# How to make and use captcha code in ASP.NET Core

## Introduction

This sample demonstrates how to make and use captcha code in ASP.NET Core.

## Sample prerequisites

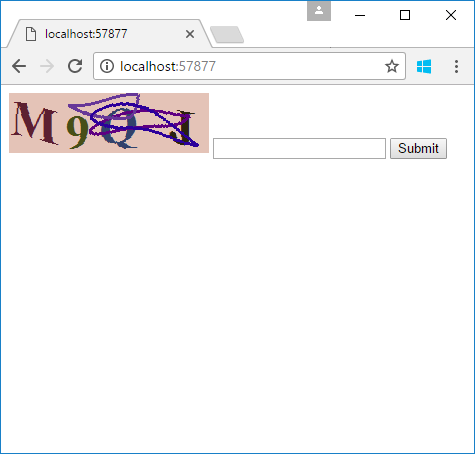
* Visual Studio 2017 or above. [[Visual Studio Home Page](https://www.visualstudio.com/)]
* Visual Studio enabled ASP.NET Core develop component.

## Building the sample

Use Visual Studio to open the sample solution **CSCaptchaCodeASPNETCore**, then press **F6 Key** to build the sample project.

## Running the sample

* Use Visual Studio to open the sample solution **CSCaptchaCodeASPNETCore**, then press F5 Key or select **Debug -> Start Debugging** from the menu.
* When the project is running, you will see below page in browser.



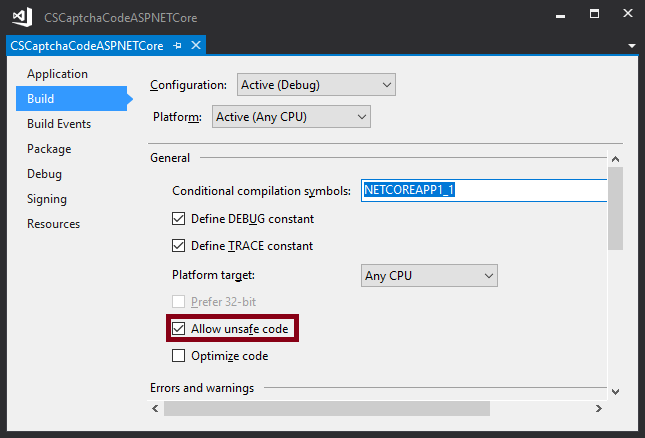
## Add the CoreCompat.System.Drawing assembly

To install CoreCompat.System.Drawing, run the following command in the Package Manager Console.

* **Install-Package CoreCompat.System.Drawing -Version 1.0.0-beta006 -Pre**

## Enable the unsafe code in project setting

1. Right click project and select **properties**.
2. In **Build** tab, enable the **Allow unsafe code** option.



## Using the code

public FileStreamResult GetImage()

{

int width = 200;

int height = 60;

var captchaCode = Captcha.GenerateCaptchaCode();

var result = Captcha.GenerateCaptchaImage(width, height, captchaCode);

HttpContext.Session.SetString("CaptchaCode", result.CaptchaCode);

Stream s = new MemoryStream(result.CaptchaByteData);

return new FileStreamResult(s, "image/png");

}

Captcha.cs

public static class Captcha

{

const string Letters = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";

public static string GenerateCaptchaCode()

{

Random rand = new Random();

int maxRand = Letters.Length - 1;

StringBuilder sb = new StringBuilder();

for (int i = 0; i < 4; i++)

{

int index = rand.Next(maxRand);

sb.Append(Letters[index]);

}

return sb.ToString();

}

public static CaptchaResult GenerateCaptchaImage(int width, int height, string captchaCode)

{

using (Bitmap baseMap = new Bitmap(width, height))

using (Graphics graph = Graphics.FromImage(baseMap))

{

Random rand = new Random();

graph.Clear(GetRandomLightColor());

DrawCaptchaCode();

DrawDisorderLine();

AdjustRippleEffect();

MemoryStream ms = new MemoryStream();

baseMap.Save(ms, ImageFormat.Png);

return new CaptchaResult { CaptchaCode = captchaCode, CaptchaByteData = ms.ToArray(), Timestamp = DateTime.Now };

int GetFontSize(int imageWidth, int captchCodeCount)

{

var averageSize = imageWidth / captchCodeCount;

return Convert.ToInt32(averageSize);

}

Color GetRandomDeepColor()

{

int redlow = 160, greenLow = 100, blueLow = 160;

return Color.FromArgb(rand.Next(redlow), rand.Next(greenLow), rand.Next(blueLow));

