AlteNET Form Designer for WinForms v7

Contents

Introduction

AlterNET Form Designer is a component library providing a quick and convenient way for creating visual user interfaces. It allows placing controls to the design surfaces, setting their initial properties and writing event handlers for their events.

What's included

The main component in the package is *FormDesignerControl*, which provides a design surface allowing users to add controls to a form, arrange them, and write code for their events.

Form Designer supports all common editing operations such as dragging, selecting and deleting components and controls; changing their size and z-order, align them horizontally or vertically, copy and paste controls. Like Visual Studio Form Designer, it serializes its content into C#/VisualBasic or TypeScript/JavaScript source code.

ToolboxControl – a visual control that displays components and controls that you can place onto the design surface. It provides a set of foldable tabs helping to organize controls by categories and allowing to specify which components and controls, including third-party controls, appear on the toolbox, on which tabs and sort order.

PropertyGridControl – a visual control allowing to view and change the design-time properties and events of the controls or components selected in the designer.

OutlineControl – a visual control allowing to show/hide controls on the form being designed.

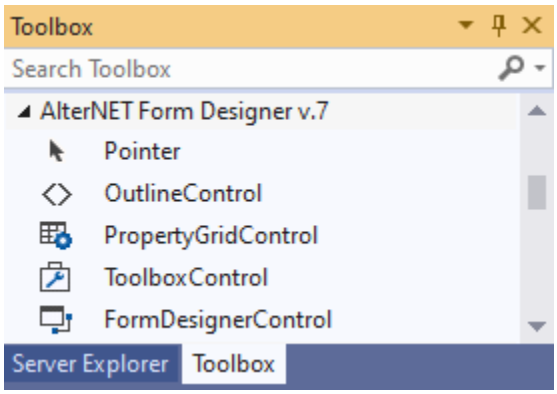
Form Designer Demo and QuickStart projects – these projects show how to place controls from the toolbox, load and save forms being designed, write code in control's event handlers, and how to run these forms.

Full Source Code - which comes with Universal edition.

Installation

AlterNET Form Designer is installed by the AlterNET Studio installer program. Advanced installation options include platform selection (WinForms, WPF or both)

Installation requires .NET Framework 4.6.1 + and Visual Studio 2015, 2017 or 2019 to be installed on the target machine. Installation program will register Visual Studio extensions and place FormDesignerControl along with ToolboxControl, PropertyGridControl and OutlineControl on the AlterNET Form Designer tab in Visual Studio toolbox.



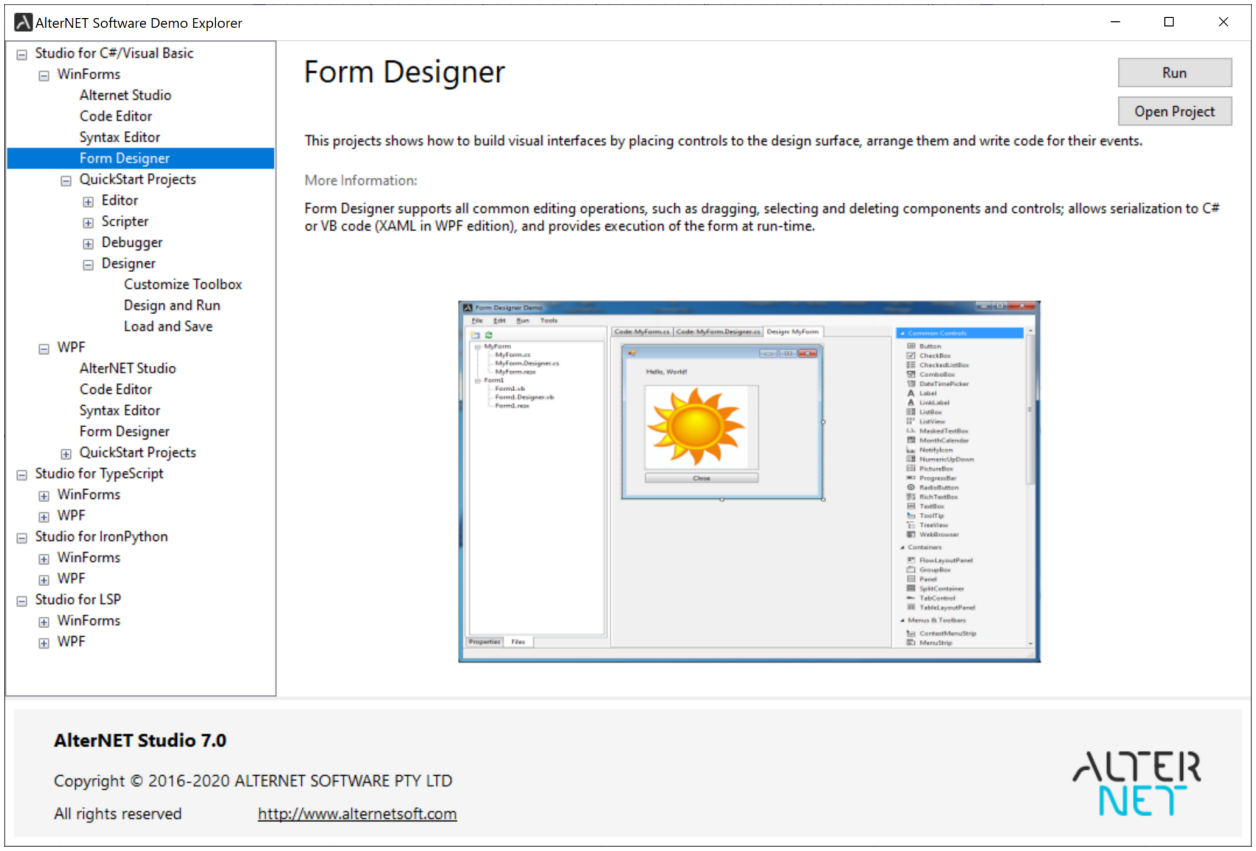
Other versions of .NET Framework 4.5.2,.NET Core 3.0,+ and Net 5.0 are available via NuGet packages. Complete list of NuGet packages can be found here:

