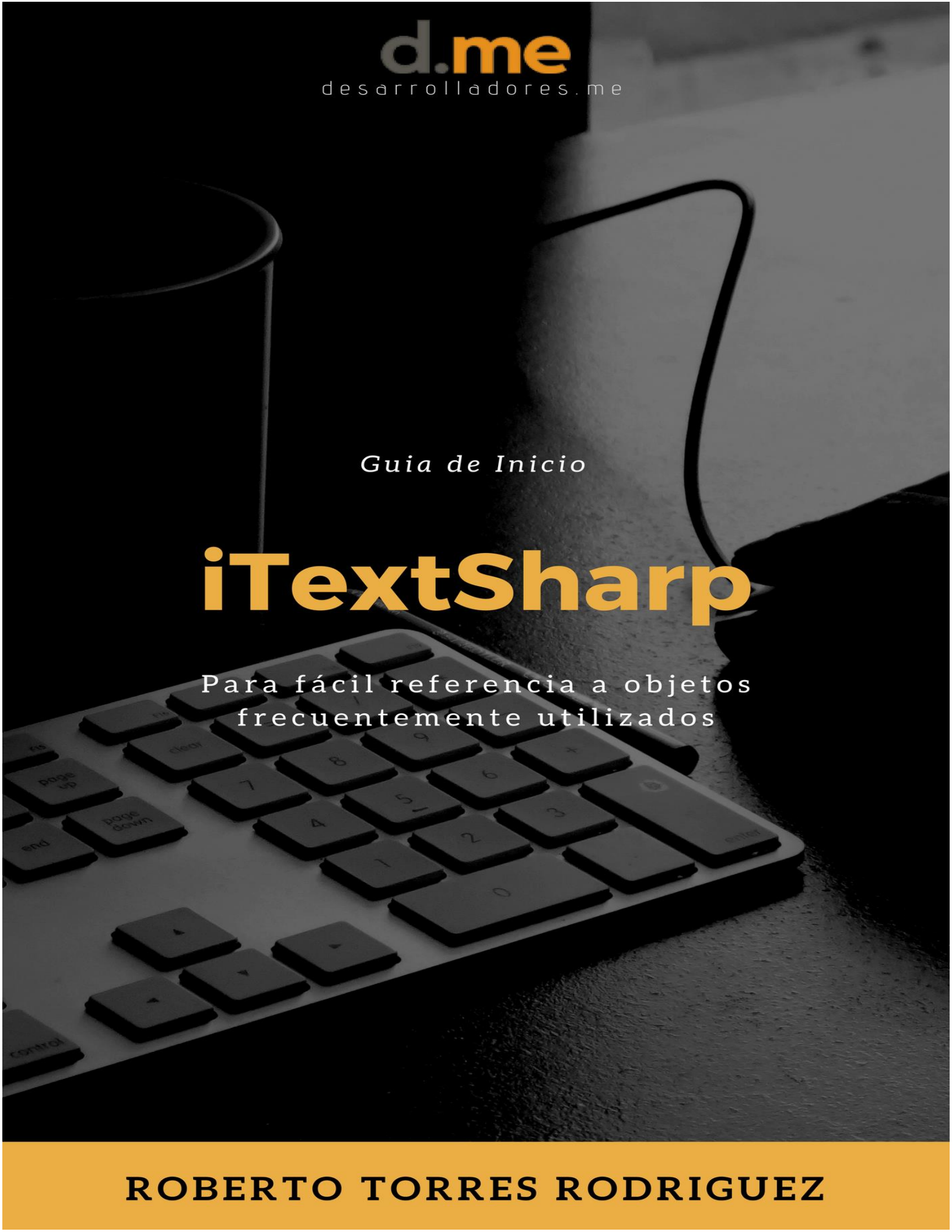

d.me

desarrolladores.me

Guia de Inicio

iTextSharp

Para fácil referencia a objetos frecuentemente utilizados

ROBERTO TORRES RODRIGUEZ

Para Comenzar

Antes de empezar a manipular elementos dentro de un PDF, es necesario entender como funciona la estructura de iTextSharp. El objeto principal que utiliza esta librería se llama *Document*. Este objeto representa el PDF cuando está en memoria, antes de ser guardado en el disco.

