

# 1 概述

---

ComPDFKit for Flutter 是一个综合型的 PDF SDK。允许您为您的 Flutter 应用程序增加 PDF 功能，比如查看、注释、PDF 内容编辑、表单、签名等。更多信息请查看：<https://www.compdf.com/>。

## 1.1 ComPDFKit for Flutter

---

ComPDFKit for Flutter 由下面两个部分组成：

- **PDF Core API**

Core API 可以独立使用，支持文档渲染、分析、文本提取、文本搜索、表单创建与填充、文档加密、文档拆分合并、注释创建和回复等一站式 PDF 功能。但不包括任何 UI 组件。

- **PDF View**

PDF View 是一个工具类，为开发人员提供根据自己的需求与 PDF 文档进行交互的功能。View 控制器提供快速、高质量的渲染、缩放、滚动和页面导航功能。视图控制派生自平台相关的查看器类，并允许通过扩展满足客户的自定义需求。

## 1.2 主要功能

---

**Viewer** 组件提供以下功能：

- 标准的页面显示模式，包括单页、双页、滚动和封面模式。
- 通过缩略图、大纲和书签进行导航。
- 文本搜索和选取。
- 缩放和自适应页面。
- 在不同的主题之间切换，包括暗黑模式、棕褐色模式、护眼模式和自定义颜色模式
- 文字重新排版。

**注释 (Annotations)** 组件提供以下功能：

- 创建、编辑、删除和回复注释，包括便签、链接、文字、直线、矩形、圆形、亮高、下划线、波浪线、删除线、图章、自由画笔和音频等注释类型。
- 支持注释外观调整。
- 导入和导出 XFDF 格式的注释数据。
- 支持注释展平。
- 预定义注释样式。

**表单 (Forms)** 组件提供以下功能：

- 创建、编辑和删除表单字段，包括按钮、复选框、单选按钮、文本字段、下拉框、列表框和签名字段。
- 填写 PDF 表单。
- 支持 PDF 表单展平。

**文档编辑器 (Document Editor)** 组件提供以下功能：

- PDF 操作，包括拆分、提取和合并PDF文件。
- 页面编辑，包括删除页面、插入页面、裁剪页面、移动页面、旋转页面、交换页面位置和不同文档的页面替换。
- 设置文档信息。
- 提取PF文件图片。
- 查找和替换。

**内容编辑器（Content Editor）** 组件提供以下功能：

- 以编程方式添加和删除 PDF 中的文本，并可以像 Word 一样编辑 PDF。允许修改文本的大小、颜色、对齐方式、文本框位置等。
- 支持编辑 PDF 图像，如移动、旋转、缩放、镜像、裁剪、替换、复制、提取。
- 撤消或重做任何更改。

**安全（Security）** 组件提供以下功能：

- 加密和解密 PDF，包括权限设置和密码保护。

**水印（Watermark）** 组件提供以下功能：

- 创建、编辑、添加、删除、更新、获取水印。
- 支持文字和图片水印。

**数字签名（Digital Signatures）** 组件提供：

- 用数字签名签署文件。
- 创建并验证数字证书。
- 创建并验证数字签名。
- 创建自签名数字 ID、编辑签名外观。
- 支持 PKCS12 证书。
- 支持添加证书到信任列表。

## 1.3 许可证

---

ComPDFKit for Flutter 是一个商业 SDK，需要许可证才能授权开发者开发或发布其应用程序。每个许可证都绑定着唯一的设备 ID 或 App ID。ComPDFKit 拥有灵活的授权模式，请联系[我们的销售团队](#)了解更多信息。然而，即使拥有了许可证，也禁止将 ComPDFKit 的任何文件、示例代码或源代码给任何第三方。

**在线许可证：**引入了在线许可证机制，使许可管理更加便捷。通过在线方式，可以更灵活地管理和更新许可证，以满足项目的需求。

**离线许可证：**在安全性较高、无法连接到互联网或离线环境的场景下，提供离线许可证选项。离线许可证允许在无法连接到互联网的情况下进行授权和使用 ComPDFKit PDF SDK。

## 2 入门指南

---

## 2.1 要求

### 2.1.1 获取许可证密钥

ComPDF 提供两种许可证密钥：免费30天试用许可证，和正式许可证。

#### 如何获取试用许可证密钥

##### 方法 1：在线申请

如果您想获得适用于 Web、Windows、Android、iOS、Flutter 和 React Native 的 PDF SDK 试用版，只需在线申请 [30 天免费试用许可证](#) 即可。

您可以在我们的[价格页](#)上查看免费试用许可证支持的功能。

##### 方法 2：联系销售

如需试用许可证之外的其他平台或功能，请随时联系我们的[销售团队](#)。

#### 如何获取正式许可证密钥

ComPDFKit PDF SDK 是商用 SDK，需要许可证才能授权开发者开发或发布其应用程序。请注意，即使拥有了许可证，也禁止将 ComPDFKit 的任何文件、示例代码或源代码给任何第三方。

##### 方法 1：在线购买

我们允许开发者在[价格页](#)在线购买 iOS 平台的 PDF SDK 的正式许可证，包括标准常用的 PDF 功能。如对套餐有任何疑问，请联系我们的销售人员。

##### 方法 2：联系销售

如需获得其他平台、高级功能、定制需求的正式许可证，或咨询报价，请随时联系我们的[销售团队](#)。

对于 Flutter PDF SDK，正式许可证必须绑定到您应用的 Android `ApplicationId`、iOS `BundleID`。

### 2.1.2 下载PDF SDK

在[Github](#) 或 [pub.dev](#) 上下载适用于ComPDFKit Flutter PDF SDK。

### 2.1.3 系统要求

#### Android

请安装以下所需的软件包：

- 最新稳定版的 [Flutter](#)。
- 最新稳定版的 [Android Studio](#)。
- [Android NDK](#)。
- [Android Virtual Device](#)。

操作环境要求：

- `minSdkVersion 21` 或更高版本。

- `compileSdkVersion` 34 或更高版本。
- `targetSdkVersion` 34 或更高版本。
- Android ABI(s): x86, x86\_64, armeabi-v7a, arm64-v8a。

## iOS

请安装以下所需的软件包：

- 最新稳定版的 [Flutter](#)。
- 最新稳定版的 [Xcode](#)。
- 最新稳定版的 [CocoaPods](#)。下载方法参考：[CocoaPods installation guide](#)。

操作环境要求：

- iOS 12.0 或更高版本。
- Xcode 12.0 或更高版本。（Objective-C 或 Swift）

## 2.2 创建一个项目

### 2.2.1 Android

1. 用 Flutter CLI 创建一个叫做 example 的 Flutter 项目：

```
flutter create --org com.compdfkit.flutter example
```

2. 在终端应用程序中，把当前的工作目录移动到您的项目中：

```
cd example
```

3. 打开 "`example/android/app/src/main/AndroidManifest.xml`" 文件，添加网络、存储权限。

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.compdfkit.flutter.example">

    <!-- 在线认证License需要使用网络权限！ -->
+    <uses-permission android:name="android.permission.INTERNET"/>

    <!-- 需要从设备存储中读取和写入文档 -->
+    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
+    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <!-- 可选配置 -->
+    <uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE"/>

    <application
+        android:requestLegacyExternalStorage="true">

        </application>
</manifest>
```

4. 打开 App 的 Gradle 构建文件 "**android/app/build.gradle**" :

```
open android/app/build.gradle
```

5. 修改 App 最低支持的系统版本，这些操作都在 "**android/app/build.gradle**" 文件中的 `android` 部分里完成（如下所示）：

```
diff --git android/app/build.gradle android/app/build.gradle
--- android/app/build.gradle
+++ android/app/build.gradle
@@ -1,10 +1 @@
 android {
     defaultConfig {
-         minSdkVersion flutter.minSdkVersion
+         minSdkVersion 21
         ...
     }
 }
```

6. 打开项目的

`android/app/src/main/java/com/example/compdfkit/flutter/example/MainActivity.java`，将 `MainActivity` 修改为继承 `FlutterFragmentActivity`：

```
diff --git android/app/src/main/java/com/example/compdfkit/flutter/example/MainActivity.java android/app/src/main/java/com/example/compdfkit/flutter/example/MainActivity.java
--- android/app/src/main/java/com/example/compdfkit/flutter/example/MainActivity.java
+++ android/app/src/main/java/com/example/compdfkit/flutter/example/MainActivity.java
@@ -1,10 +1 @@
-import io.flutter.embedding.android.FlutterActivity;
+import io.flutter.embedding.android.FlutterFragmentActivity;

-public class MainActivity extends FlutterActivity {
+public class MainActivity extends FlutterFragmentActivity {
}
```

或者，您可以更新 `AndroidManifest.xml` 文件，将 `FlutterFragmentActivity` 作为启动活动：

```
<activity
-    android:name=".MainActivity"
+    android:name="io.flutter.embedding.android.FlutterFragmentActivity"
        android:exported="true"
        android:hardwareAccelerated="true"
        android:launchMode="singleTop"
        android:theme="@style/LaunchTheme"
        android:windowSoftInputMode="adjustPan">
```

**注意：**`FlutterFragmentActivity` 不是 Flutter SDK 的官方部分。如果您需要在 ComPDFKit for Flutter 中使用 `CPDFReaderWidget`，您需要使用这部分代码。如果您不需要使用，可以跳过这一步。

7. 打开 "**pubspec.yaml**" 文件，添加 ComPDFKit 插件。

```
dependencies:
  flutter:
    sdk: flutter
+  compdfkit_flutter: ^2.5.0
```

8. 在终端应用程序中，运行下面的命令以获取所有的包：

```
flutter pub get
```

## 2.2.2 iOS

- 用 Flutter CLI 创建一个叫做 example 的 Flutter 项目：

```
flutter create --org com.compdfkit.flutter example
```

- 在终端应用程序中，把当前的工作目录移动到您的项目中：

```
cd example
```

- 在 "*pubspec.yaml*" 中添加 ComPDFKit 依赖：

```
dependencies:  
  flutter:  
    sdk: flutter  
+  compdfkit_flutter: ^2.5.0
```

- 在终端应用程序中，运行下面的命令以获取所有的包：

```
flutter pub get
```

- 在文本编辑器中打开您项目的 "*Podfile*" 文件：

```
open ios/Podfile
```

- 更新平台为 iOS 12 并添加 "*compdfkit\_flutter.podspec*"：

```
- platform :ios, '9.0'  
+ platform :ios, '12.0'  
...  
target 'Runner' do  
  use_frameworks!  
  use_modular_headers!  
  
  flutter_install_all_ios_pods File.dirname(File.realpath(__FILE__))  
+ pod "ComPDFKit",  
podspec:'https://file.compdf.com/cocoapods/ios/compdfkit_pdf_sdk/2.5.0/ComPDFKit.podspec'  
+ pod "ComPDFKit_Tools",  
podspec:'https://file.compdf.com/cocoapods/ios/compdfkit_pdf_sdk/2.5.0/ComPDFKit_Tools.pod  
spec'  
  
end
```

- 转到 "*example/ios*" 文件夹并运行 `pod install` 命令：

```
pod install
```

**注意:** 如果SSL网络请求在运行' pod install '时下载 ComPDFKit '库失败, 您可以查看[故障排除](#)中的处理方法。

#### 8. 保护用户隐私:

为了保护用户隐私, 在访问敏感的隐私数据之前, 需要找到 iOS 10.0 或更高版本 iOS 项目中的 "**Info**" 配置, 并按照下图所示配置相关隐私条款。

