Technical Note

Defining a Double Selection List

Author: R&D Department

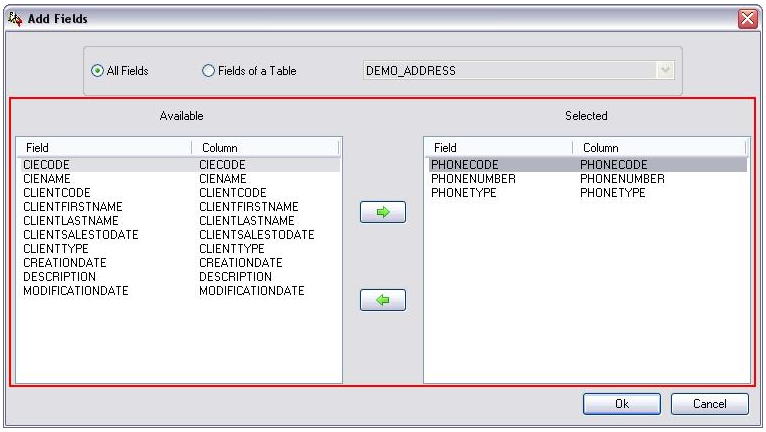
Publication date: December 22, 2006

Revision date: May 2010

Defining a Double Selection List

Overview

A double selection list is a user interface that displays a list of available items and a list of selected items, and that allows a user to move items between the two lists. An example taken from REP++*studio* appears below, showing a dialog box for selecting the fields to add to an existing **RowsetDef**:



The fields on the left are those available, while the fields on the right are those selected for the **RowsetDef**. The user can move the fields from one list to the other using the arrow buttons.

The REP++*toolkit* for Windows® applications provides a helper class named **DoubleSelectList** that turns two lists and a pair of buttons into a double selection list.

This article describes how to define a double selection list by using the **DoubleSelectList** class provided by the REP++*toolkit* for Windows.

How does it work?

The REP++*toolkit* for Windows® applications provides a helper class named **DoubleSelectList** that lets you specify two **ListViewRowset** objects and a pair of buttons to create a double selection list. As a result, all you have to do is set the layout of your controls and populate the initial content of the two lists. The REP++*toolkit* helper class will manage the movement of items between the two lists. The **DoubleSelectList** class will:

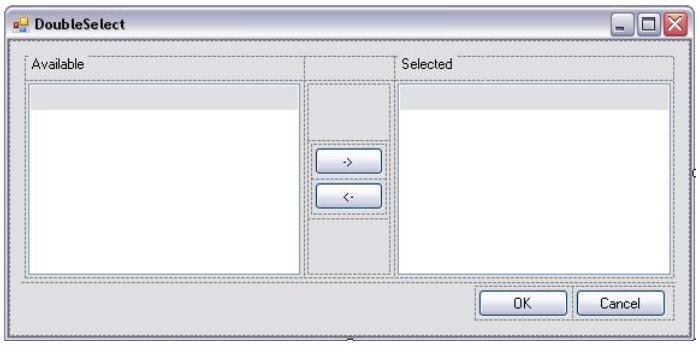
- Handle the ***DoubleClick*** event on a selected item to move it to the other list.
- Handle the ***Click*** event on each button to move a selected item from one list to another.
- Enable and disable the buttons depending on the item currently selected in the source list.

At the end of the selection process, the content of the **ListViewRowset** objects and their attached **Rowset** objects will reflect the moves made by the end user.

Defining a DoubleSelectList object

In this section, you will go over the tasks that you have to complete before using a **DoubleSelectList**. You will start by creating a sample form that displays a list of available clients and a list of selected clients.

