

Forum:[Hack Forums](#)

Thread: [\[GDI Development \] #1 : Control Designer](#)

Narwin - September 6, 2017, 12:10 pm

Intro

Maybe most of you noticed that some default controls that already exist in VS has some properties while others don't, for i.e : Panel is a common control that maybe everyone of us used even for once, it inherits ScrollableControl that has some properties different than control properties, in short when you add a panel to your project forms you simply see a small triangle in right top corner of the panel which is "Dock in Parent Container" property, have you ever thought how can we do this in Custom for our controls ? , in 3 lessons I'll try to teach you some GDI Techniques, and I'll start with control designer class, which has a lot of benefits for GDI Developers , unfortunately I don't see anyone here use it even for once (includes me).

[GDI Development]

Contains Three Lessons :

Spoiler [\(Click to View\)](#)
Control Designer.

Action List.

Child Classes.

Control Designer

From MSDN : ControlDesigner provides a base class for designers of components that derive from [Control](#). In addition to the methods and functionality inherited from the [ComponentDesigner](#) class, ControlDesigner provides additional methods to support extending and altering the behavior of an associated Control at design time.

As I said Control Designer has a lot of benefits and lot of features that can be found [here](#).

In this tutorial I will show one of the methods of Control Designer effect which is really important for GDI Developers and the users also and its [PostFilterProperties](#)

With PostFilterProperties we can change and remove any properties of a custom control.

Q : what is the need of removing properties ?

- for the most custom controls you just inherit the control base class which has a lot of none-needed

properties , like we don't need the "Background Image" property in evrey control's PropertyGrid , by this we can simply remove the un-wanted properties.

Let's turn this into the code

guess we have a custom button

C#.NET

Spoiler ([Click to View](#))

Visual Basic.NET

Spoiler ([Click to View](#))

Let's say we want to remove 4 properties of this control.
BackgroundImage

BackgroundImageLayout

Visible

Dock

we will use "PostFilterProperties" method.

for being much easy we will store all those 4 properties above in array.

C#.NET

```
Code private readonly string[] _PropertiesToRemove = { "BackgroundImage" ,  
"BackgroundImageLayout", "Visible", "Dock" };  
Visual Basic.NET
```

```
Code Private ReadOnly _PropertiesToRemove As String() = new String(){ "BackgroundImage" ,  
"BackgroundImageLayout", "Visible", "Dock" }
```

now we will create a class for the designer and inherit the ControlDsigner

Note : You need you import "System.Design" as referense.

C#.NET

Spoiler ([Click to View](#))

Visual Basic.NET

Spoiler [\(Click to View\)](#)

now let's join this class with the Custom Control above. for that we will go to the top of custom control and add this line :

C#.NET

Code [System.ComponentModel.Designer(typeof(NButtonDesigner))]
Visual Basic.NET

Code <System.ComponentModel.Designer(GetType(NButtonDesigner))>

The Whole Code :

C#.NET

Spoiler [\(Click to View\)](#)

Visual Basic.NET

Spoiler [\(Click to View\)](#)

Result

Spoiler [\(Click to View\)](#)

you can remove all properties except those you want to remain

C#.NET

Spoiler [\(Click to View\)](#)

Visual Basic.NET

Spoiler [\(Click to View\)](#)

Result

Spoiler [\(Click to View\)](#)

There are a lot of useful usings of ControlDsigner but this is enough for now.

Enjoy!

tweety - September 6, 2017, 12:12 pm

Who the hell still uses winforms lol

Narwin - September 6, 2017, 12:14 pm

(Sep 6, 2017 12:12 PM)Tweety. Wrote: [\[CITE\]](#) Who the hell still uses winforms lol
What's wrong with WinForms ?

xxdarklord65xx - September 6, 2017, 12:17 pm

thanks for sharing nice stuff mate .
i hope if u do more gdi posts would be great

Cuthbert - September 6, 2017, 12:33 pm

(Sep 6, 2017 12:12 PM)Tweety. Wrote: [\[CITE\]](#) Who the hell still uses winforms lol
WinForms is the best for quick GUI development.
Could you recommend something better?

sdk - September 6, 2017, 12:44 pm

(Sep 6, 2017 12:33 PM)SomeRandom Wrote: [\[CITE\]](#) WinForms is the best for quick GUI development.
Could you recommend something better?
Personally I love to fuck with WPF it's so easy to design something simple.

About 5-10mins of design work and thats cause I kept getting lost in a Netflix show :)
Spoiler [\(Click to View\)](#)
Sorry for the blurs still a work in progress (for a new thing not an already out thing :))

But to op nice tut keep up the good work

Cuthbert - September 6, 2017, 1:14 pm

(Sep 6, 2017 12:44 PM)sdk? Wrote: [\[CITE\]](#) Personally I love to fuck with WPF it's so easy to design something simple.