https://alternetsoft.com/download

If you have a previous major version of AlterNET Studio, and decide to install the new one side-by-side, you will have two sets of Visual Studio Extensions and two sets of tabs, each one clearly displaying version number.

Getting Started

Once the product is installed, it might be good idea to explore quick start projects and Form Designer demo first, either by compiling Alternet.Studio.AllDemos.sln solution or accessing these demos through Demo Explorer tool which is added to Windows Start menu.



Below is brief overview of these projects:

FormDesigner - This project shows how to build visual interfaces by placing controls to the design surface, arrange them and write code for their events.

LoadAndSave – demonstrates how to save/load forms being designed.

DesignAndRun – shows how to write event handlers code and run the form being designed.

CustomizeToolbox – shows how to rearrange toolbox tabs and install third-party assemblies on the toolbox.

Once you've done with demo projects, it's time to create your own project with form designing functionality.

Essential steps are: Place FormDesignerControl control on the form, place ToolboxControl on the form, assign ToolboxControl.FormDesignerControl to the one you've just placed. Run the application.

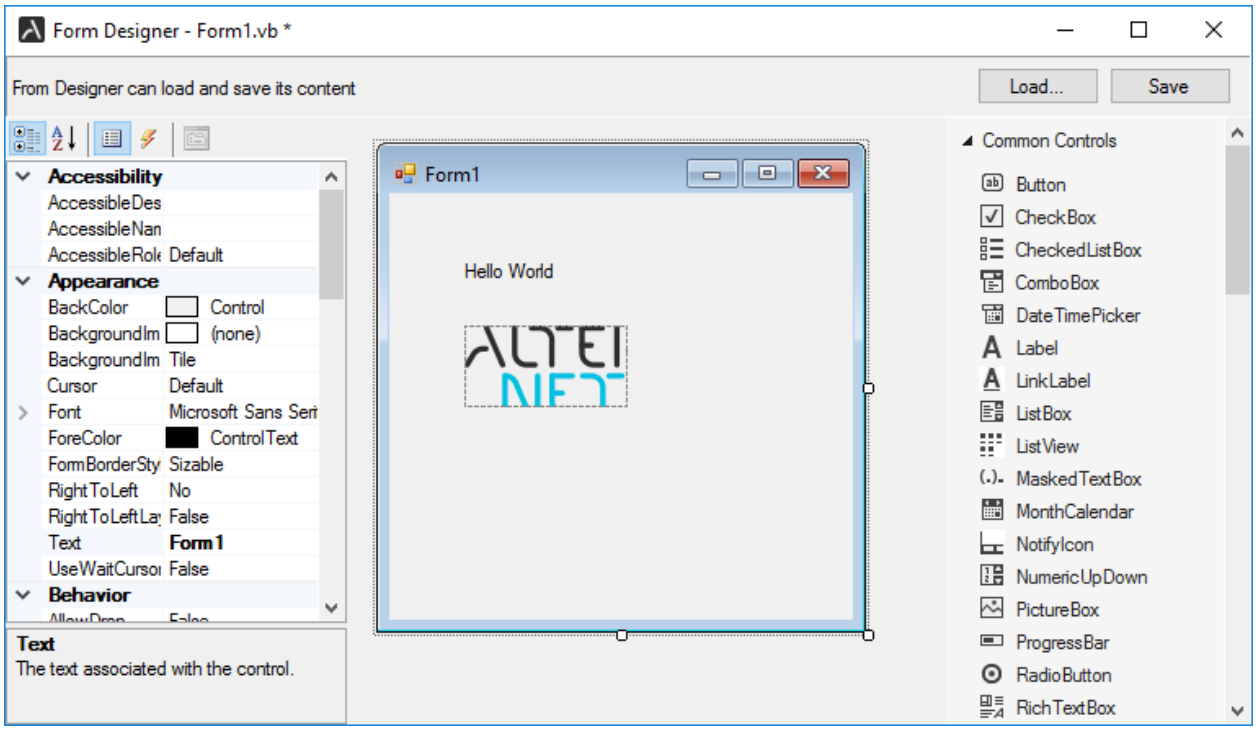
You will see a new empty form appearing on design surfaces, where you can drag controls from the toolbox, resize, arrange them, etc.

If you'd like to examine and change properties of the selected controls, you will need to place PropertyGridControl, set its FormDesignerControl property, so it displays a component or a control being currently selected in the designer.

Similarly, for navigation through form's control you can place OutlineViewControl and set its FormDesignerControl property.

Under the bonnet

FormDesigner control is based on .NET Framework FormDesigner services, which are implemented in System.ComponentModel.Design namespace and included in .NET framework. This allows AlterNET FormDesigner to look and feel very similar to Visual Studio WinForms Form Designer.



Below are most essential properties, methods and events of FormDesignerControl class:

Properties:

DesignerCommands – provides an interface to Form Designer commands, such as Copy/Paste, Undo/Redo, Aligning and Arranging controls, etc.