Objeto	Descripción	Código
	Luego de añadir la referencia a la solución de Visual Studio, necesitamos utilizar la librería de iTextSharp.	`using iTextSharp.text;` `using iTextSharp.text.pdf;`
Document	Es el objeto principal de iTextSharp. Es necesario crear una instancia de este objeto para poder trabajar con el PDF en memoria. Esto crea una nueva instancia con la configuración predefinida. El tamaño es A4 (210mm X 297mm, o 8.26'' X 11.69'') y los márgenes están configurados a media pulgada.	`var doc = new Document();`
PdfWriter	Para poder grabar el documento al disco, necesitamos la clase PdfWriter.	`string path = Server.MapPath("PDFs");` `PdfWriter.GetInstance(doc, new FileStream(path + "/Doc.pdf", FileMode.Create));`
	Abrir documento PDF	`doc.Open();`
	Cerrar documento PDF	`doc.Close();`
	Para cambiar el tamaño de la página, necesitamos establecerlo al momento de inicializar el objeto *document*.	`var doc = new Document(PageSize.A5);`

PageSize	Este objeto contiene el tamaño de las páginas más utilizadas.	`PageSize.A0; // hasta A10`
		`PageSize.B0; // hasta B10`
		`PageSize.LEGAL`
		`PageSize.LEDGER`
		`PageSize.LETTER`
		`PageSize.POSTCARD`
		`PageSize.TABLOID`

Rectangle	Esta clase se usa para aplicar un tamaño de página personalizado.
	
	Recuerda que cada 72 puntos equivalen a una pulgada.
		`new Rectangle(100f, 300f);`
		
		`var doc = new Document(new Rectangle(100f, 300f));`

	Para cambiar el color de la página del PDF.	`Rectangle r = new Rectangle(100f, 300f);`
		`r.BackgroundColor = new CMYKColor(25, 90, 25, 0); // código CMYK`
		`r.BackgroundColor = new Color(191, 64, 124); // código RGB`

	Añadir objetos al documento PDF	`doc.Add(objeto);`

Añadiendo Texto

Insertar texto en los archivos PDF que creamos es una de las cosas más básica que podemos hacer. iTextSharp provee varios objetos para este propósito, cada uno con su peculiaridad. Debemos utilizar el que más nos convenga para el propósito que queramos alcanzar.

Objeto	Descripción	Código
Chunck	Es la pieza más pequeña de texto dentro de un documento PDF. **No** crean una nueva línea por cada *chunck* creado. Es comparable con un *label* en ASP.Net	`Chunk c1 = new Chunk("Esto es un pedazo de texto aislado.");`

Phrase	Es un arreglo de objetos *Chunck*. Este objeto sí toma en consideración el espacio entre ellos y crea nuevas líneas.	`Phrase p = new Phrase();`
		`p.Add(new Chunk("Esto es un pedazo de texto aislado."));`
		`p.Add(new Chunk("Esto es otro pedazo de texto aislado."));`

Paragraph	Es un conjunto de objetos *Chunk* y *Phrase*. Se rige por los márgenes de la página y siempre hay una nueva línea entre objetos Paragraph	`Chunk c = new Chunk("Texto");`
		`Phrase p1 = new Phrase(c);`
		
		`Paragraph p = new Paragraph();`
		`p.Add(p1);`
		`Paragraph p2 = new Paragraph(new Phrase("Texto"));`
		`p.Add(p2);`

	Los objetos tipo *Paragraph* tienen propiedades sumamente útiles.	`Paragraph p = new Paragraph();`
		
		`p.FirstLineIndent // permite aplicar un valor numérico para añadir sangrado a la primera línea`
		`p.IndentationLeft // permite añadir sangrado al lado izquierdo`
		`p.IndentationRight // permite añadir sangrado al lado derecho`
		`p.SpacingBefore // define la cantidad de espacio antes del párrafo`
		`p.SpacingAfter // define la cantidad de espacio después del párrafo`

Trabajando Con Tipos de Letra

Crear un documento con varios tipos de letra es posible en iTextSharp. No solamente es posible, sino que es sencillo de hacer y hay varias formas de hacerlo.

