

Forum:[Hack Forums](#)

Thread: [Win8Form](#)

maruata - November 30, 2014, 5:47 am

Here is another silly WinForm Theme.

Just for those who started to learn GDI Code.

I am presenting you very simple code for winform theme.

The form is too simple...please dont expect me to decorate this one.. I will release nice (style) form after someday!

The form is movable. But only with the TitleBar !

Give me a suggestion....

VB Code

```
Code Imports System.Collections.Generic
Imports System.ComponentModel
Imports System.Data
Imports System.Drawing
Imports System.Linq
Imports System.Text
Imports System.Windows.Forms
Imports System.Runtime.InteropServices
```

Namespace ZoSoft

Public Partial Class Win8Form

Inherits Form

'Variable

Public Const WM_NCLBUTTONDOWN As Integer = &Ha1

Public Const HT_CAPTION As Integer = &H2

<DllImportAttribute("user32.dll")> _

Public Shared Function SendMessage(hWnd As IntPtr, Msg As Integer, wParam As Integer, lParam As Integer) As Integer

End Function

<DllImportAttribute("user32.dll")> _

Public Shared Function ReleaseCapture() As Boolean

End Function

Private _backColor As Color = Color.FromArgb(235, 237, 238)

Private _borderColor As Color = Color.Black

Private _titleBar As New TitleBar()

'Enum

```

'Properties
Public Property BackgroundColor() As Color
    Get
        Return _backColor
    End Get
    Set
        _backColor = value
        Invalidate()
    End Set
End Property

Public Property BorderColor() As Color
    Get
        Return _borderColor
    End Get
    Set
        _borderColor = value
        Invalidate()
    End Set
End Property

Public Sub New()
    SetStyle(ControlStyles.SupportsTransparentBackColor Or
ControlStyles.OptimizedDoubleBuffer Or ControlStyles.Selectable, True)
    BackColor = Color.Fuchsia
    TransparencyKey = Color.Fuchsia

    'Controls
    Me.Controls.Add(_titleBar)
    AddHandler _titleBar.MouseDown, AddressOf TitleMouseDown

    InitializeComponent()
End Sub

Protected Overrides Sub OnPaint(e As PaintEventArgs)
    Dim g As Graphics = e.Graphics
    g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality

    'Size - Rectangle
    Dim fSize As New Rectangle(1, 1, Width - 2, Height - 2)

    'Fill BackColor
    g.FillRectangle(New SolidBrush(BackColor), fSize)

    'Draw border
    g.DrawRectangle(New Pen(New SolidBrush(BorderColor), 1F), fSize)

    'Draw Controls
    _titleBar.Location = New Point(1, 1)
    _titleBar.Size = New Size(Width - 2, 30)
    _titleBar.Font = Me.Font
    _titleBar.ForeColor = Me.ForeColor
    _titleBar.Text = Me.Text

```

```
_titleBar.Icon = Me.Icon
```

```
MyBase.OnPaint(e)  
End Sub
```

```
Private Sub TitleMouseDown(sender As [Object], e As MouseEventArgs)  
    ReleaseCapture()  
    SendMessage(Me.Handle, WM_NCLBUTTONDOWN, HT_CAPTION, 0)  
End Sub
```

```
Private Class TitleBar  
    Inherits Control  
    'Variable  
    Private _icon As Icon  
    Private _closeButton As New CloseButton()
```

```
    'Properties  
    Public Overrides Property Text() As String  
        Get  
            Return MyBase.Text  
        End Get  
        Set  
            MyBase.Text = value  
        End Set  
    End Property
```

```
    Public Overrides Property ForeColor() As Color  
        Get  
            Return MyBase.ForeColor  
        End Get  
        Set  
            MyBase.ForeColor = value  
        End Set  
    End Property
```

```
    Public Overrides Property Font() As Font  
        Get  
            Return MyBase.Font  
        End Get  
        Set  
            MyBase.Font = value  
        End Set  
    End Property
```

```
    Public Property Icon() As Icon  
        Get  
            Return _icon  
        End Get  
        Set  
            _icon = value  
        End Set  
    End Property
```

```

'Constructor
Public Sub New()
    SetStyle(ControlStyles.SupportsTransparentBackColor Or
ControlStyles.OptimizedDoubleBuffer Or ControlStyles.Selectable, True)

    BackColor = Color.Transparent
End Sub

Protected Overrides Sub OnPaint(e As PaintEventArgs)
    Dim g As Graphics = e.Graphics
    g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality

    'Size - Rectangle
    Dim fSize As New Rectangle(1, 1, Width - 1, Height - 1)

    'Fill
    g.FillRectangle(New SolidBrush(Color.FromArgb(189, 195, 199)), fSize)

    'Draw Title
    Dim fontHeight As Integer = Font.Height / 2
    Dim meHeight As Integer = Height / 2
    g.DrawString(Text, Font, New SolidBrush(ForeColor), 35, meHeight - fontHeight)

    'Draw Icon
    g.DrawIcon(Icon, New Rectangle(6, 4, 24, 24))

    'Border
    'g.DrawRectangle(new Pen(new SolidBrush(Color.Black), 1f), fSize);

    MyBase.OnPaint(e)
End Sub

End Class