DesignerHost – Provides an interface for managing designer transactions and components.

IsModified – Indicates whether designer content has been modified since last save.

SelectedComponents – contains list of selected components or controls

PrimarySelection – gets first selected component or control.

Source – gets or sets FormDesigner Source.

ToolboxControl – gets or sets toolbox control associated with the designer.

ReferencedAssemblies – gets collection of assemblies where the controls and components used on the form being designed are declared.

ImportedNamespaces - in the case of Visual Basic, gets a collection of globally available namespaces.

Options – allows to change Form Designer appearance by specifying whether to display snap lines, smart tags, form designer grid and change grid size.

Events:

DesignerHostChanged – occurs when the designer host changes, for example if a new form is loaded.

NavigateToUserMethodRequested – occurs when form designer is requested to navigate to the event handler. For example, when a user double clicks on the control.

SelectionChanged – occurs when a user selects a different control in the designer.

CommandStateChanged – occurs when state of designer commands changes (for example when undo stack becomes available)

DesignedContentChanged – occurs when a user modifies any aspect of the control being designed.

LoadingErrorOccured – occurs when there is a parse error of the design code during loading.

CompilerErrorClick – occurs when a user clicks on a compiler error on the Form Designer surface.

DesignSurfaceKeyDown - when a user presses a key when the design surface is focused.

Methods:

Reload – reloads form to be designed from the source.

Save – serializes designer to C# / Visual Basic file or TypeScript / JavaScript code.

Form Designer works with three different source files simultaneously: one containing design-time code, another one containing user-written event-handlers and resource file for saving/loading form's resources

(such as images). Most often users will need to edit at least a file containing event handlers, which will require setting up FormDesigner controls's source to the one supporting integration with the editor. Please refer to *FormDesignerTextSource* class for the implementation of the source which integrates with SyntaxEdit control included in our demo projects.