Objeto	Descripción	Código

BaseFont	Permite definir un tipo de letra. Con este objeto definir letras se hace un poco limitado.	`BaseFont bf`
		`= BaseFont.CreateFont(BaseFont.TIMES_ROMAN, BaseFont.CP1252, false);`
		`Font times = new Font(bfTimes, 12, Font.ITALIC, Color.RED);`
		`Paragraph p = new Paragraph("Esto es una prueba", times);`

FontFactory	Con este objeto puedes configurar muchos otros aspectos de la letra que quieres definir. Aquí unos ejemplos de cómo lo puedes usar.	`Font arial = FontFactory.GetFont("Arial", 28, Color.GRAY);`
		`Font verdana = FontFactory.GetFont("Verdana",`
		`16, Font.BOLDITALIC, new Color(125, 88, 15));`
		`Font palatino = FontFactory.GetFont(`
		`"palatino linotype italique",`
		`BaseFont.CP1252,`
		`BaseFont.EMBEDDED,`
		`10,`
		`Font.ITALIC,`

		Color.GREEN
		);
		Font smallfont = FontFactory.GetFont("Arial", 7);
		Font x = FontFactory.GetFont("nina fett");
		x.Size = 10;
		x.SetStyle("Italic");
		x.SetColor(100, 50, 200);

FontFactory	Para identificar todos los tipos de letras que puedes utilizar, puedes usar la propiedad ***RegisteredFonts*** dentro del objeto FontFactory.	int totalfonts
		= FontFactory.RegisterDirectory("C:\\WINDOWS\\Fonts");
		foreach (string fontname in FontFactory.RegisteredFonts)
		{
		// tu código aquí…
		}

	Registrando un tipo de letra directamente dentro del PDF.	string fontpath = Server.MapPath(".");
		BaseFont letra = BaseFont.CreateFont(fontpath
		+ "myspecial.ttf", BaseFont.CP1252, BaseFont.EMBEDDED);
		Font font = new Font(letra, 12);
		string s = "Esto es una prueba.";
		doc.Add(new Paragraph(s, font));

Tablas

Las tablas son unos de los elementos más utilizado cuando vamos a crear un PDF con iTextSharp. Especialmente si lo estamos utilizando en nuestras aplicaciones web con ASP.Net. Podemos exportar facturas, ordenes, contenido estructurado y reportes financieros, entre muchas otras cosas.

Objeto	Descripción	Código

PdfPTable	Es el objeto utilizado por iTextSharp para crear tablas en documentos PDF.
	
	Este objeto es fácil de utilizar ya que fue diseádo para que las propiedades sean lo más parecido posible a las de las tablas en HTML y CSS.
		`PdfPTable tabla = new PdfPTable(3);`
		`PdfPCell cell = new PdfPCell(new Phrase("Encabezado que abarca 3 columnas"));`
		`cell.Colspan = 3;`
		`cell.HorizontalAlignment = 1; //0=Izquierda, 1=Centro, 2=Derecha`
		`tabla.AddCell(cell);`
		`tabla.AddCell("Row 1 Col 1");`
		`tabla.AddCell("Row 1 Col 2");`
		`tabla.AddCell("Row 1 Col 3");`
		`tabla.AddCell("Row 2 Col 1");`
		`tabla.AddCell("Row 2 Col 2");`
		`tabla.AddCell("Row 2 Col 3");`
		`doc.Add(tabla);`

	Para asignar el ancho de la tabla.	`// ancho de la tabla en puntos` `tabla.TotalWidth = 216f;`

	Para asignar el ancho de cada columna dentro de la tabla.	`// ancho de las columnas en proporciones - 1/3 and 2/3`
		`float[] tamanos = new float[] { 1f, 2f };`
		`tabla.SetWidths(tamanos);`

	Configurar espacio antes y después de la tabla	`table.SpacingBefore = 20f; // dejar espacio antes de la tabla`
		`table.SpacingAfter = 30f; // dejar espacio después de la tabla`

PdfPCell	Objeto del cual está compuesto la tabla ***PdfPTable***. Hace referencia a una celda en una tabla.
	
