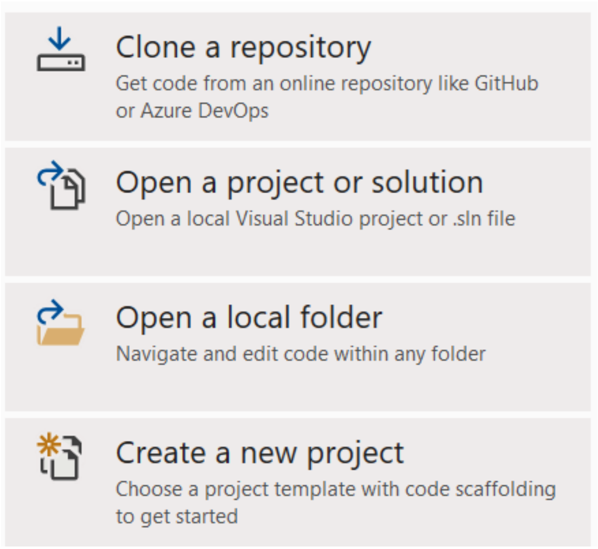
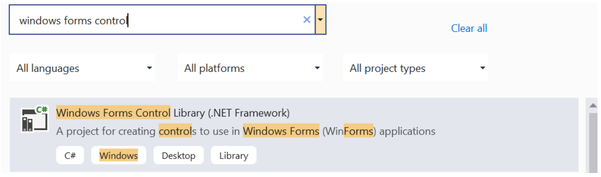
Introduction

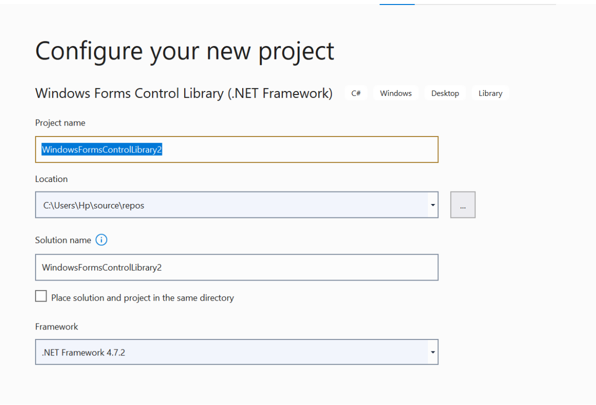
After a week of rigorous coding, Welcome back!

You have learned all about Event-Driven Programming and Desktop Applications in the previous lab manuals. Let's move on to the next, new, and exciting concepts.

In this Lab, We will learn about Custom Control Development.

Let's then move on with it.

 Clone a repository Get code from an online repository like GitHub or Azure DevOps Open a project or solution Open a local Visual Studio project or .sln file Open a local folder Navigate and edit code within any folder Create a new project Choose a project template with code scaffolding to get started	Create a New Project
 windows forms control Clear all All languages All platforms All project types Windows Forms Control Library (.NET Framework) A project for creating controls to use in Windows Forms (WinForms) applications C# Windows Desktop Library	Search and choose the Windows form Control Library with .Net Framework



Configure your new project

Windows Forms Control Library (.NET Framework) C# Windows Desktop Library

Project name: WindowsFormsControlLibrary2

Location: C:\Users\Hp\source\repos

Solution name: WindowsFormsControlLibrary2

☐ Place solution and project in the same directory

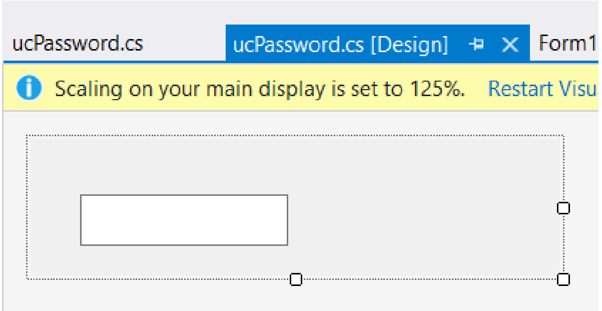
Framework: .NET Framework 4.7.2

Rename the Project and Click on Create.

Things to Remember:
This will create a similar window to what we have been working on within the previous labs in the case of windows forms.
However, the difference is that this code is not testable, we can only test the custom control functionality by “Building the solution” in the working window and then use the newly appended control in other projects.

Congratulations !!! You have learned how to create a project for creating Custom Controls in the .NET framework.

