MuPDFCore

1.1.0

Generated by Doxygen 1.8.18

Chapter 1

MuPDFCore: Multiplatform .NET Core bindings for MuPDF

MuPDFCore is a set of multiplatform .NET Core bindings for `MuPDF`. It can render PDF, XPS, EPUB and other formats to raster images returned either as raw bytes, or as image files in multiple formats (including PNG and PSD). It also supports multithreading.

It also includes **MuPDFCore.MuPDFRenderer**, an Avalonia control to display documents compatible with MuPDFCore in Avalonia windows (with multithreaded rendering).

The library is released under the `AGPLv3` licence.

1.1 Getting started

The MuPDFCore library targets .NET Standard 2.0, thus it can be used in projects that target .NET Standard 2.0+, .NET Core 2.0+, .NET Framework 4.6.1 and possibly others. MuPDFCore includes a pre-compiled native library, thus projects using it can only run on Windows, macOS and Linux x64 operating systems.

To use the library in your project, you should install the `MuPDFCore NuGet package` and/or the `MuPDFCore.PDFRenderer NuGet package`.

1.2 Usage

1.2.1 Documentation

Interactive documentation for the library can be accessed from the `documentation website`. A `PDF reference manual` is also available.

1.2.2 Examples

The `Demo` folder in the repository contains some examples of how the library can be used to extract pages from a PDF or XPS document, render them to a raster image, or combine them in a new document

The `PDFViewerDemo` folder contains a complete (though minimal) example of a PDF viewer program built around the `MuPDFCore.MuPDFRenderer.PDFRenderer` control.

Note that these examples intentionally avoid any error handling code: in a production setting, you should typically make sure that calls to MuPDFCore library functions are within a `try...catch` block to handle any resulting `MuPDFException`s.

1.2.3 MuPDFCore library

The first step when using MuPDFCore is to create a `MuPDFCore.MuPDFContext` object that is used internally by the MuPDF library to store various things:

```
MuPDFContext context = new MuPDFContext();
```

This object is `IDisposable`, therefore you should always call the `Dispose()` method on it once you are done with it (or, better yet, wrap it in a `using` directive). In most instances, you will only need one instance of `MuPDFContext` for your whole application.

Amongst other things, MuPDF uses this context to store a cache of "assets" (e.g. images or fonts) that have been used while rendering documents and that may be needed in future. This requires some memory: by default, the maximum size of this cache store is 256MB; however, if you want to restrict how much memory can be used, you can alter this by providing a `long` value to constructor, indicating the size in bites for the store. A value of `0` means that the store can grow up to an unlimited size. Furthermore, you can clear the cache completely by using the `MuPDFContext.ClearCache` method, or partially by using the `MuPDFContext.ShrinkCache` method.

Once you have obtained a `MuPDFContext`, you can use it to open a `MuPDFDocument`. A document can be opened from a file on disk:

```
MuPDFDocument document = new MuPDFDocument(context, "path/to/file");
```

Or from a `byte[]` array (in this case, you will have to specify the format of the document):

```
byte[] data;
...
MuPDFDocument document = new MuPDFDocument(context, data, InputFileTypes.PDF);
```

Or from a `MemoryStream` (in this case too, you will have to specify the format of the document):

```
MemoryStream stream;
...
MuPDFDocument document = new MuPDFDocument(context, ref stream, InputFileTypes.PDF);
```

The `MemoryStream` is passed with the `ref` keyword to indicate that the `MuPDFDocument` will take care of appropriately disposing it once it finishes using it.

A `MuPDFDocument` is also `IDisposable` and should be properly disposed of to avoid memory leaks.

Important note: the constructor taking a `byte[]` and the one taking a `MemoryStream` will not copy the data bytes before sending them to the native MuPDF library functions. Rather, they will *pin them in place*. This is a **bad thing** because it will mess up with the Garbage Collector's management of memory. Therefore, this is only suitable for short-lived objects. If you need to initialise a long-lived document object from memory, you should first copy the data to unmanaged memory and then use one of the constructors that take an `IntPtr` parameter, e.g.:

```
byte[] data;
...
//Allocate enough unmanaged memory
IntPtr ptr = Marshal.AllocHGlobal(data.Length);
//Copy the byte array to unmanaged memory
Marshal.Copy(data, 0, ptr, data.Length);
//Wrap the pointer in an IDisposable
IDisposable dispIntPtr = new DisposableIntPtr(ptr);
//Create the document
MuPDFDocument document = new MuPDFDocument(ctx, ptr, data.Length, InputFileTypes.PDF, ref dispIntPtr);
```

The `DisposableIntPtr` class is a wrapper around a pointer that calls `Marshal.FreeHGlobal` on it once it is disposed. Passing it as the final optional parameter of `MuPDFDocument` constructor (again by reference, to indicate that the document takes ownership of the object) makes sure that the memory is properly freed once the document is disposed.

After having obtained a document, you can do many things with it: for example, you can render a page and save the results to a file on disk, or you can collect multiple pages and combine them in a new document. Code to do this can be found in the `Program.cs` file of the Demo project.

Furthermore, you can render a page directly to memory:

```
byte[] pixelData = document.Render(0, 1, PixelFormats.RGBA);
```

This method renders page 0 (i.e. the first page of the document) at a 1x resolution (1pt in the document is equivalent to 1px in the image), preserving alpha (transparency) information, and returns the image as an array of the bytes that constitute the pixel data (four bytes per pixel). A variation of this method allows you to supply a rectangular region of the page that you would like to render, rather than the whole page.

Alternatively, if you already know where the image data should be put (e.g. because you are using some kind of graphics library that lets you manipulate the pixel data of its images), you can use the methods that take an `IntPtr` destination:

```
IntPtr destination;
...
document.Render(0, 1, PixelFormats.RGBA, destination);
```

In this case, **you have to make sure that there is enough memory to hold the resulting image**! Otherwise, an `AccessViolationException` will occur and your program will usually fail catastrophically. Since it may sometimes be hard to determine how much memory a particular image will need (especially because of subtle differences in the rounding routines, which can cause images to be 1px larger or shorter than expected), the `GetRenderedSize` method is provided, which returns the number of bytes that will be needed to render a certain page. For example:

```
//Get the number of bytes that will be necessary to hold the rendered page at the given resolution.
int sizeInBytes = document.GetRenderedSize(0, 1, PixelFormats.RGBA);
//Allocate an appropriate amount of memory.
IntPtr destination = Marshal.AllocHGlobal(sizeInBytes);
//Again, we use a DisposableIntPtr to make sure that we are freeing the memory when we are done with it.
using (DisposableIntPtr holder = new DisposableIntPtr(destination))
{
    //Make sure that all the parameters match those of the call to GetRenderedSize, or the size of the
    //resulting image may be different than expected! Even a translation of 1px could have catastrophic
    //consequences.
    document.Render(0, 1, PixelFormats.RGBA, destination);
}
```

Finally, **none of these methods are inherently thread-safe**! E.g. you cannot render multiple pages of the same document (nor multiple regions of a single page) by simply performing multiple calls to `MuPDFDocument.Render` in parallel. For multi-threaded operation, you must instead use a `MuPDFMultiThreadedPageRender`. You can obtain one from a document:

```
MuPDFMultiThreadedPageRenderer renderer = document.GetMultiThreadedRenderer(0, 2);
```

This method obtains an object that can be used to render the first page of the document using two threads. By using the `Render` method of this object, the page can be rendered. The page will be rendered to a number of separate tiles equal to the number of threads, which will then be your responsibility to appropriately "stitch up" (e.g. if you want to display them on screen, you could just place them appropriately). The size of each tile (and the position it should occupy) can be computed by using the `Split` method of the `RoundedSize` struct.

Furthermore, multiple `MuPDFMultiThreadedPageRenderer`s can be used in parallel, which makes it possible e.g. to render every page in the document at the same time (while also using multiple threads to render each page). The following example will render all the pages in a document at the same time in RGBA format at a 1.5x zoom, using 2 threads for each page:

```
//Create a MuPDFContext with a using statement, so that it gets disposed at the right time.
using MuPDFContext context = new MuPDFContext();
//Open the document also with a using statement.
using MuPDFDocument document = new MuPDFDocument(context, "path/to/file.pdf");
//Create arrays to hold the objects for the various pages
//Renderers: one per page
MuPDFMultiThreadedPageRenderer[] renderers = new MuPDFMultiThreadedPageRenderer[document.Pages.Count];
//Page size: one per page
RoundedSize[] renderedPageSizes = new RoundedSize[document.Pages.Count];
//Boundaries of the tiles that make up each page: one array per page, with one element per thread
RoundedRectangle[][] tileBounds = new RoundedRectangle[document.Pages.Count][];
//Addresses of the memory areas where the image data of the tiles will be stored: one array per page, with
      one element per thread
IntPtr[][] destinations = new IntPtr[document.Pages.Count][];
//Cycle through the pages in the document to initialise everything
for (int i = 0; i < document.Pages.Count; i++)
{
    //Initialise the renderer for the current page, using two threads (total number of threads: number of
       pages x 2
    renderers[i] = document.GetMultiThreadedRenderer(i, 2);
    //Determine the boundaries of the page when it is rendered with a 1.5x zoom factor
    RoundedRectangle roundedBounds = document.Pages[i].Bounds.Round(1.5);
    renderedPageSizes[i] = new RoundedSize(roundedBounds.Width, roundedBounds.Height);
    //Determine the boundaries of each tile by splitting the total size of the page by the number of
       threads.
```

```
    tileBounds[i] = renderedPageSizes[i].Split(renderers[i].ThreadCount);
    destinations[i] = new IntPtr[renderers[i].ThreadCount];
    for (int j = 0; j < renderers[i].ThreadCount; j++)
    {
        //Allocate the required memory for the j-th tile of the i-th page.
        //Since we will be rendering with a 24-bit-per-pixel format, the required memory in bytes is height
       x width x 3.
        destinations[i][j] = Marshal.AllocHGlobal(tileBounds[i][j].Height * tileBounds[i][j].Width * 3);
    }
}
//Start the actual rendering operations in parallel.
Parallel.For(0, document.Pages.Count, i =>
{
    renderers[i].Render(renderedPageSizes[i], document.Pages[i].Bounds, destinations[i], PixelFormats.RGB);
});
//The code in this for-loop is not really part of MuPDFCore - it just shows an example of using
       SixLabors.ImageSharp to "stitch" the tiles up and produce the full image.
for (int i = 0; i < document.Pages.Count; i++)
{
    //Create a new (empty) image to hold the whole page.
    SixLabors.ImageSharp.Image renderedPage = new
       SixLabors.ImageSharp.Image<SixLabors.ImageSharp.PixelFormats.Rgb24>(renderedPageSizes[i].Width,
       renderedPageSizes[i].Height);
    //Draw each tile onto the image.
    for (int j = 0; j < renderers[i].ThreadCount; j++)
    {
        ReadOnlySpan<byte> imageData;
        //By using unsafe code, we can avoid having to marshal the image data around.
        unsafe
        {
            //Create a new ReadOnlySpan that reads the unmanaged memory where the image data is located.
            imageData = new ReadOnlySpan<byte>((void*)destinations[i][j], tileBounds[i][j].Height *
       tileBounds[i][j].Width * 3);
        }
        //Load the image data in the tile by using the ReadOnlySpan.
        SixLabors.ImageSharp.Image tile =
       SixLabors.ImageSharp.Image.LoadPixelData<SixLabors.ImageSharp.PixelFormats.Rgb24>(imageData,
       tileBounds[i][j].Width, tileBounds[i][j].Height);
        //Draw the tile on the main image page.
        renderedPage.Mutate(x => x.DrawImage(tile, new SixLabors.ImageSharp.Point(tileBounds[i][j].X0,
       tileBounds[i][j].Y0), 1));
        //Release the resources held by the tile.
        tile.Dispose();
    }
    //Save the full page as a JPG image.
    using (FileStream fs = new FileStream("page" + i.ToString() + ".jpg", FileMode.Create))
    {
        renderedPage.SaveAsJpeg(fs);
    }
    //Release the resources held by the image.
    renderedPage.Dispose();
}
//Clean-up code.
for (int i = 0; i < document.Pages.Count; i++)
{
    //Release the allocated memory.
    for (int j = 0; j < renderers[i].ThreadCount; j++)
    {
        Marshal.FreeHGlobal(destinations[i][j]);
    }
    //Release the renderer (if you skip this, the quiescent renderer's threads will not be stopped, and your
       application will never exit!
    renderers[i].Dispose();
}
```