About 5-10mins of design work and thats cause I kept getting lost in a Netflix show :)
Spoiler [\(Click to View\)](#)
Sorry for the blurs still a work in progress (for a new thing not an already out thing :))

But to op nice tut keep up the good work

Isn't WPF quite time consuming to learn and develop, it's essentially learning a whole different language just for a GUI?

sdk - September 6, 2017, 1:22 pm

(Sep 6, 2017 01:14 PM)SomeRandom Wrote: [\[CITE\]](#) Isn't WPF quite time consuming to learn and develop, it's essentially learning a whole different language just for a GUI?
It's just XAML, took me a day or two to pick it up. I used to be big into Android dev so the whole XML view based development wasn't new to me.

In my eyes it's worth it to be able to make such easy designs and they are unique, especially if you do work for people, you can provide unique "themes" per person

uint32 - September 6, 2017, 2:01 pm

I usually prefer to avoid ControlDesigner, you can achieve the same effect using the EditorBrowsable ([MSDN](#)) andBrowsable ([MSDN](#)) attributes without over-complicating your code with long string arrays. This also allows you more functionality for runtime handling and such.

Narwin - September 6, 2017, 2:15 pm

(Sep 6, 2017 02:01 PM)Coefficient Wrote: [\[CITE\]](#) I usually prefer to avoid ControlDesigner, you can achieve the same effect using the EditorBrowsable ([MSDN](#)) andBrowsable ([MSDN](#)) attributes without over-complicating your code with long string arrays. This also allows you more functionality for runtime handling and such.

if you use EditorBrowsable or justBrowsable you need to declare every property you want to hide , which is make the job more difficult as I see.

and also there is another useful trick when you use ControlDesigner (Action Lists) which I mention it in the next tutorial.

uint32 - September 6, 2017, 2:41 pm

(Sep 6, 2017 02:15 PM)Narwin Wrote: [\[CITE\]](#) if you use EditorBrowsable or justBrowsable you need to declare every property you want to hide , which is make the job more difficult as I see.

and also there is another useful trick when you use ControlDesigner (Action Lists) which I mention it in the next tutorial.

It isn't much more difficult, just takes some more space. I do prefer to take the long path because if you need to rename a member and it's placed in a string literal, IntelliSense won't be able to find the reference and won't perform the rename in the string and that can lead to other errors that are harder to trace.

Recreating the property also allows you to control it's functionality at runtime; throwing exceptions, controlling input...

The only advantage I see to using ControlDesigner is to conditionally hide members based on user input and such.

Narwin - September 6, 2017, 3:06 pm

(Sep 6, 2017 02:41 PM)Coefficient Wrote: [\[CITE\]](#) It isn't much more difficult, just takes some more space. I do prefer to take the long path because if you need to rename a member and it's placed in a string literal, IntelliSense won't be able to find the reference and won't perform the rename in the string and that can lead to other errors that are harder to trace.

Recreating the property also allows you to control it's functionality at runtime; throwing exceptions, controlling input...

The only advantage I see to using ControlDesigner is to conditionally hide members based on user input and such.

let's you have a lot of custom controls and which everyone of them inherit a Control and you want to hide some shared properties in every control, you need to declare them separately or try to create a base class in for then inherit it but it still makes no difference with ControlDesigner, every property you hide inside it you can use it in runtime or change it, this just clarify the PropertyGrid from un-wanted properties, and as I said you need the ControlDesigner for implementing the ActionLists with make the jobs much easier and the user doesn't need to go for PropertyGrid to change a property value.

0xDEAD10CC - September 6, 2017, 3:29 pm

(Sep 6, 2017 02:41 PM)Coefficient Wrote: [\[CITE\]](#) It isn't much more difficult, just takes some more space. I do prefer to take the long path because if you need to rename a member and it's placed in a string literal, IntelliSense won't be able to find the reference and won't perform the rename in the string and that can lead to other errors that are harder to trace.

Recreating the property also allows you to control it's functionality at runtime; throwing exceptions, controlling input...

The only advantage I see to using ControlDesigner is to conditionally hide members based on user input and such.

Just as a note, this is why C# has nameof().

cristlol - September 6, 2017, 4:04 pm

Bookmarked!

Nice tutorial would be great if you post more =p

Thanks for sharing man.

sh@rp - September 7, 2017, 3:50 am

Very nice tut. keep up the good work.

thank you very much. ;)

?DiViN3 - September 7, 2017, 3:06 pm

Thats a great quality tutorial. I especially like how you include both languages.

myst3rio_666 - September 7, 2017, 5:26 pm

wow.. nice tutorial. dont mind the other people that dont even share their knowledge.

Rhez - September 9, 2017, 6:46 pm

You have typographical error.

(Sep 6, 2017 12:10 PM)Narwin Wrote: [\[CITE\]](#) Let's say we want to remove 4 properties of this control.

BackgroundImage

BackgroundImage

Visible

Dock

It should be BackgroundImageLayout, right?

Anyway, awesome tutorial. Well detailed information. ?

Narwin - September 10, 2017, 9:19 am

(Sep 9, 2017 06:46 PM)Rhez Wrote: [\[CITE\]](#) You have typographical error.

It should be BackgroundImageLayout, right?

Anyway, awesome tutorial. Well detailed information. ?

Right , I may typed quickly so that I was not aware of that but in the code there is the right one, fixed.

