# Creating a Desktop GUI with C# and Windows Forms

# Project Set Up

**Note this demo relies on the Windows operating system and the .NET Framework. It WILL NOT work on Linux or Apple machines.**

## Setup Steps for Visual Studio

From within VS (using point and click):

* Open the UsingHighLevelAppFrameworks solution
* Add a new Windows Forms App project called WinFormsDemo.csproj to the solution

## Now do the following:

* Make sure you are looking at the Form1.cs Design view and that you can see the Toolbox window (typically on the left hand side of the IDE) .
* Drag a Button and a Label control from the Toolbox and locate them at a site of your choosing on the form.
* Use the Properties window (usually located on the right hand side of the IDE beneath the Solution Explorer window) to set the Name property of the Button control to “CountButton” and its Text property to “OK”.
* Set the Name property of the Label control to CountLabel and its Text property to “0” (zero).
* Go to the Solution Explorer window, right-click on the Form1.cs file and select the View Code option.
* Replace all of the code in the file with the following:

namespace WinFormsDemo

{

public partial class Form1 : Form

{

private int count = 0;

public Form1()

{

InitializeComponent();

}

private void UpdateCount()

{

CountLabel.Text = $"Count: {count}";

}

private void CountButton\_Click(object sender, EventArgs e)

{

count++;

UpdateCount();

}

private void Form1\_Load(object sender, EventArgs e)

{

UpdateCount();

}

}

}

* Return to the Form1.cs Design window and click on the CountButton. Then select the lightning bolt symbol at the top of the Properties window. The window will now show you the events supported by Button controls.
* Click on the “Click” event, select the dropdown button that will appear to the right and select “CountButton\_Click”. This will ensure the code in the CountButton\_Click method (see above code listing) will be invoked whenever the CountButton is clicked.
* Now Select the Form itself in the Form1.cs Design window and return to the Properties window’s events list which will be displaying all the events supported by the underlying Form.
* Select the Load event and then Form1\_Load from the right hand drop down.
* Take a look at the code and try to work out what will happen when the application is run and the user interacts with the form.
* Run the application and confirm the app responds as you expect.

**Enhance the app so it retrieves data from some input controls**

* Add the following controls to the form:
  + A Label
    - Name => NameLabel
    - Text => “Name: “
  + A TextBox (alongside the NameLabel)
    - Name => NameTextBox
  + A Label
    - Name => AgeLabel
    - Text => “Age: “
  + A NumericUpdown (alongside the AgeLabel)
    - Name => NudAge
  + A Label
    - Name => MessageLabel
    - Text => “” (an empty string)
* Replace everything in Form1.cs (code view) file with the following:

namespace WinFormsDemo

{

public partial class Form1 : Form

{

private int count = 0;

public Form1()

{

InitializeComponent();

}

private void UpdateCount()

{

CountLabel.Text = $"Count: {count}";

}

private void CountButton\_Click(object sender, EventArgs e)

{

count++;

UpdateCount();

}

private void Form1\_Load(object sender, EventArgs e)

{

UpdateCount();

UpdateMessage();

}

private void NameTextBox\_TextChanged(object sender, EventArgs e)

{

UpdateMessage();

}

private void UpdateMessage()

{

if (!string.IsNullOrEmpty(NameTextBox.Text))

{

MessageLabel.Text =

$"Hello {NameTextBox.Text}, you are {NudAge.Value} years old!";

}

else

{

MessageLabel.Text = "";

}

}

private void NudAge\_ValueChanged(object sender, EventArgs e)

{

UpdateMessage();

}

}

}

* Make sure the CountButton’s Click event is still hooked up to the CountButton\_Click method and the For’s Load event is still hooked up to the Form\_Load method.
* Hook the NameTextBox’s TextChanged event up to the NameTextBox\_Changed method.
* Hook the NudAge control’s ValueChanged event to the NudAge\_ValueChanged method.
* Take a look at the code and try to work out what will happen when the application is run and the user interacts with the form.
* Run the application and confirm the app responds as you expect.

