

Testing Windows.Forms views

A bit ugly but we can write integration tests involving Windows.Forms controls 😊

```
[Test]
[ NUnit.Framework.Category("Windows.Forms")]
public void CreateAndShowTransectDataView()
{
    var app = mocks.Stub<IApplication>();
    var gui = mocks.Stub<IGui>();
    var project = new Project();
    app.Project = project;
    gui.Application = app;

    var provider = new DurosPlusModelViewProvider {Gui = gui};

    var transectData = DurosPlusModelMockHelper.CreateSampleTransectData();

    project.RootFolder.Add(transectData); // <-- project must contain TransectData in order to create
DurosPlus model

    var view = (Control)provider.CreateView(typeof (TransectDataView), transectData);

    // show a message box when item is added to the project
    project.RootFolder.CollectionChanged += delegate(object sender, NotifyCollectionChangedEventArgs e)
    {
        log.DebugFormat("New item has been added: {0}", e.Item);

        // assert
        e.Item.Should("check if newly added item is Duros+ model").Be.TypeOf<DurosPlusModel>();
    };

    // select 2 checkboxes and click on a "Create Duros+ Models" button.
    var groupBoxSelection = (GroupBox)view.Controls["groupBoxSelection"];
    var checkedListBoxTransect = (CheckedListBox)groupBoxSelection.Controls["checkedListBoxTransect"];
    var checkedListBoxYear = (CheckedListBox)groupBoxSelection.Controls["checkedListBoxYear"];
    var buttonCreateDurosPlusModels = (Button)view.Controls["buttonCreateDurosPlusModels"];

    // do something after form is displayed
    Action<Form> formShownAction = delegate
    {
        checkedListBoxTransect.SetItemChecked(0, true);
        checkedListBoxYear.SetItemChecked(0, true);
        buttonCreateDurosPlusModels.PerformClick();
    };

    WindowsFormsTestHelper.ShowModal(view, formShownAction);
}
```

After test runs:

- 2 checkboxes are checked
- button is clicked
- model is created and added to the project
- assert checks if added item is of a proper type

The screenshot shows the WindowsFormsTestHelper application. On the left is a tree view with 'RootNode' and 'TransectDataView'. Below it is a properties window with 'Accessibility' and 'Appearance' sections. The 'Appearance' section shows 'BackColor' as 'Control', 'BackgroundImage' as '(none)', 'BackgroundImageLayout' as 'Tile', and 'BorderStyle' as 'None'. The main area is titled 'Selectie:' and contains a 'Kustvak:' dropdown menu set to 'Schier'. Below this are two list boxes: 'Raai:' with items 1 (checked) and 2, and 'Jaar:' with items 2010 (checked) and 2011. To the right of these is a 'Selectie eigenschappen:' panel showing 'Selected transects: 1' and 'Selected years: 2010'. At the bottom right are two buttons: 'Create Duros+ Models' and 'Create RT Models'.

WindowsFormsTestHelper

RootNode
TransectDataView

Selectie:

Kustvak:
Schier

Raai:
☒ 1
☐ 2

Jaar:
☒ 2010
☐ 2011

Selectie eigenschappen:
Selected transects:
1
Selected years:
2010

Accessibility
AccessibleDescrip
AccessibleName
AccessibleRole Default

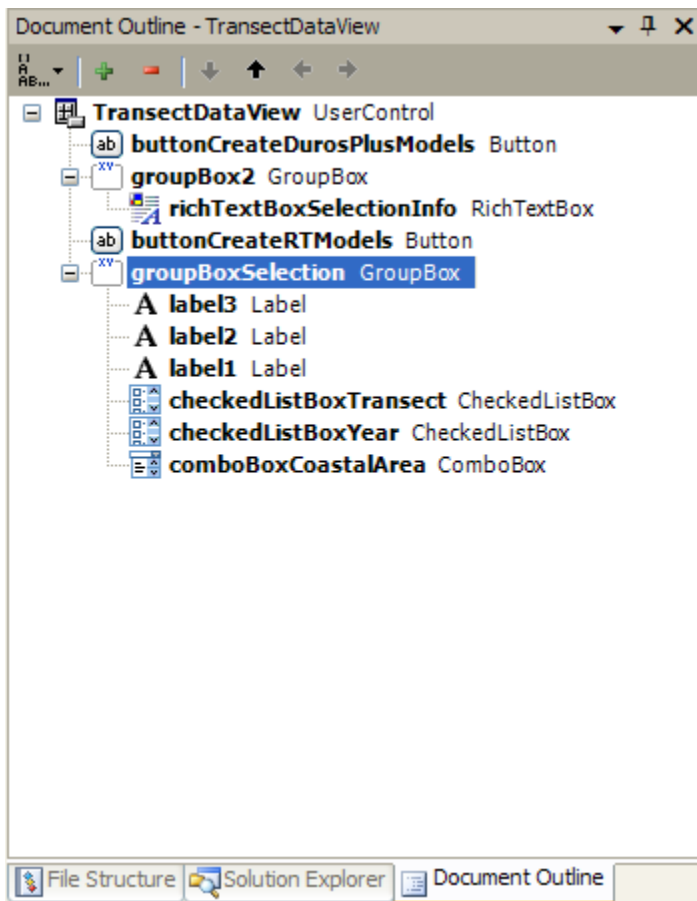
Appearance
BackColor Control
BackgroundImage (none)
BackgroundImageLayout Tile
BorderStyle None

Accessibility

Create Duros+ Models
Create RT Models

Some tips on how to use it:

- Make sure to name all controls on the form correctly
- Use **Document Outline** in Visual Studio to check names of controls, **make sure names are readable and intuitive!**



Probably we should refactor WindowsFormsTestHelper a little to make it more usable. Add any ideas as comments to the current blog post.