1.2.4 MuPDFCore.MuPDFRenderer control

To use the `PDFRenderer` control in an Avalonia application, first of all you need to add it to you Avalonia `Window`, e.g. in the XAML:

```
<Window xmlns="https://github.com/avaloniaui"
        ...
        xmlns:mupdf="clr-namespace:MuPDFCore.MuPDFRenderer;assembly=MuPDFCore.MuPDFRenderer"
        Opened="WindowOpened"
        ... >
    <mupdf:PDFRenderer Name="MuPDFRenderer" />
</Window>
```

You then need to initialise it from the backing code, e.g. in a `WindowOpened` event:

```
private void WindowOpened(object sender, EventArgs e)
{
    this.FindControl<PDFRenderer>("MuPDFRenderer").Initialize("path/to/file.pdf");
```

```
}
```

This way, the renderer will start showing the first page of the specified document, using a number of rendering threads that is decided based on the number of processors in the computer. There are many other ways to initialise a PDFRenderer, so make sure to look at the documentation to see the other possibilities!

1.3 Building from source

Building the MuPDFCore library from source requires the following steps:

1. Building the `libmupdf` native library
2. Building the `MuPDFWrapper` native library
3. Creating the `MuPDFCore` library NuGet package

Steps 1 and 2 need to be performed on all of Windows, macOS and Linux (no cross-compiling)! Otherwise, some native assets will be missing and it will not be possible to build the NuGet package.

1.3.1 1. Building libmupdf

You can download the open-source (GNU AGPL) MuPDF source code from here. You will need to uncompress the source file and compile the library on Windows, macOS and Linux. You need the following files:

- From Windows:
  - libmupdf.lib
  - libthirdparty.lib
- From macOS:
  - libmupdf.a
  - libmupdf-third.a
- From Linux:
  - libmupdf.a
  - libmupdf-third.a

Note that the files from macOS and Linux are different, despite sharing the same name.

Depending on your system, on Linux and/or macOS you may need to enable the `-fPIC` compiler option to generate library files that can be included in the MuPDFWrapper shared library, otherwise a later step may fail. You can do this in multiple ways, e.g. by opening the `Makefile` included in the MuPDF source and adding `-fPIC` at the end of the line specifying `CFLAGS` (line 23 in the MuPDF 1.17.0 source).

For convenience, these compiled files for MuPDF 1.17.0 are included in the `native/MuPDFWrapper/lib` folder of this repository.

1.3.2 2. Building MuPDFWrapper

Once you have the required static library files, you should download the MuPDFCore source code: `MuPDFCore-1.0.0.tar.gz` (or clone the repository) and place the library files in the appropriate subdirectories in the `native/MuPDFWrapper/lib/` folder.

To compile `MuPDFWrapper` you will need `CMake` and (on Windows) `Ninja`.

On Windows, the easiest way to get all the required tools is probably to install `Visual Studio`. By selecting the "Desktop development with C++" workload you should get everything you need.

On macOS, you will need to install at least the Command-Line Tools for Xcode (if necessary, you should be prompted to do this while you perform the following steps) and CMake.

Once you have everything at the ready, you will have to build MuPDFWrapper on the three platforms.

1.3.2.1 Windows

1. Assuming you have installed Visual Studio, you should open the "__x64__ Native Tools Command Prompt for VS" (you should be able to find this in the Start menu). Take care to open the x64 version, otherwise you will not be able to compile the library. A normal command propmpt will not work, either.
2. `CD` to the directory where you have downloaded the MuPDFCore source code.
3. `CD` into the `native` directory.
4. Type `build`. This will start the `build.cmd` batch script that will delete any previous build and compile the library.

After this finishes, you should find a file named `MuPDFWrapper.dll` in the `native/out/build/win-x64/MuPDFWrapper/` directory. Leave it there.

1.3.2.2 macOS and Linux

1. Assuming you have everything ready, open a terminal in the folder where you have downloaded the MuPDFCore source code.
2. `cd` into the `native` directory.
3. Type `chmod +x build.sh`.
4. Type `./build.sh`. This will delete any previous build and compile the library.

After this finishes, you should find a file named `libMuPDFWrapper.dylib` in the `native/out/build/mac-x64/MuPDFWrapper/` directory (on macOS) and a file named `libMuPDFWrapper.so` in the `native/out/build/linux-x64/MuPDFWrapper/` directory (on Linux). Leave it there.

1.3.3 3. Creating the MuPDFCore NuGet package

Once you have the `MuPDFWrapper.dll`, `libMuPDFWrapper.dylib` and `libMuPDFWrapper.so` files, make sure they are in the correct folders (`native/out/build/xxx-x64/MuPDFWrapper/`), **all on the same machine**.

To create the MuPDFCore NuGet package, you will need the `.NET Core 2.0 SDK or higher` for your platform. Once you have installed it and have everything ready, open a terminal in the folder where you have downloaded the MuPDFCore source code and type:

```
cd MuPDFCore
dotnet pack -c Release
```

This will create a NuGet package in `MuPDFCore/bin/Release`. You can install this package on your projects by adding a local NuGet source.

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Avalonia Namespace Reference

5.2 Avalonia.Animation Namespace Reference

Classes

- class RectTransition

  Transition class that handles AvaloniaProperty with Rect types.

5.3 MuPDFCore Namespace Reference

Classes

- class DisposableIntPtr

  An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.
- class MuPDFContext

  A wrapper around a MuPDF context object, which contains the exception stack and the resource cache store.
- class MuPDFDocument

  A wrapper over a MuPDF document object, which contains possibly multiple pages.
- class MuPDFException

  The exception that is thrown when a MuPDF operation fails.
- class MuPDFMultiThreadedPageRenderer

  A class that holds the necessary resources to render a page of a MuPDF document using multiple threads.
- class MuPDFPage

  A wrapper over a MuPDF page object, which contains information about the page's boundaries.
- class MuPDFPageCollection

  A lazy collection of MuPDFPages. Each page is loaded from the document as it is requested for the first time.
- struct Rectangle

  Represents a rectangle.
- class RenderProgress

  Holds a summery of the progress of the current rendering operation.
- struct RoundedRectangle

  Represents a rectangle using only integer numbers.
- struct RoundedSize

  Represents the size of a rectangle using only integer numbers.
- struct Size

  Represents the size of a rectangle.

Enumerations

- enum ExitCodes {
  ExitCodes.ERR_CANNOT_CREATE_CONTEXT = 129, ExitCodes.ERR_CANNOT_REGISTER_HANDLERS = 130, ExitCodes.ERR_CANNOT_OPEN_FILE = 131, ExitCodes.ERR_CANNOT_COUNT_PAGES = 132, ExitCodes.ERR_CANNOT_RENDER = 134, ExitCodes.ERR_CANNOT_OPEN_STREAM = 135, ExitCodes.ERR_CANNOT_L = 136, ExitCodes.ERR_CANNOT_COMPUTE_BOUNDS = 137, ExitCodes.ERR_CANNOT_INIT_MUTEX = 138, ExitCodes.ERR_CANNOT_CLONE_CONTEXT = 139, ExitCodes.ERR_CANNOT_SAVE = 140, ExitCodes.ERR_CANNOT_CREATE_BUFFER = 141, ExitCodes.ERR_CANNOT_CREATE_WRITER = 142, ExitCodes.ERR_CANNOT_CLOSE_DOCUMENT = 143, ExitCodes.EXIT_SUCCESS = 0 }

  Exit codes returned by native methods describing various errors that can occur.
- enum InputFileTypes {
  InputFileTypes.PDF = 0, InputFileTypes.XPS = 1, InputFileTypes.CBZ = 2, InputFileTypes.PNG = 3, InputFileTypes.JPEG = 4, InputFileTypes.BMP = 5, InputFileTypes.GIF = 6, InputFileTypes.TIFF = 7, InputFileTypes.PNM = 8, InputFileTypes.PAM = 9, InputFileTypes.EPUB = 10, InputFileTypes.FB2 = 11 }

  File types supported in input by the library.
- enum RasterOutputFileTypes { RasterOutputFileTypes.PNM = 0, RasterOutputFileTypes.PAM = 1, RasterOutputFileTypes.PNG = 2, RasterOutputFileTypes.PSD = 3 }

  Raster image file types supported in output by the library.
- enum DocumentOutputFileTypes { DocumentOutputFileTypes.PDF = 0, DocumentOutputFileTypes.SVG = 1, DocumentOutputFileTypes.CBZ = 2 }

  Document file types supported in output by the library.
- enum PixelFormats { PixelFormats.RGB = 0, PixelFormats.RGBA = 1, PixelFormats.BGR = 2, PixelFormats.BGRA = 3 }

  Pixel formats supported by the library.

5.3.1 Enumeration Type Documentation

5.3.1.1 DocumentOutputFileTypes

`enum MuPDFCore.DocumentOutputFileTypes [strong]`

Document file types supported in output by the library.