```


```objective-c
<key>NSCameraUsageDescription</key>
<string>Your consent is required before you could access the function.</string>

<key>NSMicrophoneUsageDescription</key>
<string>Your consent is required before you could access the function.</string>

<key>NSPhotoLibraryAddUsageDescription</key>
<string>Your consent is required before you could access the function.</string>

<key>NSPhotoLibraryUsageDescription</key>
<string>Your consent is required before you could access the function.</string>

<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```
```
```

## 2.3 应用许可证密钥

如果您还没有获取许可证密钥, 请查看[如何获取许可证密钥](#)。

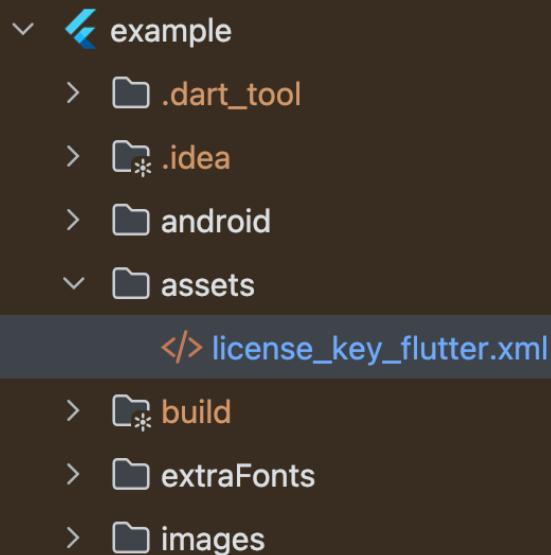
ComPDFKit PDF SDK 目前支持两种许可证密钥验证方式: 在线认证和离线认证。

了解关于:

- [ComPDFKit的许可证认证机制是什么?](#)
- [在线认证和离线认证有什么区别?](#)

准确获取 License Key 对于应用许可证至关重要。

1. 在您收到的邮件中, 找到包含 License Key 的 XML 文件。
2. 将 License.xml 文件复制到 Flutter 项目的 **assets** 目录 中。



3. 打开项目中的 pubspec.yaml 文件，在 flutter: 配置项下启用 assets 目录。

```
52      # The following section is specific to Flutter packages.
53  flutter:
54
55    # The following line ensures that the Material Icons font is
56    # included with your application, so that you can use the icons in
57    # the material Icons class.
58    uses-material-design: true
59    generate: true
60  assets:
61    - assets/
62    - pdfs/
63    - images/
64    - extraFonts/
```

4. 初始化 SDK:

```
// 将 License 文件包含在 Flutter assets 中并复制到设备存储
// 将 `license_key_flutter.xml` 添加到 Flutter 项目的 assets 目录中
File licenseFile = await CPDFFileUtil.extractAsset(context,
'assets/license_key_flutter.xml');
await ComPDFKit.initWithPath(licenseFile.path);
```

其他方式（可选）

```
// Android  
// 将 license_key_flutter.xml 文件复制到 Android 项目的 `android/app/src/main/assets` 目录下:  
await ComPDFKit.initWithPath('assets://license_key_flutter.xml');  
  
// iOS  
// 将 license_key_flutter.xml 文件复制到 iOS 项目目录（或可读路径）下:  
await ComPDFKit.initWithPath('license_key_flutter.xml');
```

## 2.4 运行APP

1. 打开 `lib/main.dart`，将所有内容替换为以下代码。并在 `ComPDFKit.init` 方法中填写提供给您的许可证，这个简单示例将从本地设备文件系统加载 PDF 文档。

有两种不同的方法可以使用 ComPDFKit Flutter API：

- 通过插件呈现文档。
- 通过 `CPDFReaderWidget` 组件显示 ComPDFKit 文档视图。

### 使用插件

打开 `lib/main.dart`，将整个文件替换为以下内容：

```
import 'dart:io';  
  
import 'package:compdfkit_flutter/compdfkit.dart';  
import 'package:compdfkit_flutter/configuration/cpdf_configuration.dart';  
import 'package:compdfkit_flutter/util/cpdf_file_util.dart';  
import 'package:flutter/material.dart';  
  
const String _documentPath = 'pdfs/PDF_Document.pdf';  
  
void main() {  
  runApp(const MyApp());  
}  
  
class MyApp extends StatefulWidget {  
  const MyApp({super.key});  
  
  @override  
  State<MyApp> createState() => _MyAppState();  
}  
  
class _MyAppState extends State<MyApp> {  
  @override  
  void initState() {  
    super.initState();  
    _init();  
  }  
}
```

```

void _init() async {
    File licenseFile = await CPDFFfileUtil.extractAsset(context,
'assets/license_key_flutter.xml');
    final result = await ComPDFKit.initWithPath(licenseFile.path);
    debugPrint('ComPDFKit Init Result: $result');
}

@Override
Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            body: SafeArea(
                child: Center(
                    child: ElevatedButton(
                        onPressed: () async {
                            showDocument(context);
                        },
                        child: const Text(
                            'Open Document'
                        )),
                )));
}

void showDocument(BuildContext context) async {
    var pdfFile = await CPDFFfileUtil.extractAsset(_documentPath);
    var configuration = CPDFConfiguration();
    // Present a document via a plugin.
    ComPDFKit.openDocument(pdfFile.path, password: '', configuration: configuration);
}
}

```

## 使用组件

打开 lib/main.dart，将整个文件替换为以下内容：

```

import 'dart:io';

import 'package:compdfkit_flutter/compdfkit.dart';
import 'package:compdfkit_flutter/configuration/cpdf_configuration.dart';
import 'package:compdfkit_flutter/widgets/cpdf_reader_widget.dart';
import 'package:compdfkit_flutter/util/cpdf_file_util.dart';
import 'package:flutter/material.dart';

const String _documentPath = 'pdfs/PDF_Document.pdf';

void main() {
    runApp(const MyApp());
}

class MyApp extends StatefulWidget {
    const MyApp({super.key});

```

```
    @override
    State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
    String? _document;

    @override
    void initState() {
        super.initState();
        _init();
        _getDocumentPath(context).then((value) {
            setState(() {
                _document = value;
            });
        });
    }

    void _init() async {
        /// 请将其替换为您的 ComPDFKit 许可证
        File licenseFile = await CPDFFileUtil.extractAsset(context,
'assets/license_key_flutter.xml');
        final result = await ComPDFKit.initWithPath(licenseFile.path);
        debugPrint('ComPDFKit Init Result: $result');
    }
}

@Override
Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            resizeToAvoidBottomInset: false,
            appBar: AppBar(
                title: const Text('CPDFReaderWidget Example'),
            ),
            body: _document == null
                ? Container()
                : CPDFReaderWidget(
                    document: _document!,
                    configuration: CPDFConfiguration(),
                    onCreated: (_create) => {}));
}

Future<String> _getDocumentPath(BuildContext context) async {
    final bytes = await DefaultAssetBundle.of(context).load(_documentPath);
    final list = bytes.buffer.asUint8List();
    final tempDir = await ComPDFKit.getTemporaryDirectory();
    var pdfsDir = Directory('${tempDir.path}/pdfs');
    pdfsDir.createSync(recursive: true);

    final tempDocumentPath = '${tempDir.path}/${_documentPath}';
    final file = File(tempDocumentPath);
```

```
    if (!file.existsSync()) {
      file.create(recursive: true);
      file.writeAsBytesSync(list);
    }
    return tempDocumentPath;
}
}
```

## 2. 将您要显示的 PDF 文档添加到项目中

- 创建一个 `pdf` 目录

```
mkdir pdfs
```

- 将您的示例文档复制到新创建的 `pdfs` 目录中，并命名为 `PDF_Document.pdf`

## 3. 在 `pubspec.yaml` 文件中指定 `assets` 目录

```
flutter:
+ assets:
+   - pdfs/
```

## 4. 启动您的 Android 或 iOS 模拟器，或连接您的设备。

```
flutter emulators --launch apple_ios_simulator
```

## 5. 运行应用程序：

```
flutter run
```

# 3 指南

欢迎阅读 ComPDFKit PDF SDK for Flutter 的开发者指南。这些指南将向您展示如何将文档功能添加到您的 Flutter 应用程序中。

如果您是 ComPDFKit 的新手，请查看我们的 [入门指南](#) 指南，快速为您的应用程序添加 PDF 查看、注释和编辑功能。

## 3.1 基本操作

### 3.1.1 概述

本节将演示一些最基本的使用示例，以帮助您快速入门。

#### 指南

[应用许可证密钥](#)

使用许可证密钥初始化 ComPDFKit PDF SDK。

## [打开文档](#)

如何从本地存储打开 PDF。

## [保存文档](#)

如何手动保存以及设置保存回调函数。

### 3.1.2 应用许可证秘钥

如果您还没有获取许可证密钥，请查看[如何获取许可证密钥](#)。

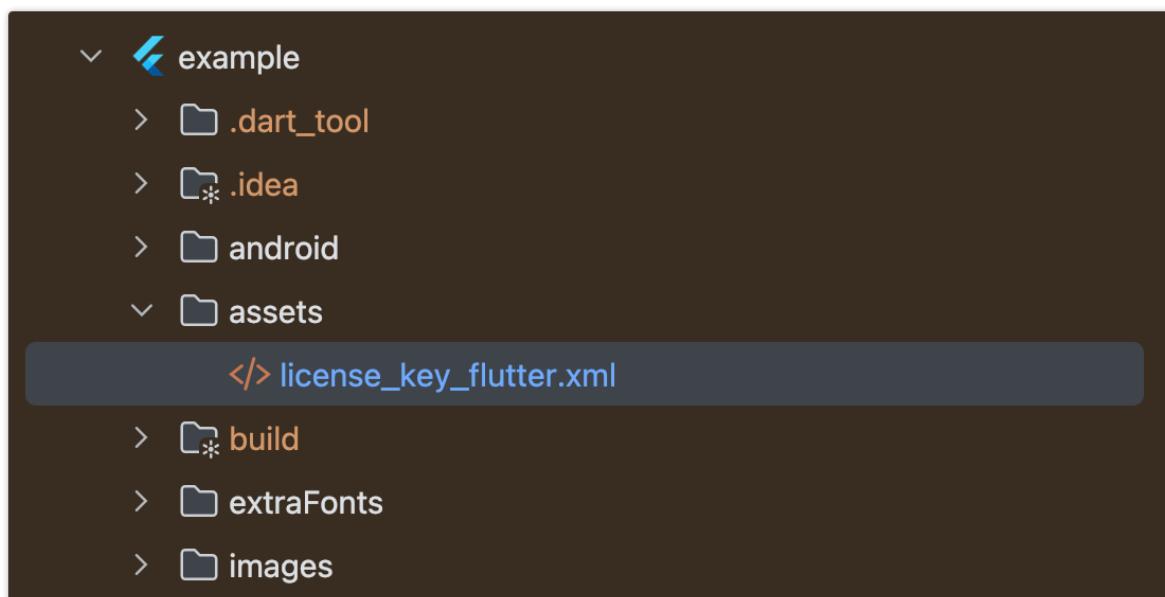
ComPDFKit PDF SDK 目前支持两种许可证密钥验证方式：在线认证和离线认证。

了解关于：

- [ComPDFKit的许可证认证机制是什么？](#)
- [在线认证和离线认证有什么区别？](#)

准确获取 License Key 对于应用许可证至关重要。

1. 在您收到的邮件中，找到包含 License Key 的 XML 文件。
2. 将 License.xml 文件复制到 Flutter 项目的 **assets** 目录 中。



3. 打开项目中的 pubspec.yaml 文件，在 flutter: 配置项下启用 assets 目录。

```
52      # The following section is specific to Flutter packages.
53  flutter:
54
55    # The following line ensures that the Material Icons font is
56    # included with your application, so that you can use the icons in
57    # the material Icons class.
58    uses-material-design: true
59    generate: true
60  assets:
61    - assets/
62    - pdfs/
63    - images/
64    - extraFonts/
```

#### 4. 初始化 SDK:

```
// 将 License 文件包含在 Flutter assets 中并复制到设备存储
// 将 `license_key_flutter.xml` 添加到 Flutter 项目的 assets 目录中
File licenseFile = await CPDFFileUtil.extractAsset(context,
'assets/license_key_flutter.xml');
await ComPDFKit.initWithPath(licenseFile.path);
```

#### 其他方式 (可选)

```
// Android
// 将 license_key_flutter.xml 文件复制到 Android 项目的 `android/app/src/main/assets` 目录下:
await ComPDFKit.initWithPath('assets://license_key_flutter.xml');

// iOS
// 将 license_key_flutter.xml 文件复制到 iOS 项目目录 (或可读路径) 下:
await ComPDFKit.initWithPath('license_key_flutter.xml');
```

### 3.1.3 打开文档

ComPDFKit PDF SDK for Flutter 提供多种方式打开 PDF 文档，适用于不同使用场景：

- 使用 `CPDFReaderWidget` 显示内嵌阅读器 UI。
- 使用 `ComPDFKit.openDocument()` 直接启动完整文档视图。
- 使用 `CPDFDocument` 进行后台处理或无 UI 操作。

## 使用 CPDFReaderWidget (推荐内嵌阅读场景)

