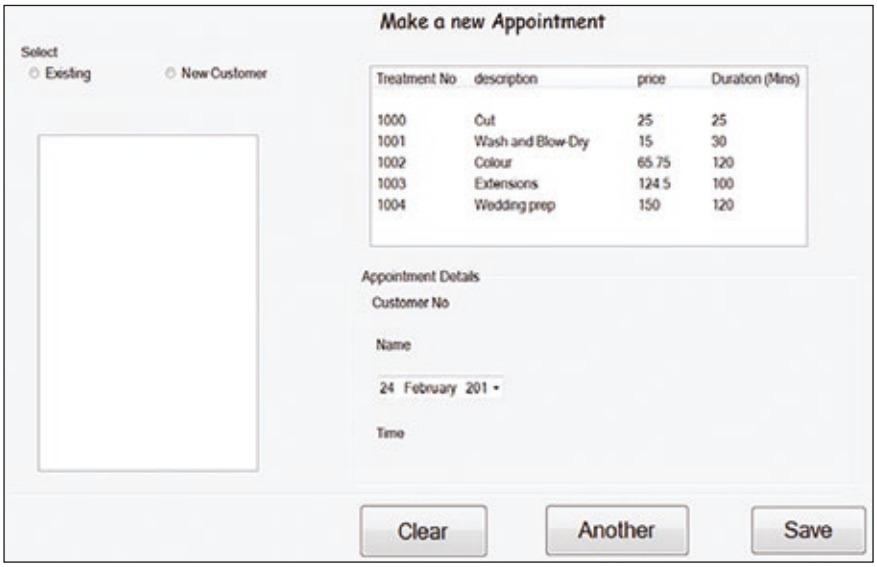

Figure 4.1 An example of a ListView component

The example below shows how Listview controls can be used for selection/display in an Order Processing activity.

Data can be selected from one Listview control (filled from a table populated by a query of 'available stock') and moved to another Listview control ('ordered stock'). It is removed from the 'available stock' and so prevents duplicate stock numbers in an order.

This is useful as the dataset table AvailableStock is not affected by the moves from one listview to another until the AddOrder button is clicked when it should be refreshed.

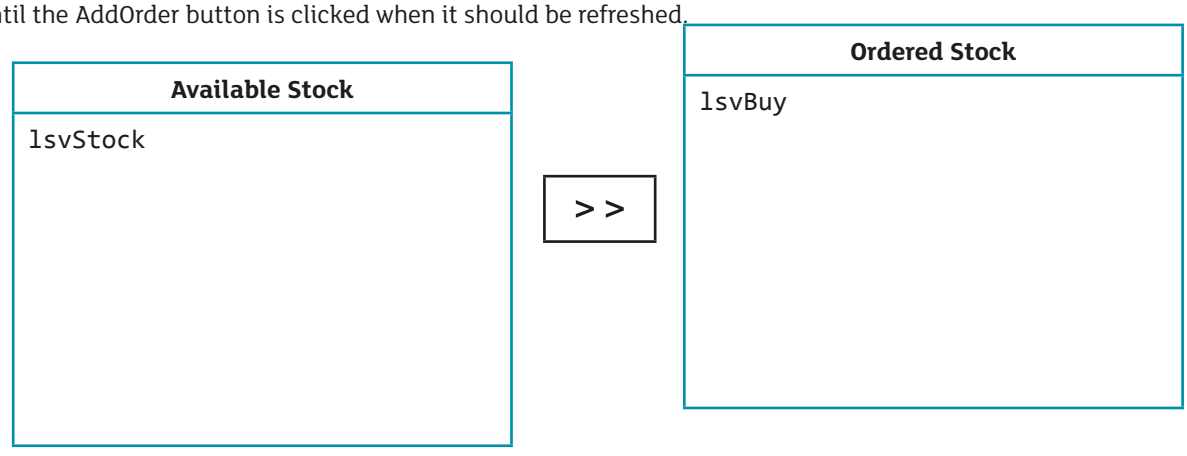

Figure 4.2 Using ListView controls to view and select items

To select an item from a ListView use the click event (or DoubleClick event when deleting)

```
private void lsvStock_Click(object sender, EventArgs e)
  {
        int no;
        ListViewItem itm = new ListViewItem();

        // get index of first selected row item and retrieve data to 'itm'
        no = lsvAvailableStock.SelectedIndices[0];
        itm = lsvAvailableStock.Items[no];

        // remove item from ListView as it has been selected for the order
        lsvStock.Items[no].Remove();

        // Add item to the 'buy' ListView
        lsvOrdered.Items.Add(itm);
        }
```

