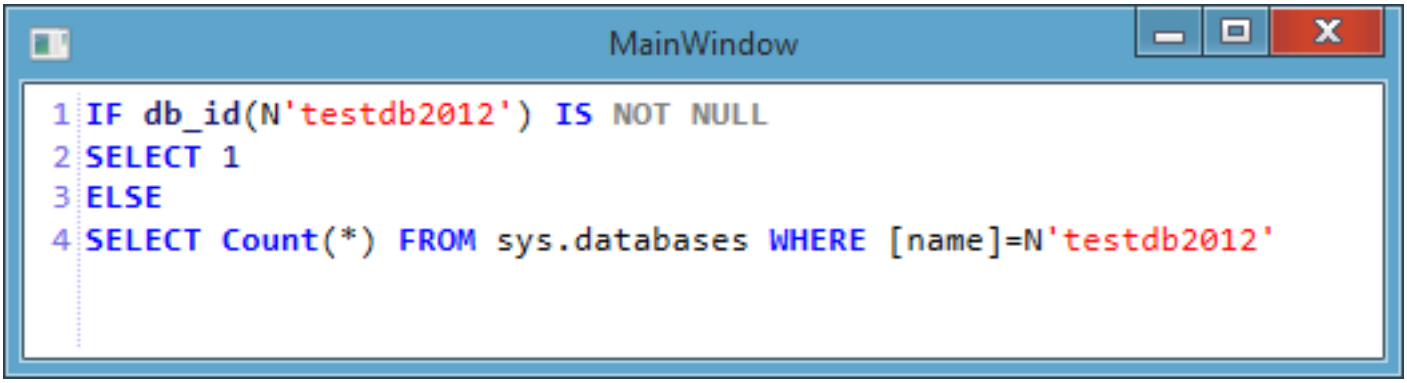
عنوان: **استفاده از AvalonEdit در WPF**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۲۷ ۱۲:۴۰
آدرس: www.dotnettips.info
گروه‌ها: WPF

AvalonEdit یکی از زیرساخت‌های برنامه‌ی SharpDevelop است که ویرایشگر متنی به همراه syntax highlighting زبان‌های مختلف را در آن پشتیبانی می‌کند. کیفیت بالایی داشته و بسیاری از برنامه‌های دیگر نیز از آن جهت ارائه ویرایشگر و یا syntax highlighting متون ارائه شده، استفاده می‌کنند. در ادامه نحوه‌ی استفاده از این ویرایشگر را در برنامه‌های WPF خصوصا با دید MVVM بررسی خواهیم کرد.



دریافت و نصب AvalonEdit

برای نصب AvalonEdit می‌توان دستور ذیل را در کنسول پاورشل نیوگت صادر کرد:

```
PM> install-package AvalonEdit
```

استفاده‌ی مقدماتی از AvalonEdit

برای استفاده از این ویرایشگر ابتدا نیاز است فضای نام xmlns:avalonEdit تعریف شود. سپس کنترل avalonEdit:TextEditor در دسترس خواهد بود:

```
<Window x:Class="SyntaxHighlighter.MainWindow"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:avalonEdit="http://icsharpcode.net/sharpdevelop/avalonedit"
   Title="MainWindow" Height="401" Width="617">
  <Grid>
<avalonEdit:TextEditor
    Name="txtCode"
    SyntaxHighlighting="C#"
  FontFamily="Consolas"
  FontSize="10pt"/>
  </Grid>
</Window>
```

توسط خاصیت SyntaxHighlighting آن می‌توان زبان مشخصی را تعریف کرد. لیست زبان‌های توکار پشتیبانی شده

استفاده از AvalonEdit در برنامه‌های MVVM

خاصیت Text این ویرایشگر به صورت معمولی تعریف شده (DependencyProperty نیست) و امکان binding دو طرفه به آن وجود ندارد. به همین جهت نیاز است یک چنین DependencyProperty را به آن اضافه کرد:

```
using System;
using System.Collections.Concurrent;
using System.Reflection;
using System.Windows;
using System.Xml;
using ICSharpCode.AvalonEdit;
using ICSharpCode.AvalonEdit.Highlighting;
using ICSharpCode.AvalonEdit.Highlighting.Xshd;

namespace AvalonEditWpfTest.Controls
{
    public class BindableAvalonTextEditor : TextEditor
    {
        public static readonly DependencyProperty BoundTextProperty =
            DependencyProperty.Register("BoundText",
                typeof(string),
                typeof(BindableAvalonTextEditor),
                new FrameworkPropertyMetadata(default(string), propertyChangedCallback));

        public static string GetBoundText(DependencyObject obj)
        {
            return (string)obj.GetValue(BoundTextProperty);
        }

        public static void SetBoundText(DependencyObject obj, string value)
        {
            obj.SetValue(BoundTextProperty, value);
        }

        protected override void OnTextChanged(EventArgs e)
        {
            SetCurrentValue(BoundTextProperty, Text);
            base.OnTextChanged(e);
        }

        private static void propertyChangedCallback(DependencyObject obj,
                                    DependencyPropertyChangedEventArgs args)
        {
            var target = (BindableAvalonTextEditor)obj;
            var value = args.NewValue;
            if (value == null)
                return;

            if (string.IsNullOrWhiteSpace(target.Text) ||
                !target.Text.Equals(args.NewValue.ToString()))
            {
                target.Text = args.NewValue.ToString();
            }
        }
    }
}
```

