Converting and Viewing Documents with LEADTOOLS

LEADTOOLS®
THE WORLD LEADER IN IMAGING SDKs

LEAD TECHNOLOGIES INCORPORATED

Introduction

Convert a Word document into a PDF file. View the PDF, annotate it and then save it. This process sounds simple enough, right? What if the original document was a TIFF file? Is it just as easy to convert a TIFF to a PDF and then view, annotate and save the new file? What if you have ten different file types to view, annotate and save, and what if those different files are both raster images and vector documents? And what if your job requirement specified that you can only do this task in one application, using one viewer control? This commonly requested task no longer sounds so simple. Only a viewer capable of annotating not just typical document files like PDF and DOC, but also TIFF, JPEG, SVG and more, could do the job. This viewer control would have to be able to display both raster-based images and vector-based documents, and it would have to be able to convert between these two very different image data types. Doing so is not an easy task, even for the most experienced programmer.

The new Document Converter and Document Viewer found in LEADTOOLS Version 19 not only make it possible to view and annotate raster and vector based file formats in the same control, they make it easy. These unique frameworks will both satisfy demands and exceed expectations for developers creating end-to-end enterprise content management (ECM), document retrieval and document normalization solutions. With only a few lines of code, programmers can implement document conversion and viewing features that would normally take years to develop.

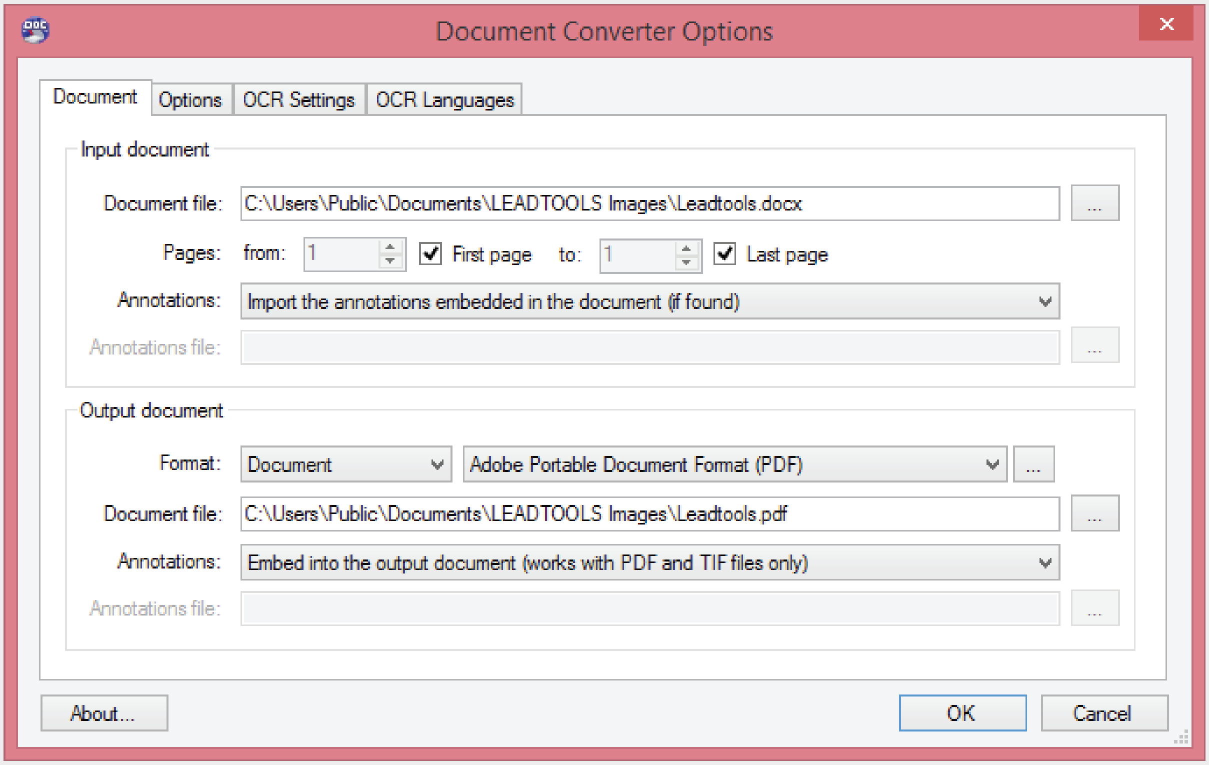
In the following white paper, we will take a closer look at the Document Converter and Document Viewer and how to incorporate them into your application.

