# TOPIC: How to set AutomationProperties for Controls

## Research Items

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Item Details** | **Status** | **URL** |
| 1 | AutomationProperty Definition | Done |  |
| 2 | How to set AutomationProperty (manually, automatically) | Done |  |

Reference Source:

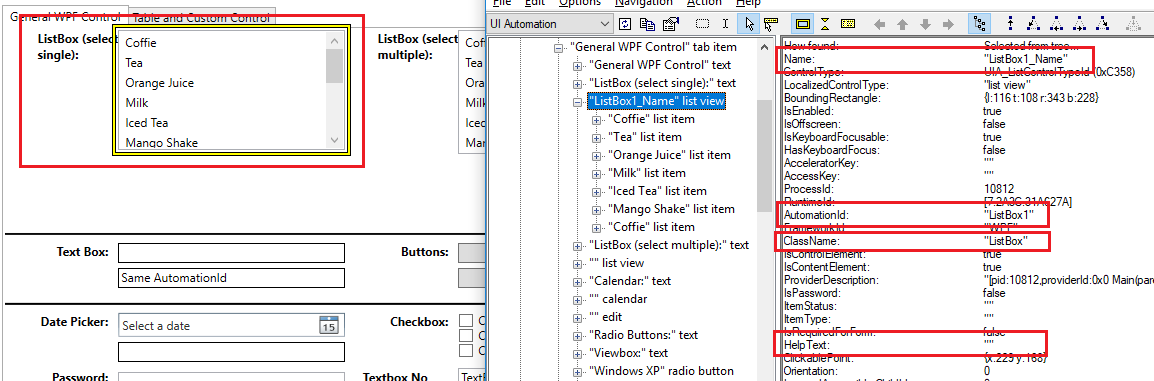
<https://docs.microsoft.com/en-us/visualstudio/test/set-a-unique-automation-property-for-windows-store-controls-for-testing?view=vs-2019>

## Research Details

1. **What is AutomationProperty?**

To automate on Window application, each control-under-test must be identified by a unique automation property. Developers can assign a unique automation property based on the type of XAML control in your application. To examine those properties on a Window application, many tools can be used such as: [Inspect](https://docs.microsoft.com/en-us/windows/desktop/winauto/inspect-objects), [VisualUIAVerify](https://docs.microsoft.com/en-us/windows/desktop/winauto/visual-ui-automation-verify)…

There are some properties which are commonly used to be identified: AutomationId, Name, ClassName and HelpText.



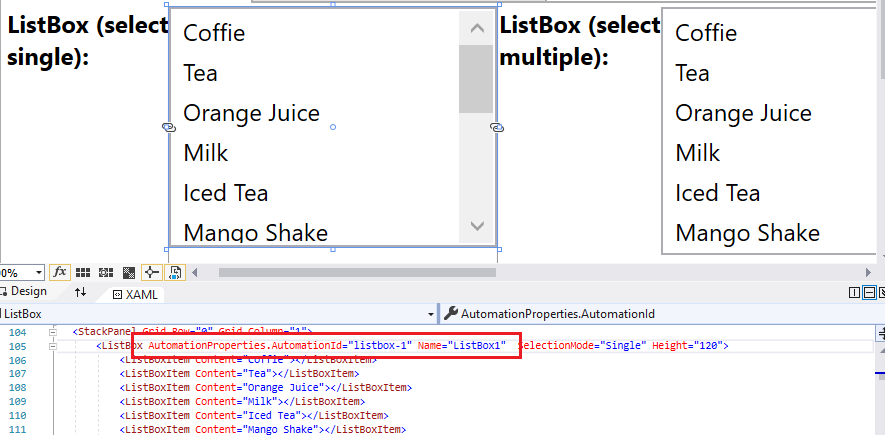
1. **How to set AutomationProperty manually?**