	Este objeto es altamente configurable siguiendo los nombres utilizados en HTML y CSS.
		`PdfPCell cell`
		`= new PdfPCell(new Phrase("Celda", new Font(Font.HELVETICA, 8f, Font.NORMAL, Color.YELLOW)));`
		`cell.BackgroundColor = new Color(0, 150, 0);`
		`cell.BorderColor = new Color(255,242,0);`
		`cell.Border = Rectangle.BOTTOM_BORDER | Rectangle.TOP_BORDER;`
		`cell.BorderWidthBottom = 3f;`
		`cell.BorderWidthTop = 3f;`
		`cell.PaddingBottom = 10f;`
		`cell.PaddingLeft = 20f;`
		`cell.PaddingTop = 4f;`
		`cell.Rotation = 90; // rotar el text 90 grados. El valor de esta propiedad tiene que ser multiplos de 90.`
		`tabla.AddCell(cell);`

Imágenes

iTextSharp permite utilizar la gran mayoría de los formatos de imágenes populares (jpg, tif, gif, bmp, png y wmf). Hay varias maneras de añadir imágenes a un documento PDF con iTextSharp.

Objeto	Descripción	Código

Image	La manera más básica de insertar una imagen es haciendo referencia a la localización de la foto en el ordenador y utilizándola con la función *GetInstance*.	`string imagepath = Server.MapPath("Images");`
		`Image gif = Image.GetInstance(imagepath + "/desarrolladores.gif");`
		`doc.Add(gif);`

Image	En este caso se está utilizando una imagen a través de un URL o a través de un objeto *System.Drawing.Image*.	`string url = "http://localhost:1805/PDF/Images/desarrolladores.jpg";`
		`Image jpg = Image.GetInstance(new Uri(url));`
		`doc.Add(jpg);`
		`doc.Add(new Paragraph("PNG"));`
		`using (FileStream fs = new FileStream(imagepath`
		`+ "/desarrolladores.png", FileMode.Open))`
		`{`
		`  Image png`
		`= Image.GetInstance(System.Drawing.Image.FromStream(fs),ImageFormat.Png);`
		`  doc.Add(png);`
		`}`

Image	Cambiar el espacio que ocupa una imagen es una tarea sencilla. En este caso, se está reduciendo el espacio que ocupa la imagen en un 24%.
	Nota: El tamaño (bytes) de la foto permanece igual, pero ocupa menos espacio en el PDF.
		`Image tif = Image.GetInstance(imagepath + "/desarrolladores.tif");`
		`tif.ScalePercent(24f);`
		`doc.Add(tif);`

Image	Podemos utilizar la función *SetAbsolutePosition* para determinar donde exactamente queremos posicionar la imagen.	`Image tif = Image.GetInstance(imagepath + "/logo.tif");`
		`tif.ScalePercent(24f);`
		`tif.SetAbsolutePosition(doc.PageSize.Width - 36f - 72f,`
		`  doc.PageSize.Height - 36f - 216.6f);`
		`doc.Add(tif);`

Image	Si lo que necesitas en insertar una imagen dentro de un párrafo, iTextSharp también provee para hacerlo.	`Image jpg = Image.GetInstance(imagepath + "/imagen.jpg");` `Paragraph paragraph = new Paragraph(@"Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Suspendisse blandit blandit turpis. Nam in lectus ut dolor consectetuer bibendum. Morbi neque ipsum, laoreet id; dignissim et, viverra id, mauris. Nulla mauris elit, consectetuer sit amet, accumsan eget, congue ac, libero. Vivamus suscipit. Nunc dignissim consectetuer lectus. Fusce elit nisi; commodo non, facilisis quis, hendrerit eu, dolor? Suspendisse eleifend nisi ut magna. Phasellus id lectus! Vivamus laoreet enim et dolor. Integer arcu mauris, ultricies vel, porta quis, venenatis at, libero. Donec nibh est, adipiscing et, ullamcorper vitae, placerat at, diam. Integer ac turpis vel ligula rutrum auctor! Morbi egestas erat sit amet diam. Ut ut ipsum? Aliquam non sem. Nulla risus eros, mollis quis, blandit ut; luctus eget, urna.");` `paragraph.Alignment = Element.ALIGN_JUSTIFIED;` `jpg.ScaleToFit(250f, 250f);` `jpg.Alignment = Image.TEXTWRAP | Image.ALIGN_RIGHT;` `jpg.IndentationLeft = 9f;` `jpg.SpacingAfter = 9f;`

