Technical Note

# Persisting the User Interface State

*Author: R&D Department*

*Publication date: December 7, 2006*

*Revision date: May 2010*

# Persisting the User Interface State

## *Overview*

Start Visual Studio® .NET, resize it, change its position, then restart it. Visual Studio .NET, like many user-friendly applications, stores its user interface state for future sessions.

The REP++*toolkit* for Windows® applications provides a component named **UIConfigurationHelper** to automate the persistence of your applications' user interface. You simply register your forms, and the configuration helper saves or loads their size, position and state. The configuration helper also saves or loads the state of a number controls within the registered form. It includes:

- Built-in support for **Form**s, **Panel**s and **DataGrid**s, and
- **Control**s that implement the **IUIConfiguration** interface.

There are two ways to extend the persistence mechanism of the user interface state:

- Derive your own configuration helper from **UIConfigurationHelper** to extend the list of built-in controls. Using this approach, you do not have to extend your controls in order to manage their user interface state.
- Extend your controls to enable them to manage their user interface state by implementing the **IUIConfiguration** interface. Using this approach, you will simply use the configuration helper as is.

This article describes how to implement both approaches, i.e. how to use the **UIConfigurationHelper** control, and how to extend a standard control into a control capable of persisting its user interface state. It also explains how to save the user interface state in a different data store.

## *Frequently used methods of the UIConfigurationHelper class*

This section introduces the methods of the **UIConfigurationHelper** class that you are most likely to use.

| Method | Description |
|---|---|
| **RegisterForm** | Registers a form in the list of forms whose user interface state will be saved by the configuration helper. |
| **SaveToFile** | Saves the user interface state information gathered by the configuration helper into an XML file. The file name was specified at construction time. |
| **GetNodeAttribute** | Gets the value of an attribute from an XML element. Used mainly to load the value of a property of a control. |
| **SetNodeAttribute** | Sets the value of an attribute in an XML element. Used mainly to save the value of a property of a control. |
| **FindNode** | Finds a node in the configuration XML file. Usually used to find/create a node to save/load the configuration of a specific control. |
| **SaveFormState** | Saves the state of a form in the in-memory XML file. The **SaveToFile** method must be called to save the state to the file system. |

## *Using the UIConfigurationHelper object to implement user interface state persistence*

You will add a **UIConfigurationHelper** object to a standard Windows® form in order to save its size, position and state.

1. Create a new *Windows Application* project.
2. Add a reference to **RepPP.Toolkit.V1B.dll**.
3. Modify the code of your form as follows.

```
using System;
using System.Windows.Forms;
using RepPP.Toolkit.Window;

namespace WindowsApplication8 {
    public partial class Form1 : Form {

        private UIConfigurationHelper m_configHelper;

        public Form1() {
            InitializeComponent();

            // Creates the configuration helper and sets the name of the file
            m_configHelper = new UIConfigurationHelper("myConfigurationFile.xml");

            // Indicates that the configuration must be saved automatically
            // when this form is closed
            m_configHelper.AttachedForm = this;

            // Registers this form in the list of forms whose user interface
            // state will be saved by the configuration helper
            m_configHelper.RegisterForm(this);
        }
    }
}
```

4. Run your application, and modify the size, position and state of your form.
5. Restart your application. You should see that the previous state of your application was successfully saved and restored.

Here is the content of the configuration file that was generated.

```
<REP>
  <FORMFORM1 VERSION="1" WINDOWSTATE="0" WIDTH="751" HEIGHT="272" LEFT="114" TOP="150">
</REP>
```

## *Extending a control to implement user interface state persistence*

In addition to handling the state of **Form**s, **Panel**s and **DataGrid**s, the **UIConfigurationHelper** object can save the state of the controls within the form that are capable of saving their user interface state. The purpose of the **IUIConfiguration** interface is to signal the **UIConfigurationHelper** object that this class implements an object that is capable of persisting its user interface state.

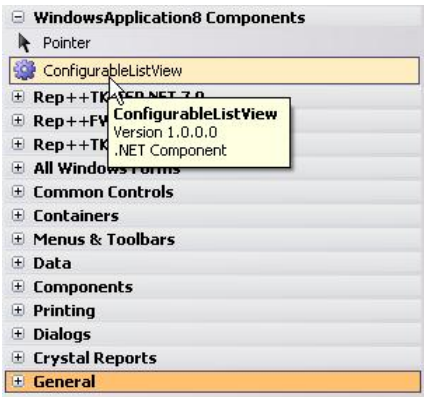
To illustrate this, you will take a standard **ListView** control and create a **ConfigurableListView** control capable of remembering the width of its columns.