***2.1. AutomationId***

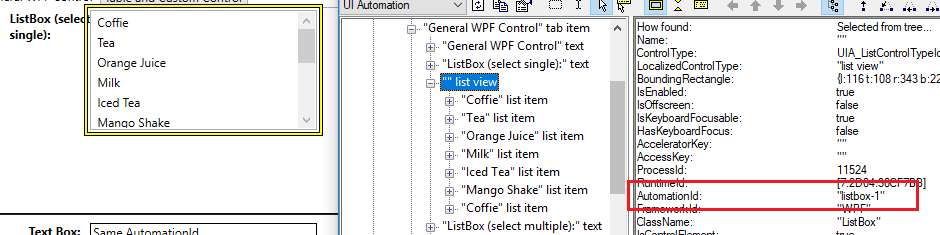
For a control to be identified by AutomationId, it should have *AutomationProperty.AutomationId* OR *Name*. If a control has both properties, AutomationId value will be *AutomationProperty.AutomationId* (higher priority). If a control only has *Name* property, AutomationId value will be *Name* (lower priority).

**Notes**: With *AutomationProperty.AutomationId*, any text value can be input. This is not the same for *Name*. *Name* should be unique throughout current scope, and should follow variable naming (as this property can also be used in CodeBehind).

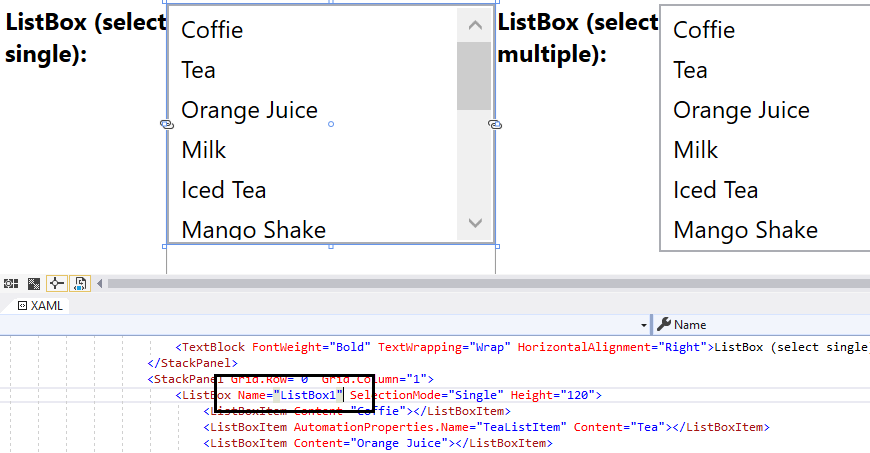
Example 1: A ListBox with both properties:



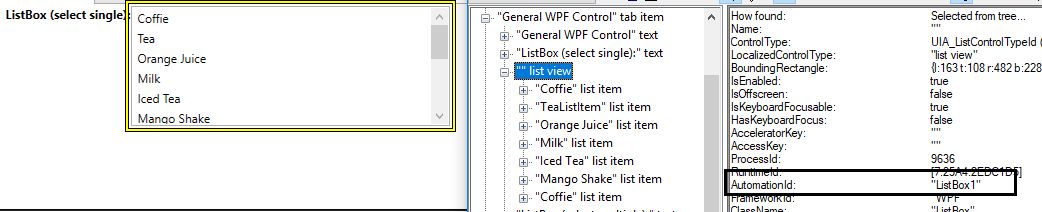
When inspecting this ListBox, AutomationId value is as below:



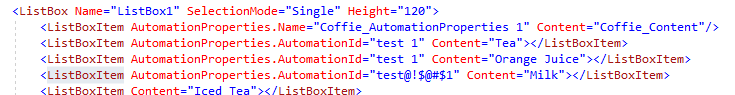
Example 2: A ListBox with only *Name* property:

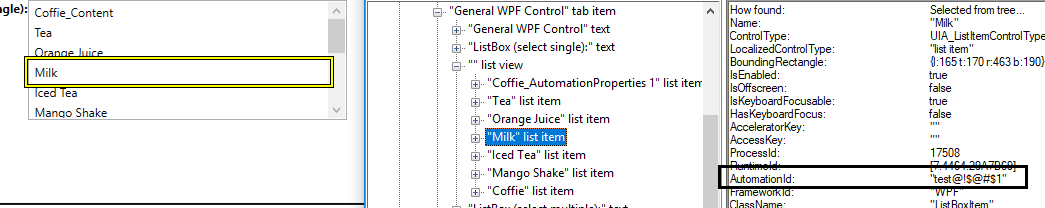


When inspecting this ListBox, AutomationId value is as below:

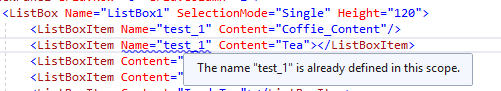


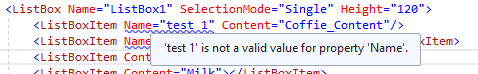
Example 3: *AutomationProperty.AutomationId* property can be specified with any text value:

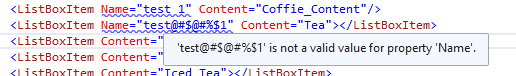


******

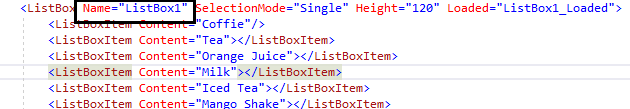
Example 4: *Name* property should be unique, following naming convention:

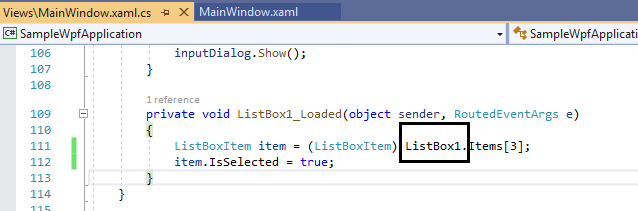
******

******

******

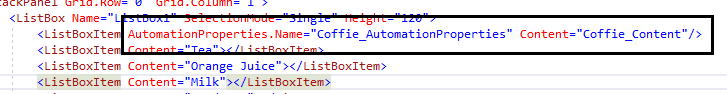
* **Reason:** *Name* property can be used in Code Behind to represent that control:

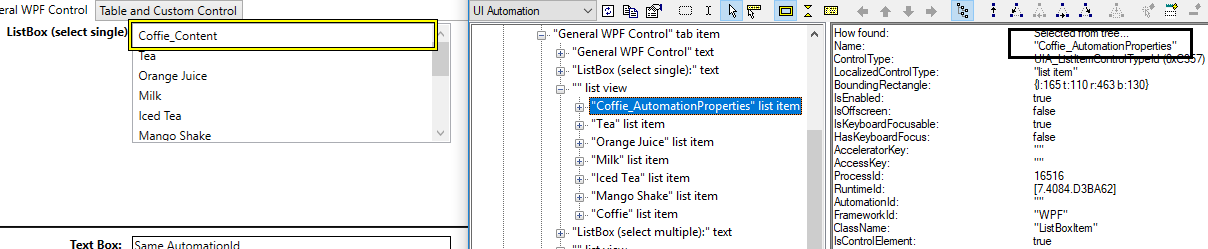
******

******

***2.2. Name***

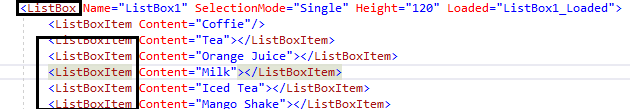
For a control to be identified by Name, it should have *AutomationProperty.Name* OR *Content* (the text of control). Same as AutomationId, higher priority is *AutomationProperty.Name*. Example: If a ListBoxItem has both properties, Name value will be *AutomationProperty.Name*.