		jpg.BorderWidthTop = 36f;
		jpg.BorderColorTop = Color.WHITE;
		doc.Add(jpg);
		doc.Add(paragraph);

Layout de la página

Hay ocasiones en que necesitamos tener más de una columna de texto en nuestro documento PDF. Para este propósito, iTextSharp provee el objeto *MultiColumnText*, el cual hace que esta tarea sea una sumamente sencilla de implementar.

Objeto	Descripción	Código

MultiColumnText	Crear 2 columnas definiendo la cantidad de espacio a la izquierda, ala derecho y entre las columnas.
	
	La función *AddRegularColumns* crea columnas exactamente del mismo tamaño.
		string texto = @"Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Suspendisse blandit blandit turpis. Nam in lectus ut dolor consectetuer bibendum. Morbi neque ipsum, laoreet id; dignissim et, viverra id, mauris. Nulla mauris elit, consectetuer sit amet, accumsan eget, congue ac, libero. Vivamus suscipit. Nunc dignissim consectetuer lectus. Fusce elit nisi; commodo non, facilisis quis, hendrerit eu, dolor? Suspendisse eleifend nisi ut magna. Phasellus id lectus! Vivamus laoreet enim et dolor. Integer arcu mauris, ultricies vel, porta quis, venenatis at, libero. Donec nibh est, adipiscing et, ullamcorper vitae, placerat at, diam. Integer ac turpis vel ligula rutrum auctor! Morbi egestas erat sit amet diam. Ut ut ipsum? Aliquam non sem. Nulla risus eros, mollis quis, blandit ut; luctus eget, urna. Vestibulum vestibulum dapibus erat. Proin egestas leo a metus?";
		MultiColumnText col = new MultiColumnText();
		//float izquierda, float derecha, float espacio entre medio, int número de columnas
		col.AddRegularColumns(36f, doc.PageSize.Width-36f, 24f, 2);
		Paragraph p = new Paragraph(texto, new Font(Font.HELVETICA, 8f));
		p.SpacingAfter = 9f;
		p.Alignment = Element.ALIGN_JUSTIFIED;
		col.AddElement(p); // añadir varias veces el párrafo para simular mucho texto
		col.AddElement(p);
		col.AddElement(p);
		col.AddElement(p);
		doc.Add(col);

MultiColumnText	Para crear columnas de diferentes tamaños se utiliza la función *AddSimpleColumn*.	col.AddSimpleColumn(36f, 170f); col.AddSimpleColumn(194f, doc.PageSize.Width - 36f);

Listas

En iTextSharp puedes añadir tanto listas ordenadas como no ordenadas. Es importante mencionar que cada ítem de la lista es creado en una línea nueva.

Objeto	Descripción	Código

List	Crea una lista ordenada o desordenada.	`using it = iTextSharp.text;`
		`…`
		`it.List list = new it.List(it.List.UNORDERED);`
		`list.Add(new it.ListItem("One"));`
		`list.Add("Uno");`
		`list.Add("Dos");`
		`list.Add("Tres");`
		`list.Add("Cuatro");`
		`doc.Add(list);`

	Para cambiar el estilo de la lista podemos utilizar el siguiente pedazo de código. En este código estamos cambiando el tamaño del sangrado, el guion de cada ítem por un punto y la cantidad de espacio entre el símbolo y el texto del ítem.	`it.List list = new it.List(it.List.UNORDERED, 10f);`
		`list.SetListSymbol("\u2022");`
		`list.IndentationLeft = 30f;`