```
const String _documentPath = 'pdfs/PDF_Document.pdf';

File document = await CPDFFileUtil.extractAsset(_documentPath, shouldOverwrite: false);
...
Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(),
  body: CPDFReaderWidget(
    document: document.path,
    configuration: CPDFConfiguration(),
    onCreated: (controller) {
      ...
    },
  )));
...
```

## 使用 ComPDFKit.openDocument() (快速跳转阅读器界面)

适合直接打开一个新的阅读界面，快速预览 PDF 文件。

```
const String _documentPath = 'pdfs/PDF_Document.pdf';

File document = await CPDFFileUtil.extractAsset(_documentPath, shouldOverwrite: false);

ComPDFKit.openDocument(
  document.path,
  password: '',
  configuration: CPDFConfiguration(),
);

```

## 使用 CPDFDocument (后台处理场景)

适合不需要 UI 的场景，例如文档注释操作、导出导入等。

```
const String _documentPath = 'pdfs/PDF_Document.pdf';

File document = await CPDFFileUtil.extractAsset(_documentPath, shouldOverwrite: false);
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
CPDFDocumentError error = await document.open(document.path);

if (error == CPDFDocumentError.success) {
  // 使用CPDFDocument对象进行其他操作...
}
```

## 3.1.4 保存文档

### 3.1.4.1 手动保存

在使用 `CPDFReaderWidget` 小部件时，文档会在执行诸如共享、扁平化或添加水印等操作时自动保存。此外，您还可以随时通过 `CPDFReaderWidgetController` 手动保存文档。以下是示例代码：

```
late CPDFReaderWidgetController _controller;

Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(
    leading: IconButton(
      onPressed: () {
        _save();
        Navigator.pop(context);
      },
      icon: const Icon(Icons.arrow_back),
    ),
  ),
  body: CPDFReaderWidget(
    document: widget.documentPath,
    configuration: CPDFConfiguration(),
    onCreated: (controller) {
      setState(() {
        _controller = controller;
      });
    },
  )));
}

void _save() async {
  bool saveResult = await _controller.document.save();
  debugPrint('ComPDFKit-Flutter: saveResult:$saveResult');
}
```

### 3.1.4.2 另存为

ComPDFKit Flutter SDK 支持通过“另存为”功能保存文档。在 Flutter 中使用 `CPDFReaderWidget` 显示文档后，您可以按照以下步骤实现“另存为”功能：

- 使用 `CPDFReaderWidget` 打开文档。
- 当自定义 Flutter 按钮被点击时，调用 `CPDFReaderWidgetController` 的 `saveAs` 接口。

Dart 代码示例如下：

```
late CPDFReaderWidgetController _controller;

@Override
Widget build(BuildContext context) {
  return Scaffold(
    resizeToAvoidBottomInset: false,
    appBar: AppBar(),
    body: Column(children: [
      TextButton(onPressed: () async{
        final tempDir = await ComPDFKit.getTemporaryDirectory();
        String savePath = '${tempDir.path}/temp/test_save_as.pdf';
      });
    ],
  );
}
```

```

        bool saveResult = await _controller.document.saveAs(savePath, removeSecurity:
false, fontSubSet: true),
        child: const Text('另存为'),
        Expanded(child: CPDFReaderWidget(
            document: widget.documentPath,
            configuration: CPDFConfiguration(),
            onCreated: (controller) {
                setState(() {
                    _controller = controller;
                });
            },
        )),
    ],
);
}

```

在 Android 平台上，`saveAs()` 方法还支持通过 Uri 保存到指定目录。例如，以下代码将文档保存到设备公共目录的 `Downloads` 文件夹中：

```

String? uri = await ComPDFKit.createUri('save_as.pdf');
if(uri != null){
    bool saveResult = await _controller.document.saveAs(uri, removeSecurity: false,
fontSubSet: true);
}

```

**参数详情：**

参数名称	类型	描述
savePath	string	“另存为”操作的目标路径
removeSecurity	bool	是否移除文档密码默认值: false
fontSubSet	bool	是否在文档中包含字体子集。该参数会影响 PDF 在其他软件中的显示效果。 如果其他软件中缺少所需字体，文本可能无法正确显示。默认值: true

### 3.1.4.3 保存回调

创建 `CPDFReaderWidget` 时，可以设置保存监听器来处理保存事件。例如：

```

CPDFReaderWidget(
    document: widget.documentPath,
    configuration: CPDFConfiguration(),
    onCreated: (controller) {

},
    onSaveCallback: (){
        debugPrint('CPDFDocument: save success');
},
)

```

## 3.2 查看器

### 3.2.1 概述

ComPDFKit for Flutter 提供了一个高质量、快速、准确且功能丰富的 PDF 查看器。它为开发人员提供了一种快速且可配置的方法，以在 Android 和 iOS 平台的 Flutter 应用程序中集成 PDF 查看器。

您可以通过 [CPDFConfiguration](#) 对象配置页面滚动方向、滚动模式、支持的使用模式等。

#### 指南

##### [查看模式](#)

如何设置打开 PDF 查看器时显示的默认模式，以及配置可切换的模式。

##### [显示模式](#)

如何配置单页、双页、书籍、翻页或滚动方向，以及裁剪和查看模式。

##### [缩放](#)

如何设置初始缩放系数。

##### [主题](#)

如何设置预设主题。

##### [页面导航](#)

如何跳转到指定页码并获取当前页码。

##### [高亮表单字段和超链接](#)

如何高亮表单字段和超链接注释。

##### [文本搜索和选取](#)

如何使用 API 在 PDF 文档中搜索特定文本。

##### [渲染 PDF 页面为图片](#)

如何使用 API 渲染指定的 PDF 页面为图片。

### 3.2.2 查看模式

当您使用 `CPDFReaderWidget` 或 `ComPDFKit.openDocument()` 打开文档时，您可以根据产品需求设置默认显示模式。例如，默认的 `Viewer Mode` 允许查看 PDF 文档和填写表单，但不允许编辑注释、文本等。

#### 设置默认模式

以下示例演示如何将注释模式设置为默认显示模式。在注释模式下，用户可以添加、删除和修改注释。

```
// CPDFReaderWidget 示例
Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(
    title: const Text('CPDFReaderWidget Example'),
```

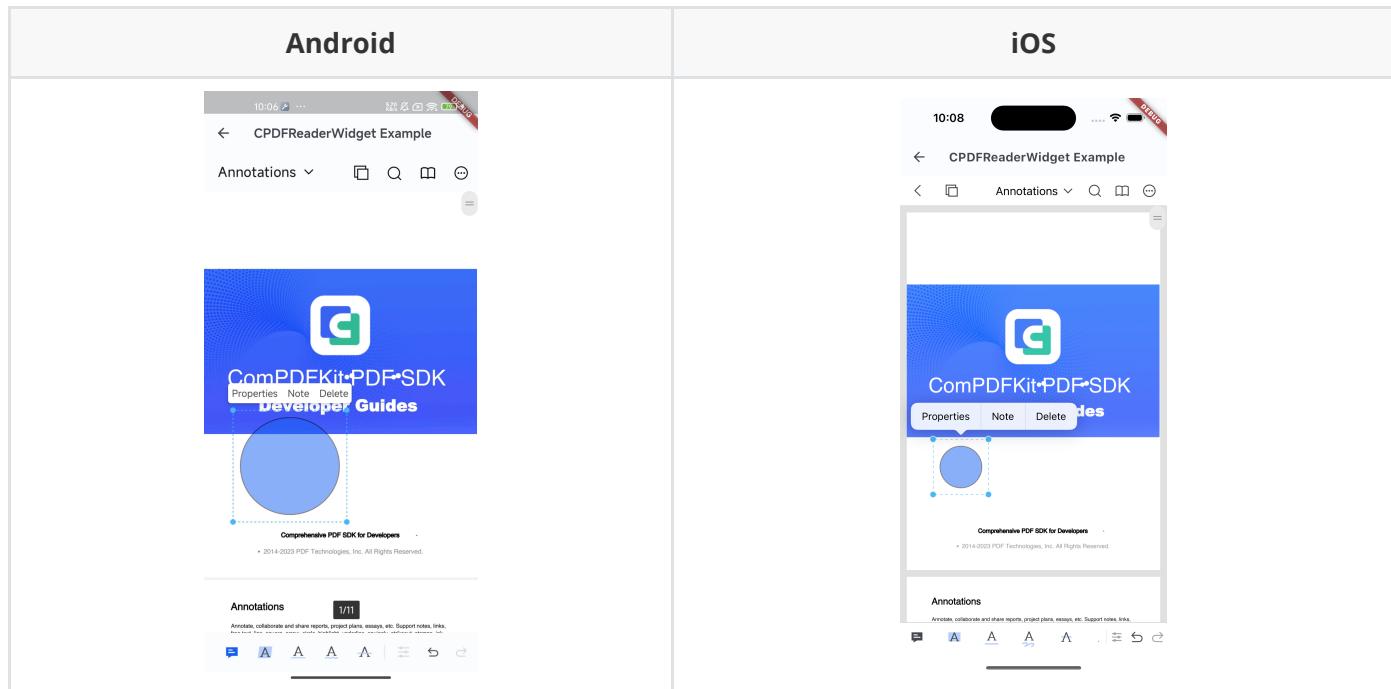
```

),
body: CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(
    modeConfig: const ModeConfig(initialViewMode: CPreviewMode.annotations)
  ),
  onCreated: (controller) {
  },
),
);

// ComPDFKit.openDocument 示例
ComPDFKit.openDocument(documentPath, '', CPDFConfiguration(
  modeConfig: const ModeConfig(initialViewMode: CPreviewMode.annotations)
));

```

结果如下：



## 设置模式列表

您可以通过点击顶部标题来切换模式。根据您的需求，您可以在 `CPDFConfiguration` 中配置所需的模式及其显示顺序。以下示例显示了如何仅配置查看模式和注释模式：

```

// CPDFReaderWidget 示例
Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(
    title: const Text('CPDFReaderWidget Example'),
  ),
  body: CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(
      modeConfig: const ModeConfig(availableViewModes: [
        CPreviewMode.viewer,

```

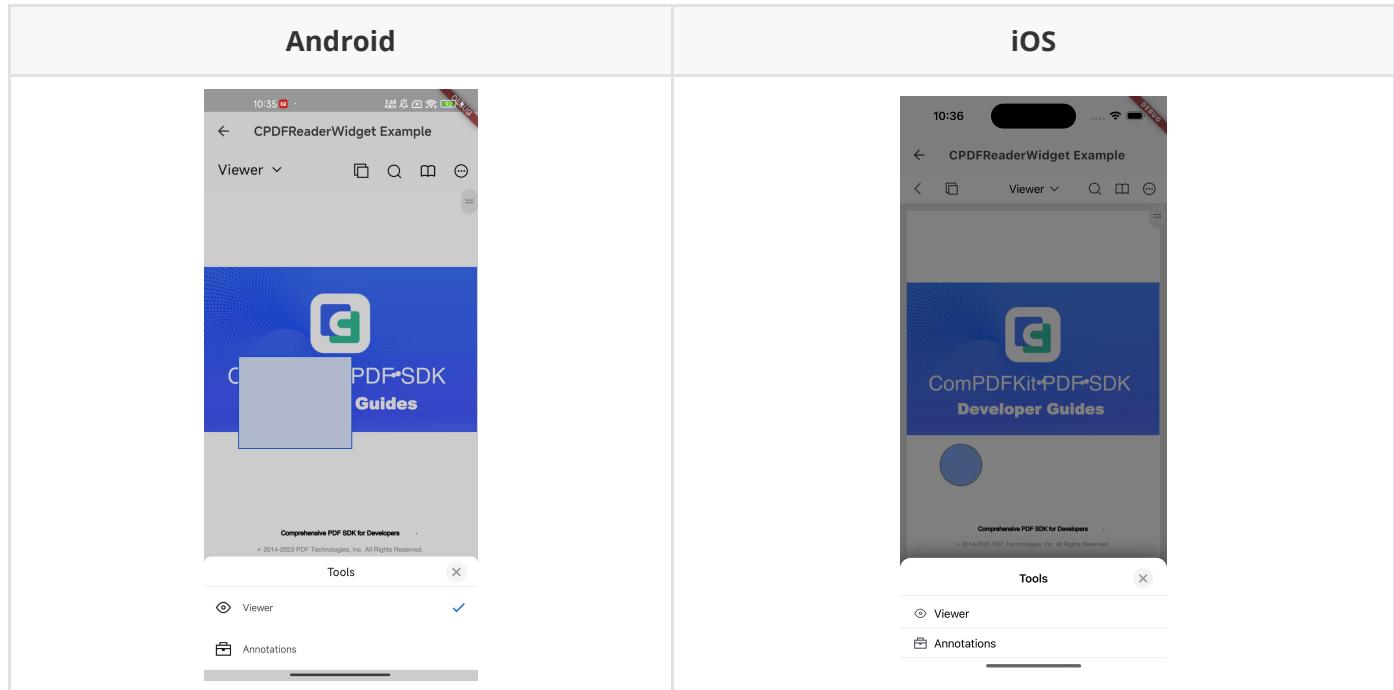
```