Enumerator

PDF	Portable Document Format.
SVG	Scalable Vector Graphics.
CBZ	Comic book archive format.

Definition at line 199 of file MuPDF.cs.

5.3.1.2 ExitCodes

`enum MuPDFCore.ExitCodes [strong]`

Exit codes returned by native methods describing various errors that can occur.

Enumerator

ERR_CANNOT_CREATE_CONTEXT	An error occurred while creating the context object.
ERR_CANNOT_REGISTER_HANDLERS	An error occurred while registering the default document handlers with the context.
ERR_CANNOT_OPEN_FILE	An error occurred while opening a file.
ERR_CANNOT_COUNT_PAGES	An error occurred while determining the total number of pages in the document.
ERR_CANNOT_RENDER	An error occurred while rendering the page.
ERR_CANNOT_OPEN_STREAM	An error occurred while opening the stream.
ERR_CANNOT_LOAD_PAGE	An error occurred while loading the page.
ERR_CANNOT_COMPUTE_BOUNDS	An error occurred while computing the page bounds.
ERR_CANNOT_INIT_MUTEX	An error occurred while initialising the mutexes for the lock mechanism.
ERR_CANNOT_CLONE_CONTEXT	An error occurred while cloning the context.
ERR_CANNOT_SAVE	An error occurred while saving the page to a raster image file.
ERR_CANNOT_CREATE_BUFFER	An error occurred while creating the output buffer.
ERR_CANNOT_CREATE_WRITER	An error occurred while creating the document writer.
ERR_CANNOT_CLOSE_DOCUMENT	An error occurred while finalising the document file.
EXIT_SUCCESS	No error occurred. All is well.

Definition at line 26 of file MuPDF.cs.

5.3.1.3 InputFileTypes

```
enum MuPDFCore.InputFileTypes  [strong]
```

File types supported in input by the library.

Enumerator

PDF	Portable Document Format.
XPS	XML Paper Specification document.
CBZ	Comic book archive file (ZIP archive containing page scans).
PNG	Portable Network Graphics format.
JPEG	Joint Photographic Experts Group image.
BMP	Bitmap image.
GIF	Graphics Interchange Format.
TIFF	Tagged Image File Format.
PNM	Portable aNyMap graphics format.
PAM	Portable Arbitrary Map graphics format.
EPUB	Electronic PUBlication document.
FB2	FictionBook document.

Definition at line 107 of file MuPDF.cs.

5.3.1.4 PixelFormats

enum MuPDFCore.PixelFormats [strong]

Pixel formats supported by the library.

Enumerator

RGB	24bpp RGB format.
RGBA	32bpp RGBA format.
BGR	24bpp BGR format.
BGRA	32bpp BGRA format.

Definition at line 220 of file MuPDF.cs.

5.3.1.5 RasterOutputFileTypes

enum MuPDFCore.RasterOutputFileTypes [strong]

Raster image file types supported in output by the library.

Enumerator

PNM	Portable aNyMap graphics format.
PAM	Portable Arbitrary Map graphics format.
PNG	Portable Network Graphics format.
PSD	PhotoShop Document format.

Definition at line 173 of file MuPDF.cs.

5.4 MuPDFCore.MuPDFRenderer Namespace Reference

Classes

- class PDFRenderer

  A control to render PDF documents (and other formats), potentially using multiple threads.

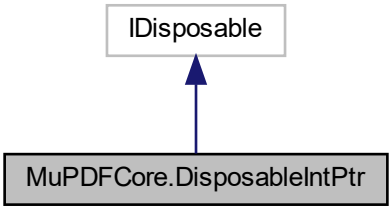
Chapter 6

Class Documentation

6.1 MuPDFCore.DisposableIntPtr Class Reference

An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.

Inheritance diagram for MuPDFCore.DisposableIntPtr:



Public Member Functions

- DisposableIntPtr (IntPtr pointer)

  Create a new DisposableIntPtr.
- void **Dispose** ()

6.1.1 Detailed Description

An IDisposable wrapper around an IntPtr that frees the allocated memory when it is disposed.

Definition at line 297 of file MuPDF.cs.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 DisposableIntPtr()

```
MuPDFCore.DisposableIntPtr.DisposableIntPtr (
            IntPtr pointer )
```

Create a new DisposableIntPtr.

Parameters

pointer	The pointer that should be freed upon disposing of this object.

Definition at line 308 of file MuPDF.cs.

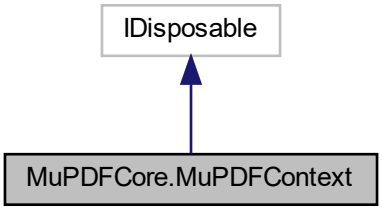
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDF.cs

6.2 MuPDFCore.MuPDFContext Class Reference

A wrapper around a MuPDF context object, which contains the exception stack and the resource cache store.

Inheritance diagram for MuPDFCore.MuPDFContext:



Public Member Functions

- MuPDFContext (long storeSize=256<< 20)

  Create a new MuPDFContext instance with the specified cache store size.
- void ClearStore ()

  Evict all items from the resource cache store (freeing the memory where they were held).
- void ShrinkStore (double fraction)

  Evict items from the resource cache store (freeing the memory where they were held) until the the size of the store drops to the specified fraction of the current size.
- void **Dispose** ()

Properties

- long StoreSize [get]

  The current size in bytes of the resource cache store. Read-only.
- long StoreMaxSize [get]

  The maximum size in bytes of the resource cache store. Read-only.

6.2.1 Detailed Description

A wrapper around a MuPDF context object, which contains the exception stack and the resource cache store.

Definition at line 25 of file MuPDFContext.cs.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 MuPDFContext()

```
MuPDFCore.MuPDFContext.MuPDFContext (
            long storeSize = 256 << 20 )
```

Create a new MuPDFContext instance with the specified cache store size.

Parameters

storeSize	The maximum size in bytes of the resource cache store. The default value is 256 MiB.

Definition at line 58 of file MuPDFContext.cs.

6.2.3 Member Function Documentation

6.2.3.1 ClearStore()

```
void MuPDFCore.MuPDFContext.ClearStore ( )
```

Evict all items from the resource cache store (freeing the memory where they were held).

Definition at line 87 of file MuPDFContext.cs.

6.2.3.2 ShrinkStore()

```
void MuPDFCore.MuPDFContext.ShrinkStore (
            double fraction )
```

Evict items from the resource cache store (freeing the memory where they were held) until the the size of the store drops to the specified fraction of the current size.

Parameters

fraction	The fraction of the current size that constitutes the target size of the store. If this is <= 0, the cache is cleared. If this is >= 1, nothing happens.

Definition at line 96 of file MuPDFContext.cs.

6.2.4 Property Documentation

6.2.4.1 StoreMaxSize

```
long MuPDFCore.MuPDFContext.StoreMaxSize  [get]
```

The maximum size in bytes of the resource cache store. Read-only.

Definition at line 46 of file MuPDFContext.cs.

6.2.4.2 StoreSize

```
long MuPDFCore.MuPDFContext.StoreSize  [get]
```

The current size in bytes of the resource cache store. Read-only.

Definition at line 35 of file MuPDFContext.cs.

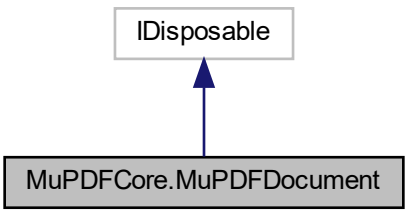
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFContext.cs

6.3 MuPDFCore.MuPDFDocument Class Reference

A wrapper over a MuPDF document object, which contains possibly multiple pages.

Inheritance diagram for MuPDFCore.MuPDFDocument:



Public Member Functions

- MuPDFDocument (MuPDFContext context, IntPtr dataAddress, long dataLength, InputFileTypes fileType)

  Create a new MuPDFDocument from data bytes accessible through the specified pointer.
- MuPDFDocument (MuPDFContext context, IntPtr dataAddress, long dataLength, InputFileTypes fileType, ref IDisposable dataHolder)

  Create a new MuPDFDocument from data bytes accessible through the specified pointer.
- MuPDFDocument (MuPDFContext context, byte[ ] data, InputFileTypes fileType)

  Create a new MuPDFDocument from an array of bytes.
- MuPDFDocument (MuPDFContext context, ref MemoryStream data, InputFileTypes fileType)

  Create a new MuPDFDocument from a MemoryStream.
- MuPDFDocument (MuPDFContext context, string fileName)

  Create a new MuPDFDocument from a file.
- void ClearCache ()

  Discard all the display lists that have been loaded from the document, possibly freeing some memory in the case of a huge document.
- byte[ ] Render (int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, bool includeAnnotations=true)

  Render (part of) a page to an array of bytes.
- byte[ ] Render (int pageNumber, double zoom, PixelFormats pixelFormat, bool includeAnnotations=true)

  Render a page to an array of bytes.
- void Render (int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, IntPtr destination, bool includeAnnotations=true)

  Render (part of) a page to the specified destination.
- void Render (int pageNumber, double zoom, PixelFormats pixelFormat, IntPtr destination, bool includeAnnotations=true)

  Render a page the specified destination.
- MuPDFMultiThreadedPageRenderer GetMultiThreadedRenderer (int pageNumber, int threadCount, bool includeAnnotations=true)

  Create a new MuPDFMultiThreadedPageRenderer that renders the specified page with the specified number of threads.
- int GetRenderedSize (int pageNumber, double zoom, PixelFormats pixelFormat)

  Determine how many bytes will be necessary to render the specified page at the specified zoom level, using the the specified pixel format.
- void SaveImage (int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, string fileName, RasterOutputFileTypes fileType, bool includeAnnotations=true)

  Save (part of) a page to an image file in the specified format.
- void SaveImage (int pageNumber, double zoom, PixelFormats pixelFormat, string fileName, RasterOutputFileTypes fileType, bool includeAnnotations=true)

  Save a page to an image file in the specified format.
- void WriteImage (int pageNumber, Rectangle region, double zoom, PixelFormats pixelFormat, Stream outputStream, RasterOutputFileTypes fileType, bool includeAnnotations=true)

  Write (part of) a page to an image stream in the specified format.
- void WriteImage (int pageNumber, double zoom, PixelFormats pixelFormat, Stream outputStream, RasterOutputFileTypes fileType, bool includeAnnotations=true)

  Write a page to an image stream in the specified format.
- void **Dispose** ()

Static Public Member Functions

- static int GetRenderedSize (Rectangle region, double zoom, PixelFormats pixelFormat)

  Determine how many bytes will be necessary to render the specified region in page units at the specified zoom level, using the the specified pixel format.
- static void CreateDocument (MuPDFContext context, string fileName, DocumentOutputFileTypes fileType, bool includeAnnotations=true, params(MuPDFPage page, Rectangle region, float zoom)[ ] pages)

  Create a new document containing the specified (parts of) pages from other documents.
- static void CreateDocument (MuPDFContext context, string fileName, DocumentOutputFileTypes fileType, bool includeAnnotations=true, params MuPDFPage[ ] pages)

  Create a new document containing the specified pages from other documents.