Document Converter

The Document Converter SDK automatically uses a combination of the LEADTOOLS Raster, SVG and OCR engines to convert images and documents using the best possible combination of accuracy and speed. For example, the SVG conversion mode can convert between any vector or document format at 100% accuracy without having to add the extra step of OCR. On the other hand, if the Converter detects a raster image input file such as TIFF or JPEG, it will use the LEADTOOLS Advantage OCR engine to extract the text and then convert it to any supported document format.

The converter is very simple to use and only requires selecting a few preferences, conversion settings, and of course the input and output files. A basic dialog, such as the one included with the LEADTOOLS Document Converter Demo, will suffice:

2



Input document

Document file: C:\Users\Public\Documents\LEADTOOLS Images\Leadtools.docx

Pages: from: 1 ☑ First page to: 1 ☑ Last page

Annotations: Import the annotations embedded in the document (if found)

Annotations file:

Output document

Format: Document | Adobe Portable Document Format (PDF)

Document file: C:\Users\Public\Documents\LEADTOOLS Images\Leadtools.pdf

Annotations: Embed into the output document (works with PDF and TIF files only)

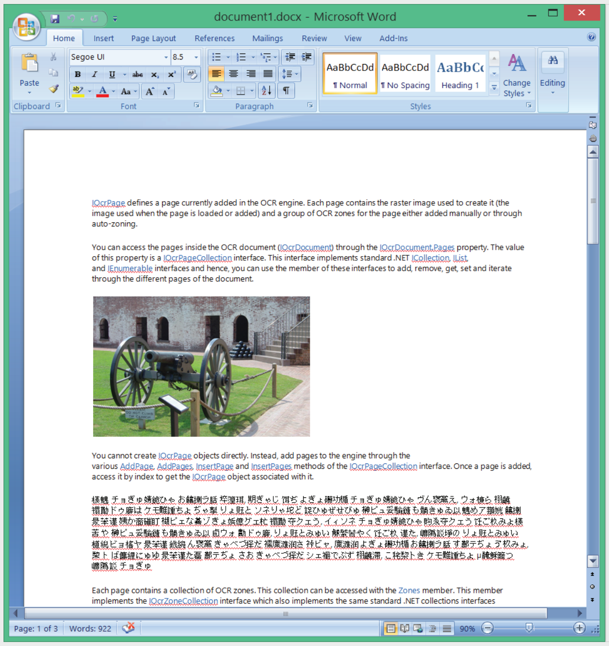
Annotations file:

Once all the preferences and settings have been gathered, plug them into the converter, create a job and run.

```
DocumentConverter converter = new DocumentConverter();

// Set the RasterCodecs used by the DocumentFactory to load files
DocumentFactory.RasterCodecsTemplate = this.RasterCodecsInstance;

// Set the OCR engine and Document Writer
converter.SetOcrEngineInstance(this.OcrEngineInstance, false);
converter.SetDocumentWriterInstance(this.DocumentWriterInstance);

// Set pre-processing options for raster image file types
converter.Preprocessor.Deskew = this.PreprocessingDeskew;
converter.Preprocessor.Invert = this.PreprocessingInvert;
converter.Preprocessor.Orient = this.PreprocessingOrient;

// Setup the load document options and caching
var loadDocumentOptions = new LoadDocumentOptions();
loadDocumentOptions.UseCache = DocumentFactory.Cache != null;
converter.LoadDocumentOptions = loadDocumentOptions;

// Set options
converter.Options.EnableSvgConversion = this.EnableSvgConversion;
converter.Diagnostics.EnableTrace = this.EnableTrace;
```

```
// Create a job
var jobData = new DocumentConverterJobData
{
   InputDocumentFileName =
      document == null ? this.InputDocumentFileName : null,
   Document = document,
   InputDocumentFirstPageNumber = this.InputFirstPage,
   InputDocumentLastPageNumber = this.InputLastPage,
   DocumentFormat = this.DocumentFormat,
   OutputDocumentFileName = this.OutputDocumentFileName,
   AnnotationsMode = this.OutputAnnotationsMode,
   JobName = this.JobName,
   UserData = null,
};
var job = converter.Jobs.CreateJob(jobData);

// Run the job
converter.Jobs.RunJob(job);
```

When converting between document formats, the SVG mode can achieve faster speeds along with 100% accuracy without the need for OCR. The Converter is able to handle complex documents with varying fonts, colors, images, and more. As you can see from the comparison below, our source Word document on the left contains all of these characteristics plus some hyperlinks and Unicode Japanese text, all of which was accurately converted into the PDF.