کار با ارث بری از TextEditor (ویرایشگر AvalonEdit) شروع می‌شود. سپس یک DependencyProperty به نام BoundText در اینجا اضافه شده‌است. هر زمان که متن داخل آن تغییر کرد، آن‌را به خاصیت متنی Text این ویرایشگر نسبت می‌دهد. به این ترتیب binding یک طرفه (از کدها به کنترل) کار می‌کند. فعال سازی binding دو طرفه با پشتیبانی از انتقال تغییرات از ویرایشگر به خواص ViewModel در متد بازنویسی شده‌ی OnTextChanged انجام می‌شود.

اکنون برای استفاده از این کنترل جدید که BindableAvalonTextEditor نام دارد، می‌توان به نحو ذیل عمل کرد:

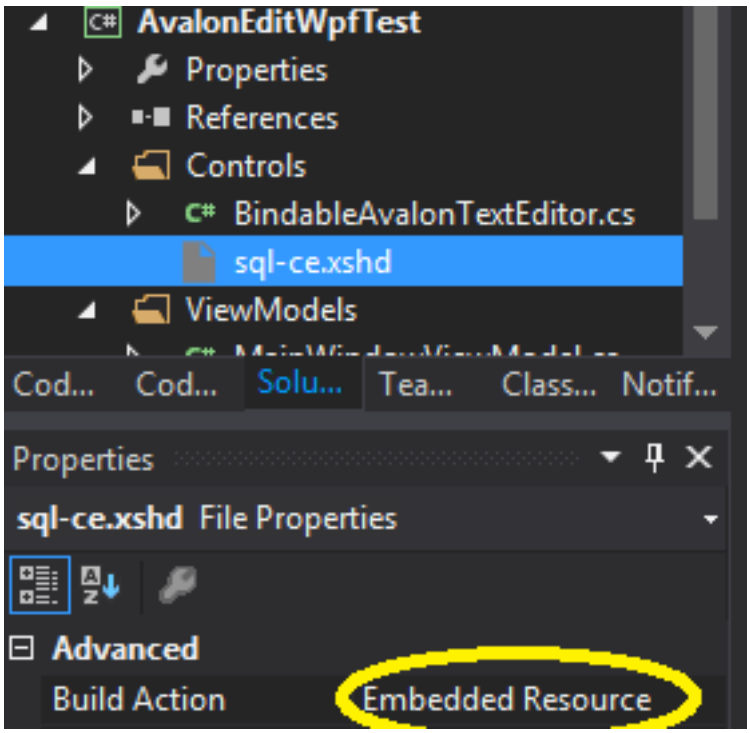
```
<Window x:Class="AvalonEditWpfTest.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:viewModels="clr-namespace:AvalonEditTests.ViewModels"
        xmlns:controls="clr-namespace:AvalonEditWpfTest.Controls"
        Title="MainWindow" Height="350" Width="525">
    <Window.Resources>
        <viewModels:MainWindowViewModel x:Key="MainWindowViewModel"/>
    </Window.Resources>
```

```
    <Grid DataContext="{Binding Source={StaticResource MainWindowViewModel}}">
        <controls:BindableAvalonTextEditor
                BoundText="{Binding SourceCode, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
                        WordWrap="True"
                        ShowLineNumbers="True"
                        LineNumbersForeground="MediumSlateBlue"
                        FontFamily="Consolas"
                        VerticalScrollBarVisibility="Auto"
                        Margin="3"
                        HorizontalScrollBarVisibility="Auto"
                        FontSize="10pt"/>
    </Grid>
</Window>
```

ابتدا فضای نام جدید کنترل BindableAvalonTextEditor مشخص می‌شود و سپس به controls:BindableAvalonTextEditor دسترسی خواهیم داشت. در اینجا نحوه‌ی استفاده از خاصیت جدید BoundText را نیز مشاهده می‌کنید.

