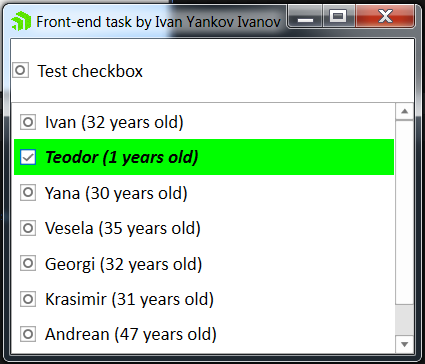
Hi there,

my name is Ivan Yankov. I am working for [Progress®](https://www.progress.com/) as WPF Front-end developer.

In this tutorial I will explain you how to create a custom WPF “CheckBoxList like” control.



Prerequisites

As a base we will use [RadListBox](https://docs.telerik.com/devtools/wpf/controls/radlistbox/overview) control, which is part of [Telerik UI for WPF suite](https://www.telerik.com/products/wpf).

For our WPF application we will apply the [Office2013theme](https://docs.telerik.com/devtools/wpf/styling-and-appearance/themes-suite/common-styling-appearance-office2013-theme), also made by [Telerik](https://www.telerik.com/).

You can download Telerik UI for WPF from [here](https://www.telerik.com/products/wpf).

If you prefer installing NuGet packages, follow this [link](https://docs.telerik.com/devtools/wpf/installation-and-deployment/installing-telerik-ui-on-your-computer/installation-installing-from-nuget-wpf).

Note that you have to [create account](https://www.telerik.com/login), if you do not have one already.

Preparing the Visual Studio project

I am assuming that you have the Telerik UI for WPF installed. Now let us start Visual Studio and create new C# WPF application (.NET Framework 4.5). Project name will be “FrontEndTask”.

From the Visual Studio file menu choose TELERIK >> Telerik UI for WPF >> Convert to Telerik WPF Application. “Convert Project Wizard” dialog will appear. Check the “Controls” checkbox and then hit “Next” button. On the next page select Office 2013 theme and then click the “Finish” button.

This wizard is very smart.

It will do several very important things for us:

* It will copy the WPF suite DLL files (into lib subfolder of our solution directory).
* It will add reference for Telerik.Windows.Controls.dll and Telerik.Windows.Themes.Office2013.dll (their NoXaml versions as explained below)
* It will add merged dictionaries to our App.xaml (needed for applying WPF themes and re-styling controls)

Note: We will use “implicit styles” mechanism for applying the Office 2013 theme. This mechanism requires NoXaml DLL references. If you do not know what I am talking about, please read the [Setting a Theme Overview](https://docs.telerik.com/devtools/wpf/styling-and-appearance/styling-apperance-setting-a-theme-overview) tutorial.

Code design and implementation

Nice! We are ready to insert the mentioned above RadListBox control. At later stage we will modify it.

As usual we have to create namespace declaration to use the Telerik controls in our XAML.

Please add the following declaration to your MainWindow.xaml and build the solution (F6):

xmlns:telerik="clr-namespace:Telerik.Windows.Controls;assembly=Telerik.Windows.Controls"

After this step, adding a RadListBox to your XAML becomes quite easy:

<telerik:RadListBox />

This control is designed to display its items, using the RadListBoxItem control.

* Our first goal is to change the behavior of the default RadListBoxItem control. This includes opacity changes (when the item is hovered) and color changes (when the item is selected). The opacity changes will be animated (fade in and fade out animations). The selected color will be hardcoded to Lime. To achieve all this, we have to modify the RadListBoxItem’s ControlTemplate property. It is always better to create style, which holds the control template inside of it. Due to we might want to use this style anywhere in our app, we will define it at App.xaml. It will be called FadingLimeRadListBoxItemStyle.
* Our second goal is to turn the original RadListBox into a listbox, which contains checkboxes. To do this, we have to modify the RadListBox’s ItemsTemplate property. We will change the RadListBox’s ItemContainerStyle property to apply the previously mentioned FadingLimeRadListBoxItemStyle. To achieve all this, we will create new style at App.xaml with the following name: CheckedRadListBoxStyle.
* Our final task is to change the default checkbox style. It will display circle image, even when the checkbox is not currently checked.

Before we proceed with actual styles writing, we have to add the following namespace declaration to our App.xaml:

xmlns:telerik1="clr-namespace:Telerik.Windows.Controls;assembly=Telerik.Windows.Controls"

We need this declaration, because we will use the RadListBoxItem style originally written by Telerik.

You can find it at \**Telerik UI for WPF installation directory*\*\Themes.Implicit\WPF40\Office2013\Themes\Telerik.Windows.Controls.xaml.

Open the file and search for “RadListBoxItemStyle”.

Code preview

You can preview how the MainWindow.xaml should looks like: [View MainWindow.xaml at paste.bin](https://pastebin.com/0tSqpGBg)

Here is how the App.xaml should looks like: [View App.xaml at paste.bin](https://pastebin.com/HAU3bwkg)

To add some example data to our “CheckBoxList like” control, I created a simple class called Person. Our modified RadListBox’s ItemsSource property is bound to a List<Person> at MainWindow’s code behind, using a dependency property called CheckedRadListBoxItemsSource.

You can view the full Visual Studio 2013 [project at Github](https://github.com/djlastnight/ProgressFrontEndTask).

If you have any questions, you can write me a letter to [mr.ivan.yankov@gmail.com](mailto:mr.ivan.yankov@gmail.com)

* Thank you for choosing our UI for WPF suite!

Greetings,

Ivan Yankov

WPF Front-end developer,

Progress Software Corp.

Mob: +359 988 808 138