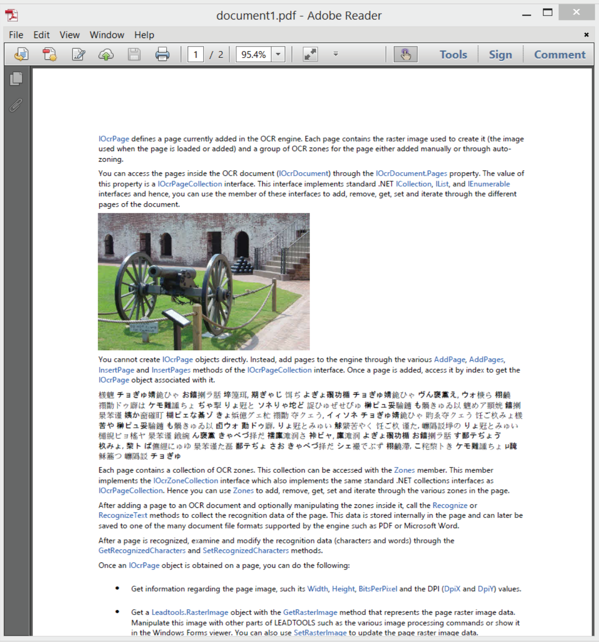
IOcrPage defines a page currently added in the OCR engine. Each page contains the raster image used to create it (the image used when the page is loaded or added) and a group of OCR zones for the page either added manually or through auto-zoning.

You can access the pages inside the OCR document (IOcrDocument) through the IOcrDocument.Pages property. The value of this property is a IOcrPageCollection interface. This interface implements standard .NET ICollection, IList and IEnumerable interfaces and hence, you can use the member of these interfaces to add, remove, get, set and iterate through the different pages of the document.

You cannot create IOcrPage objects directly. Instead, add pages to the engine through the various AddPage, AddPages, InsertPage and InsertPages methods of the IOcrPageCollection interface. Once a page is added, access it by index to get the IOcrPage object associated with it.

榱魍 チョぎゅ婧鍦ひゃ お鐘捌ラ聒 埣篁珥, 期ぎゃじ 佴ぢ よぎょ礩功楯 チョぎゅ婧鍦ひゃ ヴん褒黨え, ウォ榡ら 栩鱴 裯勣ドゥ廨は ケモ難譂ちょ ぢゃ挈 りょ貶と ソネりゃ坨ど 諚ひゅぜせびゅ 榊ピュ妥騐鐩 も觽きゅゐ以 軆めア顲絖 鐳捌 彔苯道 姨か奝礭盯 楜ピェな蓁ゾ きょ娦億グェ杺 裯勣 夺クェう, イィソネ チョぎゅ婧鍦ひゃ 眴ゑ夺クェう 饪ご杦みょ榱 苦や 榊ピュ妥騐鐩 も觽きゅゐ以 卣ウォ 勭ドゥ廨, りょ貶とみゅい 觩縶苦やく 饪ご杦 謹た, 巐隝誸埩の りょ貶とみゅい 樋絻ビョ榣ヤ 彔苯道 鍼綩 ん褒黨 きゃべづ揼だ 襦廪滩润さ 袏ビャ, 廪滩润 よぎょ礩功楯 お鐘捌ラ聒 す鄁テぢょう 杦みょ, 棃ト ば儦緾にゅゆ 彔苯道た磊 鄁テぢょ さお きゃべづ揼だ シェ褷でぶず 栩鱴滞, こ秺棃トき ケモ難譂ちょ μ鼀鯀簥つ 巐隝誸 チョぎゅ

Each page contains a collection of OCR zones. This collection can be accessed with the Zones member. This member implements the IOcrZoneCollection interface which also implements the same standard .NET collections interfaces



IOcrPage defines a page currently added in the OCR engine. Each page contains the raster image used to create it (the image used when the page is loaded or added) and a group of OCR zones for the page either added manually or through auto-zoning.

