native_pdf_view 4.2.0

Published Sep 18, 2021 • serge.software Null safety

FLUTTER | ANDROID IOS MACOS WEB WINDOWS

267

Metadata →

Flutter plugin to render PDF files on Web, MacOS, Windows, Android and iOS.

More...

native_pdf_view

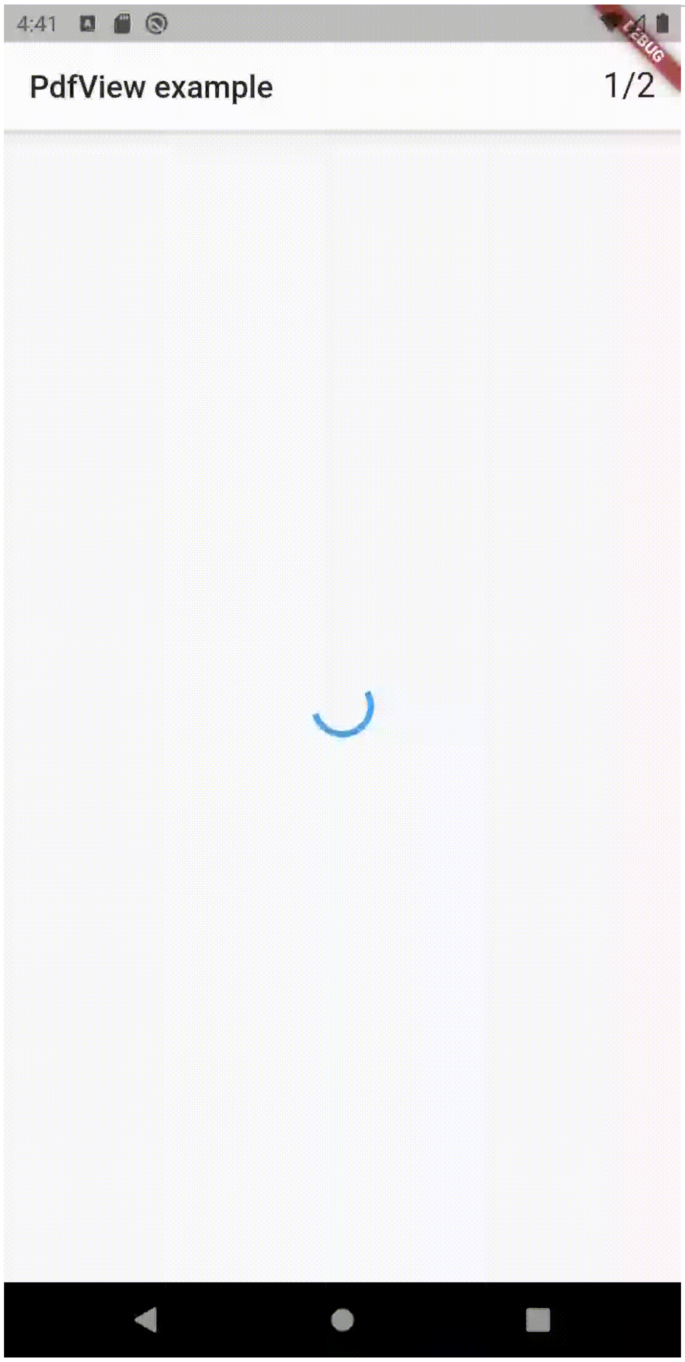
`Flutter` Plugin to render PDF and show a PDF file on **Web**, **MacOs 10.11+**, **Android 5.0+**, **iOS** and **Windows**.