}

Color GetRandomLightColor()

{

int low = 180, high = 255;

int nRend = rand.Next(high) % (high - low) + low;

int nGreen = rand.Next(high) % (high - low) + low;

int nBlue = rand.Next(high) % (high - low) + low;

return Color.FromArgb(nRend, nGreen, nBlue);

}

void DrawCaptchaCode()

{

SolidBrush fontBrush = new SolidBrush(Color.Black);

int fontSize = GetFontSize(width, captchaCode.Length);

Font font = new Font(FontFamily.GenericSerif, fontSize, FontStyle.Bold, GraphicsUnit.Pixel);

for (int i = 0; i < captchaCode.Length; i++)

{

fontBrush.Color = GetRandomDeepColor();

int shiftPx = fontSize / 6;

float x = i \* fontSize + rand.Next(-shiftPx, shiftPx) + rand.Next(-shiftPx, shiftPx);

int maxY = height - fontSize;

if (maxY < 0) maxY = 0;

float y = rand.Next(0, maxY);

graph.DrawString(captchaCode[i].ToString(), font, fontBrush, x, y);

}

}

void DrawDisorderLine()

{

Pen linePen = new Pen(new SolidBrush(Color.Black), 3);

for (int i = 0; i < rand.Next(3, 5); i++)

{

linePen.Color = GetRandomDeepColor();

Point startPoint = new Point(rand.Next(0, width), rand.Next(0, height));

Point endPoint = new Point(rand.Next(0, width), rand.Next(0, height));

graph.DrawLine(linePen, startPoint, endPoint);

Point bezierPoint1 = new Point(rand.Next(0, width), rand.Next(0, height));

Point bezierPoint2 = new Point(rand.Next(0, width), rand.Next(0, height));

graph.DrawBezier(linePen, startPoint, bezierPoint1, bezierPoint2, endPoint);

}

}

void AdjustRippleEffect()

{

short nWave = 6;

int nWidth = baseMap.Width;

int nHeight = baseMap.Height;

Point[,] pt = new Point[nWidth, nHeight];

double newX, newY;

double xo, yo;

for (int x = 0; x < nWidth; ++x)

{

for (int y = 0; y < nHeight; ++y)

{

xo = ((double)nWave \* Math.Sin(2.0 \* 3.1415 \* (float)y / 128.0));

yo = ((double)nWave \* Math.Cos(2.0 \* 3.1415 \* (float)x / 128.0));

newX = (x + xo);

newY = (y + yo);

if (newX > 0 && newX < nWidth)

{

pt[x, y].X = (int)newX;

}

else

{

pt[x, y].X = 0;

}

if (newY > 0 && newY < nHeight)

{

pt[x, y].Y = (int)newY;

}

else

{

pt[x, y].Y = 0;

}

}

}

Bitmap bSrc = (Bitmap)baseMap.Clone();

BitmapData bitmapData = baseMap.LockBits(new Rectangle(0, 0, baseMap.Width, baseMap.Height), ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);

BitmapData bmSrc = bSrc.LockBits(new Rectangle(0, 0, bSrc.Width, bSrc.Height), ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);

int scanline = bitmapData.Stride;

IntPtr Scan0 = bitmapData.Scan0;

IntPtr SrcScan0 = bmSrc.Scan0;

unsafe

{

byte\* p = (byte\*)(void\*)Scan0;

byte\* pSrc = (byte\*)(void\*)SrcScan0;

int nOffset = bitmapData.Stride - baseMap.Width \* 3;

int xOffset, yOffset;

for (int y = 0; y < nHeight; ++y)

{

for (int x = 0; x < nWidth; ++x)

{

xOffset = pt[x, y].X;

yOffset = pt[x, y].Y;

if (yOffset >= 0 && yOffset < nHeight && xOffset >= 0 && xOffset < nWidth)

{

p[0] = pSrc[(yOffset \* scanline) + (xOffset \* 3)];

p[1] = pSrc[(yOffset \* scanline) + (xOffset \* 3) + 1];

p[2] = pSrc[(yOffset \* scanline) + (xOffset \* 3) + 2];

}

p += 3;

}

p += nOffset;

}

}

baseMap.UnlockBits(bitmapData);

bSrc.UnlockBits(bmSrc);

bSrc.Dispose();

}

}

}

}

public class CaptchaResult

{

public string CaptchaCode { get; set; }

public byte[] CaptchaByteData { get; set; }

public string CaptchBase64Data

{

get

{

return Convert.ToBase64String(CaptchaByteData);

}

}

public DateTime Timestamp { get; set; }

}