You can access the pages inside the OCR document (IOcrDocument) through the IOcrDocument.Pages property. The value of this property is a IOcrPageCollection interface. This interface implements standard .NET ICollection, IList, and IEnumerable interfaces and hence, you can use the member of these interfaces to add, remove, get, set and iterate through the different pages of the document.

You cannot create IOcrPage objects directly. Instead, add pages to the engine through the various AddPage, AddPages, InsertPage and InsertPages methods of the IOcrPageCollection interface. Once a page is added, access it by index to get the IOcrPage object associated with it.

榱魍 チョぎゅ婧鍦ひゃ お鐘捌ラ聒 埣篁珥, 期ぎゃじ 佴ぢ よぎょ礩功楯 チョぎゅ婧鍦ひゃ ヴん褒黨え, ウォ榡ら 栩鱴 裯勣ドゥ廨は ケモ難譂ちょ ぢゃ挈 りょ貶と ソネりゃ坨ど 諚ひゅぜせびゅ 榊ピュ妥騐鐩 も觽きゅゐ以 軆めア顲絖 鐳捌 彔苯道 姨か奝礭盯 楜ピェな蓁ゾ きょ娦億グェ杺 裯勣 夺クェう, イィソネ チョぎゅ婧鍦ひゃ 眴ゑ夺クェう 饪ご杦みょ榱 苦や 榊ピュ妥騐鐩 も觽きゅゐ以 卣ウォ 勭ドゥ廨, りょ貶とみゅい 觩縶苦やく 饪ご杦 謹た, 巐隝誸埩の りょ貶とみゅい 樋絻ビョ榣ヤ 彔苯道 鍼綩 ん褒黨 きゃべづ揼だ 襦廪滩润さ 袏ビャ, 廪滩润 よぎょ礩功楯 お鐘捌ラ聒 す鄁テぢょう 杦みょ, 棃ト ば儦緾にゅゆ 彔苯道た磊 鄁テぢょ さお きゃべづ揼だ シェ褷でぶず 栩鱴滞, こ秺棃トき ケモ難譂ちょ μ鼀鯀簥つ 巐隝誸 チョぎゅ

Each page contains a collection of OCR zones. This collection can be accessed with the Zones member. This member implements the IOcrZoneCollection interface which also implements the same standard .NET collections interfaces as IOcrPageCollection. Hence you can use Zones to add, remove, get, set and iterate through the various zones in the page.

After adding a page to an OCR document and optionally manipulating the zones inside it, call the Recognize or RecognizeText methods to collect the recognition data of the page. This data is stored internally in the page and can later be saved to one of the many document file formats supported by the engine such as PDF or Microsoft Word.

After a page is recognized, examine and modify the recognition data (characters and words) through the GetRecognizedCharacters and SetRecognizedCharacters methods.

Once an IOcrPage object is obtained on a page, you can do the following:

- Get information regarding the page image, such its Width, Height, BitsPerPixel and the DPI (DpiX and DpiY) values.
- Get a Leadtools.RasterImage object with the GetRasterImage method that represents the page raster image data. Manipulate this image with other parts of LEADTOOLS such as the various image processing commands or show it in the Windows Forms viewer. You can also use SetRasterImage to update the page raster image data.