Showcase

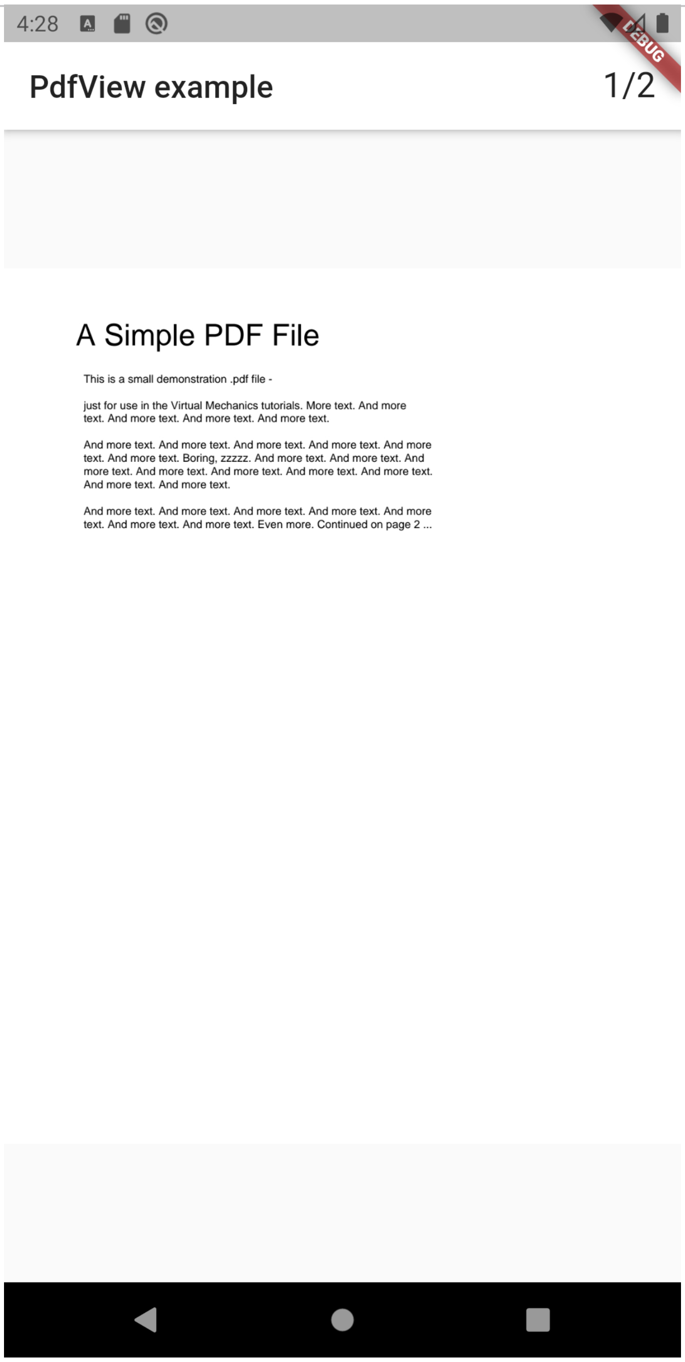
Live	Screenshot

Live

Screenshot



PdfView example 1/2



PdfView example 1/2

A Simple PDF File

This is a small demonstration .pdf file -

just for use in the Virtual Mechanics tutorials. More text. And more text. And more text. And more text. And more text.

And more text. And more text. And more text. And more text. And more text. And more text. Boring, zzzzz. And more text. And more text. And more text. And more text. And more text. And more text. And more text. And more text. And more text.

And more text. And more text. And more text. And more text. And more text. And more text. And more text. Even more. Continued on page 2 ...

Getting Started

In your flutter project add the dependency:

pub v4.2.0

```
dependencies:
  native_pdf_view: any
```

For web add lines in index.html before importing main.dart.js:

```
<script src="//cdnjs.cloudflare.com/ajax/libs/pdf.js/2.4.456/pdf.min.js"></script>
<script type="text/javascript">
  pdfjsLib.GlobalWorkerOptions.workerSrc = "//cdnjs.cloudflare.com/ajax/libs/pdf.j
</script>
```

Usage example

It very simple!

```
import 'package:native_pdf_view/native_pdf_view.dart';

final pdfController = PdfController(
  document: PdfDocument.openAsset('assets/sample.pdf'),
);

Widget pdfView() => PdfView(
  controller: pdfController,
);
```

Api

PdfController

Parameter	Description	Default
document	The document to be displayed	-
initialPage	The page to show when first creating the [PdfView]	1
viewportFraction	The fraction of the viewport that each page should occupy.	1.0

PdfView

Parameter	Description	Default

Parameter	Description	Default
controller	Pages control. See page control and additional pdf info	-
onPageChanged	Called whenever the page in the center of the viewport changes. See Document callbacks	-
onDocumentLoaded	Called when a document is loaded. See Document callbacks	-
onDocumentError	Called when a document loading error. Exception is passed in the attributes	-
documentLoader	Widget showing when pdf document loading	SizedBox()
pageLoader	Widget showing when pdf page loading	SizedBox()
builder	Callback called to render a widget for each page. See custom page builder	Default builder
errorBuilder	Show document loading error message inside `PdfView`	Centered error text
renderer	Custom PdfRenderer library options. See custom renderer options	width: page.width * 2 height: page.height * 2 format: PdfPageFormat.JPEG backgroundColor: '#ffffff'
scrollDirection	Page turning direction	Axis.horizontal
physics	How the widgets should respond to user input	-
pageSnapping	Set to false for mouse wheel scroll on web	true

Additional examples

Open another document

```
pdfController.openDocument(PdfDocument.openAsset('assets/sample.pdf'));
```

Page control:

```
// Jump to specified page
pdfController.jumpTo(3);

// Animate to specified page
_pdfController.animateToPage(3, duration: Duration(milliseconds: 250), curve: Curv

// Animate to next page
_pdfController.nextPage(duration: Duration(milliseconds: 250), curve: Curves.easeI

// Animate to previous page
_pdfController.previousPage(duration: Duration(milliseconds: 250), curve: Curves.e
```

Additional pdf info:

```
// Actual showed page
pdfController.page;

// Count of all pages in document
pdfController.pagesCount;
```

Document callbacks

```
int _actualPageNumber = 0, _allPagesCount = 0;

PdfView(
  controller: pdfController,
  onDocumentLoaded: (document) {
    setState(() {
      _allPagesCount = document.pagesCount;
    });
  },
  onPageChanged: (page) {
    setState(() {
      _actualPageNumber = page;
    });
  },
);

/// Now you can use these values to display the reading status of the document.
Text('Read: $_actualPageNumber of $_allPagesCount');
```

Custom renderer options

```
PdfView(
  controller: pdfController,
  renderer: (PdfPage page) => page.render(
    width: page.width * 2,
    height: page.height * 2,
    format: PdfPageFormat.JPEG,
    backgroundColor: '#FFFFFF',
  ),
);
```

Custom page builder:

```
PdfView(
  controller: pdfController,
  document: snapshot.data,
  pageBuilder: (
    Future<PdfPageImage> pageImage,
    int index,
    PdfDocument document,
  ) => PhotoViewGalleryPageOptions(
      imageProvider: PdfPageImageProvider(
        pageImage,
        index,
        document.id,
      ),
      minScale: PhotoViewComputedScale.contained * 1,
      maxScale: PhotoViewComputedScale.contained * 3.0,
      initialScale: PhotoViewComputedScale.contained * 1.0,
      heroAttributes: PhotoViewHeroAttributes(tag: '${document.id}-$index'),
    ),
);
```

Rendering additional info

On Web

This plugin uses the PDF.js

On Android

This plugin uses the Android native PdfRenderer

On Ios & MacOs

This plugin uses the IOS native CGPDFPage

On Windows

This plugin use PDFium