```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;
using System.Drawing.Text;

namespace Server.Web
{
    public class CaptchaCode
    {
        private const int _ndigits = 4;

        public static string GenerateRandomCode()
        {
            return GenerateRandomCode(_ndigits);
        }

        public static string GenerateRandomCode(int ndigits)
        {
            Random random = new Random();
            string s = "";
            for (int i = 0; i < ndigits; i++)
            {
                s = String.Concat(s, random.Next(10).ToString());
            }
            return s;
        }
    }

    public class CaptchaImage
    {
        private string _text;
        private int _width;
        private int _height;
        private string _familyName;
        private Bitmap _image;
        private Random _random = new Random();

        public string Text
        {
            get
            {
                return _text;
            }
        }

        public Bitmap Image
        {
            get
            {
                return _image;
    }
}

public int Width
{
    get
    {
        return _width;
    }
}

public int Height
{
    get
    {
        return _height;
    }
}

public CaptchaImage(string s, int width, int height)
{
    _text = s;
    SetDimensions(width, height);
    GenerateImage();
}

public CaptchaImage(string s, int width, int height, string familyName)
{
    _text = s;
    SetDimensions(width, height);
    SetFamilyName(familyName);
    GenerateImage();
}

~CaptchaImage()
{
    Dispose(false);
}

public void Dispose()
{
    GC.SuppressFinalize(this);
    Dispose(true);
}

protected virtual void Dispose(bool disposing)
{
    if (disposing)
    {
        _image.Dispose();
    }
```

```
        }

        private void SetDimensions(int width, int height)
        {
            if (width <= 0)
            {
                throw new ArgumentOutOfRangeException("width", width,
                   "Argument out of range, must be greater than zero.");
            }
            if (height <= 0)
            {
                throw new ArgumentOutOfRangeException("height", height,
                   "Argument out of range, must be greater than zero.");
            }
            _width = width;
            _height = height;
        }

        private void SetFamilyName(string familyName)
        {
            try
            {
                Font font = new Font(_familyName, 12F);
                _familyName = familyName;
                font.Dispose();
            }
            catch
            {
                _familyName = System.Drawing.FontFamily.GenericSerif.Name;
            }
        }

        private void GenerateImage()
        {
            // Create a new 32-bit bitmap image
            Bitmap bitmap = new Bitmap(_width, _height, PixelFormat.Format32bppArgb);

            // Create a graphics object for drawing
            Graphics g = Graphics.FromImage(bitmap);
            g.SmoothingMode = SmoothingMode.AntiAlias;
            Rectangle rect = new Rectangle(0, 0, _width, _height);

            // Fill in the background
            HatchBrush hatchBrush = new HatchBrush(
                   HatchStyle.SmallConfetti,
                   Color.LightGray,
                   Color.White);
            g.FillRectangle(hatchBrush, rect);
```

```
// Set up the text font
SizeF size;
float fontSize = rect.Height + 1;
Font font;
// Adjust the font size until the text fits within the image.
do
{
    fontSize--;
    font = new Font(_familyName, fontSize, FontStyle.Bold);
    size = g.MeasureString(_text, font);
}
while (size.Width > rect.Width);

// Set up the text format
StringFormat format = new StringFormat();
format.Alignment = StringAlignment.Center;
format.LineAlignment = StringAlignment.Center;

// Create a path using the text and warp it randomly
GraphicsPath path = new GraphicsPath();
path.AddString(_text, font.FontFamily, (int)font.Style, font.Size, rect, format);
float v = 4F;
PointF[] points =
{
    new PointF(_random.Next(rect.Width) / v, _random.Next(rect.Height) / v),
    new PointF(rect.Width - _random.Next(rect.Width) / v,
       _random.Next(rect.Height) / v),
    new PointF(_random.Next(rect.Width) / v,
       rect.Height - _random.Next(rect.Height) / v),
    new PointF(rect.Width - _random.Next(rect.Width) / v,
       rect.Height - _random.Next(rect.Height) / v)
};
Matrix matrix = new Matrix();
matrix.Translate(0F, 0F);
path.Warp(points, rect, matrix, WarpMode.Perspective, 0F);

// Draw the text
hatchBrush = new HatchBrush(HatchStyle.LargeConfetti, Color.LightGray, Color.DarkGray);
g.FillPath(hatchBrush, path);

// Add some random noise
int m = Math.Max(rect.Width, rect.Height);
for (int i = 0; i < (int) (rect.Width * rect.Height / 30F); i++)
{
    int x = _random.Next(rect.Width);
    int y = _random.Next(rect.Height);
    int w = _random.Next(m / 50);
    int h = _random.Next(m / 50);
    g.FillEllipse(hatchBrush, x, y, w, h);
}
```

```
            // Clean up
            font.Dispose();
            hatchBrush.Dispose();
            g.Dispose();

            // Set the image
            _image = bitmap;
        }
    }
}
```