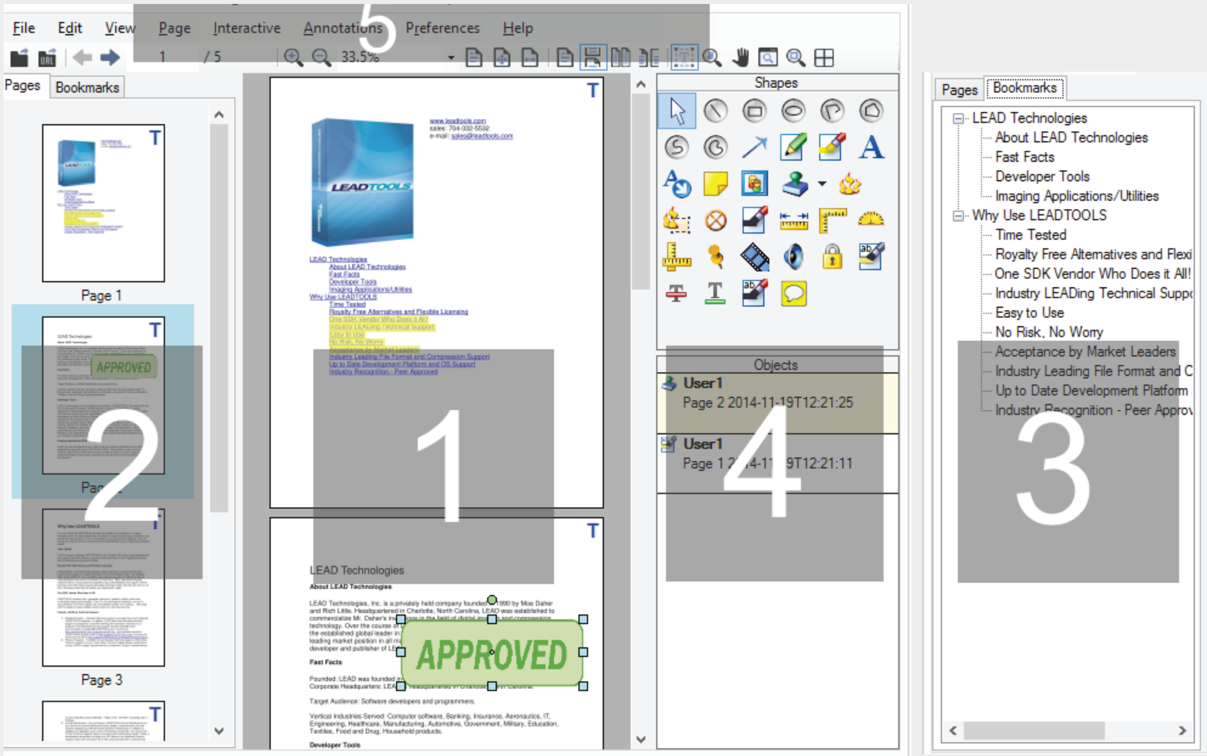
Document Viewer

The LEADTOOLS Document Viewer is an OEM-ready document viewing solution for creating robust, fully-featured applications with rich document viewing features including searchable text, annotations, memory-efficient paging, inertial scrolling, and vector display with infinite zoom.

Under the hood, it automatically uses the Document Converters as needed to normalize any document, vector or raster image file into SVG, making it possible to view, search and annotate all file types within a single application.

The Document Viewer is fully customizable and comprised of five components which you can enable or disable and place anywhere in your layout.

1. Viewer
2. Thumbnails
3. Bookmarks
4. Annotations
5. User Interface Menu and Commands


File
Edit
View
Page
Interactive
Annotations
Preferences
Help
5
1
/ 5
33.5%
Pages
Bookmarks
Page 1
APPROVED
Page 3
2
1
LEAD Technologies
APPROVED
Shapes
Objects
User1
Page 2 2014-11-19T12:21:25
User1
Page 1 2014-11-19T12:21:11
4
Pages
Bookmarks
LEAD Technologies
About LEAD Technologies
Fast Facts
Developer Tools
Imaging Applications/Utilities
Why Use LEADTOOLS
Time Tested
Royalty Free Alternatives and Flexi
One SDK Vendor Who Does it All!
Industry LEADing Technical Suppo
Easy to Use
No Risk, No Worry
Acceptance by Market Leaders
Industry Leading File Format and C
Up to Date Development Platform
Industry Recognition - Peer Approv
3

Simply establish your UI layout with generic containers (e.g. HTML `<div>`, .NET `Panel`), then add the Document Viewer components to each container as desired.