```

```

End Class
End Namespace
C# Code
Code using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;

```

```

namespace ZoSoft
{
    public partial class Win8Form : Form
    {
        //Variable
        public const int WM_NCLBUTTONDOWN = 0xA1;
    }
}

```

```

public const int HT_CAPTION = 0x2;
[DllImportAttribute("user32.dll")]
public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);
[DllImportAttribute("user32.dll")]
public static extern bool ReleaseCapture();

private Color _backColor = Color.FromArgb(235,237,238);
private Color _borderColor = Color.Black;
private TitleBar _titleBar = new TitleBar();

//Enum

//Properties
public Color BackgroundColor
{
    get { return _backColor; }
    set { _backColor = value; Invalidate(); }
}

public Color BorderColor
{
    get { return _borderColor; }
    set { _borderColor = value; Invalidate(); }
}

public Win8Form()
{
    SetStyle(ControlStyles.SupportsTransparentBackColor | ControlStyles.OptimizedDoubleBuffer |
ControlStyles.Selectable, true);
    BackColor = Color.Fuchsia;
    TransparencyKey = Color.Fuchsia;

//Controls
this.Controls.Add(_titleBar);
_titleBar.MouseDown += TitleMouseDown;

InitializeComponent();
}

protected override void OnPaint(PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;

//Size - Rectangle
Rectangle fSize = new Rectangle(1, 1, Width - 2, Height - 2);

//Fill BackColor
g.FillRectangle(new SolidBrush(BackgroundColor), fSize);

//Draw border
g.DrawRectangle(new Pen(new SolidBrush(BorderColor), 1f), fSize);

```

```

//Draw Controls
_titleBar.Location = new Point(1, 1);
_titleBar.Size = new Size(Width - 2, 30);
_titleBar.Font = this.Font;
_titleBar.ForeColor = this.ForeColor;
_titleBar.Text = this.Text;
_titleBar.Icon = this.Icon;

base.OnPaint(e);
}

private void TitleMouseDown(Object sender, MouseEventArgs e)
{
ReleaseCapture();
SendMessage(this.Handle, WM_NCLBUTTONDOWN, HT_CAPTION, 0);
}

private class TitleBar : Control
{
//Variable
private Icon _icon;
private CloseButton _closeButton = new CloseButton();

//Properties
public override string Text
{
get
{
return base.Text;
}
set
{
base.Text = value;
}
}

public override Color ForeColor
{
get
{
return base.ForeColor;
}
set
{
base.ForeColor = value;
}
}

public override Font Font
{
get
{
return base.Font;
}
}
}

```

```

    }
    set
    {
        base.Font = value;
    }
}

public Icon Icon
{
    get { return _icon; }
    set { _icon = value; }
}

//Constructor
public TitleBar()
{
    SetStyle(ControlStyles.SupportsTransparentBackColor | ControlStyles.OptimizedDoubleBuffer |
ControlStyles.Selectable, true);
    BackColor = Color.Transparent;

}

protected override void OnPaint(PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;

    //Size - Rectangle
    Rectangle fSize = new Rectangle(1, 1, Width - 1, Height - 1);

    //Fill
    g.FillRectangle(new SolidBrush(Color.FromArgb(189, 195, 199)), fSize);

    //Draw Title
    int fontHeight = Font.Height / 2;
    int meHeight = Height / 2;
    g.DrawString(Text, Font, new SolidBrush(ForeColor), 35, meHeight - fontHeight);

    //Draw Icon
    g.DrawIcon(Icon, new Rectangle(6, 4, 24, 24));

    //Border
    //g.DrawRectangle(new Pen(new SolidBrush(Color.Black), 1f), fSize);

    base.OnPaint(e);
}

}
}