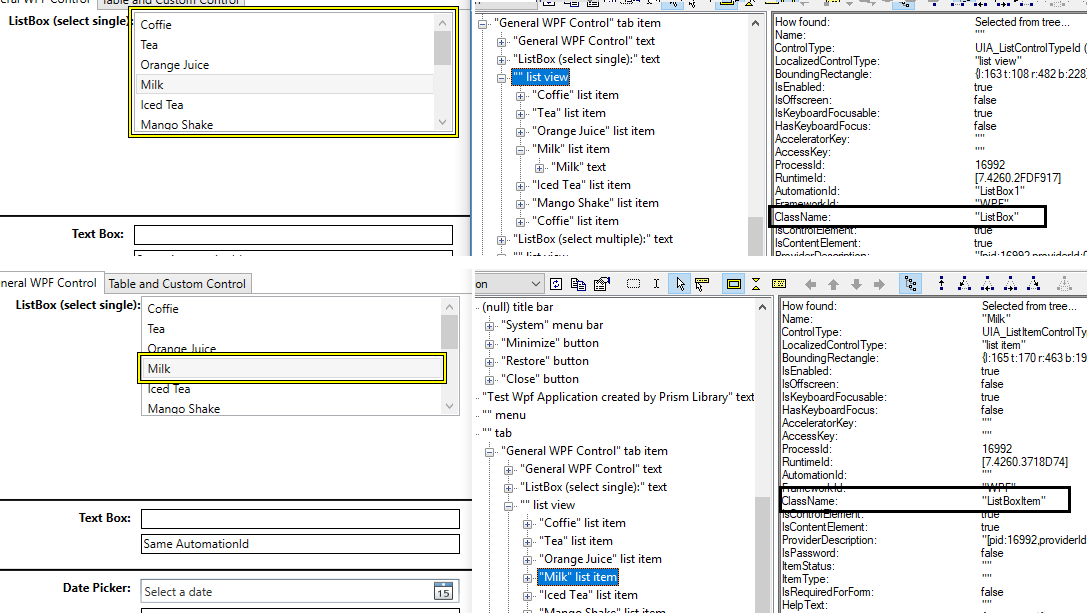




***2.3. ClassName***

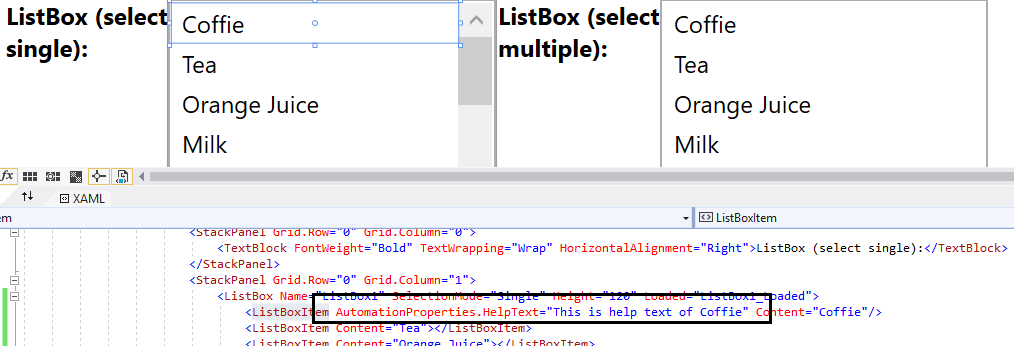
ClassName value is the type of control. For example:

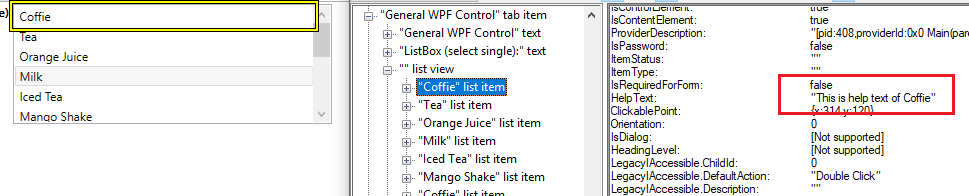




***2.4. HelpText***

HelpText value can be set with *AutomationProperties.HelpText* property. For example:





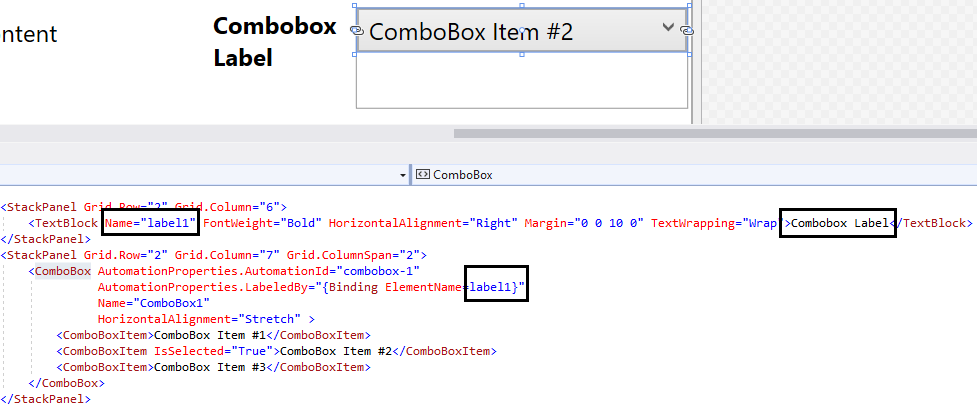
1. **How to set AutomationProperty automatically?**

AutomationProperty can be set automatically through Binding.

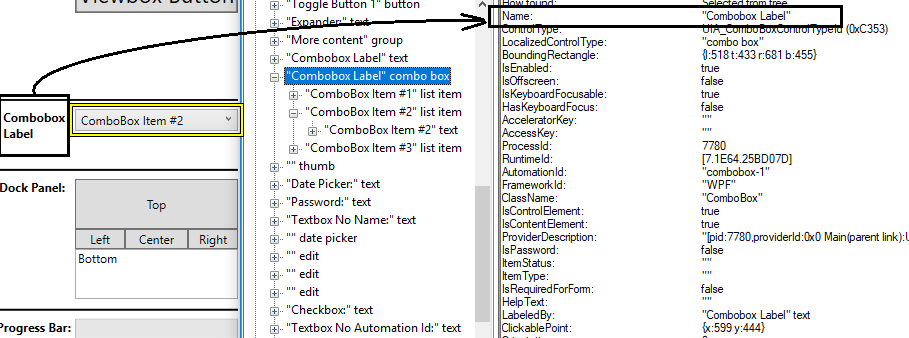
***3.1. Binding “Name” with AutomationProperty.LabeledBy***

This automation property is used to set an element that contains the text label for the control. Usually the text element should be a Label or TextBlock control. With this way, changing the label content will also change Name value of control.

Example: To identify below Combobox by Name without specify *AutomationProperties.Name*, developers can binding LabeledBy property to a TextBlock.