    CPreviewMode.annotations
  ])
),
onCreated: (controller) {
},
));
}

// ComPDFKit.openDocument 示例
ComPDFKit.openDocument(documentPath, '', CPDFConfiguration(
  modeConfig: const ModeConfig(availableViewModes: [
    CPreviewMode.viewer,
    CPreviewMode.annotations
  ])
));

```

结果如下：



切换视图模式：

在使用 `CPDFReaderWidget` 显示 PDF 时，可以通过 `CPDFReaderWidgetController` 提供的 API 以编程方式切换当前的视图模式。

```

// 切换到注释模式
controller.setPreviewMode(CPDFViewMode.annotations);

// 获取当前的视图模式
CPDFViewMode viewMode = await controller.getPreviewMode();

```

### 3.2.3 显示模式

滚动方向

页面的滚动方向可以是 `horizontal` 或 `vertical`。

如果 `verticalMode` 是 `true`，表示垂直滚动；否则是水平滚动。

```
CPDFReaderWidgetController? _controller;

Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(),
  body: CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(
      readerViewConfig: const ReaderViewConfig(verticalMode: true)),
    onCreated: (controller) {
      setState(() {
        this._controller = controller;
      });
    },
  ),
);
```

您还可以通过 `CPDFReaderWidgetController` 设置滚动方向：

```
_controller.setVerticalMode(true)
```

## 显示模式

页面显示模式可以是 `singlePage`、`doublePage` 或 `coverPage`。

```
CPDFReaderWidgetController? _controller;

Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(),
  body: CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(
      readerViewConfig: const ReaderViewConfig(displayMode: CPDFDisplayMode.doublePage)),
    onCreated: (controller) {
      setState(() {
        this._controller = controller;
      });
    },
  ),
);
```

通过 `CPDFReaderWidgetController` 设置显示模式：

- `singlePage`

```
_controller.setDoublePageMode(false);
```

- doublePage

```
_controller.setDoublePageMode(true);
```

- coverPage

```
_controller.setCoverPageMode(true);
```

## 滚动模式

滚动模式可以设置为连续滚动或翻页模式。当 `continueMode` 是 `true` 时，表示连续滚动；否则是翻页滚动。

```
CPDFReaderWidgetController? _controller;

Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(),
  body: CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(
      readerViewConfig: const ReaderViewConfig(continueMode: true)),
    onCreated: (controller) {
      setState(() {
        this._controller = controller;
      });
    },
  ),
);
```

通过 `CPDFReaderWidgetController` 设置滚动模式：

```
_controller.setContinueMode(true);
```

## 裁剪模式

要在裁剪 PDF 周围空白区域后显示文档，当 `cropMode` 是 `true` 时，表示启用裁剪模式；否则不裁剪。

```
CPDFReaderWidgetController? _controller;

Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(),
  body: CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(
      readerViewConfig: const ReaderViewConfig(cropMode: true)),
    onCreated: (controller) {
      setState(() {
        this._controller = controller;
      });
    },
);
```

```
},
));
```

通过 `CPDFReaderWidgetController` 设置裁剪模式：

```
_controller.setCropMode(true);
```

### 3.2.4 缩放

设置显示 PDF 文档的默认缩放因子， 默认为 **1.0**， 限制范围为 **1.0 到 5.0**。

```
CPDFReaderWidgetController? _controller;

Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(),
  body: CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(
      readerViewConfig: const ReaderViewConfig(pageScale: 1.0)),
    onCreated: (controller) {
      setState(() {
        this._controller = controller;
      });
    },
  ),
);
```

通过 `CPDFReaderWidgetController` 设置缩放：

```
_controller.setScale(1.0);
```

### 3.2.5 主题

主题是指在显示 PDF 文件时使用不同的背景颜色来渲染 PDF 文档页面，以适应用户偏好和场景需求，增强阅读体验。

在修改主题时，仅更改文档显示时的视觉效果，并不会修改磁盘上的 PDF 文档数据。主题设置不会保存在 PDF 文档数据中。

以下示例显示如何设置黑暗主题：

```
CPDFReaderWidgetController? _controller;

Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(),
  body: CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(
```

```
    readerViewConfig: const ReaderViewConfig(themes: CPDFThemes.dark)),
onCreated: (controller) {
  setState(() {
    this._controller = controller;
  });
},
));
```

通过 `CPDFReaderWidgetController` 设置背景颜色：

```
await _controller.setReadBackgroundColor(theme: CPDFThemes.light);
```

### 主题说明

模式	描述	选项值
light	使用白色背景和黑色文字，适合在光线充足的环境中阅读。	CPDFThemes.light
dark	使用深色背景和浅色文字，适合在光线较暗的环境中阅读。	CPDFThemes.dark
sepia	使用米色背景，适合习惯在纸上阅读的用户。	CPDFThemes.sepia
reseda	使用柔和的浅绿色背景，减少阅读时高亮度和强对比带来的不适，有效缓解视觉疲劳。	CPDFThemes.reseda

## 3.2.6 页面导航

在使用 `CPDFReaderWidget` 组件展示 PDF 时，可以通过 `CPDFReaderWidgetController` 执行以下操作：

### 跳转到指定页码

```
await _controller.setDisplayPageIndex(pageIndex);
```

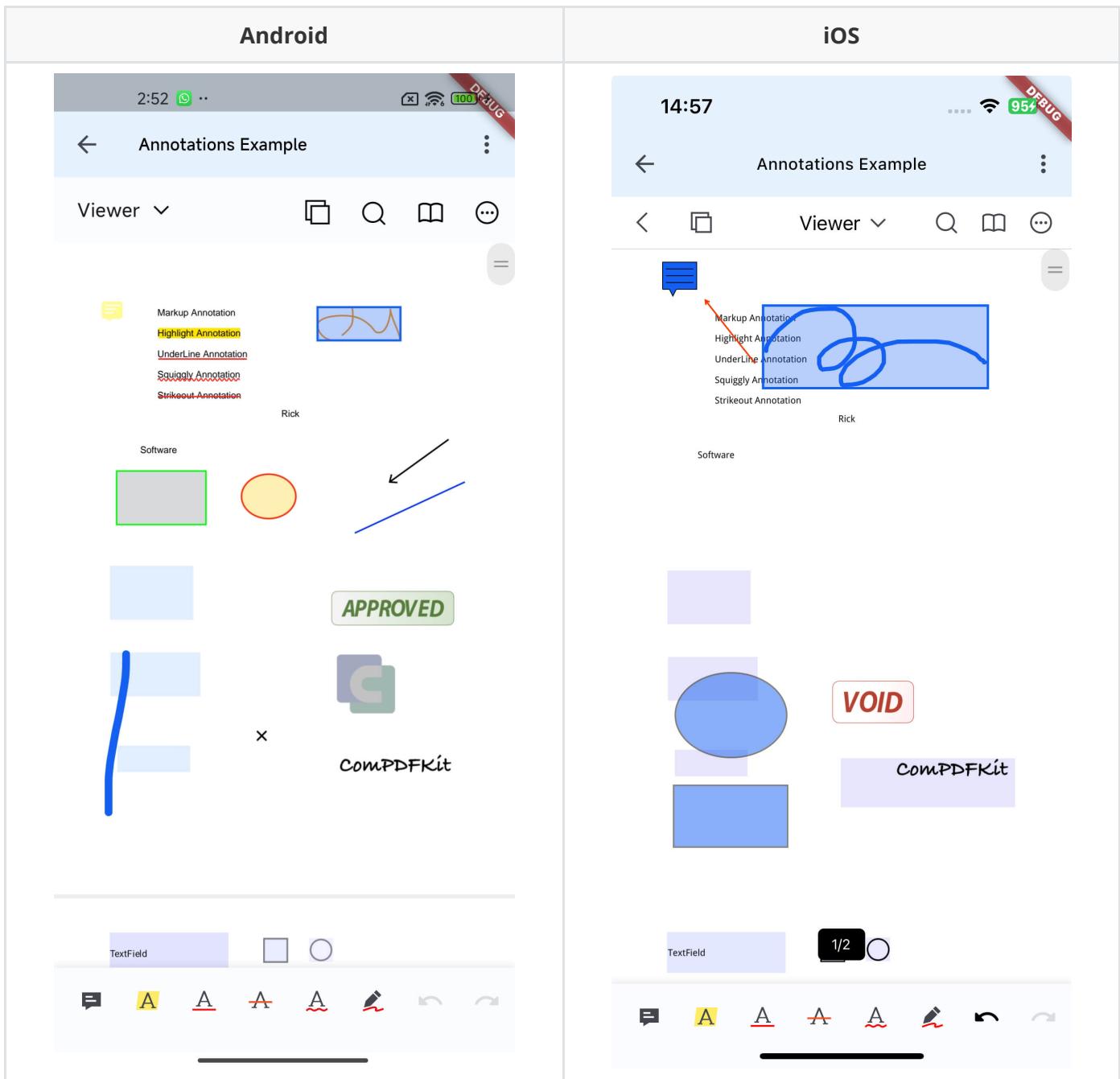
当需要跳转到指定页码并同时突出显示某个内容（例如点击注释时高亮该注释），可以通过 `setDisplayPageIndex` 的 `rectList` 参数在页面上绘制矩形框。

下面示例展示了在点击注释后跳转到注释所在页，并用矩形框高亮注释区域：

```
final pageIndex = 0;
CPDFPage page = controller.document.pageAtIndex(pageIndex);
final pageAnnotations = await page.getAnnotations();
final annotation = pageAnnotations[0];
await controller.setDisplayPageIndex(pageIndex: annotation.page, rectList:
[annotation.rect]);

// 取消高亮
await controller.clearDisplayRect();
```

效果示例：



## 获取当前页码

```
int currentPageIndex = await _controller.getCurrentPageIndex();
```

您还可以在使用 `CPDFReaderWidget` 时设置页面监听器，实时获取当前滑动的页码。

```
CPDFReaderWidgetController? _controller;  
  
CPDFReaderWidget(  
  document: widget.documentPath,  
  configuration: CPDFConfiguration(),  
  onCreated: (controller) {  
    setState(() {  
      this._controller = controller;  
    });  
  },
```

```
},
onPageChanged: (pageIndex){
  debugPrint('当前页码: ${pageIndex}');
},
)
```

### 3.2.7 高亮表单字段和超链接

您可以使用 `CPDFConfiguration` 或 `CPDFReaderWidgetController` 来设置当前文档中的表单和超链接注释为高亮显示。

PDF 表单字段的高亮功能可以帮助用户快速定位并填写表单，显著提高涉及大量表单填写的场景的效率。

超链接的高亮功能使用户能够在 PDF 文档中为关键信息添加超链接，帮助其他用户迅速找到并理解信息。这不仅提高了 PDF 文档的可读性，还增强了其交互性。

以下是如何高亮表单字段和超链接的示例：

```
CPDFReaderWidgetController? _controller;

/// 使用 CPDFConfiguration
CPDFReaderWidget(
  document: widget.documentPath,
  configuration: CPDFConfiguration(
    readerViewConfig: CPDFReaderViewConfig(linkHighlight: true, formFieldHighlight:
true)),
  onCreated: (controller) {
    setState(() {
      this._controller = controller;
    });
  },
)