Properties

- MuPDFPageCollection Pages `[get]`

  The pages contained in the document.
- bool ClipToPageBounds = true `[get, set]`

  Defines whether the images resulting from rendering operations should be clipped to the page boundaries.

6.3.1 Detailed Description

A wrapper over a MuPDF document object, which contains possibly multiple pages.

Definition at line 27 of file MuPDFDocument.cs.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 MuPDFDocument() [1/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
            MuPDFContext context,
            IntPtr dataAddress,
            long dataLength,
            InputFileTypes fileType )
```

Create a new MuPDFDocument from data bytes accessible through the specified pointer.

Parameters

context	The context that will own this document.
dataAddress	A pointer to the data bytes that make up the document.
dataLength	The number of bytes to read from the specified address.
fileType	The type of the document to read.

Definition at line 105 of file MuPDFDocument.cs.

6.3.2.2 MuPDFDocument() [2/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
            MuPDFContext context,
            IntPtr dataAddress,
            long dataLength,
            InputFileTypes fileType,
            ref IDisposable dataHolder )
```

Create a new MuPDFDocument from data bytes accessible through the specified pointer.

Parameters

context	The context that will own this document.
dataAddress	A pointer to the data bytes that make up the document.
dataLength	The number of bytes to read from the specified address.
fileType	The type of the document to read.
dataHolder	An IDisposable that will be disposed when the MuPDFDocument is disposed.

Definition at line 115 of file MuPDFDocument.cs.

6.3.2.3 MuPDFDocument() [3/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
            MuPDFContext context,
            byte[] data,
            InputFileTypes fileType )
```

Create a new MuPDFDocument from an array of bytes.

Parameters

context	The context that will own this document.
data	An array containing the data bytes that make up the document. This must not be altered until after the MuPDFDocument has been disposed! The address of the array will be pinned, which may cause degradation in the Garbage Collector's performance, and is thus only advised for short-lived documents. To avoid this issue, marshal the bytes to unmanaged memory and use one of the IntPtr constructors.
fileType	The type of the document to read.

Definition at line 148 of file MuPDFDocument.cs.

6.3.2.4 MuPDFDocument() [4/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
```

```
            MuPDFContext context,
            ref MemoryStream data,
            InputFileTypes fileType )
```

Create a new MuPDFDocument from a MemoryStream.

Parameters

context	The context that will own this document.
data	The MemoryStream containing the data that makes up the document. This will be disposed when the MuPDFDocument has been disposed and must not be disposed externally! The address of the MemoryStream's buffer will be pinned, which may cause degradation in the Garbage Collector's performance, and is thus only advised for short-lived documents. To avoid this issue, marshal the bytes to unmanaged memory and use one of the IntPtr constructors.
fileType	The type of the document to read.

Definition at line 183 of file MuPDFDocument.cs.

6.3.2.5 MuPDFDocument() [5/5]

```
MuPDFCore.MuPDFDocument.MuPDFDocument (
            MuPDFContext context,
            string fileName )
```

Create a new MuPDFDocument from a file.

Parameters

context	The context that will own this document.
fileName	The path to the file to open.

Definition at line 221 of file MuPDFDocument.cs.

6.3.3 Member Function Documentation

6.3.3.1 ClearCache()

```
void MuPDFCore.MuPDFDocument.ClearCache ( )
```

Discard all the display lists that have been loaded from the document, possibly freeing some memory in the case of a huge document.

Definition at line 246 of file MuPDFDocument.cs.

6.3.3.2 CreateDocument() [1/2]

```
static void MuPDFCore.MuPDFDocument.CreateDocument (
            MuPDFContext context,
            string fileName,
            DocumentOutputFileTypes fileType,
            bool includeAnnotations = true,
            params MuPDFPage[] pages )  [static]
```

Create a new document containing the specified pages from other documents.

Parameters

context	The context that was used to open the documents.
fileName	The output file name.
fileType	The output file format.
pages	The pages to include in the document.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 655 of file MuPDFDocument.cs.

6.3.3.3 CreateDocument() [2/2]

```
static void MuPDFCore.MuPDFDocument.CreateDocument (
            MuPDFContext context,
            string fileName,
            DocumentOutputFileTypes fileType,
            bool includeAnnotations = true,
            params(MuPDFPage page, Rectangle region, float zoom)[] pages )  [static]
```

Create a new document containing the specified (parts of) pages from other documents.

Parameters

context	The context that was used to open the documents.
fileName	The output file name.
fileType	The output file format.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.
pages	The pages to include in the document. The "page" element specifies the page, the "region" element the area of the page that should be included in the document, and the "zoom" element how much the region should be scaled.

Definition at line 562 of file MuPDFDocument.cs.

6.3.3.4 GetMultiThreadedRenderer()

```
MuPDFMultiThreadedPageRenderer MuPDFCore.MuPDFDocument.GetMultiThreadedRenderer (
            int pageNumber,
            int threadCount,
            bool includeAnnotations = true )
```

Create a new MuPDFMultiThreadedPageRenderer that renders the specified page with the specified number of threads.

Parameters

pageNumber	The number of the page to render (starting at 0).
threadCount	The number of threads to use. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than *threadCount* that satisfies this condition is used.

Returns

A MuPDFMultiThreadedPageRenderer that can be used to render the specified page with the specified number of threads.

Parameters

includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 362 of file MuPDFDocument.cs.

6.3.3.5 GetRenderedSize() [1/2]

```
int MuPDFCore.MuPDFDocument.GetRenderedSize (
            int pageNumber,
            double zoom,
            PixelFormats pixelFormat )
```

Determine how many bytes will be necessary to render the specified page at the specified zoom level, using the the specified pixel format.

Parameters

pageNumber	The number of the page to render (starting at 0).
zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixels data.

Returns

An integer representing the number of bytes that will be necessary to store the pixel data of the rendered image.

Definition at line 379 of file MuPDFDocument.cs.

6.3.3.6 GetRenderedSize() [2/2]

```
static int MuPDFCore.MuPDFDocument.GetRenderedSize (
            Rectangle region,
            double zoom,
            PixelFormats pixelFormat )  [static]
```

Determine how many bytes will be necessary to render the specified region in page units at the specified zoom level, using the the specified pixel format.

Parameters

region	The region that will be rendered.
zoom	The scale at which the region will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixels data.

Returns

An integer representing the number of bytes that will be necessary to store the pixel data of the rendered image.

Definition at line 391 of file MuPDFDocument.cs.

6.3.3.7 Render() [1/4]

```
byte [] MuPDFCore.MuPDFDocument.Render (
            int pageNumber,
            double zoom,
            PixelFormats pixelFormat,
            bool includeAnnotations = true )
```

Render a page to an array of bytes.

Parameters

pageNumber	The number of the page to render (starting at 0).
zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixel data.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Returns

A byte array containing the raw values for the pixels of the rendered image.

Definition at line 293 of file MuPDFDocument.cs.

6.3.3.8 Render() [2/4]

```
void MuPDFCore.MuPDFDocument.Render (
            int pageNumber,
            double zoom,
            PixelFormats pixelFormat,
            IntPtr destination,
            bool includeAnnotations = true )
```

Render a page the specified destination.

Parameters

pageNumber	The number of the page to render (starting at 0).
zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixel data.
destination	The address of the buffer where the pixel data will be written. There must be enough space available to write the values for all the pixels, otherwise this will fail catastrophically!
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 349 of file MuPDFDocument.cs.

6.3.3.9 Render() [3/4]

```
byte [] MuPDFCore.MuPDFDocument.Render (
            int pageNumber,
            Rectangle region,
            double zoom,
            PixelFormats pixelFormat,
            bool includeAnnotations = true )
```

Render (part of) a page to an array of bytes.

Parameters

pageNumber	The number of the page to render (starting at 0).
region	The region of the page to render in page units.
zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixel data.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Returns

A byte array containing the raw values for the pixels of the rendered image.

Definition at line 264 of file MuPDFDocument.cs.

6.3.3.10 Render() [4/4]

```
void MuPDFCore.MuPDFDocument.Render (
            int pageNumber,
            Rectangle region,
            double zoom,
            PixelFormats pixelFormat,
            IntPtr destination,
            bool includeAnnotations = true )
```

Render (part of) a page to the specified destination.

Parameters

pageNumber	The number of the page to render (starting at 0).
region	The region of the page to render in page units.
zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixel data.
destination	The address of the buffer where the pixel data will be written. There must be enough space available to write the values for all the pixels, otherwise this will fail catastrophically!
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 308 of file MuPDFDocument.cs.

6.3.3.11 SaveImage() [1/2]

```
void MuPDFCore.MuPDFDocument.SaveImage (
            int pageNumber,
            double zoom,
            PixelFormats pixelFormat,
            string fileName,
            RasterOutputFileTypes fileType,
            bool includeAnnotations = true )
```

Save a page to an image file in the specified format.

Parameters

pageNumber	The number of the page to render (starting at 0).

Parameters

zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixel data.
fileName	The path to the output file.
fileType	The output format of the file.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 472 of file MuPDFDocument.cs.

6.3.3.12 SaveImage() [2/2]

```
void MuPDFCore.MuPDFDocument.SaveImage (
            int pageNumber,
            Rectangle region,
            double zoom,
            PixelFormats pixelFormat,
            string fileName,
            RasterOutputFileTypes fileType,
            bool includeAnnotations = true )
```

Save (part of) a page to an image file in the specified format.

Parameters

pageNumber	The number of the page to render (starting at 0).
region	The region of the page to render in page units.
zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixel data.
fileName	The path to the output file.
fileType	The output format of the file.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 429 of file MuPDFDocument.cs.

6.3.3.13 WriteImage() [1/2]

```
void MuPDFCore.MuPDFDocument.WriteImage (
            int pageNumber,
            double zoom,
            PixelFormats pixelFormat,
```

```
            Stream outputStream,
            RasterOutputFileTypes fileType,
            bool includeAnnotations = true )
```

Write a page to an image stream in the specified format.

Parameters

pageNumber	The number of the page to render (starting at 0).
zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixel data.
outputStream	The stream to which the image data will be written.
fileType	The output format of the image.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 548 of file MuPDFDocument.cs.