To retrieve a value from a subItem on the row use the following code

Note. The property **FullRowSelect** must be set to **true** (the default accesses the first item value only)

E.g. to reference subItem 2

```
qty = lsvAvailableStock.Items[no].SubItems[2].Text.toString()
price = lsvAvailableStock.Items[no].SubItems[3].Text.toString()
```

Setting the value of a subItem

Another column can be set to a value or a calculation e.g. qty * price

```
lsvStock.Items[no].SubItems[4].Text.toString()  = qty * price;
```

Calculating total from a field in the listView

To go through **each row of a listView** showing the selected treatments for a client to add to the dataset table "PatientTreatments" for **update to the database**.

- BookingNo is taken from a txtBox
- treatmentNo comes from a row in the selected Treatments ListView
- qty comes from a row in the selected Treatments ListView

```
int qty, price, cost, total =0;
for ( int x = 0; x < lsvSelectedTreatments.Items.Count; x++ )
{
  ListViewItem itm = lsvTreatments.Items [x];
cost = int.parse ( itm.SubItems[4].Text.ToString()  );
total+= cost;
}
```

Updating data from a listView to the database

After adding the booking to the Booking table (BookingNo, BookingDate, CustomerNo) the treatment details must be added to the BookingDetails table.

This entails going through **each row of a listView**, showing the selected treatments for a customer, adding data to the dataset table "BookingDetails" for **update to the database table**.

- BookingNo is taken from a textbox on the form
- treatmentNo comes from a subItem in the SelectedTreatments ListView
- qty comes from a subItem in the Selected Treatments ListView

```
int BookingNo = int.parse ( txtBookingNo.Text.ToString()  );
int treatNo, qty;
for (int x = 0; x < lsvSelectedTreatments.Items.Count; x++ )
{
  ListViewItem itm = lsvSelectedTreatments.Items [x];
  treatNo = int.parse (itm.Text.ToString()  );
  qty = int.parse (itm.SubItems[2].Text.ToString()  );

//values ready to be moved to new row of dataset table "BookingDetails"and //updated to
the database.
 …
 …
 …
}
```

DataGridView

A DataGridView can be used to display data from the database in columns and also to allow columns to be used for update and for data entry.

Display of data - using **bound** fields

```
// fill the dataGrid on the ALL tab with details
dgvCustomers.DataSource = dsSampleDatabase.Tables["Customer"];

// change properties of the columns as required
dgvCustomers.Columns["Name"].Visible=false;
// example make this column invisible as it is repetition (surname and forename
// in previous two columns)

// reset the column widths if needed e.g. code below widens the address1 column
// (counts from 0)
DataGridViewColumn column;
column = dgCustomers.Columns[5]; //Or use name of column ["Address1"]
column.Width = 150;
```

Update of data – using unbound DataGrid fields

A sample DataGridView used for updating deliveries is shown below.

The columns' ItemNo', 'Description' and 'Qty Ordered' are used to display information about an order.

The columns 'QtyDelivered' and 'Reason' can be used to enter data to reflect the delivery status.
Note that the **Enable Editing** and **Enable Adding** options must be active.