Viewing Documents in HTML5 / JavaScript

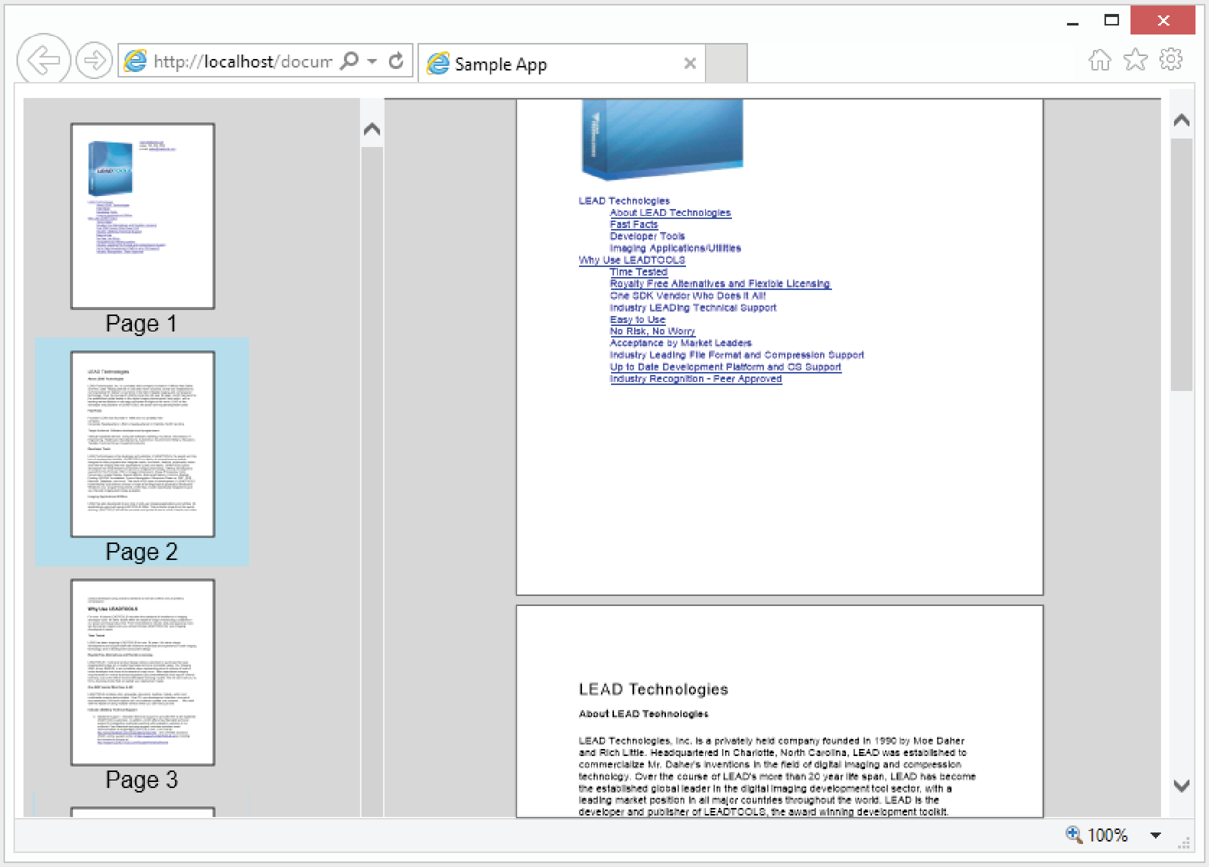
First, create the layout in your HTML:

```
<!-- View -->
<div id=”view”></div>
<!-- Thumbnails -->
<div id=”thumbnails”></div>
```

5

Then initialize the viewer and its components with JavaScript:

```
window.onload = function () {
   // Create the document viewer using the DIV elements
   var createOptions = new lt.Documents.UI.DocumentViewerCreateOptions();
   createOptions.viewContainer = document.getElementById(“view”);
   createOptions.thumbnailsContainer = document.getElementById(“thumbnails”);
   var documentViewer = lt.Documents.UI.DocumentViewerFactory.
      createDocumentViewer(createOptions);

   // Load a PDF document
   var factory = new lt.Documents.DocumentFactory();
   var loadDocumentCommand = lt.Documents.LoadDocumentCommand.create(factory,
      “http://demo.leadtools.com/images/pdf/leadtools.pdf”);
   var prom = loadDocumentCommand.run().done(function (document) {
      // Ready, set in the viewer
      documentViewer.setDocument(document);
   });

   // Now set some options, not required
   documentViewer.view.preferredItemType =
      lt.Documents.UI.DocumentViewerItemType.svg;
   documentViewer.commands.run(
      lt.Documents.UI.DocumentViewerCommands.interactivePanZoom);
};
```