	Para añadir listas ordenadas por números romanos.	`RomanList romanlist = new RomanList(true, 20);`
		`romanlist.IndentationLeft = 30f;`
		`romanlist.Add("Uno");`
		`romanlist.Add("Dos");`
		`romanlist.Add("Tres");`
		`romanlist.Add("Cuatro");`
		`romanlist.Add("Cinco");`
		`doc.Add(romanlist);`

	También es posible crear listas anidadas.	`RomanList romanlist = new RomanList(true, 20);`
		`romanlist.IndentationLeft = 10f;`
		`romanlist.Add("Uno");`
		`romanlist.Add("Dos");`
		`romanlist.Add("Tres");`
		`romanlist.Add("Cuatro");`
		`romanlist.Add("Cinco");`
		
		`List list = new List(List.ORDERED, 20f);`
		`list.SetListSymbol("\u2022");`
		`list.IndentationLeft = 20f;`
		`list.Add("Uno");`
		`list.Add("Dos");`
		`list.Add("Tres");`
		`list.Add("Lista Romana");`
		`list.Add(romanlist);`
		`list.Add("Cuatro");`
		`list.Add("Cinco");`
		
		`doc.Add(list);`

Enlaces y Marcadores

Añadir enlaces en iTextSharp es sumamente parecido a cómo se hace en HTML, el enlace puede ser a algún área interna del documento como a un recurso externo al PDF. La gran diferencia entre añadir un enlace en el PDF vs en HTML es que, en el PDF, el tipo de letra no cambia automaticamente para identificar el enlace. Por esto es sugerido subrayar la palabra enlace y cambiarle el color al típico azul.

Objeto	Descripción	Código

Anchor	De esta manera se crean enlaces que hacen referencia a recursos externos. Por ejemplo, una página web.	`// estilo del tipo de letra del enlace`
		`Font f = FontFactory.GetFont("Arial", 12, Font.UNDERLINE, new Color(0, 0, 255));`
		`Anchor enlace = new Anchor("desarrolladores.me", f);`
		`enlace.Reference = "http://desarrolladores.me";`
		`doc.Add(enlace);`

Anchor	De esta manera se crean enlaces a recursos internos del PDF. Por ejemplo, un enlace hace referencia a otro enlace dentro del mismo documento.
	
	Cuando haces clic en el primer enlace, programa donde esté viendo el PDF, te va a llevar al lugar donde se encuentra el enlace cuyo nombre es *destino*.
		`Anchor clic = new Anchor("Haga clic para ir al destino");`
		`clic.Reference = "#destino";`
		`Paragraph p1 = new Paragraph();`
		`p1.Add(clic);`
		`doc.Add(p1);`
		
		`Paragraph p2 = new Paragraph();`
		`p2.Add(new Chunk("\n\n\n\n\n\n\n\n"));`
		`doc.Add(p2);`
		
		`Anchor dest = new Anchor("Este es el destino");`
		`dest.Name = "destino";`
		`Paragraph p3 = new Paragraph();`
		`p3.Add(dest);`
		`doc.Add(p3);`

Marcadores: iTextSharp permite crear las secciones que vemos en algunos PDFs a mano derecha. Estos son los marcadores. Es buenos pensarlo como si se pudiera dividir un PDF por capítulos y secciones.		

Chapter	Es el objeto del nivel más alto y siempre comienzan en una nueva página.	`Chapter capitulo1 = new Chapter(new Paragraph("Este es el capítulo 1"),1);`
		`capitulo1.BookmarkOpen = true;`
		`doc.Add(capitulo1);`

Section	Este objeto no puede ser añadido sin un ***Chapter***, sino que tiene que ser añadido a un objeto tipo ***Chapter*** o a otro objeto padre tipo ***Section***.	`Chapter capitulo1 = new Chapter(new Paragraph("Este es el capítulo 1"),1);`
		`Section sec1 = capitulo1.AddSection(20f, "Sección 1.1", 2);`
		`Section sec2 = capitulo1.AddSection(20f, "Sección 1.2", 2);`
		`Section subsec1 = sec2.AddSection(20f, "Sub sección 1.2.1", 3);`
		`capitulo1.BookmarkOpen = false;`
		`doc.Add(capitulo1);`