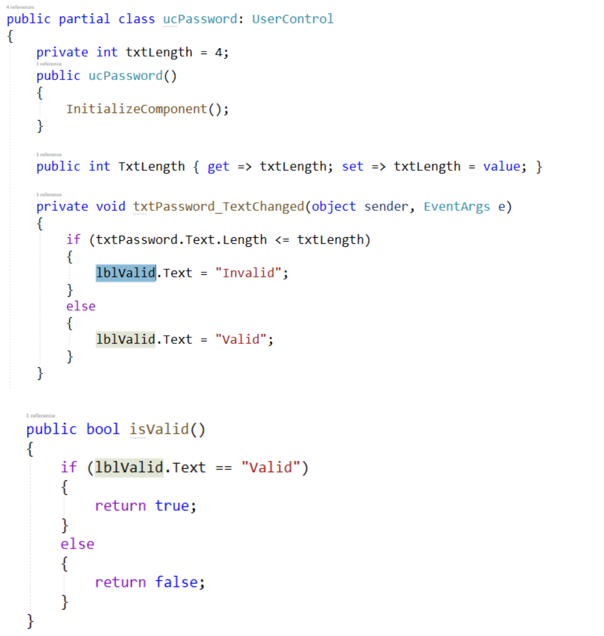
Task 01: Create a Custom Control Text Box that takes password of a certain length from the user and displays “Valid” or “Invalid” based on the text length constraint.
Solution:



Scaling on your main display is set to 125%. Restart Visu

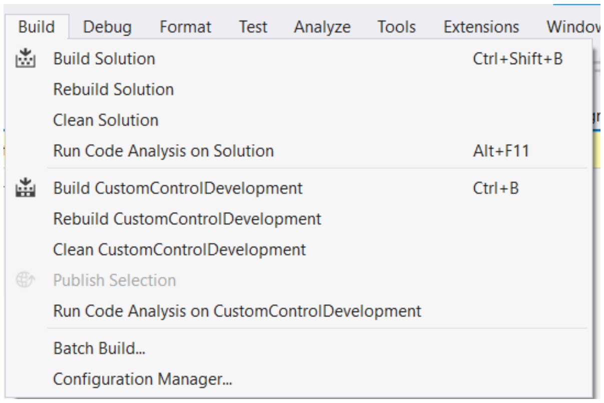
- Resize the main window.
- Drag and Drop a Text Box and a Label from the ToolBox.
- Rename the control components accordingly.

Note: We have removed the text from the label in spite of it is added in the Custom Control.



```
4 references
public partial class ucPassword: UserControl
{
    private int txtLength = 4;
    1 reference
    public ucPassword()
    {
        InitializeComponent();
    }

    1 reference
    public int TxtLength { get => txtLength; set => txtLength = value; }

    1 reference
    private void txtPassword_TextChanged(object sender, EventArgs e)
    {
        if (txtPassword.Text.Length <= txtLength)
        {
            lblValid.Text = "Invalid";
        }
        else
        {
            lblValid.Text = "Valid";
        }
    }


   1 reference
   public bool isValid()
   {
       if (lblValid.Text == "Valid")
       {
           return true;
       }
       else
       {
           return false;
       }
   }
```

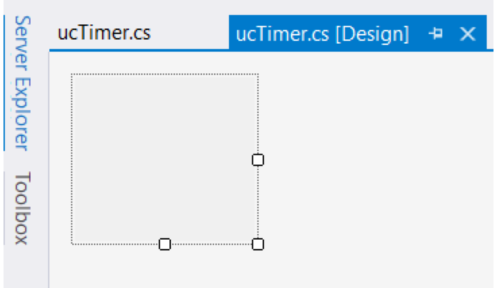
- Develop the Code Logic behind the Control Component.
- The data members you define here will appear at the end of the Property Window while using the control components.


Build
Debug
Format
Test
Analyze
Tools
Extensions
Window
Build Solution Ctrl+Shift+B
Rebuild Solution
Clean Solution
Run Code Analysis on Solution Alt+F11
Build CustomControlDevelopment Ctrl+B
Rebuild CustomControlDevelopment
Clean CustomControlDevelopment
Publish Selection
Run Code Analysis on CustomControlDevelopment
Batch Build...
Configuration Manager...

Build the Solution and Use the Control in Some Working Projects.

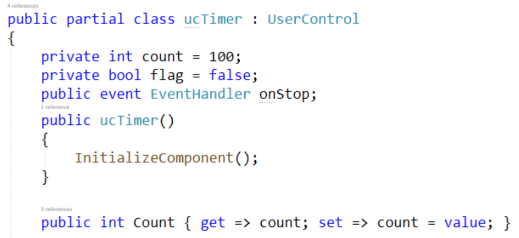
Task 02: Create a Custom Control Timer that displays a message when it reaches zero.

Solution:



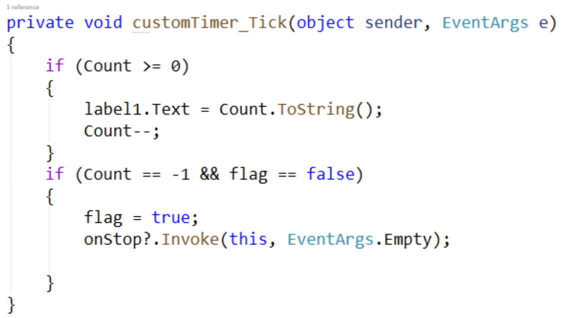
- Resize the Main Window
- Drag and Drop a Timer Control from the ToolBox Window.
- Rename the control component accordingly.