```

Save the code or add it into your project. You just have to inherit the Class

For example:

```
private class Form2 : Form
```

replace this to

```
private class Form2 : Win8Form
```

SaketSaket - November 30, 2014, 6:49 am

ohh.... You inherited Form for your theme instead of container control.... Nice start... keep coding... ;)

Cornflower Blue ???-? — - November 30, 2014, 7:10 am

(Nov 30, 2014 07:49 AM)SaketSaket Wrote: [\[CITE\]](#) ohh.... You inherited Form for your theme instead of container control.... Nice start... keep coding... ;)

Isn't the ContainerControl part of Aeons theme base? If it is then what he is doing it right.

Only3 - November 30, 2014, 7:23 am

(Nov 30, 2014 08:10 AM)Cornflower Blue ???-?— Wrote: [\[CITE\]](#) Isn't the ContainerControl part of Aeons theme base? If it is then what he is doing it right.

?

<http://msdn.microsoft.com/en-us/library/system.windows.forms.containercontrol%28v=vs.110%29.aspx>

u can use both form or containercontrol depend on what u use it for

SaketSaket - November 30, 2014, 10:19 am

(Nov 30, 2014 08:10 AM)Cornflower Blue ???-?— Wrote: [\[CITE\]](#) Isn't the ContainerControl part of Aeons theme base? If it is then what he is doing it right.

Container control is used by Aeonhack for themebase but it's not written by him... It's written by microsoft...

Form itself is inherited from container control... ;)

So what OP is doing is little weird i think, i may be wrong but i think he should inherit from container control instead of Form...

Regards

Saket

Nikey646 - November 30, 2014, 10:24 am

(Nov 30, 2014 11:19 AM)SaketSaket Wrote: [\[CITE\]](#) Container control is used by Aeonhack for themebase but it's not written by him... It's written by microsoft...

Form itself is inherited from container control... ;)

So what OP is doing is little weird i think, i may be wrong but i think he should inherit from container control instead of Form...

Regards

Saket

Except that rather than inheriting from a Control Type, You Inherit from the Form Type, and replace "Form" in a C# 'Form1.cs' file. This will make it use the custom Form Type, without adding a control on top of the form.

Eternal Darkness - November 30, 2014, 10:38 am

@OP: Take a Look at these (may be useful) [Don't just copy ant paste, but they may contain elements which assist with a Windows 8 theme]:

<https://hackforums.net/showthread.php?tid=2924071>

<https://hackforums.net/showthread.php?tid=2929512>

(Nov 30, 2014 11:19 AM)SaketSaket Wrote: [\[CITE\]](#) Container control is used by Aeonhack for themebase but it's not written by him... It's written by microsoft...

Form itself is inherited from container control... ;)

So what OP is doing is little weird i think, i may be wrong but i think he should inherit from container control instead of Form...

Regards

Saket

Container Control is a themeable class in the themebase. The themebase has a ton of code which simplifies skinning controls with GDI+ (code that "should" eventually be dropped in favor of more-so GDI+ hard/hand coding). I do not believe he claims to have coded any control in the Themebase (ie. TextBox, GroupBox, etc).

The Container Class is inherited by a new form you mean?

OP is doing some pretty normal stuff... It is the logical next step of GDI theme-ing. Replace the form, rather than adding a new random container control "as the form" (setting the actual form style to be as unnoticeable as possible, ex. form border style none), ability to actually simplify the code, and it allows the user to learn something about inheritance [as well as the fact that it is arguably easier to convert the "Form" value to "CustomFormName", rather than operate some custom container control's (that don't set form border style to none, that don't dock/anchor in the container, etc)]...

This is a long post to simply say, What he is doing is normal. If I were to release a program that were themed I would do the same ting. What we have gotten used to is weird... Adding another control, to make the theme drag and drop-able... Somewhat Inefficient, I would rather see OP's method become the new standard (even though I doubt it will happen)...

inb4 the hate train rolls by and regurgitates that changing one line of code is "too difficult"/weird, and they want to just drag and drop the unnecessary control onto the form... ?