6.3.3.14 WriteImage() [2/2]

```
void MuPDFCore.MuPDFDocument.WriteImage (
            int pageNumber,
            Rectangle region,
            double zoom,
            PixelFormats pixelFormat,
            Stream outputStream,
            RasterOutputFileTypes fileType,
            bool includeAnnotations = true )
```

Write (part of) a page to an image stream in the specified format.

Parameters

pageNumber	The number of the page to render (starting at 0).
region	The region of the page to render in page units.
zoom	The scale at which the page will be rendered. This will determine the size in pixel of the image.
pixelFormat	The format of the pixel data.
outputStream	The stream to which the image data will be written.
fileType	The output format of the image.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the display list that is generated. Otherwise, only the page contents are included.

Definition at line 488 of file MuPDFDocument.cs.

6.3.4 Property Documentation

6.3.4.1 ClipToPageBounds

```
bool MuPDFCore.MuPDFDocument.ClipToPageBounds = true  [get], [set]
```

Defines whether the images resulting from rendering operations should be clipped to the page boundaries.

Definition at line 96 of file MuPDFDocument.cs.

6.3.4.2 Pages

```
MuPDFPageCollection MuPDFCore.MuPDFDocument.Pages  [get]
```

The pages contained in the document.

Definition at line 91 of file MuPDFDocument.cs.

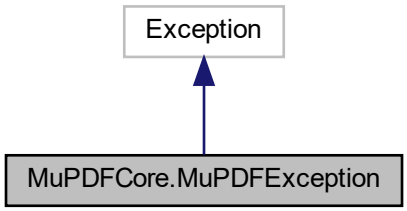
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFDocument.cs

6.4 MuPDFCore.MuPDFException Class Reference

The exception that is thrown when a MuPDF operation fails.

Inheritance diagram for MuPDFCore.MuPDFException:



Public Attributes

- readonly ExitCodes ErrorCode

  The ExitCodes returned by the native function.

6.4.1 Detailed Description

The exception that is thrown when a MuPDF operation fails.

Definition at line 342 of file MuPDF.cs.

6.4.2 Member Data Documentation

6.4.2.1 ErrorCode

```
readonly ExitCodes MuPDFCore.MuPDFException.ErrorCode
```

The ExitCodes returned by the native function.

Definition at line 347 of file MuPDF.cs.

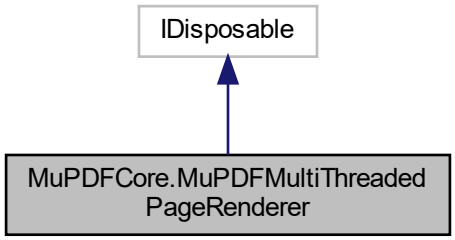
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDF.cs

6.5 MuPDFCore.MuPDFMultiThreadedPageRenderer Class Reference

A class that holds the necessary resources to render a page of a MuPDF document using multiple threads.

Inheritance diagram for MuPDFCore.MuPDFMultiThreadedPageRenderer:



Public Member Functions

- void Render (RoundedSize targetSize, Rectangle region, IntPtr[ ] destinations, PixelFormats pixelFormat)

  Render the specified region to an image of the specified size, split in a number of tiles equal to the number of threads used by this MuPDFMultiThreadedPageRenderer, without marshaling. This method will not return until all the rendering threads have finished.
- void Abort ()

  Signal to the rendering threads that they should abort rendering as soon as possible.
- RenderProgress GetProgress ()

  Get the current rendering progress of all the threads.
- void **Dispose** ()

Properties

- int ThreadCount `[get]`

  The number of threads that are used to render the image.

6.5.1 Detailed Description

A class that holds the necessary resources to render a page of a MuPDF document using multiple threads.

Definition at line 268 of file MuPDFMultiThreadedPageRenderer.cs.

6.5.2 Member Function Documentation

6.5.2.1 Abort()

```
void MuPDFCore.MuPDFMultiThreadedPageRenderer.Abort ( )
```

Signal to the rendering threads that they should abort rendering as soon as possible.

Definition at line 465 of file MuPDFMultiThreadedPageRenderer.cs.

6.5.2.2 GetProgress()

```
RenderProgress MuPDFCore.MuPDFMultiThreadedPageRenderer.GetProgress ( )
```

Get the current rendering progress of all the threads.

Returns

A RenderProgress object containing the rendering progress of all the threads.

Definition at line 477 of file MuPDFMultiThreadedPageRenderer.cs.

6.5.2.3 Render()

```
void MuPDFCore.MuPDFMultiThreadedPageRenderer.Render (
            RoundedSize targetSize,
            Rectangle region,
            IntPtr[] destinations,
            PixelFormats pixelFormat )
```

Render the specified region to an image of the specified size, split in a number of tiles equal to the number of threads used by this MuPDFMultiThreadedPageRenderer, without marshaling. This method will not return until all the rendering threads have finished.

Parameters

targetSize	The total size of the image that should be rendered.
region	The region in page units that should be rendered.
destinations	An array containing the addresses of the buffers where the rendered tiles will be written. There must be enough space available in each buffer to write the values for all the pixels of the tile, otherwise this will fail catastrophically! As long as the *targetSize* is the same, the size in pixel of the tiles is guaranteed to also be the same.
pixelFormat	The format of the pixel data.

Definition at line 360 of file MuPDFMultiThreadedPageRenderer.cs.

6.5.3 Property Documentation

6.5.3.1 ThreadCount

```
int MuPDFCore.MuPDFMultiThreadedPageRenderer.ThreadCount  [get]
```

The number of threads that are used to render the image.

Definition at line 298 of file MuPDFMultiThreadedPageRenderer.cs.

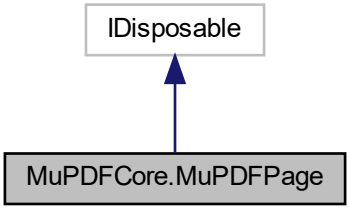
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFMultiThreadedPageRenderer.cs

6.6 MuPDFCore.MuPDFPage Class Reference

A wrapper over a MuPDF page object, which contains information about the page's boundaries.

Inheritance diagram for MuPDFCore.MuPDFPage:



Public Member Functions

- void **Dispose** ()

Properties

- Rectangle Bounds `[get]`

  The page's bounds in page units. Read-only.
- int PageNumber `[get]`

  The number of this page in the original document.

6.6.1 Detailed Description

A wrapper over a MuPDF page object, which contains information about the page's boundaries.

Definition at line 27 of file MuPDFPage.cs.

6.6.2 Property Documentation

6.6.2.1 Bounds

```
Rectangle MuPDFCore.MuPDFPage.Bounds  [get]
```

The page's bounds in page units. Read-only.

Definition at line 32 of file MuPDFPage.cs.

6.6.2.2 PageNumber

```
int MuPDFCore.MuPDFPage.PageNumber  [get]
```

The number of this page in the original document.

Definition at line 37 of file MuPDFPage.cs.

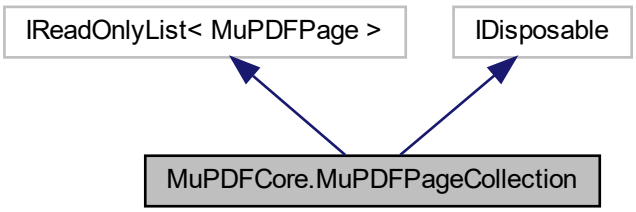
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFPage.cs

6.7 MuPDFCore.MuPDFPageCollection Class Reference

A lazy collection of MuPDFPages. Each page is loaded from the document as it is requested for the first time.

Inheritance diagram for MuPDFCore.MuPDFPageCollection:



Public Member Functions

- IEnumerator< MuPDFPage > GetEnumerator ()
  inheritdoc/>
- void **Dispose** ()

Properties

- int Length [get]
  The number of pages in the collection.
- int Count [get]
  The number of pages in the collection.
- MuPDFPage this[int index] [get]
  Get a page from the collection.

6.7.1 Detailed Description

A lazy collection of MuPDFPages. Each page is loaded from the document as it is requested for the first time.

Definition at line 119 of file MuPDFPage.cs.

6.7.2 Property Documentation

6.7.2.1 Count

```
int MuPDFCore.MuPDFPageCollection.Count  [get]
```

The number of pages in the collection.

Definition at line 144 of file MuPDFPage.cs.

6.7.2.2 Length

```
int MuPDFCore.MuPDFPageCollection.Length  [get]
```

The number of pages in the collection.

Definition at line 139 of file MuPDFPage.cs.

6.7.2.3 this[int index]

```
MuPDFPage MuPDFCore.MuPDFPageCollection.this[int index]  [get]
```

Get a page from the collection.

Parameters

index	The number of the page (starting at 0).

Returns

The specified MuPDFPage.

Definition at line 151 of file MuPDFPage.cs.

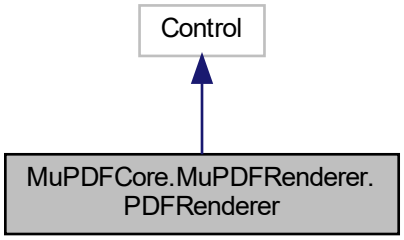
The documentation for this class was generated from the following file:

- MuPDFCore/MuPDFPage.cs

6.8 MuPDFCore.MuPDFRenderer.PDFRenderer Class Reference

A control to render PDF documents (and other formats), potentally using multiple threads.

Inheritance diagram for MuPDFCore.MuPDFRenderer.PDFRenderer:



Public Member Functions

- PDFRenderer ()

  Initializes a new instance of the PDFRenderer class.
- void Initialize (MuPDFDocument document, int threadCount=0, int pageNumber=0, double resolution↩ Multiplier=1, bool includeAnnotations=true)

  Set up the PDFRenderer to display a page of a MuPDFDocument.
- void Initialize (string fileName, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true)

  Set up the PDFRenderer to display a page of a document that will be loaded from disk.
- void Initialize (MemoryStream ms, InputFileTypes fileType, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true)

  Set up the PDFRenderer to display a page of a document that will be loaded from a MemoryStream.
- void Initialize (byte[ ] dataBytes, InputFileTypes fileType, int offset=0, int length=-1, int threadCount=0, int pageNumber=0, double resolutionMultiplier=1, bool includeAnnotations=true)

  Set up the PDFRenderer to display a page of a document that will be loaded from an array of bytes.
- void ReleaseResources ()

  Release resources held by this PDFRenderer. This is not an irreversible step: using one of the Initialize overloads after calling this method will restore functionality.
- void SetDisplayAreaNow (Rect value)

  Set the current display area to the specified value , skipping all transitions.
- void ZoomStep (double count, Point? center=null)

  Zoom around a point.
- void Contain ()

  Alter the display area so that the whole page fits on screen.
- void Cover ()

  Alter the display area so that the page covers the whole surface of the PDFRenderer (even though parts of the page may be outside it).
- RenderProgress GetProgress ()

  Get the current rendering progress.
- override void Render (DrawingContext context)

  Draw the rendered document.