/// 使用 CPDFReaderWidgetController
/// 启用超链接高亮。
await _controller.setLinkHighlight(true);
/// 启用表单字段高亮。
await _controller.setFormFieldHighlight(true);
```

### 3.2.8 文本搜索和选取

在 Flutter SDK 中，ComPDFKit 提供了 `CPDFTextSearcher` API，用于在 PDF 文档中搜索指定文本、高亮显示结果，并获取上下文片段。

#### 1. 获取 `CPDFTextSearcher` 实例

在执行搜索之前，需要通过 `CPDFDocument` 获取一个 `CPDFTextSearcher` 对象：

```
late CPDFReaderWidgetController _controller;

Scaffold(
```

```

resizeToAvoidBottomInset: false,
body: CPDFReaderWidget(
  document: widget.documentPath,
  configuration: CPDFConfiguration(),
  onCreated: (controller) {
    setState(() {
      _controller = controller;
    });
  },
));
CPDFTextSearcher? textSearcher = _controller?.document.getTextSearcher();

```

## 2. 文本搜索

使用 searchText 方法可以在 PDF 文档中搜索关键字。您可以指定搜索选项，例如是否区分大小写或是否匹配整个单词。

```

// 在 PDF 文档中搜索关键字
List<CPDFTextRange> results = await searcher.searchText(
  'keywords',
  searchOptions: CPDFSearchOptions.caseInsensitive, // 搜索选项
);

```

搜索结果将以 CPDFTextRange 列表的形式返回，每个对象表示文档中匹配关键字的位置范围。

### 搜索选项说明

枚举值	描述
CPDFSearchOptions.caseInsensitive	不区分大小写（默认）
CPDFSearchOptions.caseSensitive	区分大小写
CPDFSearchOptions.matchWholeWord	匹配整个单词

## 3. 选择搜索结果

```

// 高亮第一个搜索结果
if (results.isNotEmpty) {
  CPDFTextRange range = results[0];
  await searcher.selection(range);
}

```

## 4. 清除搜索结果

如需重置搜索状态并清空结果：

```
await searcher.clearSearch();
```

## 5. 获取上下文文本

使用 CPDFPage.getText 可以提取指定范围内的文本片段，并可选择性地扩展范围以包含上下文。

```
// 假设我们已经获取到一个搜索结果
CPDFTextRange range = results[0];

// 扩展范围，在前后各增加 20 个字符
final expandedRange = range.expanded(before: 20, after: 20);

// 从对应页面中获取上下文文本
final contextText = await page.getText(expandedRange);
print("带上下文的文本: $contextText");
```

### 设置搜索高亮样式

在 PDF 搜索中，SDK 允许通过 CPDFConfiguration 自定义搜索结果的高亮显示方式。

开发者可以分别配置：

- **普通匹配结果 (normalKeyword)**：显示所有匹配到的候选文本，仅支持Android平台。
- **当前选中结果 (focusKeyword)**：突出显示用户正在定位的搜索结果。

### 示例代码

```
CPDFConfiguration(globalConfig: const CPDFGlobalConfig(
  search: CPDFSearchConfig(
    normalKeyword: CPDFKeywordConfig(
      borderColor: Colors.transparent,
      fillColor: Color(0x77FFFF00),
    ),
    focusKeyword: CPDFKeywordConfig(
      borderColor: Colors.transparent,
      fillColor: Color(0xCCFD7338),
    ),
  )
))
```

在 CPDFReaderWidget 隐藏了顶部的工具栏情况下，可通过API显示、隐藏搜索视图

```
// 显示搜索视图
await controller.showTextSearchView();

// 隐藏搜索视图
await controller.hideTextSearchView();
```

## 3.2.9 PDF 页面转图片

此接口用于将指定的 PDF 页面渲染为图片，可用于生成预览图或导出页面内容。

返回的数据类型为 Uint8List，可直接用于显示或保存为图片文件。

## 接口声明

```
Future<Uint8List> renderPage({  
    required int pageIndex,  
    required int width,  
    required int height,  
    Color backgroundColor = Colors.white,  
    bool drawAnnot = true,  
    bool drawForm = true  
})
```

## 参数说明

参数名	类型	必填	描述
pageIndex	int	是	需要渲染的页面索引，从 0 开始
width	int	是	渲染图片的宽度（像素）
height	int	是	渲染图片的高度（像素）
backgroundColor	Color	否	页面背景颜色，默认白色，仅支持Android平台
drawAnnot	bool	否	是否渲染注释，默认 true，仅支持Android平台
drawForm	bool	否	是否渲染表单，默认 true，仅支持Android平台

## 使用示例

```
final pageImage = await controller.renderPage(  
    pageIndex: 0,  
    width: 1080,  
    height: 1920,  
    backgroundColor: Colors.white,  
    drawAnnot: true,  
    drawForm: true,  
);  
  
// 显示图片  
Image.memory(pageImage);
```

## 3.3 注释

### 3.3.1 概述

ComPDFKit Flutter SDK 提供了操作注释的相关API，允许您管理文档中的注释。我们正在不断增强这些功能，以适应更广泛的使用场景。

## 指南

### 获取注释

获取注释列表和注释对象。

### 创建注释

通过 `CPDFReaderWidget` 创建各种类型注释。

### 导入和导出

管理 XFDF 文件中的注释。

### 删除注释

如何删除指定的注释或文档中的所有注释。

### 注释扁平化

将文档现有注释以图片方式永久保存到文档中。

### 注释历史管理

用于管理当前文档中注释的编辑历史，提供撤销与重做功能。

## 3.3.2 获取注释

要获取 PDF 中的所有注释，需要按照以下步骤操作：

基本流程：

1. 获取文档对象（`CPDFReaderWidget` 的控制器或 `CPDFDocument` 实例）。
2. 获取页面总数，以便遍历每一页。
3. 逐页获取页面对象（`CPDFPage`）。
4. 从页面对象中获取注释列表。
5. 遍历并收集所有页面的注释对象。

### 使用 `CPDFReaderWidget` 获取注释

适用于 Flutter UI 集成组件场景。你需要先获取 `CPDFReaderWidgetController`，再通过文档对象访问页面和注释。

```
CPDFReaderWidgetController? _controller;

// 初始化 CPDFReaderWidget，并在 onCreated 回调中获取 controller
CPDFReaderWidget(
  document: widget.documentPath,
  password: widget.password,
  configuration: widget.configuration,
  onCreated: (controller) {
    setState(() {
      _controller = controller;
    });
  }
);
```

```

    });
},
);

// 获取注释列表
List<CPDFAnnotation> annotations = [];

if (_controller != null) {
    int pageCount = await _controller!.document.getPageCount();

    for (int i = 0; i < pageCount; i++) {
        CPDFPage page = _controller!.document.pageAtIndex(i);
        List<CPDFAnnotation> pageAnnotations = await page.getAnnotations();
        annotations.addAll(pageAnnotations);
    }
}

```

### 使用 `CPDFDocument` 获取注释

适用于纯逻辑或后台处理场景（无需 UI 组件）。

```

// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
CPDFDocumentError error = await document.open(pdfFilePath);

if (error == CPDFDocumentError.success) {
    List<CPDFAnnotation> annotations = [];

    int pageCount = await document.getPageCount();

    for (int i = 0; i < pageCount; i++) {
        CPDFPage page = document.pageAtIndex(i);
        List<CPDFAnnotation> pageAnnotations = await page.getAnnotations();
        annotations.addAll(pageAnnotations);
    }

    // annotations 现在包含了文档中所有页面的注释对象
}

```

## 3.3.3 创建注释

ComPDFKit 支持多种类型的注释，包括文本注释、链接、图形、高亮、图章、手写标注以及音频注释，全面满足不同的注释需求。

### 创建注释

通过 `CPDFReaderWidget`，用户可以进入注释模式并通过触摸操作添加注释。注释的类型通过 `setAnnotationMode()` 方法设置。

```
CPDFReaderWidgetController? _controller;
```

```
// Initialize CPDFReaderWidget and get the controller in the onCreate callback
CPDFReaderWidget(
  document: widget.documentPath,
  password: widget.password,
  configuration: widget.configuration,
  onCreate: (controller) {
    setState(() {
      _controller = controller;
    });
  },
);

await _controller?.setAnnotationMode(CPDFAnnotationType.highlight);
```

完整支持的注释类型 (CPDFAnnotationType 枚举) 包括:

注释类型	枚举值
文本注释	note
高亮	highlight
下划线	underline
波浪线	squiggly
删除线	strikeout
墨水	ink
墨水橡皮擦	ink_eraser
铅笔	pencil
圆形	circle
矩形	square
箭头	arrow
线段	line
文本	freetext
电子签名	signature
图章注释	stamp
图片	pictures
链接	link
音频	sound
退出绘制模式	unknown

### 退出注释创建模式

完成注释后，可调用以下方法退出注释状态：

```
await _controller?.setAnnotationMode(CPDFAnnotationType.unknown);
```

### 获取当前绘制注释类型

可用于判断当前是否处于注释状态或获取当前选中的注释工具：

```
CPDFAnnotationType currentType = await _controller?.getAnnotationMode();
```

## 3.3.4 导入和导出

XFDF 是一种类似 XML 的标准，来自 Adobe XFDF，用于编码注释和表单字段值。它与 Adobe Acrobat 以及其他一些第三方框架兼容。

ComPDFKit Flutter SDK 支持读取和写入 XFDF 格式来导入和导出注释。以下指南展示了如何使用 XFDF 格式导入和导出注释。

## 导入注释

您可以通过调用 `importAnnotations(xfdfFile)` 方法将 XFDF 文件导入到文档中。这将允许您将 XFDF 格式的注释数据应用到现有的 PDF 文档。

- **xfdfFile**: 指定要导入的 XFDF 文件路径。

在 Android 平台上，`xfdfFile` 可以是：

- 文件 URI (例如，`content://` 格式)
- 文件路径 (如 `/storage/emulated/0/Download/annotations.xfdf`)
- 位于 `assets` 目录下的文件

示例代码：

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreate 回调中获取 controller
CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(),
  onCreate: (controller) {
    setState(() {
      this._controller = controller;
    });
  },
)
// 导入注释
bool result = await _controller.document.importAnnotations(xfdfFile);
```

使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(filePath);
if (error == CPDFDocumentError.success) {
  // 导入注释
  bool result = await document.importAnnotations(xfdfFile);
}
```

## 导出注释

您可以通过调用 `exportAnnotations()` 方法将当前文档中的注释导出为 XFDF 格式文件。这对于将注释数据保存为文件或与其他系统共享注释数据非常有用。

- 此方法不需要路径参数。系统会自动生成并返回导出的 XFDF 文件路径。

示例代码：

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreateed 回调中获取 controller
CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(),
  onCreateed: (controller) {
    setState(() {
      this._controller = controller;
    });
  },
)
// 导出注释
String xfdfFilePath = await _controller.document.exportAnnotations();
```

使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
CPDFDocumentError error = await document.open(pdfFilePath);
if (error == CPDFDocumentError.success) {
  // 导出注释
  String xfdfFilePath = await document.exportAnnotations();
}
```

### 3.3.5 删除注释

ComPDFKit Flutter SDK支持通过api删除选定的注释，删除注释步骤如下：

1. 获取文档对象（`CPDFReaderWidget` 的控制器或 `CPDFDocument` 实例）。
2. 获取需要删除注释的页面对象。
3. 获取该页面的注释列表。
4. 在注释列表中寻找想要删除的注释。
5. 删除该注释。

示例代码：

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreateed 回调中获取 controller
CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(),
```

```
onCreated: (controller) {
    setState(() {
        this._controller = controller;
    });
},
)
// 删除第一页第一个注释
CPDFPage page = _controller.document.pageAtIndex(0);
var pageAnnotations = await page.getAnnotations();
await page.removeAnnotation(pageAnnotations[0]);
```

使用 `CPDFDocument` :

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(filePath);
if (error == CPDFDocumentError.success) {
    // 删除第一页第一个注释
    CPDFPage page = document.pageAtIndex(0);
    var pageAnnotations = await page.getAnnotations();
    await page.removeAnnotation(pageAnnotations[0]);
}
```

或者您也可以通过调用 `removeAllAnnotations()` 方法删除当前文档中的所有注释。

- 返回值为 `boolean`，表示操作是否成功。

示例代码：

使用 `CPDFReaderWidget` :

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreated 回调中获取 controller
CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(),
    onCreated: (controller) {
        setState(() {
            this._controller = controller;
        });
    },
)
// 删除所有注释
bool deleteResult = await _controller.document.removeAllAnnotations();
```