**Enhance the app so it has two forms, the second form showing a film that was released in the year the user was born**

* Add the following controls to form1:
  + A Button
    - Name => ShowMoviesButton
    - Text => “Show Movies: “
* Add the following method to the Form1.cs code file:

private void ShowMoviesButton\_Click(object sender, EventArgs e)

{

int YearOfBirth = DateTime.Now.Year - (int)(NudAge.Value);

Form form = new Form2(YearOfBirth);

form.ShowDialog();

}

* Hook the ShowMoviesButton’s click event to the ShowMoviesButton\_Click method.
* Amend the UpdateMessage method to look like the following:

// Replaces existing UpdateMessage code

private void UpdateMessage()

{

if (!string.IsNullOrEmpty(NameTextBox.Text))

{

MessageLabel.Text =

$"Hello {NameTextBox.Text}, you are {NudAge.Value} years old!";

**ShowMoviesButton.Enabled = true;**

}

else

{

MessageLabel.Text = "";

**ShowMoviesButton.Enabled = false;**

}

}

* Add a new Form to the project (calling it Form2)
* Add the following controls to the form:
  + A Button
    - Name => LoadMoviesButton
    - Text => “Load Movies: “
  + A DataGridView (stretch this out to make it nice and large)
    - Name => MoviesDGV
  + A NumericUpDown
    - Name => MovieYearNumberUpDown
* The form will need to access the Movies database so we’ll need to configure the app to use the Entity Framework.
* Use NuGet to add references to the following:
  + Microsoft.EntityFrameworkCore
  + Microsoft.EntityFrameworkCore.Design
  + Microsoft.EntityFrameworkCore.SqlServer
  + Microsoft.EntityFrameworkCore.Tools
  + Microsoft.Extensions.Configuration.Json
* Generate the Model classes by running the following line in the Package Manager Console. **NOTE, you may need to edit the Data Source to match your server credentials**:

dotnet-ef dbcontext scaffold "Data Source=(local); Initial Catalog=Movies; trusted\_connection=true" Microsoft.EntityFrameworkCore.SqlServer -o Movies

* Replace the lines in the Form2.cs code file with the following.

using System.Data;

using WinFormsDemo.Movies;

namespace WinFormsDemo

{

public partial class Form2 : Form

{

private MoviesContext context = null;

public Form2(int yearOfBirth = 1960)

{

InitializeComponent();

context = new MoviesContext();

if (yearOfBirth < NudMovieYear.Minimum || yearOfBirth > NudMovieYear.Maximum)

yearOfBirth = (int)NudMovieYear.Minimum;

NudMovieYear.Value = yearOfBirth;

}

private void Form2\_Load(object sender, EventArgs e)

{

List<Movie> movies =

context.Movies.Where(m => ((DateTime)(m.ReleaseDate)).Year

== (int)NudMovieYear.Value).OrderByDescending(m => m.Revenue).ToList();

MoviesDGV.DataSource = movies;

}

private void LoadMoviesButton\_Click(object sender, EventArgs e)

{

List<Movie> movies =

context.Movies.Where(m => ((DateTime)(m.ReleaseDate)).Year

== (int)NudMovieYear.Value).OrderByDescending(m => m.Revenue).ToList();

MoviesDGV.DataSource = movies;

}

private void NudMovieYear\_ValueChanged(object sender, EventArgs e)

{

LoadMoviesButton\_Click(sender, e);

}

}

}

* Hook the LoadMoviesButton control’s click event up to the LoadMoviesButton\_Click method.
* Hook the the NudMovieYear control’s ValueChanged event to the NudMovieYear\_ValueChanged method.
* Take a look at the code and try to work out what will happen when the application is run and the user interacts with the forms.
* Run the application and confirm the app responds as you expect. **Note: there are quite a few years for which there are no corresponding movies in the database**