Static Public Attributes

- static readonly DirectProperty< PDFRenderer, int > RenderThreadCountProperty = AvaloniaProperty.↩ RegisterDirect<PDFRenderer, int>(nameof(RenderThreadCount), o => o.RenderThreadCount)

  Defines the RenderThreadCount property.
- static readonly DirectProperty< PDFRenderer, int > PageNumberProperty = AvaloniaProperty.Register↩ Direct<PDFRenderer, int>(nameof(PageNumber), o => o.PageNumber)

  Defines the PageNumber property.
- static readonly DirectProperty< PDFRenderer, bool > IsViewerInitializedProperty = AvaloniaProperty.↩ RegisterDirect<PDFRenderer, bool>(nameof(IsViewerInitialized), o => o.IsViewerInitialized)

  Defines the IsViewerInitialized property.
- static readonly DirectProperty< PDFRenderer, Rect > PageSizeProperty = AvaloniaProperty.Register↩ Direct<PDFRenderer, Rect>(nameof(PageSize), o => o.PageSize)

  Defines the PageSize property.
- static readonly StyledProperty< Rect > DisplayAreaProperty = AvaloniaProperty.Register<PDFRenderer, Rect>(nameof(DisplayArea))

  Defines the DisplayArea property.
- static readonly StyledProperty< double > ZoomIncrementProperty = AvaloniaProperty.Register<PDFRenderer, double>(nameof(ZoomIncrement), Math.Pow(2, 1.0 / 3.0), defaultBindingMode: Avalonia.Data.Binding↩ Mode.TwoWay)

  Defines the ZoomIncrement property.
- static readonly StyledProperty< IBrush > BackgroundProperty = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(Background))

  Defines the Background property.
- static readonly StyledProperty< IBrush > PageBackgroundProperty = AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(PageBackground))

  Defines the PageBackground property.
- static readonly DirectProperty< PDFRenderer, double > ZoomProperty = AvaloniaProperty.Register↩ Direct<PDFRenderer, double>(nameof(Zoom), o => o.Zoom, (o, v) => o.Zoom = v, defaultBindingMode: Avalonia.Data.BindingMode.TwoWay)

  Defines the Zoom property.
- static readonly StyledProperty< bool > PanEnabledProperty = AvaloniaProperty.Register<PDFRenderer, bool>(nameof(PanEnabled), true)

  Defines the PanEnabled property.
- static readonly StyledProperty< bool > ZoomEnabledProperty = AvaloniaProperty.Register<PDFRenderer, bool>(nameof(ZoomEnabled), true)

  Defines the ZoomEnabled property.

Properties

- int RenderThreadCount `[get]`

  Exposes the number of threads that the current instance is using to render the document. Read-only.
- int PageNumber `[get]`

  Exposes the number of the page that the current instance is rendering. Read-only.
- bool IsViewerInitialized `[get]`

  Whether the current instance has been initialised with a document to render or not. Read-only.
- Rect PageSize `[get]`

  Exposes the size of the page that is drawn by the current instance (in page units).
- Rect DisplayArea `[get, set]`

  The region of the page (in page units) that is currently displayed by the current instance. This always has the same aspect ratio of the bounds of this control. When this is set, the value is sanitised so that the smallest rectangle with the correct aspect ratio containing the requested value is chosen.

- double ZoomIncrement `[get, set]`

  Determines by how much the scale will be increased/decreased by the ZoomStep(double, Point?) method. Set this to a value smaller than 1 to invert the zoom in/out direction.
- IBrush Background `[get, set]`

  The background colour of the control.
- IBrush PageBackground `[get, set]`

  The background colour to use for the page drawn by the control.
- double??? Zoom `[get, set]`

  The current zoom level. Setting this will change the DisplayArea appropriately, zooming around the center of the DisplayArea.
- bool PanEnabled `[get, set]`

  Whether the default handlers for pointer events (which are used for panning around the page) should be enabled. If this is false, you will have to implement your own way to pan around the document by changing the DisplayArea.
- bool ZoomEnabled `[get, set]`

  Whether the default handlers for pointer wheel events (which are used for zooming in/out) should be enabled. If this is false, you will have to implement your own way to zoom by changing the DisplayArea.

6.8.1 Detailed Description

A control to render PDF documents (and other formats), potentally using multiple threads.

Definition at line 38 of file PDFRenderer.cs.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 PDFRenderer()

```
MuPDFCore.MuPDFRenderer.PDFRenderer.PDFRenderer ( )
```

Initializes a new instance of the PDFRenderer class.

Definition at line 164 of file PDFRenderer.cs.

6.8.3 Member Function Documentation

6.8.3.1 Contain()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Contain ( )
```

Alter the display area so that the whole page fits on screen.

Definition at line 459 of file PDFRenderer.cs.

6.8.3.2 Cover()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Cover ( )
```

Alter the display area so that the page covers the whole surface of the PDFRenderer (even though parts of the page may be outside it).

Definition at line 468 of file PDFRenderer.cs.

6.8.3.3 GetProgress()

```
RenderProgress MuPDFCore.MuPDFRenderer.PDFRenderer.GetProgress ( )
```

Get the current rendering progress.

Returns

A RenderProgress object with information about the rendering progress of each thread.

Definition at line 489 of file PDFRenderer.cs.

6.8.3.4 Initialize() [1/4]

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Initialize (
            byte[] dataBytes,
            InputFileTypes fileType,
            int offset = 0,
            int length = -1,
            int threadCount = 0,
            int pageNumber = 0,
            double resolutionMultiplier = 1,
            bool includeAnnotations = true )
```

Set up the PDFRenderer to display a page of a document that will be loaded from an array of bytes.

Parameters

dataBytes	The bytes of the document that should be opened. The array will be copied and can be safely discarded/altered after this method returns.
fileType	The format of the document.
offset	The offset in the byte array at which the document starts.
length	The length of the document in bytes. If this is < 0, the whole array is used.
threadCount	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than *threadCount* that satisfies this condition is used.
pageNumber	The index of the page that should be rendered. The first page has index 0.
resolutionMultiplier	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If *resolutionMultiplier* is 1, the resolution will match the size (in screen units) of the PDFRenderer.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.

Definition at line 286 of file PDFRenderer.cs.

6.8.3.5 Initialize() [2/4]

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Initialize (
            MemoryStream ms,
            InputFileTypes fileType,
            int threadCount = 0,
            int pageNumber = 0,
            double resolutionMultiplier = 1,
            bool includeAnnotations = true )
```

Set up the PDFRenderer to display a page of a document that will be loaded from a MemoryStream.

Parameters

ms	The MemoryStream containing the document that should be opened. This can be safely disposed after this method returns.
fileType	The format of the document.
threadCount	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than *threadCount* that satisfies this condition is used.
pageNumber	The index of the page that should be rendered. The first page has index 0.
resolutionMultiplier	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If *resolutionMultiplier* is 1, the resolution will match the size (in screen units) of the PDFRenderer.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.

Definition at line 265 of file PDFRenderer.cs.

6.8.3.6 Initialize() [3/4]

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Initialize (
            MuPDFDocument document,
            int threadCount = 0,
            int pageNumber = 0,
            double resolutionMultiplier = 1,
            bool includeAnnotations = true )
```

Set up the PDFRenderer to display a page of a MuPDFDocument.

Parameters

document	The MuPDFDocument to render.
threadCount	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than *threadCount* that satisfies this condition is used.

Parameters

pageNumber	The index of the page that should be rendered. The first page has index 0.
resolutionMultiplier	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If *resolutionMultiplier* is 1, the resolution will match the size (in screen units) of the PDFRenderer.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.

Definition at line 218 of file PDFRenderer.cs.

6.8.3.7 Initialize() [4/4]

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.Initialize (
            string fileName,
            int threadCount = 0,
            int pageNumber = 0,
            double resolutionMultiplier = 1,
            bool includeAnnotations = true )
```

Set up the PDFRenderer to display a page of a document that will be loaded from disk.

Parameters

fileName	The path to the document that should be opened.
threadCount	The number of threads to use in the rendering. If this is 0, an appropriate number of threads based on the number of processors in the computer will be used. Otherwise, this must be factorisable using only powers of 2, 3, 5 or 7. If this is not the case, the biggest number smaller than *threadCount* that satisfies this condition is used.
pageNumber	The index of the page that should be rendered. The first page has index 0.
resolutionMultiplier	This value can be used to increase or decrease the resolution at which the static renderisation of the page will be produced. If *resolutionMultiplier* is 1, the resolution will match the size (in screen units) of the PDFRenderer.
includeAnnotations	If this is `true`, annotations (e.g. signatures) are included in the rendering. Otherwise, only the page contents are included.

Definition at line 241 of file PDFRenderer.cs.

6.8.3.8 ReleaseResources()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.ReleaseResources ( )
```

Release resources held by this PDFRenderer. This is not an irreversible step: using one of the Initialize overloads after calling this method will restore functionality.

Definition at line 383 of file PDFRenderer.cs.

6.8.3.9 Render()

```
override void MuPDFCore.MuPDFRenderer.PDFRenderer.Render (
            DrawingContext context )
```

Draw the rendered document.

Parameters

context	The drawing context on which to draw.

Definition at line 871 of file PDFRenderer.cs.

6.8.3.10 SetDisplayAreaNow()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.SetDisplayAreaNow (
            Rect value )
```

Set the current display area to the specified *value* , skipping all transitions.

Parameters

value	The new display area.

Definition at line 421 of file PDFRenderer.cs.

6.8.3.11 ZoomStep()

```
void MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomStep (
            double count,
            Point?  center = null )
```

Zoom around a point.

Parameters

count	Number of steps to zoom. Positive values indicate a zoom in, negative values a zoom out.
center	The point around which to center the zoom operation. If this is null, the center of the control is used.

Definition at line 434 of file PDFRenderer.cs.

6.8.4 Member Data Documentation

6.8.4.1 BackgroundProperty

```
readonly StyledProperty<IBrush> MuPDFCore.MuPDFRenderer.PDFRenderer.BackgroundProperty =
AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(Background))  [static]
```

Defines the Background property.

Definition at line 181 of file PDFRenderer.Properties.cs.

6.8.4.2 DisplayAreaProperty

```
readonly StyledProperty<Rect> MuPDFCore.MuPDFRenderer.PDFRenderer.DisplayAreaProperty = Avalonia↩
Property.Register<PDFRenderer, Rect>(nameof(DisplayArea))  [static]
```

Defines the DisplayArea property.

Definition at line 127 of file PDFRenderer.Properties.cs.