使用 `CPDFDocument` :

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(filePath);
if (error == CPDFDocumentError.success) {
    // 删除所有注释
    bool deleteResult = await document.removeAllAnnotations();
}
```

注意：此方法不会删除超链接注释。

### 3.3.6 注释扁平化

注释扁平化是指将可编辑的注释转换为不可编辑、不可修改的静态图像或纯文本形式。当对注释进行扁平化时，整个文档所有可编辑元素（包含注释和表单）都将执行扁平化，所以注释扁平化又称为文档扁平化。

以下是注释扁平化的示例代码：

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreated 回调中获取 controller
CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(),
    onCreated: (controller) {
        setState(() {
            this._controller = controller;
        });
    },
)

final savePath = 'file:///storage/emulated/0/Download/flatten.pdf';
// or use a Uri on Android:
// final savePath = await ComPDFKit.createUri('flatten_test.pdf', 'compdfkit',
// 'application/pdf');
final fontSubset = true;
final result = await _controller.document.flattenAllPages(savePath, fontSubset);
```

使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(pdfFilePath);
if (error == CPDFDocumentError.success) {
    final savePath = 'file:///storage/emulated/0/Download/flatten.pdf';
    // or use a Uri on Android:
    // final savePath = await ComPDFKit.createUri('flatten_test.pdf', 'compdfkit',
'application/pdf');
    final fontSubset = true;
    final result = await _controller.document.flattenAllPages(savePath, fontSubset);
}
```

## 什么是文档扁平化？

文档扁平化是指将一个包含注释、表单字段或图层的 PDF 文档中的这些可编辑元素转换为不可编辑、不可修改的静态图像或纯文本形式的过程。这个过程的目的是将文档的最终状态锁定，使其不再包含可编辑的元素。

文档扁平化通常在以下情境下应用：

1. **保护文档内容：** 扁平化可用于保护文档内容，确保文档在传播或分享过程中不被更改。这对于确保文档的完整性和保密性非常重要。
2. **表单提交：** 在表单处理中，扁平化可以用于将用户填写的表单字段和注释转换为静态图像，以便方便传输、存档或打印，同时防止在后续阶段对表单内容的修改。
3. **兼容性和显示：** 一些 PDF 阅读器或浏览器对于包含许多注释或图层的 PDF 文档的显示和交互可能存在问题。文档扁平化可以帮助解决这些兼容性问题，提高文档在各种环境中的可视化效果。
4. **减小文件大小：** 扁平化后的文档通常会减小文件大小，因为可编辑元素被转换为静态图像或文本，不再需要保存编辑信息的附加数据。

## 3.3.7 注释历史管理

`CPDFAnnotationHistoryManager` 类用于管理当前文档中注释的编辑历史，提供撤销与重做功能，支持对注释的添加、修改、删除等操作进行回退或恢复。

### 获取注释历史管理器

你可以通过 `CPDFReaderWidgetController` 获取 `CPDFAnnotationHistoryManager` 实例：

```
CPDFReaderWidgetController? _controller;
// Initialize CPDFReaderWidget and obtain the controller in the onCreated callback
CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(),
    onCreated: (controller) {
        setState(() {
            this._controller = controller;
        });
    },
)
CPDFAnnotationHistoryManager historyManager = _controller.annotationHistoryManager;
```

## 设置历史状态变更监听器

调用 `setOnHistoryChangedListener()` 方法设置监听器，以便在注释历史状态变化时（如用户新增或撤销了注释）收到通知：

```
historyManager.setOnHistoryChangedListener((canUndo, canRedo) {  
    print('Can Undo: $canUndo, Can Redo: $canRedo');  
    // 根据状态更新撤销/重做按钮的启用状态  
});
```

## 判断是否可以撤销或重做

使用以下方法判断当前是否存在可撤销或可重做的操作：

```
bool canUndo = await historyManager.canUndo();  
bool canRedo = await historyManager.canRedo();
```

这些方法返回布尔值，表示是否存在对应的历史记录。

## 执行撤销和重做操作

若存在可撤销或重做的历史记录，可调用以下方法进行操作：

```
await historyManager.undo(); // 撤销上一步注释操作  
await historyManager.redo(); // 重做上一步被撤销的注释操作
```

# 3.4 表单

## 3.4.1 概述

ComPDFKit Flutter SDK 支持通过API管理表单相关数据。

### 指南

#### 获取表单

如何获取PDF文档中的表单数据

#### 创建表单域

如何切换到创建表单模式创建表单

#### 删除表单域

删除指定的表单域。

#### 导入和导出

将表单数据导出为 XFDF 格式文件进行管理。

## 3.4.2 表单域类型说明

ComPDFKit 支持多种符合 PDF 标准的表单域类型，可以由许多程序读取和写入，包括 Adobe Acrobat 和其他符合标准的 PDF 处理程序，支持的表单域类型如下：

类型	描述	类名
Check Box(复选框)	从预定义选项中选择一个或多个选项	CPDFCheckBoxWidget
Radio Button(单选按钮)	从预定义选项中选择一个选项	CPDFRadioButtonWidget
Push Button(按钮)	在文档上创建自定义按钮，按下按钮时执行操作	CPDFPushButtonWidget
List Box(列表框)	从预定义选项列表中选择一个或多个选项	CPDFListboxWidget
Combo Box(下拉菜单)	从可用文本选项的下拉列表中选择一个选项	CPDFComboBoxWidget
Text(文本域)	输入文本内容，如姓名，地址，电子邮箱等	CPDFTextWidget
Signature(签名域)	对 PDF 文档进行数字签名或电子签名	CPDFSignatureWidget

### 3.4.3 获取表单

获取表单列表和表单对象的步骤如下：

1. 获取文档对象（`CPDFReaderWidget` 的控制器或 `CPDFDocument` 实例）。
2. 获取页面对象。
3. 取得页面对象中的表单列表。

以下是对应的代码：

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreated 回调中获取 controller
CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(),
  onCreated: (controller) {
    setState(() {
      this._controller = controller;
    });
  },
)
// 获取文档所有表单数据
List<CPDFWidget> widgets = [];
int pageCount = await _controller.document.getPageCount();
for (int i = 0; i < pageCount; i++) {
  CPDFPage page = _controller.document.pageAtIndex(i);
  var pageWidgets = await page.getWidgets();
  widgets.addAll(pageWidgets);
}
```

使用 `CPDFDocument` :

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(filePath);
if (error == CPDFDocumentError.success) {
    // 获取文档所有表单数据
    List<CPDFWidget> widgets = [];
    int pageCount = await document.getPageCount();
    for (int i = 0; i < pageCount; i++) {
        CPDFPage page = document.pageAtIndex(i);
        var pageWidgets = await page.getWidgets();
        widgets.addAll(pageWidgets);
    }
}
```

### 3.4.4 创建表单域

#### 创建表单域

通过 `CPDFReaderWidget`， 用户可以进入表单模式并通过触摸操作添加表单域。

```
CPDFReaderWidgetController? _controller;

// Initialize CPDFReaderWidget and get the controller in the onCreated callback
CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(
        modeConfig: const CPDFModeConfig(
            initialViewMode: CPDFViewMode.forms)),
    onCreated: (controller) {

});

await controller.setFormCreationMode(CPDFFormType.textField);
```

#### 退出创建模式

```
await controller.exitFormCreationMode();
```

#### 支持的表单域类型

类型
textField
checkBox
radioButton
listBox
comboBox
signaturesFields
pushButton

### 3.4.5 删 除 表 单 域

ComPDFKit Flutter SDK支持通过api删除选定的表单域，删除表单域步骤如下：

1. 获取文档对象（`CPDFReaderWidget` 的控制器或 `CPDFDocument` 实例）。
2. 获取需要删除表单的页面对象。
3. 获取该页面的表单列表。
4. 在表单域列表中寻找想要删除的表单。
5. 删除该表单。

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreated 回调中获取 controller
CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(),
  onCreated: (controller) {
    setState(() {
      this._controller = controller;
    });
  },
)
// 删除第一页第一个表单
CPDFPage page = _controller.document.pageAtIndex(0);
var pageWidgets = await page.getWidgets();
await page.removeWidget(pageWidgets[0]);
```

使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(pdfFilePath);
if (error == CPDFDocumentError.success) {
    // 删除第一页第一个表单
    CPDFPage page = document.pageAtIndex(0);
    var pageWidgets = await page.getWidgets();
    await page.removeWidget(pageWidgets[0]);
}
```

## 3.4.6 导入和导出

导入和导出 XFDF 表单的方法使用户能够保存和还原 PDF 文档表单数据，方便填写表单数据。

### 导入表单

导入表单的代码如下：

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreate 回调中获取 controller
CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(),
    onCreate: (controller) {
        setState(() {
            this._controller = controller;
        });
    },
)
bool importResult = await
_controller.document.importWidgets('data/your_package_name/files/xxx.xfdf');
// or use Uri on the Android Platform.
const xfdfFile = 'content://media/external/file/1000045118';
const importResult = await _controller.document.importWidgets(xfdfFile);
```

使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(pdfFilePath);
if (error == CPDFDocumentError.success) {
    bool importResult = await
document.importWidgets('data/your_package_name/files/xxx.xfdf');
    // or use Uri on the Android Platform.
    const xfdfFile = 'content://media/external/file/1000045118';
    const importResult = await document.importWidgets(xfdfFile);
}
```

## 导出表单

导出表单的代码如下：

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreate 回调中获取 controller
CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(),
    onCreate: (controller) {
        setState(() {
            this._controller = controller;
        });
    },
)
const exportPath = await _controller.document.exportWidgets();
```

使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(pdfFilePath);
if (error == CPDFDocumentError.success) {
    const exportPath = await document.exportWidgets();
}
```

## 什么是XFDF

XFDF (XML Forms Data Format) 是一种用于描述和传输 PDF 表单数据的 XML 格式。它通常与 PDF 文件一起使用，用于存储和传递表单字段的值、状态和操作。

XFDF 文件包含了对应 PDF 表单的数据，其中包括表单字段的名称、值、选项、格式等。

XFDF 是一种用于描述表单数据的格式，并不包含 PDF 文件本身。它用于存储和传输表单数据，以便在不同系统和应用程序之间进行交互和共享。

## 3.5 文档编辑

## 3.5.1 概述

文档编辑功能提供了一系列的操作页面的能力，使用户能够控制文档结构，并调整文档的布局和格式，确保文档内容以合理有序的方式精准呈现。

### 指南

#### [插入页面](#)

向目标文档中插入空白页面或另一个文档中的页面。

#### [拆分页面](#)

将多页文档的一部分拆分成独立文档。

## 3.5.2 插入页面

插入空白页面，或插入其他 PDF 页面到目标页面中。

### 插入空白页面

以下是插入空白页面的示例代码：

#### 使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreated 回调中获取 controller
CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(),
  onCreated: (controller) {
    setState(() {
      this._controller = controller;
    });
  },
)

CPDFPageSize pageSize = CPDFPageSize.a4;
// 自定义页面尺寸
// CPDFPageSize.custom(500, 800);
bool insertResult = await _controller.document.insertBlankPage(pageIndex: 0, pageSize =
pageSize);
```

#### 使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(pdfFilePath);
if (error == CPDFDocumentError.success) {
    CPDFPageSize pageSize = CPDFPageSize.a4;
    // 自定义页面尺寸
    // CPDFPageSize.custom(500, 800);
    bool insertResult = await document.insertBlankPage(pageIndex: 0, pageSize = pageSize);
}
```

## 插入其他 PDF 页面

以下是插入其他 PDF 页面的示例代码：

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreated 回调中获取 controller
CPDFReaderWidget(
    document: documentPath,
    configuration: CPDFConfiguration(),
    onCreated: (controller) {
        setState(() {
            this._controller = controller;
        });
    },
)

final filePath = '/data/user/0/com.compdfkit.flutter.example/cache/temp/PDF_Document.pdf';
final pages = [0,1,2]; // 从文档导入的页码下标集合
final insertPosition = 0; // 文档导入的位置
final result = await _controller.document.importDocument(
    filePath: filePath,
    pages: pages,
    insertPosition: insertPosition,
);
```

