C#

Convertire un file .doc in PDF

C# Code Snippet

```
string dataDir = RunExamples.GetDataDir_QuickStart();
 Document doc = new Document(dataDir + "Template.doc");
 dataDir = dataDir + "Template_out_.pdf";
doc.Save(dataDir);
```

Convertire un PDF in .doc

```
Document pdfDocument = new Document("input.pdf");
pdfDocument.Save("D://pdftest//TableHeightIssue.doc", SaveFormat.Doc);
```

Come impostare la spaziatura di un paragrafo all'interno di un Textbox

```
string dataDir =
RunExamples.GetDataDir(System.Reflection.MethodBase.GetCurrentMethod().DeclaringType);
Workbook wb = new Workbook();
Worksheet ws = wb.Worksheets[0];
ws.Shapes.AddTextBox(2, 0, 2, 0, 100, 200);
Shape shape = ws.Shapes[0];
shape.Text = "Sign up for your free phone number.\nCall and text online for free.";
TextParagraph p = shape.TextBody.TextParagraphs[1];
p.LineSpaceSizeType = LineSpaceSizeType.Points;
p.LineSpace = 20;
p.SpaceAfterSizeType = LineSpaceSizeType.Points;
p.SpaceAfter = 10;
p.SpaceBeforeSizeType = LineSpaceSizeType.Points;
p.SpaceBefore = 10;
wb.Save(dataDir + "output_out_.xlsx", SaveFormat.Xlsx);
```

Rimuovere le informazioni GPS dai metadati di una immagine

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using GroupDocs.Metadata.Tools;
```

```
using GroupDocs.Metadata.Standards.Exif;
using System.IO;

namespace GroupDocs.Metadata.Examples.Utilities.CSharp
{
    //ExStart:PhotoCleaner
    public class PhotoCleaner
    {
        // absolute path to the GroupDocs.Metadata license file.
        private const string LicensePath = @"GroupDocs.Metadata.lic";

        // absolute path to the photos directory.
        public string CleanerPath { get; set; }

        static PhotoCleaner()
        {
            /* set product license
             * uncomment following function if you have product license
             * */
            //SetInternalLicense();
        }

        public PhotoCleaner(string cleanerPath)
        {
            // check if directory exists
            if (!Directory.Exists(Common.MapSourceFilePath( cleanerPath)))
            {
                throw new DirectoryNotFoundException("Directory not found: " + cleanerPath);
            }
            // set property
            this.CleanerPath = cleanerPath;
        }
        /// <summary>
        /// Applies the product license
        /// </summary>
        private static void SetInternalLicense()
        {
            License license = new License();
            license.SetLicense(LicensePath);
        }
        public void RemoveExifLocation()
        {
            string sourceDirectoryPath = Common.MapSourceFilePath(this.CleanerPath);

            string[] files = Directory.GetFiles(sourceDirectoryPath);
```

```
        foreach (string path in files)
        {
          // get EXIF data if exists
          ExifMetadata exifMetadata = (ExifMetadata)MetadataUtility.ExtractSpecificMetadata(path,
MetadataType.EXIF);

          if (exifMetadata != null)
          {
            ExifInfo exifInfo = exifMetadata.Data;

            if (exifInfo.GPSData != null)
            {
              // set altitude, latitude and longitude to null values
              exifInfo.GPSData.Altitude = null;
              exifInfo.GPSData.Latitude = null;
              exifInfo.GPSData.LatitudeRef = null;
              exifInfo.GPSData.Longitude = null;
              exifInfo.GPSData.LongitudeRef = null;
            }

            // and update file
            MetadataUtility.UpdateMetadata(path, exifMetadata);
          }
        }
        Console.WriteLine("Press any key to exit.");
      }
    }
    //ExEnd:PhotoCleaner
}
```

Rimuovere i commenti da un documento Word

```
public static void RemoveComments()
        {
          try
          {

            // initialize DocFormat
            DocFormat docFormat = new DocFormat(Common.MapSourceFilePath(filePath));

            // remove comments
            docFormat.ClearComments();

            // save file in destination folder
            docFormat.Save(Common.MapDestinationFilePath(filePath));

            Console.WriteLine("File saved in destination folder.");
```

```
            }
            catch (Exception exp)
            {
                Console.WriteLine(exp.Message);
            }
        }
```

Effettuare il motion detection delle immagini riprese da una webcam

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Ozeki.Media.MediaHandlers;
using Ozeki.Media.MediaHandlers.Video;

namespace Tripwire_WF
{
    public partial class MainForm : Form
    {
        private WebCamera _camera;
        private DrawingImageProvider _provider;
        private MediaConnector _connector;

        private Tripwire tripwire;

        private Point _p1, _p2;

        public MainForm()
        {
            InitializeComponent();

            tripwire = new Tripwire();

            _provider = new DrawingImageProvider();
            _connector = new MediaConnector();
        }

        private void connectBt_Click(object sender, EventArgs e)
        {
            _camera = WebCamera.GetDefaultDevice();
            if (_camera == null) return;

            videoViewerWF1.SetImageProvider(_provider);

            _connector.Connect(_camera, tripwire);
```

```
    _connector.Connect(tripwire, _provider);

    _camera.Start();

    videoViewerWF1.Start();
}

private void startBt_Click(object sender, EventArgs e)
{
    tripwire.Line.LineWidth = 3;
    tripwire.LineColor = Color.Red;

    tripwire.SetPoints(new Point(300, 100), new Point(150, 300));
    tripwire.HighlightMotion = HighlightMotion.Highlight;

    tripwire.MotionColor = Color.Blue;
    tripwire.TripwireMotionEnteredToLine += TripwireTripwireMotionEnteredToLine;
    tripwire.TripwireMotionLeaveFromLine += TripwireTripwireMotionLeaveFromLine;

    tripwire.Start();
}

private void stopBt_Click(object sender, EventArgs e)
{
    tripwire.Stop();
}

void InvokeThread(Action action)
{
    Invoke(action);
}

void TripwireTripwireMotionLeaveFromLine(object sender, TripwireMotionCrossedArgs e)
{
    InvokeThread(() => { crossedText.Text = @"EXIT!!!"; });
}

void TripwireTripwireMotionEnteredToLine(object sender, TripwireMotionCrossedArgs e)
{
    InvokeThread(() => { crossedText.Text = @"ENTER!!!"; });
}

private void videoViewerWF1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button != MouseButtons.Left) return;
    _p1 = e.Location;
```

```
        videoViewerWF1.MouseMove += videoViewerWF1_MouseMove;
      }

      private void videoViewerWF1_MouseUp(object sender, MouseEventArgs e)
      {
        if (e.Button != MouseButtons.Left) return;
        _p2 = e.Location;
        tripwire.SetPoints(_p1, _p2);
        videoViewerWF1.MouseMove -= videoViewerWF1_MouseMove;
      }

      private void videoViewerWF1_MouseMove(object sender, MouseEventArgs e)
      {
        _p2 = e.Location;
        tripwire.SetPoints(_p1, _p2);
      }
    }
}
```