Note: We have removed the text from the label and timer in spite of these are added in the Custom Control.



```
4 references
public partial class ucTimer : UserControl
{
    private int count = 100;
    private bool flag = false;
    public event EventHandler onStop;
    1 reference
    public ucTimer()
    {
        InitializeComponent();
    }

    5 references
    public int Count { get => count; set => count = value; }
```

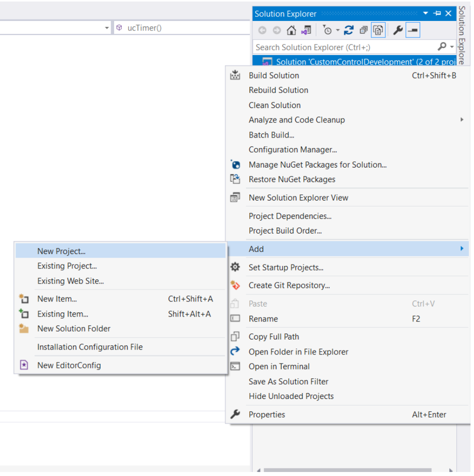
- Create the Property to start the Timer time.
- Create an EventHandler type variable to handle the event when the timer hits zero.



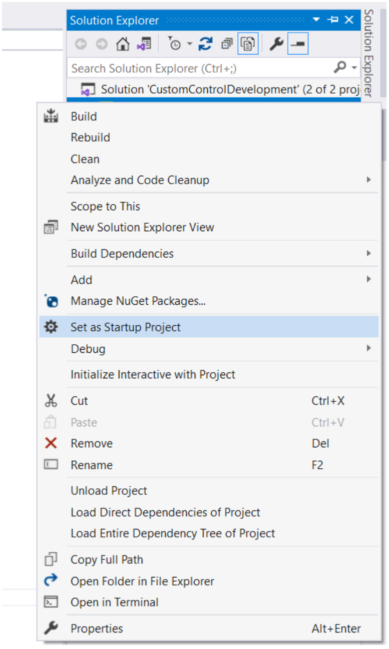
```
1 reference
private void customTimer_Tick(object sender, EventArgs e)
{
    if (Count >= 0)
    {
        label1.Text = Count.ToString();
        Count--;
    }
    if (Count == -1 && flag == false)
    {
        flag = true;
        onStop?.Invoke(this, EventArgs.Empty);

    }
}
```

- Set the control logic and implement it
- Build the solution and test the control in some other project

Tip of the Day:



- You can add a project into the existing namespace by following these steps.

<table>
<tr>
<td>


</td>
<td>

- Right Click on the added project and choose “set as startup project”

</td>
</tr>
<tr>
<td colspan="2">You can test the control into working forms code while building the control components at the same time.</td>
</tr>
</table>

There are some challenges on the next page. Give them a try.

Note: The submission is till Sunday midnight.

Custom Control Development

Challenge 01:

Create a custom **ComboBox.** This custom control will be the composition of 2 controls

1. **Combo Box**
2. **Text Box**

Make a property named **dataList** of string type that will be publicly available. User will set this property in the form while using the custom combo box. Whenever any item from the comboBox is selected it will be displayed in the textBox as well.

Challenge 02:

Note: You can only use textbox, buttons, and labels. You **can not** use Grid View to complete this challenge.

Create a custom **DataGridView**.

Make a property named **dataList** of **MUser** type that will be publicly available in the container (form). MUser contains 3 attributes of string type. Username, Password and Role.

1. Add 1 **Delete** Button in the user Control.
   When the user clicks the Delete button, custom control will check the selected Row of the grid and then delete the record from the list and then update the data grid. The custom control will also invoke an event that the data is successfully deleted from the datagrid view. And a message will be displayed from the form that the data is successfully deleted.

2. Similarly, add the **Search** button and a **Text Box** in the custom Control. When the search button is clicked, the custom control will read the username from the textBox that the user has entered and then search the whole list and find the username in the list. If the username is present then the custom control will invoke an event telling the container (form) that username is present in the list.

Challenge 03:

Previously, you had difficulty in adding the menu strips in your project as there were alot of separate forms and you had to copy paste the code in each form to add the menuStrip.
Now, make a **custom menuStrip** for your project. Maybe 2 menuStrips will be required.
1 for Admin and 1 for User.

Add it in your projects to avoid code repetition.

Good Luck and Best Wishes ‼

Happy Coding ahead :)

Department of Computer Science, UET Lahore.