Разработка Капчи на WPF

1. Создаем новое приложение WPF (.NET Framework)
2. Открываем наше окно и создаем в <Grid> новый

<Grid Width=”400” Height=”200”></Grid>

1. Внутри нашего нового Grid создаем Пишем следующий код:

<StackPanel Orientation="Horizontal" Name="SPanelSymbols" HorizontalAlignment="Center" VerticalAlignment="Center">

</StackPanel>

<Canvas Name="CanvasNoise" HorizontalAlignment="Center" Width="380" Height="200" VerticalAlignment="Center" ClipToBounds="True">

</Canvas>

<Button x:Name="BtnUpdateCaptcha" Click="BtnUpdateCaptcha\_Click" VerticalAlignment="Bottom">Обновить капчу</Button>

Этот код позволит нам создать оболочку для нашей капчи.

Перейдем к коду.

Для начала в Public partial Class Страницы Добавляем код:

Random \_random = new Random(); - строчка для создания рандома.

После строчки кода InitializeComponent(); Вставляем строчку кода UpdateCaptcha(); для формирования капчи при запуске приложения.

Создаем Функцию – private void UpdateCaptcha() со следующим кодом

{

SPanelSymbols.Children.Clear(); // очистка Стак Панела

CanvasNoise.Children.Clear();//очистка Канвас

GenerateSymbols(4); // наша функция на создание 4 символов (можно сделать больше)

GenerateNoise(30); // Функция на создание «Шума»

}

Переходим к созданию функции GenerateSymbols. Код функции выглядит так:

private void GenerateSymbols(int count) // создаем функцию и получаем число символов

{

string alphabet = "WERPASFHKXVBM234578"; // задаем алфавит

for (int i = 0; i < count; i++) // запускаем цикл

{

string symbol = alphabet.ElementAt(\_random.Next(0, alphabet.Length)).ToString(); // с помощью рандома выбираем случаный символ из алфавита

TextBlock lbl = new TextBlock(); //создаем текст блок в который поместим наш символ

lbl.Text = symbol;

lbl.FontSize = \_random.Next(40, 80); //задаем случайный размер шрифта в диапазоне от 40 до 80

lbl.RenderTransform = new RotateTransform(\_random.Next(-45, 45)); // задаем случайный угол

lbl.Margin = new Thickness(20, 0, 20, 0); // задаем случайные отступы

SPanelSymbols.Children.Add(lbl); // добавляем новый текст блок на наш стак панел

}

}

Создаем функцию Создания «Шума»:

private void GenerateNoise(int volumeNoise) // создаем функцию и получаем число элементов

{

for (int i = 0; i < volumeNoise; i++) // цикл создания элементов

{

Border border = new Border(); //задаем новую рамку

border.Background = new SolidColorBrush(Color.FromArgb((byte)\_random.Next(100, 200),

(byte)\_random.Next(0, 256),

(byte)\_random.Next(0, 256),

(byte)\_random.Next(0, 256))); // создаем случайный цвет для нашего элемента

border.Height = \_random.Next(2, 10); // создаем рандом высоты

border.Width = \_random.Next(10, 400);//создаем рандом ширины

border.RenderTransform = new RotateTransform(\_random.Next(0, 360));

CanvasNoise.Children.Add(border); //добавляем наш элемент

Canvas.SetLeft(border, \_random.Next(0, 300));

Canvas.SetTop(border, \_random.Next(0, 150)); // добавляем отступы от точки создания

Ellipse ellipse = new Ellipse(); // создаем кружок

ellipse.Fill = new SolidColorBrush(Color.FromArgb((byte)\_random.Next(100, 200),

(byte)\_random.Next(0, 256),

(byte)\_random.Next(0, 256),

(byte)\_random.Next(0, 256))); // так же как и с Border создаем случайный цвет для элемента (кружка)

ellipse.Height = ellipse.Width = \_random.Next(20, 40); //задаем радиус

CanvasNoise.Children.Add(ellipse); // добавляем

Canvas.SetLeft(ellipse, \_random.Next(0, 300));

Canvas.SetTop(ellipse, \_random.Next(0, 150));

}

}

Последний код который нужно добавить – обработчик нажатия на кнопку:

private void BtnUpdateCaptcha\_Click(object sender, RoutedEventArgs e)

{

UpdateCaptcha(); //запускаем обновление капчи

}