使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(pdfFilePath);
if (error == CPDFDocumentError.success) {
  final filePath =
'./data/user/0/com.compdfkit.flutter.example/cache/temp/PDF_Document.pdf';
  final pages = [0,1,2]; // 从文档导入的页码下标集合
  final insertPosition = 0; // 文档导入的位置
  final result = await document.importDocument(
    filePath: filePath,
    pages: pages,
    insertPosition: insertPosition,
  );
}
```

### 3.5.3 拆分页面

拆分页面可以将当前文档指定的页码拆分重组为一份新的PDF文档，以下是拆分页面的步骤：

1. 指定要拆分的页码
2. 指定拆分后的页面PDF文档保存路径

使用 `CPDFReaderWidget`：

```
CPDFReaderWidgetController? _controller;
// 初始化 CPDFReaderWidget，并在 onCreated 回调中获取 controller
CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(),
  onCreated: (controller) {
    setState(() {
      this._controller = controller;
    });
  },
)

var pages = [0,1,2];
var savePath = './data/user/0/com.compdfkit.flutter.example/cache/temp/PDF_Document.pdf';
// 在 Android 上，您可以使用 ComPDFKit.createUri() 来创建 Uri
// var savePath = ComPDFKit.createUri('split_document.pdf');
bool splitResult = await _controller.document.splitDocumentPages(savePath, pages);
```

使用 `CPDFDocument`：

```
// 创建并打开文档
CPDFDocument document = await CPDFDocument.createInstance();
var error = await document.open(pdfFilePath);
if (error == CPDFDocumentError.success) {
    var pages = [0,1,2];
    var savePath = '/data/user/0/com.compdfkit.flutter.example/cache/temp/PDF_Document.pdf';
    // 在 Android 上，您可以使用 ComPDFKit.createUri() 来创建 Uri
    // var savePath = ComPDFKit.createUri('split_document.pdf');
    bool splitResult = await document.splitDocumentPages(savePath, pages);
}
```

## 3.6 安全

### 3.6.1 概述

安全模块提供了包括密码保护、权限设置、Bates 编码、背景以及页眉和页脚功能。通过管理文档密码与权限，并添加标志和版权信息，保障文档的安全性。

#### 主要功能

- **密码管理：** 设置、修改或删除文档密码和权限。
- **加密标准：** 支持标准的 PDF 安全处理器（40 位和 128 位 RC4 加密），以及 128 位和 256 位 AES（高级加密标准）加密。

#### 指南

##### 设置密码

通过管理文档密码和权限设置，防止未经授权的访问，并可控制用户对文档的操作。

##### 删除密码

通过 API 移除文档的密码和权限。

### 3.6.2 设置密码

PDF 权限模块通过提供加密、文档权限、解密和密码移除功能来确保文档安全性，允许用户对文档进行安全的控制和管理。

#### 加密

加密功能由两部分组成：用户密码和所有者密码。

- **用户密码** 用于打开文档，确保只有授权用户可以访问内容。当设置用户密码时，通常会限制某些文档权限，例如修改、复制或打印。
- **所有者密码** 不仅可以打开文档，还可以解锁所有受限权限，允许用户修改、复制或打印文档。

这种双重密码系统旨在提供更加灵活且安全的文档访问与管理方式。

ComPDFKit 提供多种加密算法与权限设置。根据需求，您可以选择适当的算法并配置自定义权限以保护文档。

加密步骤如下：

1. 设置不同的用户密码和所有者密码。
2. 创建权限信息类。
3. 指定加密算法。
4. 使用用户密码、所有者密码、权限信息和指定的算法加密文档。
5. 保存文档。

以下示例展示了如何加密文档：

```
CPDFReaderWidgetController? _controller;

@Override
Widget build(BuildContext context) {
    return Scaffold(
        resizeToAvoidBottomInset: false,
        appBar: AppBar(),
        body: Column(children: [
            TextButton(onPressed: () async {
                // 设置密码
                bool setPasswordResult = await _controller.document.setPassword(
                    userPassword: '1234',
                    ownerPassword: '12345',
                    allowsPrinting: false,
                    allowsCopying: false,
                    encryptAlgo: CPDFDocumentEncryptAlgo.aes256);
                debugPrint('ComPDFKit-Flutter: setPasswordResult:$setPasswordResult');
                // 保存 PDF
                await _controller.document.save();
            }, child: const Text('加密 PDF')),
            Expanded(child: CPDFReaderWidget(
                document: widget.filePath,
                configuration: CPDFConfiguration(),
                onCreated: (controller) {
                    setState(() {
                        _controller = controller;
                    });
                },
            )));
        ],
    );
}
```

## 不同加密算法

加密算法	描述	枚举值
No Encrypt Algo	不进行加密	CPDFDocumentEncryptAlgo.noEncryptAlgo
RC4	基于密钥的异或加密明文	CPDFDocumentEncryptAlgo.rc4
AES-128	使用 128 位密钥的 AES 加密	CPDFDocumentEncryptAlgo.aes128
AES-256	使用 256 位密钥的 AES 加密	CPDFDocumentEncryptAlgo.aes256

### 3.6.3 删 除 密 码

删除密码是指拥有所有者权限的用户移除用户密码和所有者密码，使文档无需密码即可打开，并默认拥有完全访问权限。

删除密码的步骤：

1. 使用 `CPDFReaderWidget` 组件打开文档。
2. 解锁文档并获取完全权限。
3. 从解锁的文档中移除密码。

以下示例展示了如何删除密码：

```
late CPDFReaderWidgetController _controller;

@Override
Widget build(BuildContext context) {
    return Scaffold(
        resizeToAvoidBottomInset: false,
        appBar: AppBar(),
        body: Column(children: [
            TextButton(onPressed: () async {
                // 删除密码
                bool removePasswordResult = await _controller.document.removePassword();
            }, child: const Text('删除密码')),
            Expanded(child: CPDFReaderWidget(
                document: widget.filePath,
                configuration: CPDFConfiguration(),
                onCreated: (controller) {
                    setState(() {
                        _controller = controller;
                    });
                },
            )));
        ],
    );
}
```

检查文档是否已加密：

```
bool isEncrypted = await controller.isEncrypted();
```

获取文档权限状态：

```
CPDFDocumentPermissions permissions = await controller.getPermissions();
```

权限状态说明：

权限状态	描述	枚举值
无权限	文档未应用任何权限	CPDFDocumentPermissions.none
用户权限	使用用户密码打开文档，可能限制打印和复制等操作	CPDFDocumentPermissions.user
所有者权限	使用所有者密码打开文档，无任何限制	CPDFDocumentPermissions.owner

检查所有者权限：

```
// 检查当前文档是否通过所有者权限解锁  
bool unlocked = await controller.checkOwnerUnlocked();  
  
// 检查所有者密码是否正确  
bool result = await controller.checkOwnerPassword('owner_password');
```

## 3.7 水印

### 3.7.1 概述

水印是添加在 PDF 文档上的标记，通常以半透明文本或图像的形式出现。通过添加水印，可以突出显示文档的所有权或其他相关信息，同时不影响文档的可读性。

#### ComPDFKit 水印的优势

- 版权标记：**通过水印显示文档来源和版权信息，防止截图或不当使用。
- 品牌宣传：**自定义水印样式以展示品牌标志或其他预设内容。
- 快速 UI 集成：**使用可扩展的 UI 组件轻松集成和自定义水印。

#### ComPDFKit 水印功能

- 添加文本水印**  
使用 API 接口添加自定义文本水印，或通过弹窗交互方式设置文本水印。
- 添加图片水印**  
使用 API 接口添加图片水印，或通过弹窗交互方式选择和配置图片水印。
- 移除水印**  
从 PDF 文档中移除水印。

## 3.7.2 添加文本水印

ComPDFKit 支持 **两种方式** 添加文本水印：

### 1. 编程方式（API 调用）

通过 `document.createWatermark` 创建文本水印，适合需要在代码中自动生成水印的场景。

```
late CPDFReaderWidgetController _controller;

@Override
Widget build(BuildContext context) {
    return Scaffold(
        resizeToAvoidBottomInset: false,
        appBar: AppBar(),
        body: Column(children: [
            TextButton(onPressed: () async {
                await _controller.document.createWatermark(CPDFWatermark.text(
                    textContent: 'ComPDFKit',
                    scale: 1.0,
                    fontSize: 60,
                    rotation: 0,
                    horizontalAlignment: CPDFWatermarkHorizontalAlignment.center,
                    verticalAlignment: CPDFWatermarkVerticalAlignment.center,
                    textColor: Colors.red,
                    pages: [0, 1, 2, 3]));
            }, child: const Text('添加文本水印')),
            Expanded(child: CPDFReaderWidget(
                document: widget.filePath,
                configuration: CPDFConfiguration(),
                onCreated: (controller) {
                    setState(() {
                        _controller = controller;
                    });
                },
            )),
        ],
    );
}
```

### 2. UI 弹窗方式

- 在 `CPDFConfiguration` 中配置水印默认样式，例如默认文本、颜色、字体大小。
- 调用 `controller.showAddWatermarkView()` 打开添加水印的弹窗界面。

```
// 在配置中设置默认文本水印样式
final configuration = CPDFConfiguration(
    globalConfig: const CPDFGlobalConfig(
        watermark: CPDFWatermarkConfig(
            text: 'ComPDFKit-Flutter',
            textColor: Colors.red,
            textSize: 36,
```

```

        opacity: 120,
        rotation: -45
    )
)
);

// 打开添加水印弹窗
controller.showAddWatermarkView(
    config:
    CPDFGlobalConfig(
        watermark: CPDFWatermarkConfig(
            text: 'ComPDFKit-Flutter',
            textColor: Colors.red,
            textSize: 36,
            opacity: 120,
            rotation: -45
        )
    )
);

```

### 3.7.3 添加图片水印

ComPDFKit 支持 **两种方式** 添加图片水印

#### 1. 编程方式 (API 调用)

```

late CPDFReaderWidgetController _controller;

@Override
Widget build(BuildContext context) {
    return Scaffold(
        resizeToAvoidBottomInset: false,
        appBar: AppBar(),
        body: Column(children: [
            TextButton(onPressed: () async {
                // 图像文件路径
                // 安卓平台支持 Uri 格式。
                // content://xxxx
                File imageFile = await extractAsset(context, 'images/logo.png');
                await controller.document.createWatermark(CPDFWatermark.image(
                    imagePath: imageFile.path,
                    opacity: 1,
                    rotation: 45,
                    pages: [0, 1, 2, 3],
                    horizontalAlignment: CPDFWatermarkHorizontalAlignment.center,
                    verticalAlignment: CPDFWatermarkVerticalAlignment.center,
                )));
            }, child: const Text('添加图片水印')),
            Expanded(child: CPDFReaderWidget(
                document: widget.filePath,
                configuration: CPDFConfiguration(),

```

```
    onCreated: (controller) {
      setState(() {
        _controller = controller;
      });
    },
  )),
],
));
}
```

## 2. UI 弹窗方式

- 在 CPDFConfiguration 中配置水印默认样式，例如图片、透明度、旋转角度。
- 调用 controller.showAddWatermarkView(CPDFWatermarkConfig config) 打开添加水印的弹窗界面。

```
// 在配置中设置默认图片水印样式
final configuration = CPDFConfiguration(
  globalConfig: const CPDFGlobalConfig(
    watermark: CPDFWatermarkConfig(
      image: 'ic_logo'
      opacity: 120,
      rotation: -45
    )
  )
);

// 打开添加水印弹窗
controller.showAddWatermarkView(
  config:
  CPDFGlobalConfig(
    watermark: CPDFWatermarkConfig(
      image: 'ic_logo'
      opacity: 120,
      rotation: -45
    )
  )
);
);
```

要自定义默认的图片，您需要在Android和iOS中以不同的方式实现它们。

### Android

1. 将图片文件添加到资源文件中

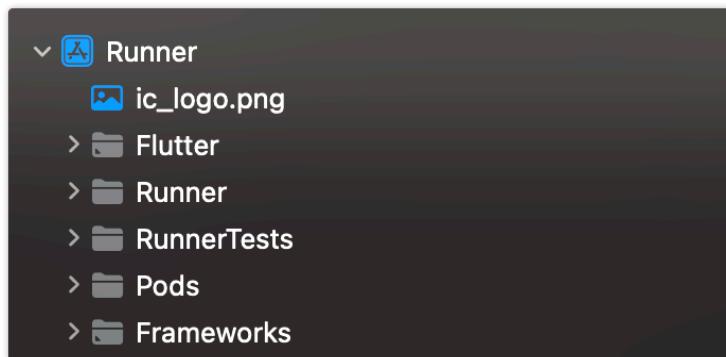
```
android/app/src/main/res/drawable/ic_logo.png
```

2. 使用不带扩展名的文件名在代码中引用它们：