1. Add a new class to your project and name it **ConfigurableListView**.

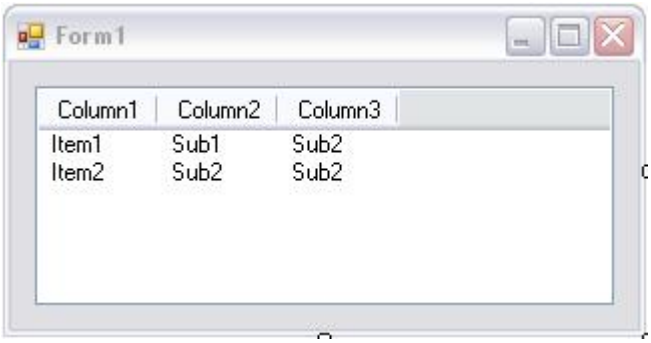
2. Derive the **ConfigurableListView** class from **ListView** and implement the **IUIConfiguration** interface.
3. Implement the **LoadState** and **SaveState** methods of the **IUIConfiguration** interface as follows.

```
using System;
using System.Windows.Forms;
using System.Xml;
using RepPP.Toolkit.Window;

namespace WindowsApplication8 {
    class ConfigurableListView : ListView, IUIConfiguration {
        #region IUIConfiguration Members

        public void LoadState(UIConfigurationHelper helper, XmlNode node) {
            XmlNode nodeCol;
            string  strWidth = null;
            int     iWidth;

            foreach (ColumnHeader col in Columns) {
                nodeCol = helper.FindNode(node, "col" + col.Text);
                if (nodeCol != null ) {
                    strWidth = helper.GetNodeAttribute(nodeCol, "WIDTH");
                }
                if (strWidth != null && Int32.TryParse(strWidth, out iWidth)) {
                    col.Width = iWidth;
                }
            }
        }

        public void SaveState(UIConfigurationHelper helper, XmlNode node) {
            XmlNode nodeCol;

            foreach (ColumnHeader col in Columns) {
                nodeCol = helper.FindNode(node, "col" + col.Text);
                if (nodeCol != null ) {
                    helper.SetNodeAttribute(nodeCol, "WIDTH", col.Width);
                }
            }
        }

        #endregion
    }
}
```

4. Build your project.
5. Pick the **ConfigurableListView** control from the toolbox and add it to the form developed in the previous section.



6. Using the *Properties* window, add a number of columns to the **ConfigurableListView** to test the persistence of the column widths.




7. Run your application, and modify the width of the columns of your **ConfigurableListView** control.
8. Restart your application. You should see that your columns have exactly the same width as in the last execution, and that the previous state of your application was successfully saved and restored.

Here is the content of the configuration file that was generated.

```
<REP>
  <FORMFORM1 VERSION="1" WINDOWSTATE="0" WIDTH="751" HEIGHT="272" LEFT="114" TOP="150">
    <CTLLISTVIEW1>
      <COLCOLUMN1 WIDTH="242" />
      <COLCOLUMN2 WIDTH="158" />
      <COLCOLUMN3 WIDTH="103" />
    </CTLLISTVIEW1>
  </FORMFORM1>
</REP>
```

Note that:

- The **ConfigurableListView** control needs the **UIConfigurationHelper** object in its containing form to do its magic.
- Although you used column widths in this example, you can persist ANY property.

## *Saving the user interface state to a different data store*

In the first section of this document, you saved the state of a standard Windows® form to an XML file. The code below shows you how to modify the form so that it saves/loads its user interface state to/from a different data store, such as a database, a registry, etc.

```
using System;
using System.Windows.Forms;
using RepPP.Toolkit.Window;
using System.IO;

namespace WindowsApplication8 {
    public partial class Form1 : Form {

        private UIConfigurationHelper m_configHelper;

        public Form1() {
            string          strUIState;
            StringReader    reader;
            InitializeComponent();
```

```
            strUIState     = LoadUIStateFromSomeWhere();
            reader         = new StringReader(strUIState);
            m_configHelper = new UIConfigurationHelper(reader);
            m_configHelper.RegisterForm(this);
        }

        protected override void OnFormClosed(FormClosedEventArgs e) {
            System.IO.StringWriter writer;

            base.OnFormClosed(e);
            writer = new System.IO.StringWriter();
            m_configHelper.SaveToWriter(writer);
            SaveUIStateToSomeWhere(writer.ToString());
        }


        private string LoadUIStateFromSomeWhere() {
            // TODO LOAD STATE FROM DATA STORE
            return (@"");
        }

        private void SaveUIStateToSomeWhere(string strUIState) {
            // TODO SAVE STATE TO DATA STORE
        }
    }
}
```