6.8.4.3 IsViewerInitializedProperty

```
readonly DirectProperty<PDFRenderer, bool> MuPDFCore.MuPDFRenderer.PDFRenderer.IsViewer↩
InitializedProperty = AvaloniaProperty.RegisterDirect<PDFRenderer, bool>(nameof(IsViewerInitialized),
o => o.IsViewerInitialized)  [static]
```

Defines the IsViewerInitialized property.

Definition at line 79 of file PDFRenderer.Properties.cs.

6.8.4.4 PageBackgroundProperty

```
readonly StyledProperty<IBrush> MuPDFCore.MuPDFRenderer.PDFRenderer.PageBackgroundProperty =
AvaloniaProperty.Register<PDFRenderer, IBrush>(nameof(PageBackground))  [static]
```

Defines the PageBackground property.

Definition at line 194 of file PDFRenderer.Properties.cs.

6.8.4.5 PageNumberProperty

```
readonly DirectProperty<PDFRenderer, int> MuPDFCore.MuPDFRenderer.PDFRenderer.PageNumber↩
Property = AvaloniaProperty.RegisterDirect<PDFRenderer, int>(nameof(PageNumber), o => o.↩
PageNumber)  [static]
```

Defines the PageNumber property.

Definition at line 55 of file PDFRenderer.Properties.cs.

6.8.4.6 PageSizeProperty

```
readonly DirectProperty<PDFRenderer, Rect> MuPDFCore.MuPDFRenderer.PDFRenderer.PageSize↩
Property = AvaloniaProperty.RegisterDirect<PDFRenderer, Rect>(nameof(PageSize), o => o.Page↩
Size)  [static]
```

Defines the PageSize property.

Definition at line 103 of file PDFRenderer.Properties.cs.

6.8.4.7 PanEnabledProperty

```
readonly StyledProperty<bool> MuPDFCore.MuPDFRenderer.PDFRenderer.PanEnabledProperty = Avalonia↩
Property.Register<PDFRenderer, bool>(nameof(PanEnabled), true)  [static]
```

Defines the PanEnabled property.

Definition at line 246 of file PDFRenderer.Properties.cs.

6.8.4.8 RenderThreadCountProperty

```
readonly DirectProperty<PDFRenderer, int> MuPDFCore.MuPDFRenderer.PDFRenderer.RenderThread↩
CountProperty = AvaloniaProperty.RegisterDirect<PDFRenderer, int>(nameof(RenderThreadCount), o
=> o.RenderThreadCount)  [static]
```

Defines the RenderThreadCount property.

Definition at line 31 of file PDFRenderer.Properties.cs.

6.8.4.9 ZoomEnabledProperty

```
readonly StyledProperty<bool> MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomEnabledProperty = Avalonia↩
Property.Register<PDFRenderer, bool>(nameof(ZoomEnabled), true)  [static]
```

Defines the ZoomEnabled property.

Definition at line 259 of file PDFRenderer.Properties.cs.

6.8.4.10 ZoomIncrementProperty

```
readonly StyledProperty<double> MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomIncrementProperty =
AvaloniaProperty.Register<PDFRenderer, double>(nameof(ZoomIncrement), Math.Pow(2, 1.0 / 3.0),
defaultBindingMode:  Avalonia.Data.BindingMode.TwoWay)  [static]
```

Defines the ZoomIncrement property.

Definition at line 159 of file PDFRenderer.Properties.cs.

6.8.4.11 ZoomProperty

```
readonly DirectProperty<PDFRenderer, double> MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomProperty
= AvaloniaProperty.RegisterDirect<PDFRenderer, double>(nameof(Zoom), o => o.Zoom, (o, v) =>
o.Zoom = v, defaultBindingMode:  Avalonia.Data.BindingMode.TwoWay)  [static]
```

Defines the Zoom property.

Definition at line 207 of file PDFRenderer.Properties.cs.

6.8.5 Property Documentation

6.8.5.1 Background

```
IBrush MuPDFCore.MuPDFRenderer.PDFRenderer.Background  [get], [set]
```

The background colour of the control.

Definition at line 185 of file PDFRenderer.Properties.cs.

6.8.5.2 DisplayArea

```
Rect MuPDFCore.MuPDFRenderer.PDFRenderer.DisplayArea  [get], [set]
```

The region of the page (in page units) that is currently displayed by the current instance. This always has the same aspect ratio of the bounds of this control. When this is set, the value is sanitised so that the smallest rectangle with the correct aspect ratio containing the requested value is chosen.

Definition at line 132 of file PDFRenderer.Properties.cs.

6.8.5.3 IsViewerInitialized

```
bool MuPDFCore.MuPDFRenderer.PDFRenderer.IsViewerInitialized [get]
```

Whether the current instance has been initialised with a document to render or not. Read-only.

Definition at line 87 of file PDFRenderer.Properties.cs.

6.8.5.4 PageBackground

```
IBrush MuPDFCore.MuPDFRenderer.PDFRenderer.PageBackground [get], [set]
```

The background colour to use for the page drawn by the control.

Definition at line 198 of file PDFRenderer.Properties.cs.

6.8.5.5 PageNumber

```
int MuPDFCore.MuPDFRenderer.PDFRenderer.PageNumber [get]
```

Exposes the number of the page that the current instance is rendering. Read-only.

Definition at line 63 of file PDFRenderer.Properties.cs.

6.8.5.6 PageSize

```
Rect MuPDFCore.MuPDFRenderer.PDFRenderer.PageSize [get]
```

Exposes the size of the page that is drawn by the current instance (in page units).

Definition at line 111 of file PDFRenderer.Properties.cs.

6.8.5.7 PanEnabled

```
bool MuPDFCore.MuPDFRenderer.PDFRenderer.PanEnabled [get], [set]
```

Whether the default handlers for pointer events (which are used for panning around the page) should be enabled. If this is false, you will have to implement your own way to pan around the document by changing the DisplayArea.

Definition at line 250 of file PDFRenderer.Properties.cs.

6.8.5.8 RenderThreadCount

```
int MuPDFCore.MuPDFRenderer.PDFRenderer.RenderThreadCount  [get]
```

Exposes the number of threads that the current instance is using to render the document. Read-only.

Definition at line 39 of file PDFRenderer.Properties.cs.

6.8.5.9 Zoom

```
double???  MuPDFCore.MuPDFRenderer.PDFRenderer.Zoom  [get], [set]
```

The current zoom level. Setting this will change the DisplayArea appropriately, zooming around the center of the DisplayArea.

Definition at line 215 of file PDFRenderer.Properties.cs.

6.8.5.10 ZoomEnabled

```
bool MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomEnabled  [get], [set]
```

Whether the default handlers for pointer wheel events (which are used for zooming in/out) should be enabled. If this is false, you will have to implement your own way to zoom by changing the DisplayArea.

Definition at line 263 of file PDFRenderer.Properties.cs.

6.8.5.11 ZoomIncrement

```
double MuPDFCore.MuPDFRenderer.PDFRenderer.ZoomIncrement  [get], [set]
```

Determines by how much the scale will be increased/decreased by the ZoomStep(double, Point?) method. Set this to a value smaller than 1 to invert the zoom in/out direction.

Definition at line 163 of file PDFRenderer.Properties.cs.

The documentation for this class was generated from the following files:

- MuPDFCore.MuPDFRenderer/PDFRenderer.cs
- MuPDFCore.MuPDFRenderer/PDFRenderer.Properties.cs

6.9 MuPDFCore.Rectangle Struct Reference

Represents a rectangle.

Public Member Functions

- Rectangle (float x0, float y0, float x1, float y1)

  Create a new Rectangle from the specified coordinates.
- Rectangle (double x0, double y0, double x1, double y1)

  Create a new Rectangle from the specified coordinates.
- RoundedRectangle Round ()

  Round the rectangle's coordinates to the closest integers.
- RoundedRectangle Round (double zoom)

  Round the rectangle's coordinates to the closest integers, applying the specified zoom factor.
- Rectangle[ ] Split (int divisions)

  Split the rectangle into the specified number of Rectangles.
- Rectangle Intersect (Rectangle other)

  Compute the intersection between this Rectangle and another one.
- bool Contains (Rectangle other)

  Checks whether this Rectangle contains another Rectangle.

Public Attributes

- float X0

  The left coordinate of the rectangle.
- float Y0

  The top coordinate of the rectangle.
- float X1

  The right coordinate of the rectangle.
- float Y1

  The bottom coordinate of the rectangle.
- float Width => X1 - X0

  The width of the rectangle.
- float Height => Y1 - Y0

  The height of the rectangle.

6.9.1 Detailed Description

Represents a rectangle.

Definition at line 326 of file Rectangles.cs.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Rectangle() [1/2]

```
MuPDFCore.Rectangle.Rectangle (
            float x0,
            float y0,
            float x1,
            float y1 )
```

Create a new Rectangle from the specified coordinates.

Parameters

x0	The left coordinate of the rectangle.
y0	The top coordinate of the rectangle.
x1	The right coordinate of the rectangle.
y1	The bottom coordinate of the rectangle.

Definition at line 365 of file Rectangles.cs.

6.9.2.2 Rectangle() [2/2]

```
MuPDFCore.Rectangle.Rectangle (
            double x0,
            double y0,
            double x1,
            double y1 )
```

Create a new Rectangle from the specified coordinates.

Parameters

x0	The left coordinate of the rectangle.
y0	The top coordinate of the rectangle.
x1	The right coordinate of the rectangle.
y1	The bottom coordinate of the rectangle.

Definition at line 380 of file Rectangles.cs.

6.9.3 Member Function Documentation

6.9.3.1 Contains()

```
bool MuPDFCore.Rectangle.Contains (
            Rectangle other )
```

Checks whether this Rectangle contains another Rectangle.

Parameters

other	The Rectangle to check.

Returns

A boolean value indicating whether this Rectangle contains the *other* Rectangle.

Definition at line 466 of file Rectangles.cs.

6.9.3.2 Intersect()

```
Rectangle MuPDFCore.Rectangle.Intersect (
            Rectangle other )
```

Compute the intersection between this Rectangle and another one.

Parameters

other	The other Rectangle to intersect with this instance.

Returns

The intersection between the two Rectangles.

Definition at line 443 of file Rectangles.cs.

6.9.3.3 Round() [1/2]

```
RoundedRectangle MuPDFCore.Rectangle.Round ( )
```

Round the rectangle's coordinates to the closest integers.

Returns

A RoundedRectangle with the rounded coordinates.

Definition at line 392 of file Rectangles.cs.

6.9.3.4 Round() [2/2]

```
RoundedRectangle MuPDFCore.Rectangle.Round (
            double zoom )
```

Round the rectangle's coordinates to the closest integers, applying the specified zoom factor.

Parameters

zoom	The zoom factor to apply.