Nikey646 - November 30, 2014, 2:56 pm

(Nov 30, 2014 11:38 AM)Eternal Darkness Wrote: [\[CITE\]](#) @OP: Take a Look at these (may be useful) [Don't just copy ant paste, but they may contain elements which assist with a Windows 8 theme]:

<https://hackforums.net/showthread.php?tid=2924071>

<https://hackforums.net/showthread.php?tid=2929512>

Container Control is a themeable class in the themebase. The themebase has a ton of code which simplifies skinning controls with GDI+ (code that "should" eventually be dropped in favor of more-so GDI+ hard/hand coding). I do not believe he claims to have coded any control in the Themebase (ie. TextBox, GroupBox, etc).

The Container Class is inherited by a new form you mean?

OP is doing some pretty normal stuff... It is the logical next step of GDI theme-ing. Replace the form, rather than adding a new random container control "as the form" (setting the actual form style to be as unnoticeable as possible, ex. form border style none), ability to actually simplify the code, and it allows the user to learn something about inheritance [as well as the fact that it is arguably easier to convert the "Form" value to "CustomFormName", rather than operate some custom container control's (that don't set form border style to none, that don't dock/anchor in the container, etc)]...

This is a long post to simply say, What he is doing is normal. If I were to release a program that were themed I would do the same thing. What we have gotten used to is weird... Adding another control, to make the theme drag and drop-able... Somewhat Inefficient, I would rather see OP's method become the new standard (even though I doubt it will happen)...

inb4 the hate train rolls by and regurgitates that changing one line of code is "too difficult"/weird, and they want to just drag and drop the unnecessary control onto the form... ?

I think the current standard of it being a drop-in control is primarily due to the fact that Themes were targeted to all users, not just users who know how to change the inheritance of the class. Also in C#, it's quite simple to change a classes inheritance, because it's in Form1.cs, but i believe in VB it's actually in Form1.Designer.vb, rather than Form1.vb, and if i recall correctly, navigating to hidden files like that in VB is a lot harder then C#. Of course, this post is based off of pure speculation and past experience.

Long story short, too many copy and paste skids who want drag and drop code (literally), rather then doing anything themselves. (Hint: AEDEAD.)

Eternal Darkness - November 30, 2014, 3:09 pm

(Nov 30, 2014 03:56 PM)Nikey646 Wrote: [\[CITE\]](#) I think the current standard of it being a drop-in control is primarily due to the fact that Themes were targeted to all users, not just users who know how to change the inheritance of the class. Also in C#, it's quite simple to change a classes inheritance, because it's in Form1.cs, but i believe in VB it's actually in Form1.Designer.vb, rather than Form1.vb, and if i recall correctly, navigating to hidden files like that in VB is a lot harder then C#. Of course, this post is based off of pure speculation and past experience.

Long story short, too many copy and paste skids who want drag and drop code (literally), rather then doing anything themselves. (Hint: AEDEAD.)

Either way it is in the designer code. Before you think about using custom controls, you should have the basic experience of seeing whats going on in the background as you build your form (hopefully, even though I assume it is usually not the case). Whether it is in C# or VB you should know how to see the Designer Generated code before you think about theme-ing (Again, I know this isn't usually the case, but it would be best if it were so). I would rather see themes teach programmers necessary skills and something as simple as changing a 10 character part of a line, rather than adding a otherwise unnecessary control, messing with the form, and making a "complicated" way to drag and drop...

People are going to skid either way. I'd rather take the "ease" of drag and drop away (for the form control), and teach users about inheriting custom classes and stuff. Inheritance should be something learned before messing with GDI themes... Mainly because it is more efficient (remove an extra control, don't have to mess around with Parent Events, its all just built in to your custom form).

I would rather not debate "how the current method became the standard" or "which is 'right' and which is 'wrong'", I am just stating for the record that OP's method of skinning makes since (The evidence provided is to state that it is logical to teach people to inherit, not that it is the "right" way [or only "right" way, rather]). ?

maruata - December 1, 2014, 6:13 am

Thanks to all comments....

I am in the way of the Hill. :-P.

I don't know why I choose the Form instead of Container Control. But I believe that the Drawing Speed is better for Form. Since Container Control need to be applied on the Form again.

So, here we need to just inherit the Win8Form Class instead of putting it as one control. :-)

Anyway, one way it is the choice of the Coder ! ;-|

maruata - December 11, 2014, 6:57 am

I have another messy version here! Please download if you want more stuff on form!

<http://pastebin.com/rS9Qc2CP>