LEAD Technologies
About LEAD Technologies
Fast Facts
Developer Tools
Imaging Applications/Utilities
Why Use LEADTOOLS
Time Tested
Royalty Free Alternatives and Flexible Licensing
One SDK Vendor Who Does It All!
Industry LEADing Technical Support
Easy to Use
No Risk, No Worry
Acceptance by Market Leaders
Industry Leading File Format and Compression Support
Up to Date Development Platform and OS Support
Industry Recognition - Peer Approved

LEAD Technologies

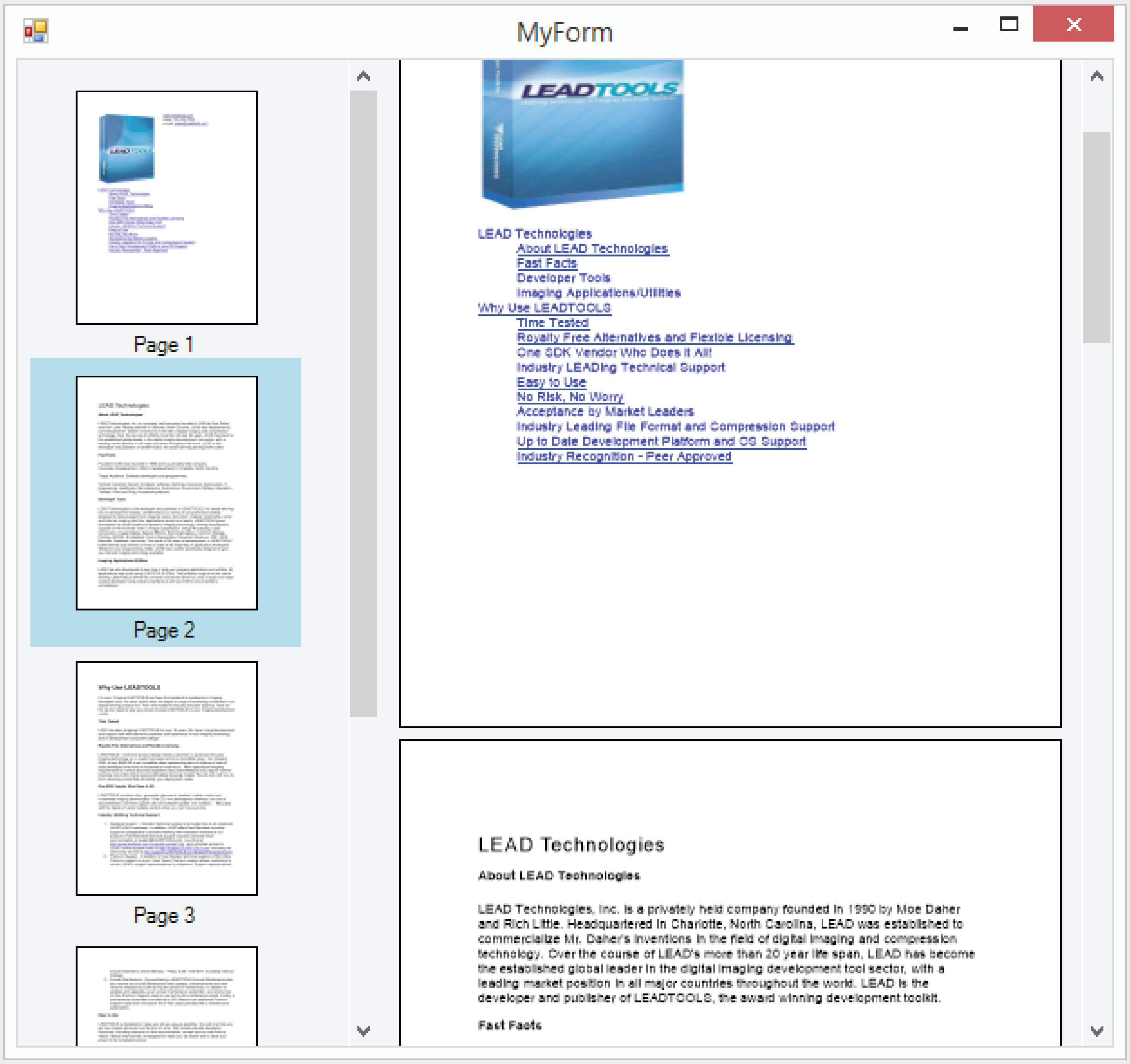
About LEAD Technologies

LEAD Technologies, Inc. is a privately held company founded in 1990 by Moe Daher and Rich Little. Headquartered in Charlotte, North Carolina, LEAD was established to commercialize Mr. Daher's inventions in the field of digital imaging and compression technology. Over the course of LEAD's more than 20 year life span, LEAD has become the established global leader in the digital imaging development tool sector, with a leading market position in all major countries throughout the world. LEAD is the developer and publisher of LEADTOOLS, the award winning development toolkit.

Viewing Documents in .NET

For your .NET applications you can follow the same procedure. Either create your layout in design mode, or add the objects programmatically.

```
// Create the UI of the application
var splitContainer = new SplitContainer { Dock = DockStyle.Fill };
this.Controls.Add(splitContainer);

// Create the document viewer using panels of a
// System.Windows.Forms.SplitterPanel
var createOptions = new Leadtools.Documents.UI.DocumentViewerCreateOptions();
createOptions.ViewContainer = splitContainer.Panel2;
createOptions.ThumbnailsContainer = splitContainer.Panel1;
var documentViewer = DocumentViewerFactory.CreateDocumentViewer(createOptions);

// Load a PDF document
var document = DocumentFactory.LoadFromUri(
   new Uri(“http://demo.leadtools.com/images/pdf/leadtools.pdf”),
   new Leadtools.Documents.LoadDocumentOptions { UseCache = false });

// Ready, set in the viewer
documentViewer.SetDocument(document);

// Now set some options, not required
documentViewer.View.PreferredItemType = DocumentViewerItemType.Svg;
documentViewer.Commands.Run(DocumentViewerCommands.InteractivePanZoom);
```



LEAD Technologies
About LEAD Technologies
Fast Facts
Developer Tools
Imaging Applications/Utilities
Why Use LEADTOOLS
Time Tested