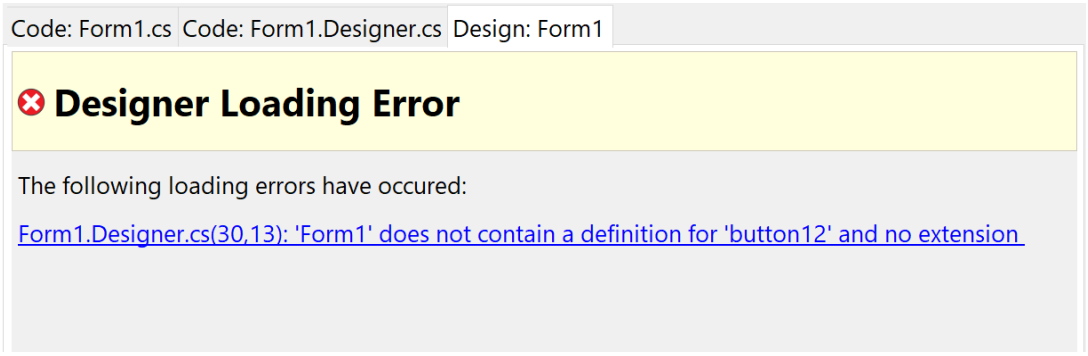
Error Handling

WinForms FormDesigner loads and saves its content into C#/VisualBasic or TypeScript/JavaScript code; this code needs to be correct both from syntax and semantic point of view. In case FormDesigner loader encounters an error in the code, it switches to error mode and displays a list of found errors, allowing a user to click and navigate to the error source by handling `CompilerErrorClick` event.

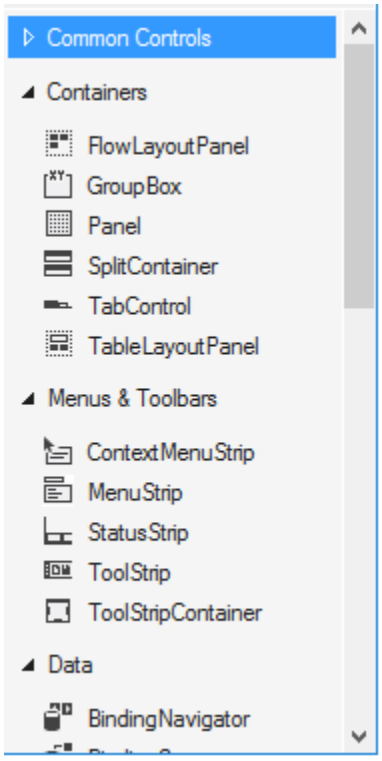
Note: specific language assembly handling code serialization needs to be included in the project to save/load Form designer content into the code.

C#/VisualBasic language services are implemented in Alternet.FormDesigner.Roslyn.v6 assembly;

TypeScript/JavaScript language services are implemented in Alternet.FormDesigner.TypeScript.v6 assembly.



ToolboxControl



The Toolbox control displays icons for controls and other items that you can add to the form being designed. It can be linked to FormDesignerControl by setting *FormDesignerControl* property which allows dragging components and controls to the Form Designer.

Below are essential properties and methods to manipulate Toolbox control:

CategoryNames – gets a collection of Categories (Tabs) displayed by the toolbox.

SelectedToolboxItem – returns currently selected toolbox item.

AddCategory – adds a new category to the toolbox.

Clear – deletes all tabs and items from the toolbox.

AddItem – places toolbox item onto specified toolbox tab.

ClearItemsInCategory – clears items in the specified tab.

GetAllTools – gets all toolbox items.

GetToolsFromCategory – gets toolbox items on the specific tab.

SelectPointer – deselects currently selected toolbox item and selects pointer tool.

RemoveCategory – removes a specific tab.

RemoveItem – removes a toolbox item from the category.

SetSelectedItem – selects toolbox item.

AddItemForType – adds toolbox item from type name

AddItemsFromAssembly – add all types that can appear on the toolbox from the assembly.

ScrollToCategory- scrolls toolbox control to the specified category.

ScrollToItem - scrolls toolbox control to the specified item.