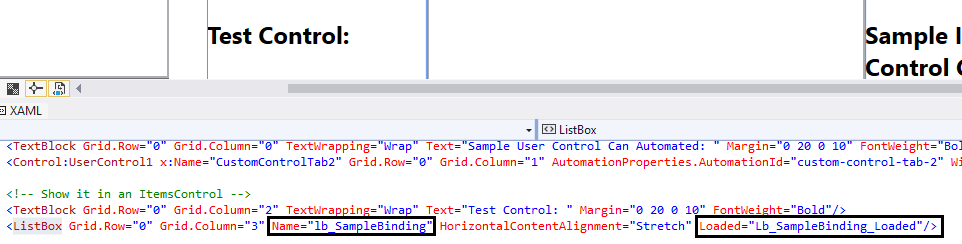


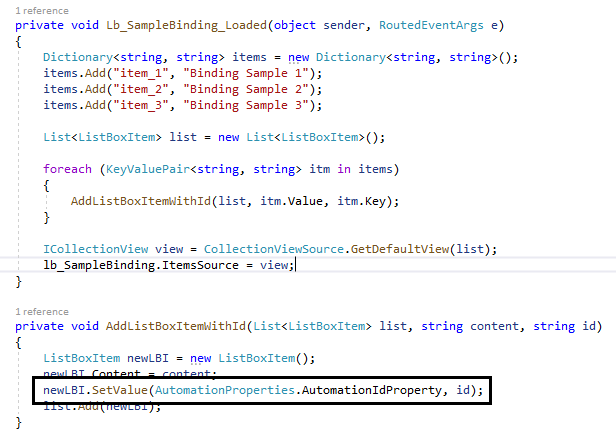
🡺 The result will be:



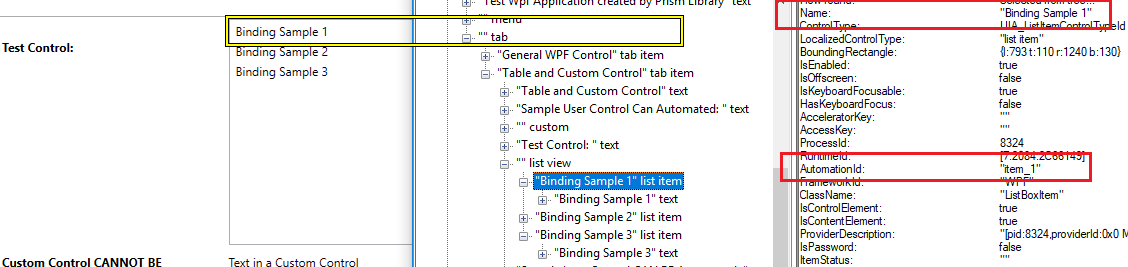
***3.2. Specify AutomationProperty with Code Behind***

AutomationProperty can be set in Code Behind, when a control is loaded. Example: Below example will add Items into a ListBox, each Item will have AutomationId and Content defined in a data source.



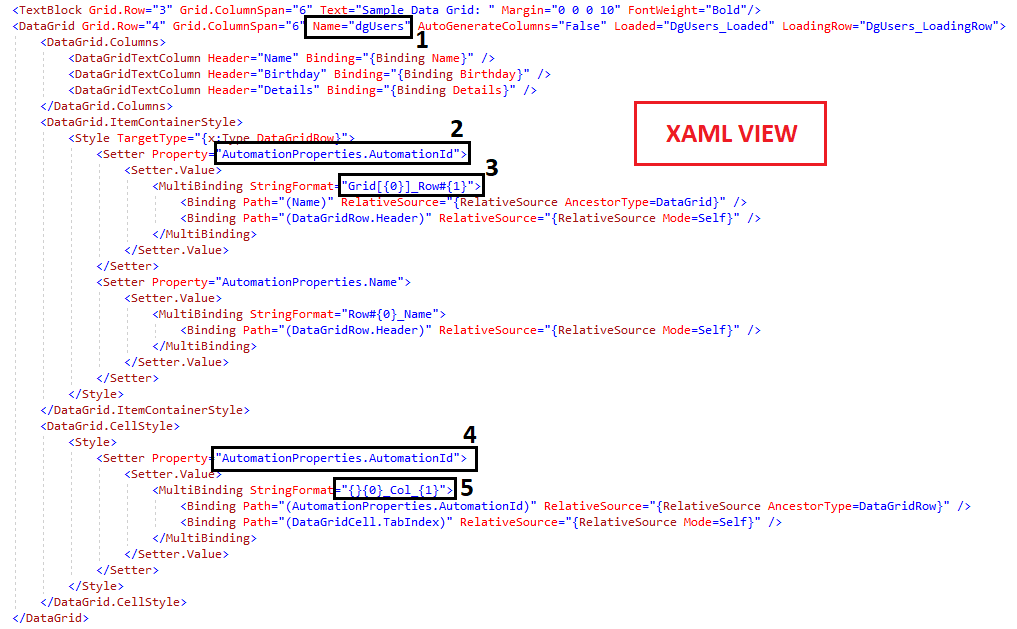


🡺 Result:



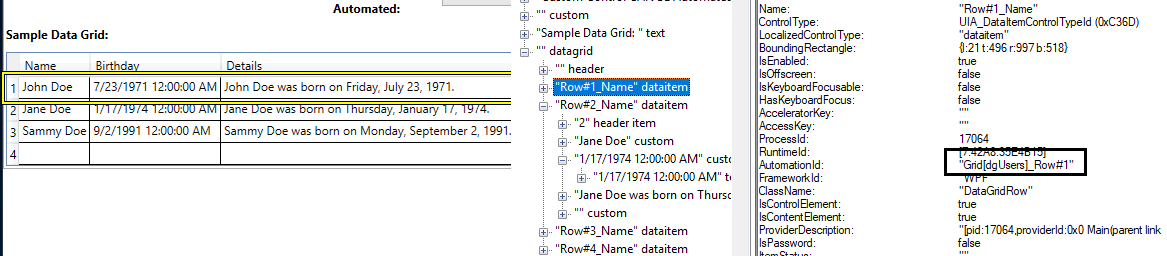
***3.3. Specify automation property on XAML View***

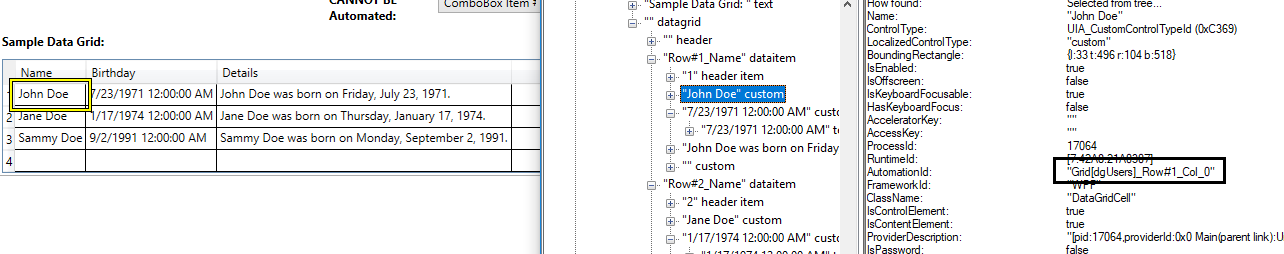
AutomationProperty can be set dynamically on XAML, by using binding. Example: Below example will add data into a DataGrid table, each Row and Cell will have **unique** AutomationId.





🡺 **Result**:





🡺 **Explanation**:

**(1)**: We set Name = “dgUsers” for DataGrid.

**(2)**: We try to set AutomationId for each DataGridRow in DataGrid by using: <Style TargetType="{x:Type DataGridRow}">

**(3)**: We want the AutomationId has format like: Grid[XXX]\_Row#YYY with XXX is the Name of DataGrid (set in 1) and YYY is the Header of current DataGridRow.

<Binding Path="(Name)" RelativeSource="{RelativeSource AncestorType=DataGrid}" /> 🡪 This will get the value of Name property of the parent control DataGrid and set it to {0}

<Binding Path="(DataGridRow.Header)" RelativeSource="{RelativeSource Mode=Self}" /> 🡪 This will get the value of Header of the current control DataGridRow (Self) and set it to {1}

* For Row 1, AutomationId will be: Grid[dgUsers]\_Row#1

**(4)**, **(5)**: Same explanation as above. The result is that each DataGridcell will have AutomationId = Grid[XXX]\_Row#YYY\_Col\_ZZZ (e.g.: Grid[dgUsers]\_Row#1\_Col\_1, Grid[dgUsers]\_Row#1\_Col\_2 …)

1. **Other mandatory Property**

Every Window control should have Title property. This is required for WhiteLibrary to switch back and ford from Window/Dialog to Window/Dialog.