افزودن syntax highlighting زبان‌هایی که به صورت رسمی پشتیبانی نمی‌شوند

به خاصیت SyntaxHighlighting این کنترل صرفا مقادیری را می‌توان نسبت داد که به صورت توکار پشتیبانی می‌شوند. برای مثال C#، XML و امثال آن.
فرض کنید نیاز است SyntaxHighlighting زبان SQL را فعال کنیم. برای اینکار نیاز به فایل‌های ویژه‌ای است، با پسوند xshd.
برای نمونه فایل sql-ce.xshd را در اینجا می‌توانید مطالعه کنید. در آن یک سری واژه‌های کلیدی و حروفی که باید با رنگی متفاوت نمایش داده شوند، مشخص می‌گردند.
برای استفاده از فایل sql-ce.xshd باید به نحو ذیل عمل کرد:
الف) فایل sql-ce.xshd را به پروژه اضافه کرده و سپس در برگه‌ی خواص آن در VS.NET، مقدار build action آن‌را به embedded resource تغییر دهید.



ب) با استفاده از متد ذیل، این فایل مدفون شده در اسمبلی را گشوده و به متد HighlightingLoader.Load ارسال می‌کنیم:

```
private static IHighlightingDefinition getHighlightingDefinition(string resourceName)
        {
            if (string.IsNullOrWhiteSpace(resourceName))
                throw new NullReferenceException("Please specify SyntaxHighlightingResourceName.");

            using (var stream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(resourceName))
            {
                if (stream == null)
                    throw new NullReferenceException(string.Format("{0} resource is null.",
resourceName));

                using (var reader = new XmlTextReader(stream))
                {
                    return HighlightingLoader.Load(reader, HighlightingManager.Instance);
                }
            }
        }
```

نحوه استفاده از آن نیز به صورت ذیل است:

```
txtCode.SyntaxHighlighting = getHighlightingDefinition(resourceName);
```

به این ترتیب می‌توان یک فایل xhsd را به صورت پویا بارگذاری و به خاصیت SyntaxHighlighting کنترل انتساب داد.

برای سهولت استفاده از این قابلیت شاید بهتر باشد یک DependencyProperty دیگر به نام SyntaxHighlightingResourceName را به کنترل جدید BindableAvalonTextEditor اضافه کنیم:

```
using System;
using System.Collections.Concurrent;
using System.Reflection;
using System.Windows;
using System.Xml;
using ICSharpCode.AvalonEdit;
using ICSharpCode.AvalonEdit.Highlighting;
using ICSharpCode.AvalonEdit.Highlighting.Xshd;

namespace AvalonEditWpfTest.Controls
{
    public class BindableAvalonTextEditor : TextEditor
    {
         public static readonly DependencyProperty SyntaxHighlightingResourceNameProperty =
           DependencyProperty.Register("SyntaxHighlightingResourceName",
               typeof(string),
               typeof(BindableAvalonTextEditor),
               new FrameworkPropertyMetadata(default(string), resourceNamePropertyChangedCallback));

        public static string GetSyntaxHighlightingResourceName(DependencyObject obj)
        {
            return (string)obj.GetValue(SyntaxHighlightingResourceNameProperty);
        }

        public static void SetSyntaxHighlightingResourceName(DependencyObject obj, string value)
        {
            obj.SetValue(SyntaxHighlightingResourceNameProperty, value);
        }

        private static void loadHighlighter(TextEditor @this, string resourceName)
        {
            if (@this.SyntaxHighlighting != null)
                return;

            @this.SyntaxHighlighting = getHighlightingDefinition(resourceName);
        }

        private static void resourceNamePropertyChangedCallback(DependencyObject obj,
                                          DependencyPropertyChangedEventArgs args)
        {
            var target = (BindableAvalonTextEditor)obj;
            var value = args.NewValue;
            if (value == null)
                return;
```

```
            loadHighlighter(target, value.ToString());
        }
    }
}
```

کاری که در اینجا انجام شده، افزودن یک خاصیت جدید به نام SyntaxHighlightingResourceName به کنترل BindableAvalonTextEditor است. هر زمانیکه مقدار آن تغییر کند، متد getHighlightingDefinition بحث شده، فراخوانی گردیده و به صورت پویا مقدار خاصیت SyntaxHighlighting این کنترل، مقدار دهی می‌شود.
استفاده از آن نیز به شکل زیر است:

```
<controls:BindableAvalonTextEditor
SyntaxHighlightingResourceName = "AvalonEditWpfTest.Controls.sql-ce.xshd"
/>
```

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

AvalonEditWpfTest.zip