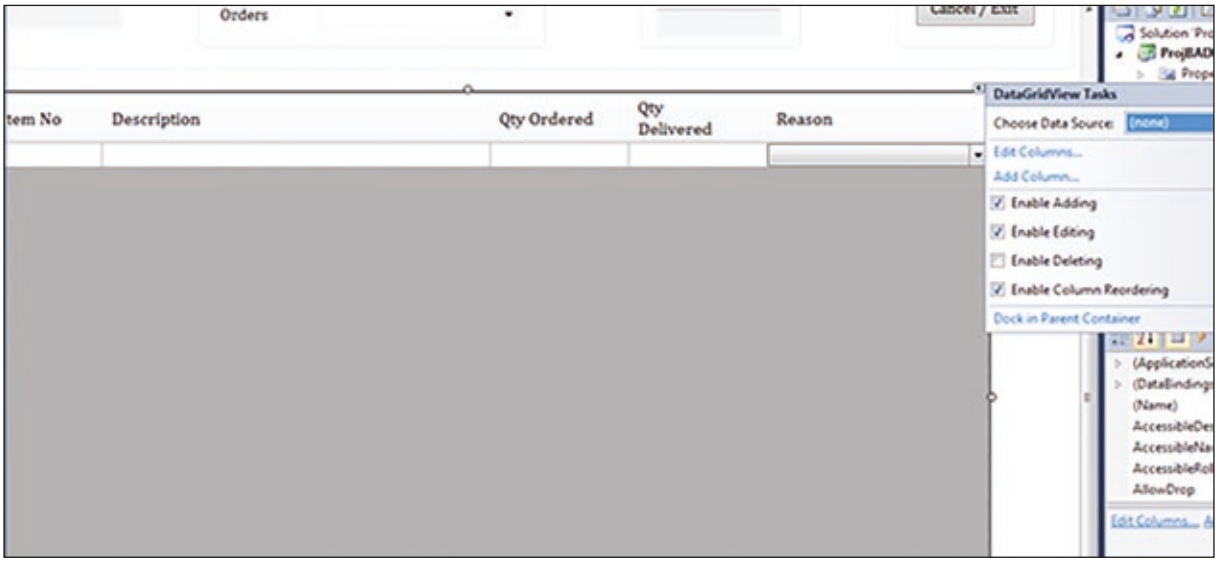

Figure 4.3 Sample dataGridView used for updating deliveries

Code to fill dataGridView with order details from a table in the DataSet - unbound fields

```
foreach (DataRow dr in dsStockControl.Tables["OrdItems"].Rows)
{
        dgvDelivery.Rows.Add(dr["ItemNo"].ToString(), dr["ItemDesc"].ToString(),
        dr["QuantityOrdered"].ToString(), dr["QuantityOrdered"].ToString());
}// end foreach
```

The repetition of the 'QuantityOrdered' value is in the column headed 'Qty Delivered' is user friendly as the delivery quantity is changed only when it does not match the QuantityOrdered. The mismatch reason can then be selected from the comboBox.

Selecting from a DataGridView

Properties to consider:

- SelectionMode - set to fullRowselect or cell select,
- MultiSelect - set false

To reference the row in the DataGridView where a CELL has been clicked

Example.

```
Private void dgvSelectStock CellContentClick
(object sender, DataGridViewCellEventArgs e)

{
        int index = e.RowIndex;
        DataGridViewRow  row = dgvSelectStock.Rows[index];
        dgvSelectStock.Rows.RemoveAt(index);
}
```

You can use SelectionChanged event if you are using FullRowSelect selection mode.
(See the properties of the DataGridView)

Inside the handler you can access SelectedRows property and get data from it.

Example.

```
foreach (DataGridViewRow row in dataGridView.SelectedRows)
{
        string value1 = row.Cells[0].Value.ToString();
        string value2 = row.Cells[1].Value.ToString();
        //...etc
}
```

To reference columns in a row in the dataGridView - where numRow is the selected row.

```
num1 = int.Parse(dgvDelivery[2, numRow].Value.ToString()); //col 3
num2 = int.Parse(dgvDelivery[3, numRow].Value.ToString()); //col 4
```

Code to update tables after delivery details have been entered.

Use the appropriate columns of the dataGridView (itemNo, delivery qty, reason Code) and the textbox values on the form (deliveryNo, orderNo) to update the delivery and delivery Details table.

Note that the Reason column (5) has been modified to be a comboBox. This can then be hardcoded with delivery discrepancy reasons or filled from a table 'Reasons' in the dataset.

To fill comboBox from a table.

```
((DataGridViewComboBoxColumn)(dgvDelivery.Columns[4])).DataSource = dsStockControl.
Tables["Reasons"];      // cast column as a ComboBox

((DataGridViewComboBoxColumn)(dgvDelivery.Columns[4])).ValueMember = "ReasonNo";

((DataGridViewComboBoxColumn)(dgvDelivery.Columns[4])).DisplayMember =
"ReasonDescription";
```

Recommended weblink:

www.c-sharpcorner.com/uploadfile/mahesh/listbox-in-C-Sharp/