Returns

A RoundedRectangle with the rounded coordinates.

Definition at line 407 of file Rectangles.cs.

6.9.3.5 Split()

```
Rectangle [] MuPDFCore.Rectangle.Split (
            int divisions )
```

Split the rectangle into the specified number of Rectangles.

Parameters

divisions	The number of rectangles in which the rectangle should be split. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than *divisions* that satisfies this condition is used.

Returns

An array of Rectangles that when positioned properly cover the same area as this object.

Definition at line 422 of file Rectangles.cs.

6.9.4 Member Data Documentation

6.9.4.1 Height

```
float MuPDFCore.Rectangle.Height => Y1 - Y0
```

The height of the rectangle.

Definition at line 356 of file Rectangles.cs.

6.9.4.2 Width

```
float MuPDFCore.Rectangle.Width => X1 - X0
```

The width of the rectangle.

Definition at line 351 of file Rectangles.cs.

6.9.4.3 X0

```
float MuPDFCore.Rectangle.X0
```

The left coordinate of the rectangle.

Definition at line 331 of file Rectangles.cs.

6.9.4.4 X1

```
float MuPDFCore.Rectangle.X1
```

The right coordinate of the rectangle.

Definition at line 341 of file Rectangles.cs.

6.9.4.5 Y0

```
float MuPDFCore.Rectangle.Y0
```

The top coordinate of the rectangle.

Definition at line 336 of file Rectangles.cs.

6.9.4.6 Y1

```
float MuPDFCore.Rectangle.Y1
```

The bottom coordinate of the rectangle.

Definition at line 346 of file Rectangles.cs.

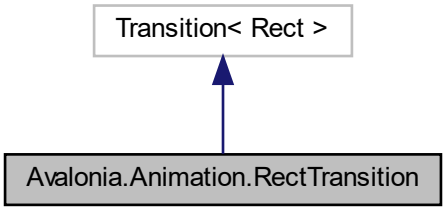
The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

6.10 Avalonia.Animation.RectTransition Class Reference

Transition class that handles AvaloniaProperty with Rect types.

Inheritance diagram for Avalonia.Animation.RectTransition:



Public Member Functions

- override IObservable< Rect > DoTransition (IObservable< double > progress, Rect oldValue, Rect new↩ Value)

  <*inheritdocs*>

6.10.1 Detailed Description

Transition class that handles AvaloniaProperty with Rect types.

Definition at line 26 of file RectTransition.cs.

The documentation for this class was generated from the following file:

- MuPDFCore.MuPDFRenderer/RectTransition.cs

6.11 MuPDFCore.RenderProgress Class Reference

Holds a summery of the progress of the current rendering operation.

Classes

- struct ThreadRenderProgress

  Holds the progress of a single thread.

Properties

- ThreadRenderProgress[] ThreadRenderProgresses [get]

  Contains the progress of all the threads used in rendering the document.

6.11.1 Detailed Description

Holds a summery of the progress of the current rendering operation.

Definition at line 259 of file MuPDF.cs.

6.11.2 Property Documentation

6.11.2.1 ThreadRenderProgresses

```
ThreadRenderProgress [] MuPDFCore.RenderProgress.ThreadRenderProgresses [get]
```

Contains the progress of all the threads used in rendering the document.

Definition at line 286 of file MuPDF.cs.

The documentation for this class was generated from the following file:

- MuPDFCore/MuPDF.cs

6.12 MuPDFCore.RoundedRectangle Struct Reference

Represents a rectangle using only integer numbers.

Public Member Functions

- RoundedRectangle (int x0, int y0, int x1, int y1)

  Create a new RoundedRectangle from the specified coordinates.
- RoundedRectangle[] Split (int divisions)

  Split the rectangle into the specified number of RoundedRectangles.

Public Attributes

- int X0

  The left coordinate of the rectangle.
- int Y0

  The top coordinate of the rectangle.
- int X1

  The right coordinate of the rectangle.
- int Y1

  The bottom coordinate of the rectangle.
- int Width => X1 - X0

  The width of the rectangle.
- int Height => Y1 - Y0

  The height of the rectangle.

6.12.1 Detailed Description

Represents a rectangle using only integer numbers.

Definition at line 475 of file Rectangles.cs.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 RoundedRectangle()

```
MuPDFCore.RoundedRectangle.RoundedRectangle (
            int x0,
            int y0,
            int x1,
            int y1 )
```

Create a new RoundedRectangle from the specified coordinates.

Parameters

x0	The left coordinate of the rectangle.
y0	The top coordinate of the rectangle.
x1	The right coordinate of the rectangle.
y1	The bottom coordinate of the rectangle.

Definition at line 514 of file Rectangles.cs.

6.12.3 Member Function Documentation

6.12.3.1 Split()

```
RoundedRectangle [] MuPDFCore.RoundedRectangle.Split (
            int divisions )
```

Split the rectangle into the specified number of RoundedRectangles.

Parameters

divisions	The number of rectangles in which the rectangle should be split. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than *divisions* that satisfies this condition is used.

Returns

An array of RoundedRectangles that when positioned properly cover the same area as this object.

Definition at line 527 of file Rectangles.cs.

6.12.4 Member Data Documentation

6.12.4.1 Height

```
int MuPDFCore.RoundedRectangle.Height => Y1 - Y0
```

The height of the rectangle.

Definition at line 505 of file Rectangles.cs.

6.12.4.2 Width

```
int MuPDFCore.RoundedRectangle.Width => X1 - X0
```

The width of the rectangle.

Definition at line 500 of file Rectangles.cs.

6.12.4.3 X0

```
int MuPDFCore.RoundedRectangle.X0
```

The left coordinate of the rectangle.

Definition at line 480 of file Rectangles.cs.

6.12.4.4 X1

```
int MuPDFCore.RoundedRectangle.X1
```

The right coordinate of the rectangle.

Definition at line 490 of file Rectangles.cs.

6.12.4.5 Y0

```
int MuPDFCore.RoundedRectangle.Y0
```

The top coordinate of the rectangle.

Definition at line 485 of file Rectangles.cs.

6.12.4.6 Y1

```
int MuPDFCore.RoundedRectangle.Y1
```

The bottom coordinate of the rectangle.

Definition at line 495 of file Rectangles.cs.

The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

6.13 MuPDFCore.RoundedSize Struct Reference

Represents the size of a rectangle using only integer numbers.

Public Member Functions

- RoundedSize (int width, int height)

  Create a new RoundedSize with the specified width and height.
- RoundedRectangle[] Split (int divisions)

  Split the size into the specified number of RoundedRectangles.

Public Attributes

- int Width

  The width of the rectangle.
- int Height

  The height of the rectangle.

6.13.1 Detailed Description

Represents the size of a rectangle using only integer numbers.

Definition at line 181 of file Rectangles.cs.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 RoundedSize()

```
MuPDFCore.RoundedSize.RoundedSize (
            int width,
            int height )
```

Create a new RoundedSize with the specified width and height.

Parameters

width	The width of the rectangle.
height	The height of the rectangle.

Definition at line 198 of file Rectangles.cs.

6.13.3 Member Function Documentation

6.13.3.1 Split()

```
RoundedRectangle [] MuPDFCore.RoundedSize.Split (
            int divisions )
```

Split the size into the specified number of RoundedRectangles.

Parameters

divisions	The number of rectangles in which the size should be split. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than *divisions* that satisfies this condition is used.

Returns

An array of RoundedRectangles that when positioned properly cover an area of the size of this object.

Definition at line 209 of file Rectangles.cs.

6.13.4 Member Data Documentation

6.13.4.1 Height

```
int MuPDFCore.RoundedSize.Height
```

The height of the rectangle.

Definition at line 191 of file Rectangles.cs.

6.13.4.2 Width

```
int MuPDFCore.RoundedSize.Width
```

The width of the rectangle.

Definition at line 186 of file Rectangles.cs.

The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

6.14 MuPDFCore.Size Struct Reference

Represents the size of a rectangle.

Public Member Functions

- Size (float width, float height)

  Create a new Size with the specified width and height.
- Size (double width, double height)

  Create a new Size with the specified width and height.
- Rectangle[ ] Split (int divisions)

  Split the size into the specified number of Rectangles.

Public Attributes

- float Width

  The width of the rectangle.
- float Height

  The height of the rectangle.

6.14.1 Detailed Description

Represents the size of a rectangle.

Definition at line 25 of file Rectangles.cs.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Size() [1/2]

```
MuPDFCore.Size.Size (
            float width,
            float height )
```

Create a new Size with the specified width and height.

Parameters

width	The width of the rectangle.
height	The height of the rectangle.

Definition at line 42 of file Rectangles.cs.

6.14.2.2 Size() [2/2]

```
MuPDFCore.Size.Size (
            double width,
            double height )
```

Create a new Size with the specified width and height.

Parameters

width	The width of the rectangle.
height	The height of the rectangle.

Definition at line 53 of file Rectangles.cs.

6.14.3 Member Function Documentation

6.14.3.1 Split()

```
Rectangle [] MuPDFCore.Size.Split (
            int divisions )
```

Split the size into the specified number of Rectangles.

Parameters

divisions	The number of rectangles in which the size should be split. This must be factorisable using only powers of 2, 3, 5 or 7. Otherwise, the biggest number smaller than *divisions* that satisfies this condition is used.

Returns

An array of Rectangles that when positioned properly cover an area of the size of this object.

Definition at line 64 of file Rectangles.cs.

6.14.4 Member Data Documentation

6.14.4.1 Height

```
float MuPDFCore.Size.Height
```

The height of the rectangle.

Definition at line 35 of file Rectangles.cs.

6.14.4.2 Width

```
float MuPDFCore.Size.Width
```

The width of the rectangle.

Definition at line 30 of file Rectangles.cs.

The documentation for this struct was generated from the following file:

- MuPDFCore/Rectangles.cs

6.15 MuPDFCore.RenderProgress.ThreadRenderProgress Struct Reference

Holds the progress of a single thread.

Public Attributes

- int Progress

  The current progress.
- long MaxProgress

  The maximum progress. If this is 0, this value could not be determined (yet).

6.15.1 Detailed Description

Holds the progress of a single thread.

Definition at line 264 of file MuPDF.cs.

6.15.2 Member Data Documentation

6.15.2.1 MaxProgress

```
long MuPDFCore.RenderProgress.ThreadRenderProgress.MaxProgress
```

The maximum progress. If this is 0, this value could not be determined (yet).

Definition at line 274 of file MuPDF.cs.

6.15.2.2 Progress

```
int MuPDFCore.RenderProgress.ThreadRenderProgress.Progress
```

The current progress.

Definition at line 269 of file MuPDF.cs.

The documentation for this struct was generated from the following file:

- MuPDFCore/MuPDF.cs

Index