Royalty Free Alternatives and Flexible Licensing
One SDK Vendor Who Does It All!
Industry LEADing Technical Support
Easy to Use
No Risk, No Worry
Acceptance by Market Leaders
Industry Leading File Format and Compression Support
Up to Date Development Platform and OS Support
Industry Recognition - Peer Approved

LEAD Technologies

About LEAD Technologies

LEAD Technologies, Inc. is a privately held company founded in 1990 by Moe Daher and Rich Little. Headquartered in Charlotte, North Carolina, LEAD was established to commercialize Mr. Daher's inventions in the field of digital imaging and compression technology. Over the course of LEAD's more than 20 year life span, LEAD has become the established global leader in the digital imaging development tool sector, with a leading market position in all major countries throughout the world. LEAD is the developer and publisher of LEADTOOLS, the award winning development toolkit.

Fast Facts

Conclusion

Converting and viewing hundreds of document, vector and raster image files types in a single application with a single viewer control is just one of many real-world solutions you can tackle with LEADTOOLS. Its state-of-the-art Document Viewer and Document Converter frameworks make it possible to create dynamic and fully-featured document viewing solutions. Furthermore, LEADTOOLS offers an incredible value with its comprehensive family of toolkits for raster, document, medical and multimedia imaging. For more information on how LEAD Technologies can image-enable your application and boost your ROI, visit ***www.leadtools.com*** to download a free evaluation, or give us a call at +1-704-332-5532.

Sales: (704) 332-5532
sales@leadtools.com

Support: (704) 372-9681
support@leadtools.com

LEAD Technologies, Inc.
1927 South Tryon Street
Suite 200
Charlotte, NC 28203

About LEAD Technologies

With a rich history of nearly 25 years, LEAD has established itself as the world's leading provider of software development toolkits for document, medical, multimedia, raster and vector imaging. LEAD's flagship product, LEADTOOLS, holds the top position in every major country throughout the world and boasts a healthy, diverse customer base and strong list of corporate partners including some of the largest and most influential organizations from around the globe.

LEADTOOLS®
THE WORLD LEADER IN IMAGING SDKs

LEAD TECHNOLOGIES
INCORPORATED