Faculty of Computers, Informatics and Microelectronics

Technical University of Moldova

Event-Driven Programming

Laboratory work #6

Developing using C#. WinRT

Author:
Vasile Drumea

Supervisor:
Mihai Coșleț

Chișinău 2018

Laboratory work #6

1 Purpose of the laboratory

Gain knowledge about basics of developement of windows apps with C#.

2 Laboratory Work Requirements

- **Mandatory Objectives**
  - Create an Win32 or WinRT application using C#;
  - Choose one of:
    - Convert a previous laboratory work to C#;
    - Create a ToDo list. Should contain :
      - A list of tasks;
      - A way to add new tasks from UI;
- **Objectives With Points:**
  - `(3pt)` Create a report;
  - `(3pt)` Use WinRT;
  - `(4pt)` Create a pull request with a meaningful fix/feature to 5th WP laboratory work of your colleagues. It should be a project in which you were not involved.

3 Laboratory work implementation

3.1 Tasks and Points

- **Mandatory Objectives**
  - ✓Create an Win32 or WinRT application using C#;
  - Choose one of:
    - Convert a previous laboratory work to C#;
    - ✓Create a ToDo list. Should contain :
      - ✓A list of tasks;
      - ✓A way to add new tasks from UI;
- **Objectives With Points:**
  - `(3pt)` ✓Create a report;
  - `(3pt)` ✓Use WinRT;
  - `(4pt)` Create a pull request with a meaningful fix/feature to 5th WP laboratory work of your colleagues. It should be a project in which you were not involved.

Total : 6pt

3.2 Laboratory work analysis

I've implemented a ToDo list based on C# and XAML. The main component part of the application is the window named MainWindow.xaml. This contains 2 buttons and a listbox. A button is for adding a task and the other one is for deleting one. The main window is controlled by a C# class. In that class we have 4 methods :

```
public MainWindow()
private void AddButton_Click(object sender, RoutedEventArgs e)
private void DeleteButton_Click(object sender, RoutedEventArgs e)
private void Window_Closing(object sender, System.ComponentModel.Can
```

First method initializes the main window. Second handles the add operation of a new task. Third method handles the deletion of a task and the last method is responsible for savind the tasks in a file named Tasks.xml when the application is closed.

As I previously mentioned there is a xml file where we store the tasks and from where we get them when needed.

The tasks have 4 properties : Name, Priority, Done(Wether is done or not), and the description.

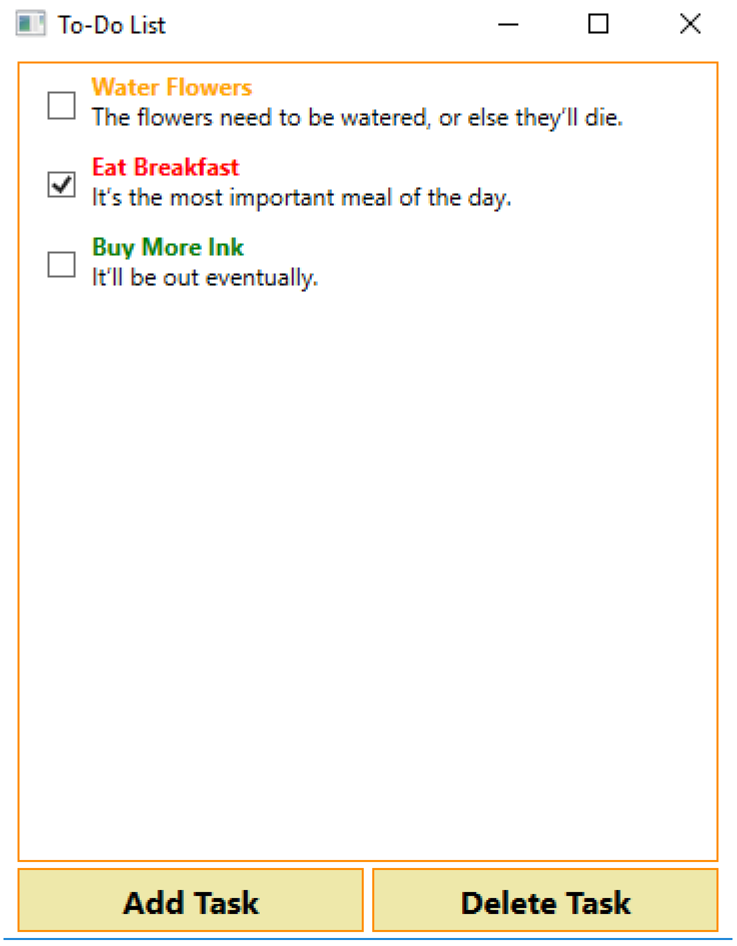
On the main window the name and descr. are shown in 2 rows. The done attribute is shown by a checkbox and the priority influences the color of name. The latest 2 are converted from text by 2 C# classes named :

```
PriorityConverter.cs // Converts to a color
StatusConverter.cs   // Converts to a checkbox
```

Additionally for the add operation we have another window, thus another xaml and a c# code that controlles it. It is a dialog box with a field for each attribute of a task in which we insert the information.

To create styles I've added Style tags in App.xaml Window.Resources tags. It makes the styles to be global. Also for each window I've modified style if needed, but also inherited the global style.

3.3 Prove your work with screens


To-Do List

Water Flowers
The flowers need to be watered, or else they'll die.

☑ Eat Breakfast
It's the most important meal of the day.

Buy More Ink
It'll be out eventually.

Figure 3.1 – ToDo List

Conclusion

In this laboratory work I've gained basic knowledge of the workflow with C# and XAML for developing windows desktop applications.

Behind every XAML which contains the resources and child windows and other controlles, the window is controlled by a C# script.

The execution shows that application works as expected.

References

1 To do List with C#, `https://www.aspfree.com/c/a/windows-scripting/completing-a-wpf-to-do-list-application/`