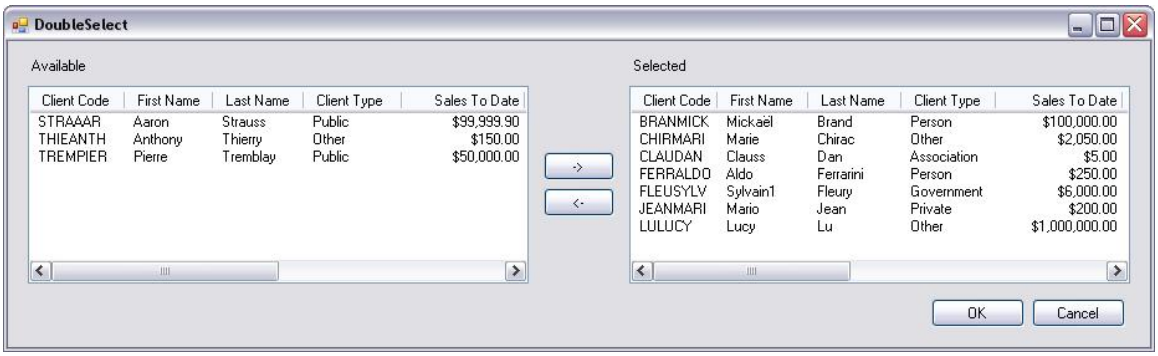
1. Create a new Windows® application and add a Windows Form named *DoubleSelect.*
2. Add a pair of **ListViewRowset** objects named *lvAvailable* and *lvSelected*.
3. Set their **View** property to *Detail*.
4. Add four buttons named **btnAdd**, **btnRemove**, **btnOK** and **btnCancel**. The layout of your form should look similar to the following figure:



5. Edit the code of the form to load the initial content of the lists. Please note that the following code separates the clients between the two lists to show that both lists can have an initial content:

```
…
using RepPP;

namespace Demo {
    public partial class DoubleSelect : Form {
        private RowsetTree        m_rstAvailable = null;
        private RowsetTree        m_rstSelected  = null;

        public DoubleSelect() {
            RepPP.Application app;
            RowsetTreeDef     rstDef;

            InitializeComponent();
            app    = RepPP.Application.CreateFromRes();
            rstDef = app.RowsetTreeDefs["CLIENT"];
            rstDef.BuildSqlCommand();
            m_rstAvailable = rstDef.RowsetTrees.Add();
            rstDef.Nodes[0].SetSelectSqlWhere("WHERE CLIENTCODE > 'M'");
            m_rstAvailable.ReadFromDb();

            m_rstSelected = rstDef.RowsetTrees.Add();
            rstDef.Nodes[0].SetSelectSqlWhere("WHERE CLIENTCODE <= 'M'");
            m_rstSelected.ReadFromDb();

            lvAvailable.Fill(m_rstAvailable.RootRowset, true);
            lvSelected.Fill(m_rstSelected.RootRowset, true);
        }
    }
}
```

6. Build and run your application.



As you can see, the lists have been populated, but items cannot be moved from one list to the other. You could implement the logic of a double selection list manually, but this process is very mechanical and error-prone. The next section will integrate a **DoubleSelectList** object to your form to automate the implementation of your double selection list.

Using a DoubleSelectList object

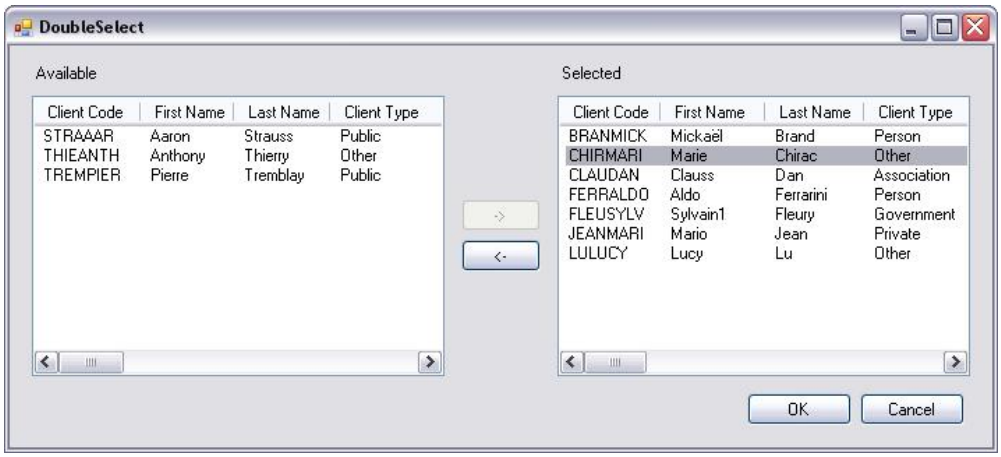
In this section, you will add a **DoubleSelectList** object to the form that you created in the previous section. In order create a **DoubleSelectList**, you have to provide:

- The **ListViewRowset** object representing the available items.
- A flag to indicate how to delete the **Rowset** line of an item moved from the list of available items. Set the value of this flag to *True* to remove the line or to *False* to mark it as deleted.
- The **ListViewRowset** object representing the selected items.
- A flag to indicate how to delete the **Rowset** line of an item moved from the list of selected items. Set the value of this flag to *True* to remove the line or to *False* to mark it as deleted.
- The button that selects an available item.
- The button that removes a selected item.

Modify the code of the form created in the previous section to add a **DoubleSelectList** object as follows:

```
…
using RepPP.Toolkit.Window;

namespace Demo {
  public partial class DoubleSelect : Form {
    …
    private DoubleSelectList m_dblSelList = null;

    public DoubleSelect() {
      …
      m_dblSelList = new DoubleSelectList(lvAvailable, false,
                                          lvSelected, false,
                                          btnAdd,
                                          btnRemove);
    }
  }
}
```

Guess what? You have just implemented a fully functional double selection list! Build and run your application, and try it out.



Note:

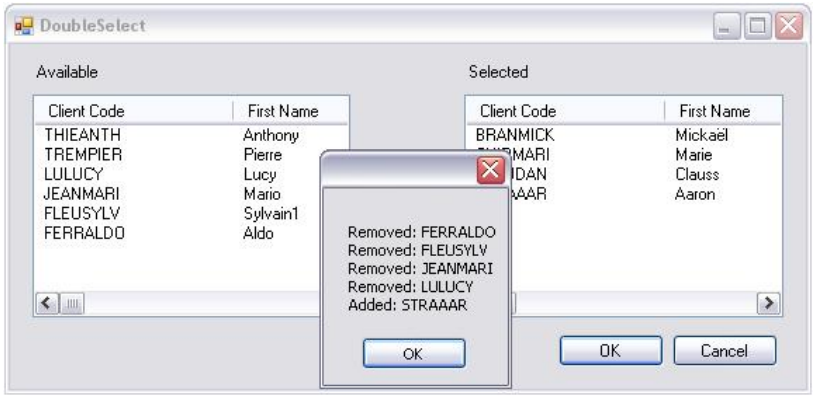
- How the buttons are enabled and disabled depending on the lists' current selection.
- That you can also double-click on an item to move it.

Now that your double selection list is up and running, you will write some code to track the changes made by the user.

1. Create an event handler that handles the ***Click*** event of the **btnOK** button.
2. Add the following code, which lists the items added to or removed from the list of selected items using the line state of the attached **Rowset**.

```
private void btnOK_Click(object sender, EventArgs e) {
  Rowset        rowset        = lvSelected.Rowset;
  Field         fldCode       = rowset.Fields["CLIENTCODE"];
  LineState     eState;
  string        strMessage    = string.Empty;
  int           iLineCount    = rowset.LineCount;

  for(int iLine = 0; iLine < iLineCount; iLine++) {
    eState = rowset.ThisLineState(iLine);
    if (eState == LineState.sdLineDeleted) {
      strMessage += "\n Removed: " + fldCode.GetValue(iLine, true);
    }if (eState == LineState.sdLineNew) {
      strMessage += "\n Added: " + fldCode.GetValue(iLine, true);
    }
  }
  if (strMessage == string.Empty) {
    strMessage = "No Change!!";
  }
  MessageBox.Show(strMessage);
}
```

Build and run your application.


DoubleSelect
Available
Client Code
First Name
THIEANTH
Anthony
TREMPIER
Pierre
LULUCY
Lucy
JEANMARI
Mario
FLEUSYLV
Sylvain1
FERRALDO
Aldo
Selected
Client Code
First Name
BRANMICK
Mickaël
Marie
Clauss
Aaron
Removed: FERRALDO
Removed: FLEUSYLV
Removed: JEANMARI
Removed: LULUCY
Added: STRAAAR
OK
OK
Cancel