

# PDFViewer

## **ZPF.PDFViewer**

- PDFHelper
- PDFInfos
- PDFPageInfo
- PDFPageOrientations

## **ZPF.PDFViewer.DataSources**

- AssetPdfSource
- FilePdfSource
- HttpPdfSource
- IPdfSource
- PdfTempFileHelper

## **ZPF.PDFViewer.Events**

- OnNeedDataEventHandler
- PageChangedEventArgs

## **ZPF.PDFViewer.Maui**

- PDFToImageHelper
- PDFViewer
- ViewExtensions

## Methods

`Single ToPT ( Single cm )`

`Single ToPT ( Double cm )`

`Single ToCM ( Single pt )`

`Single ToCM ( Double pt )`

*Represents metadata and basic information about a Portable Document Format (PDF) document, including its file name, page count, file size, title, and password protection status.*

## Properties

**string FileName ( get; set; )**

*Gets the name of the file associated with this instance.*

**int PageCount ( get; set; )**

*Gets the total number of pages in the document. A value of -1 indicates that the page count has not been determined.*

**Int64 FileSizeInBytes ( get; set; )**

*Gets the size of the file, in bytes.*

**string Title ( get; )**

*Gets the title of the item. If the title is not set, returns the file name without its extension.*

**bool IsPasswordProtected ( get; set; )**

*Gets whether the Portable Document Format (PDF) document is password-protected.*

## Methods

**string ToString ( )**

*Represents information about a single page in a PDF document, including its dimensions, image source, and display status.*

## Properties

**int PageNumber** ( get; set; )

**Double Width** ( get; set; )  
*Width in cm*

**Double Height** ( get; set; )  
*Height in cm*

**ImageSource ImageSource** ( get; set; )

**bool IsCurrentPage** ( get; set; )

**Color SelectedColor** ( get; )

**int RealWidth** ( get; set; )

**int RealHeight** ( get; set; )

**Double WidthRequest** ( get; )

**Double HeightRequest** ( get; )

**PDFPageOrientations Rotation** ( get; set; )

**Double Scale** ( get; set; )  
*Gets or sets the scale factor applied to the element.  
Default value is 1.0.*

*Remarks:*

## Methods

**void SetValues** ( PDFPageInfo pdfPageInfo )

**string ToString** ( )

## Members

**Portrait**

**Landscape**

**FlippedPortrait**

**FlippedLandscape**

*Represents a source for PDF files that are embedded as resources within the application.*

## Properties

**string LastMessage** ( get; set; )

## Methods

**Task<string> GetFilePathAsync** ( )

**Task<string> LoadPDF** ( string resourcePath )

*Load PDF from URL wo rendering it.  
(ImageFileName) are requested.*

*The rendering is done when pages*

*Represents a PDF source that provides access to PDF files via a specified file path.*

## Properties

**string LastMessage** ( get; set; )

## Methods

**Task<string> GetFilePathAsync** ( )

**Task<string> LoadPDF** ( string filePath )

*Load PDF from URL wo rendering it.  
(ImageFileName) are requested.*

*The rendering is done when pages*

*Provides a PDF source that retrieves PDF files asynchronously from a specified HTTP URL.*

## Properties

**string LastMessage** ( get; set; )

## Methods

**Task<string> GetFilePathAsync** ( )

**Task<string> LoadPDF** ( string url )

*Load PDF from URL wo rendering it.  
(ImageFileName) are requested.*

*The rendering is done when pages*



## Properties

`string LastMessage ( get; )`

## Methods

`Task<string> GetFilePathAsync ( )`

`Task<string> LoadPDF ( string url )`

*Load PDF from URL wo rendering it.  
(ImageFileName) are requested.*

*The rendering is done when pages*

*Provides utility methods for creating and managing temporary file paths for PDF documents and related page files.*

## Methods

**string CreateTempPdfFilePath ( )**

*Creates a unique temporary file path for a PDF file.*

**string CreateTempPageFilePath ( string filename )**

*Creates a unique temporary file path for a page file.*

**void DeleteTempFiles ( )**

*Deletes all temporary files from the application's temporary folder.*

## Methods

```
void Invoke ( Object sender )
```

```
IAsyncResult BeginInvoke ( Object sender, AsyncCallback  
                           callback, Object object )
```

```
void EndInvoke ( IAsyncResult result )
```

## Properties

`int CurrentPage ( get; )`

## Methods

**Task SaveFirstPageAsImageAsync** ( string pdfPath, string  
outputImagePath )

*Asynchronously saves the first page of the document as an image file at the specified path.*

**Task SavePageAsImageAsync** ( string pdfPath, string  
outputImagePath, UInt32 pageNumber

*Asynchronously saves a specified page of a PDF document as an image file.*

**Task<ImageSource> GetFirstPageAsImageSourceAsync** ( string  
pdfPath )

*!!! Experimental !!!*

**Task<ImageSource> GetPageAsImageSourceAsync** ( string  
pdfPath, int

*!!! Experimental !!!*

## Properties

**string LastMessage** ( get; set; )

**int CurrentPageNumber** ( get; set; )

*Gets the current page number in the pagination system.*

**ObservableCollection<PDFPageInfo> Pages** ( get; set; )

*Gets or sets the collection of pages contained in the PDF document.*

**bool IsBusy** ( get; set; )

*Gets a value indicating whether the component is currently performing a background operation.*

**bool IsToolbarVisible** ( get; set; )

*Gets or sets a value indicating whether the toolbar is visible.*

**Color ToolbarColor** ( get; set; )

*Gets or sets the Color which will fill the background of the toolbar.*

**PDFInfos Infos** ( get; set; )

*Gets the PDF information associated with the document.*

**Double ZoomFactor** ( get; set; )

## Methods

**void UnloadPDF** ( )

**Task SavePageAsImageAsync** ( string outputPath, UInt32

*Asynchronously saves a specified page of a PDF document as an image file.*

**Task<ImageSource> RenderPageToImageSource** ( int pageNumber )

**Task<bool> LoadPDF** ( string filename, bool genPages )

*Load PDF from URL wo rendering it. The rendering is done when pages (ImageFileName) are requested.*

**Task<bool> LoadPDF** ( string filename, string password, bool

*Load PDF from URL wo rendering it. The rendering is done when pages (ImageFileName) are requested.*

**Task<bool> LoadPDF** ( IPdfSource pdfSource, string url, string password, bool genPages )

*Load PDF from URL wo rendering it. The rendering is done when pages (ImageFileName) are requested.*

...

**Task SaveFirstPageAsImageAsync ( string outputPath )**

*Asynchronously saves the first page of the document as an image file at the specified path.*

**void ClearPages ( )**

*Removes all pages from the collection and updates the associated view to reflect the cleared state.*

**Task ScrollToFirstAsync ( )**

*Scrolls to first page*

**Task ScrollToAsync ( UInt32 pageNumber )**

*Scrolls to 'pageNumber'*

**Task ScrollToLastAsync ( )**

*Scrolls to last page*

**Task DoZoom ( Double zoom )**

## Methods

`Rect GetBoundingBox ( VisualElement view )`