PDF for .NET is a library that allows you to create PDF documents from your applications.

The PDF (Portable Document Format) format was created by Adobe in 1993 and has since become the industry standard for the representation of printed material in electronic systems. PDF is popular because it is high-quality, portable, device-independent, and because there are many tools available for creating, manipulating, and viewing PDF documents. These tools include the free multi-platform document viewer Adobe Acrobat. The portability and high quality of the PDF documents make PDF an excellent choice for Web publishing. For more information on the PDF format and PDF utilities from Adobe and other sources, visit the Adobe Web site.

Key Features 

PDF for .NET supports most of the advanced features included in the PDF specification, including security, compression, outlining, hyperlinking, and attachments.

But the main feature in PDF for .NET is ease of use. The commands provided for adding content to documents are similar to the ones available in the .NET Graphics class. If you know how to display text and graphics in .NET, you already know how to use PDF for .NET.

The following are some of the features of PDF for .NET that you may find useful:

Fast rendering and compression of images in Metafiles

Metafiles are not converted into bitmaps; they are parsed and converted into vector graphics commands and thus retain the best possible resolution. If you want to add charts or technical drawings to your PDF document, metafiles are better than bitmap images.

AcroForms support

Use the new AddField method to add Acrobat form fields to your PDF documents. PDF for .NET supports the following field types: textbox, checkbox, radio button, push button, combo box, and list box.

HTML Support

Use the new DrawStringHtml method to render HTML into your PDF documents. You can flow HTML content into multiple pages or columns, use existing style sheets, and mix HTML with other types of content (images, RTF, plain text, form fields, and so on).

Familiar syntax using the DrawImage method

Adding images to PDF documents is easy; all the work is done by the DrawImage method. DrawImage draws a given image at a specified location and has parameters that provide control over the image alignment and scaling. You can render any regular .NET Image object, including metafiles.

Manage document restrictions

Allow users to copy and edit content, restrict users from printing the document, set annotation edit permission to the user, and more.

Add attachments to PDF files

Attachments can contain any kind of file, including spreadsheets with detailed information that would clutter the main document, multimedia files with movies and sound, sample code, and more. Adding an attachment to your PDF file is easy. Simply specify which file you want to attach, what area of the page should contain the attachment, and optionally, the appearance of the attachment.

Owner and user password protection

If your PDF documents contain sensitive information, you can encrypt them so that only authorized users can access it. There is a separate password for the owner of the document and for all other users. The user's access can be selectively restricted to allow only certain operations, such as viewing, printing, or editing the document.

Add graphical elements

Add lines, rectangles, ellipses, pies, arcs, rounded rectangles, polygons, Bezier curves, and more.

Create an Outline structure

Most long PDF documents contain an outline structure that is displayed on a pane on the left of the reader. The outline makes it easy to browse through a document's structure and find specific topics. With PDF for .NET, you can build this outline structure by adding outline entries (bookmarks).

Add hyperlinks and local links

PDF provides methods for adding hyperlinks and hyperlink targets to your PDF documents. You can also add local links, that when clicked take the user to another location within the same PDF document. This type of link is useful when you want to implement some type of cross-referencing within the document, such as a table of contents or an index.

Control document information and viewer preferences

PDF allows you to add meta data to the PDF documents you create. Specify author, creation date, keywords, and so on. You can also provide default viewer preferences to be applied when the document is opened in the Adobe Reader. Specify the initial page layout, window position, as well as reader toolbar and menu visibility.

Support for PDF/A

PDF/A is commonly used by users creating invoices, brochures, manuals or research reports to store their reports to PDF/A formats. It enables export of JPEG2000 Images, provisions for digital signatures, and support for embedded fonts.

PDF for WinForms

Creating Documents

Using ComponentOne PDF for .NET > Creating Documents

Show AllShow All

To create PDF documents using PDF for .NET, you simply create a C1PdfDocument object, add content to the document, and save the document. To follow tradition, here's how to create a "hello world" document using PDF for .NET.

Add the following Imports or using directive to your code.

To write code in Visual Basic

Visual Basic

Copy Code

Imports System.Drawing

To write code in C#

C#

Copy Code

using System.drawing;

Create a C1PdfDocument object.

To write code in Visual Basic

Visual Basic

Copy Code

' Create the C1PdfDocument object.

Dim pdf As New C1.C1Pdf.C1PdfDocument()

To write code in C#

C#

Copy Code

// Create the C1PdfDocument object.

C1.C1Pdf.C1PdfDocument pdf = new C1.C1Pdf.C1PdfDocument();

Add your content to the Form_Load event. This usually involves calling the DrawString method.

To write code in Visual Basic

To write code in C#

Save the document to a file or to a stream using the Save method.

To write code in Visual Basic

To write code in C#

After the application is run, the hello world.pdf document will look like this:

Step 3 is the most interesting one. The code starts by retrieving the page rectangle, expressed in points. It then adds a one-inch margin around the page (72 points). Finally, the code creates a Font object and calls the DrawString method to write "Hello World!" on the page. This is exactly what you would do if you were writing to a Graphics object in .NET and is what makes PDF for .NET so easy to use.

One important thing to remember is that PDF for .NET uses a point-based coordinate system with the origin at the top-left corner of the page. This is similar to the default coordinate system used by

.NET, but is different from the default PDF coordinate system (where the origin is on the bottom-left corner of the page). In this example, the top left point of the "H" in "Hello World" is located at [1,1].

Because the coordinate system is based on points, rather than pixels, PDF for .NET uses RectangleF, SizeF, and PointF structures, which have members of type float, rather than Rectangle, Size, and Point, which have members of type int.