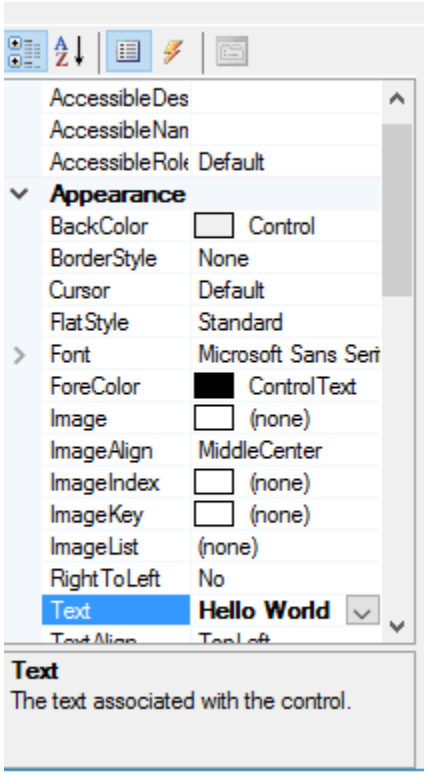
BeginUpdate – prevents repainting of the toolbox until EndUpdate is called

EndUpdate – re-enables toolbox repainting.

Save - saves the toolbox content to the specified stream.

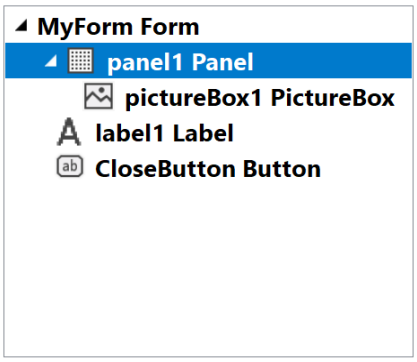
Load - loads the toolbox content from the specified stream.

PropertyGridControl



OutlineControl

OutlineControl displays Form's layout as a tree view, providing an easy way to navigate and re-arrange controls on the form.



Integration with other AlterNET Studio components

Most likely you would need a text editor of some sort in order to write custom code for the event handlers. The AlterNET Studio package includes Code Editor specialy tailored to edit C#, Visual Basic, TypeScript and JavaScript code.It also provides some additional features like modification only of portions of the design-time code which was affected by changes in form being designed. However you're not required to use it; you can choose any other external text editing components instead. Our demos are written the way it should be relatively easy to integrate them with other text editors by implementing *IScriptEdit* interface.

Form Designer comes with simple form runner based on C#/VisualBasic code compiler which is included in .NET framework; if you need to design multiple forms support or need a more powerful tool to run and debug code with your UI, a Scripter package might be more suitable.

Please refer to our AlterNET Studio demo to see how Code Editor, Form Designer and Scripter/Debugger work together.

Licensing

Form Designer requires a valid license to be installed for developing a .NET application that uses its components. We supply evaluation-licenses upon AlterNET Studio installation; these licenses are based on licx files technology provided by Microsoft and are valid for 30 days since first use.

Upon ordering a paid version of our product, you will receive a License key and will be able to activate it on your computer. This key will support a number of activations and it's not transferable between development machines otherwise.

Below is more information about Microsoft license compiler and some discussions related to intent and purpose of licx files, using them with source code control systems, etc.

https://msdn.microsoft.com/en-us/library/ha0k3c9f(v=vs.110).aspx

http://stackoverflow.com/questions/5628969/how-licenses-licx-file-is-used

http://stackoverflow.com/questions/51363/how-does-the-licenses-licx-based-net-component-licensing-model-work

In a nut-shell, once you drag FormDesigner control onto your Form, the licx file with the following content will be added to your project under Properties folder:

Alternet.FormDesigner.WinForms.FormDesignerControl, Alternet.FormDesigner.v7, Version=7.1.0.0, Culture=neutral, PublicKeyToken=8032721e70924a63

In case you create FormDesignerControl from code, please make sure licx file with such content is added to your project – you can use one from one of our demo projects if needed.

The design-time license is being checked when you work with this component at design-time, or when you compile your project; and the nag screen reminding you about the evaluation mode will appear once in a while. If you run a project compiled with an evaluation version of Code Editor (without launching it from Microsoft Visual Studio debugger), you will see a screen suggesting that the application was created with an evaluation version of Code Editor. Once you activate a paid license using LicenseActivation tool, nag screen will no longer appear.

When the evaluation period expires, you will still be able to compile and run your application from within Visual Studio, however applications created with expired license will not be run in standalone mode.