```
image: 'ic_logo'
```

### iOS

- 将图片文件通过Xcode导入到项目中



- 使用不带扩展名的文件名在代码中引用它们：

```
image: 'ic_logo'
```

## Flutter 跨平台方案

也可以直接将图片资源放入 Flutter 项目中，然后在运行时复制到设备本地存储，再传入图片的文件路径：

```
final imagePath = await extractAsset(context, 'images/ic_logo.png');
await controller.showAddWatermarkView(
  config: CPDFWatermarkConfig(
    image: imagePath.path,
    rotation: -45,
    opacity: 200,
    scale: 2.0
),
);
```

## 3.7.4 移除水印

使用以下方法移除水印：

```
// 从文档中移除所有水印
document.removeAllWatermarks();
```

## 3.8 自定义UI

### 3.8.1 概述

#### 在 Flutter 中自定义 PDF 查看器

ComPDFKit PDF SDK for Flutter 允许您通过隐藏按钮或工具栏来自定义应用程序的用户界面。

有关详细的可配置选项，请参阅 [CONFIGURATION.md](#)。

## 主要功能

- 工具栏 - 隐藏或添加功能按钮。
- 打开页面进行编辑、添加水印、配置安全设置、捕获屏幕截图和其他功能视图。
- 国际化和本地化

## 自定义 UI 的指南

### 主工具栏

如何显示或隐藏工具栏按钮

### 注释工具栏

如何自定义注释工具栏

### 内容编辑工具栏

如何自定义内容编辑工具栏

### 表单工具栏

如何自定义表单工具栏

### 签名工具栏

如何自定义签名工具栏

## 国际化与本地化

如何更改 Flutter 版 ComPDFKit PDF SDK 中的语言

### 工具

工具集包括页面编辑对话框、水印添加对话框、安全设置等相关 API。用户可以通过 `CPDFReaderWidgetController` 直接调用并打开相应功能，以在 Flutter 中实现这些功能。

### 上下文菜单

用于配置选择评论、文本、图片等时弹出的上下文菜单选项。

### UI 可见性

如何显示或隐藏用户界面。

### BOTA界面配置

如何通过配置显示隐藏BOTA界面功能与菜单选项。

### 页面编辑菜单

如何配置页面编辑界面底部菜单选项。

## 3.8.2 主工具栏

ComPDFKit 的主工具栏设计灵活且高度可配置。本指南展示如何自定义主工具栏。

### 默认工具栏

默认工具栏包含以下工具：

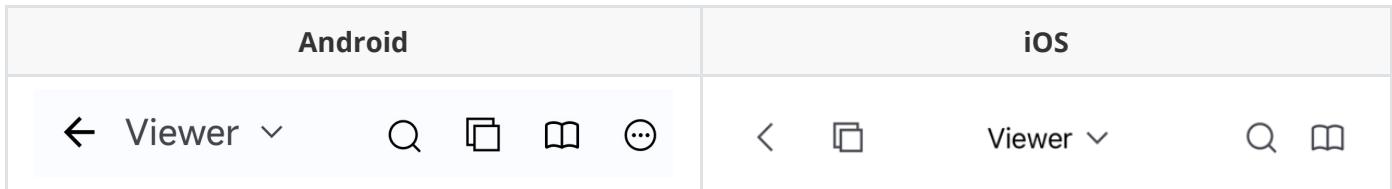
Android	iOS
Viewer ▾    	<  Viewer ▾   

## 自定义工具栏按钮

您可以在显示 PDF 时使用 iOS 的 `iosLeftBarAvailableActions` 或 `iosRightBarAvailableActions` 属性和 Android 的 `androidAvailableActions` 属性自定义主工具栏按钮。以下示例展示了如何在 iOS 中隐藏导航栏（主工具栏）中的菜单按钮，以及如何自定义 Android 工具栏菜单项：

```
CPDFConfiguration configuration = CPDFConfiguration(toolbarConfig: const
CPDFToolbarConfig(
  mainToolbarVisible: true,
  androidAvailableActions: [
    ToolbarAction.back,
    ToolbarAction.search,
    ToolbarAction.thumbnail,
    ToolbarAction.bota,
    ToolbarAction.menu,
  ],
  iosLeftBarAvailableActions: [
    ToolbarAction.back,
    ToolbarAction.thumbnail
  ],
  iosRightBarAvailableActions: [
    ToolbarAction.search,
    ToolbarAction.bota,
  ],
));
// CPDFReaderWidget 示例
Scaffold(
  resizeToAvoidBottomInset: false,
  appBar: AppBar(),
  body: CPDFReaderWidget(
    document: documentPath,
    configuration: configuration,
    onCreated: (controller) {},
  ));
// ComPDFKit.openDocument 示例
ComPDFKit.openDocument(documentPath, '', configuration)
```

自定义工具栏将如下所示：



### 可用的工具栏自定义选项

工具栏按钮项	描述
back	显示关闭按钮项。
thumbnail	显示缩略图按钮项。
search	显示搜索按钮项。
bota	显示大纲、书签、注释列表按钮项。
menu	显示菜单按钮项。

注意：请参考 `ToolbarAction` 获取相关选项。

### 可用的菜单自定义选项

如果配置了 `ToolbarAction.menu`，您可以在菜单中访问更多功能按钮。有关可配置选项，请参阅以下列表：

菜单按钮项	描述
viewSettings	打开设置视图并设置滚动方向、显示模式、主题颜色等相关设置。
documentEditor	打开文档缩略图列表，可以在视图中删除、旋转和添加文档页面。
documentInfo	打开文档信息视图以显示基本文档信息和权限信息。
watermark	打开水印编辑视图以添加文本和图像水印，并将其保存为新文档。
security	打开安全设置视图，设置文档打开密码和权限密码。
flattened	将文档中的注释扁平化，注释将不可编辑。
save	保存 PDF 文档。
share	打开系统共享功能。
openDocument	打开系统文件选择器并打开新 PDF 文档。

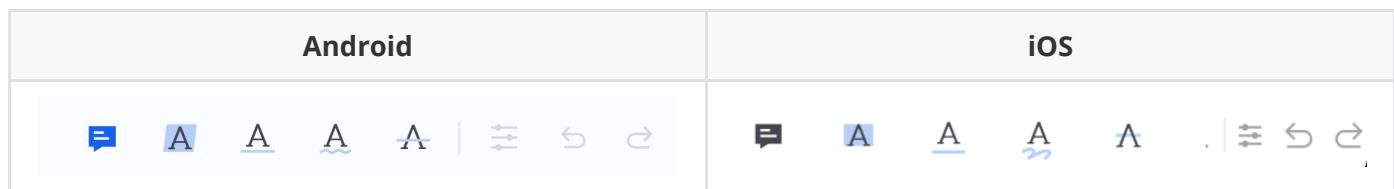
注意：请参考 `ToolbarMenuAction` 获取相关选项。

## 3.8.3 注释工具栏

在 ComPDFKit 中，注释工具栏可以灵活配置，使用户能够选择注释类型和工具。本节将介绍如何自定义注释工具栏。

## 默认注释工具栏

默认的注释工具栏包含以下注释类型和工具：

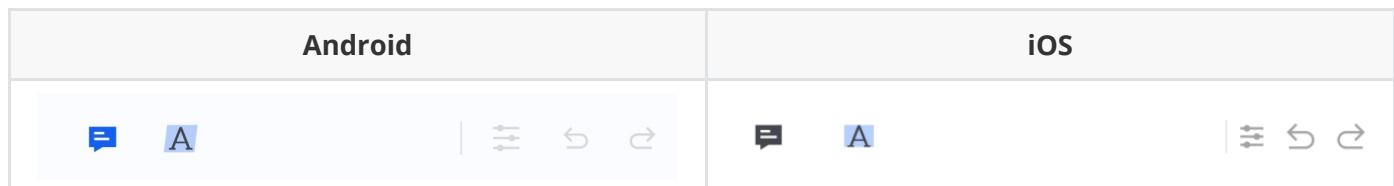


## 自定义注释工具栏按钮

您可以通过在 `annotationsConfig` 对象中设置 `availableTypes` 属性来启用或隐藏特定的注释类型。以下示例演示了如何只显示注释和高亮注释类型。

```
CPDFConfiguration configuration = CPDFConfiguration(  
    annotationsConfig: const CPDFAnnotationsConfig(availableTypes: [  
        CPDFAnnotationType.note,  
        CPDFAnnotationType.highlight  
    ], availableTools: [  
        CPDFConfigTool.setting,  
        CPDFConfigTool.undo,  
        CPDFConfigTool.redo,  
    ]);  
  
// CPDFReaderWidget 示例  
Scaffold(  
    resizeToAvoidBottomInset: false,  
    appBar: AppBar(),  
    body: CPDFReaderWidget(  
        document: documentPath,  
        configuration: configuration,  
        onCreated: (controller) {},  
    ));  
  
// ComPDFKit.openDocument 示例  
ComPDFKit.openDocument(documentPath, '', configuration)
```

自定义的工具栏将如下所示。



## 可用的注释工具栏自定义选项

Type					
note (注释)	highlight (高亮)	underline (下划线)	squiggly (波浪线)	strikeout (删除线)	ink (墨迹)
pencil (铅笔)	circle (圆圈)	square (方框)	arrow (箭头)	line (直线)	freetext (文本)
signature (签名)	stamp (图章)	pictures (图片)	link (链接)	sound (声音)	

注意：请参考 `CPDFAnnotationType` 获取相关选项。`pencil` 只在 iOS 上可用。

#### 可用的注释工具栏工具自定义选项

工具	描述
setting	设置按钮，对应打开所选注释、文本或图片属性面板。
undo	撤销注释、内容编辑、表单操作。
redo	重做被撤销的操作。

注意：请参考 `CPDFConfigTool` 获取相关选项。

#### 显示或隐藏工具栏

可以通过配置 `CPDFConfiguration` 来控制注释模式下底部工具栏的显示状态。

```
CPDFConfiguration configuration = CPDFConfiguration(
    // 设置注释模式底部工具栏是否可见
    toolbarConfig: const CPDFToolbarConfig(annotationToolbarVisible: false),
);
```

### 3.8.4 内容编辑工具栏

在 ComPDFKit 中，内容编辑工具栏可以灵活配置，使用户能够选择编辑类型和工具。本节将介绍如何自定义内容编辑工具栏。

#### 默认内容编辑工具栏

默认的内容编辑工具栏包含以下编辑类型和工具：

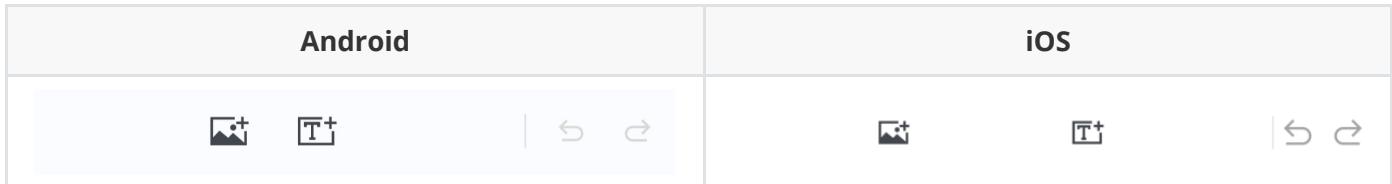
Android	iOS
    	    

## 自定义内容编辑工具栏按钮

您可以通过在 `contentEditorConfig` 对象中设置 `availableTypes` 属性来启用或隐藏特定的编辑类型。以下示例演示了如何调整编辑类型的顺序，并隐藏内容编辑工具的设置按钮。

```
CPDFConfiguration configuration = CPDFConfiguration(  
    contentEditorConfig: const CPDFContentEditorConfig(availableTypes: [  
        CPDFContentEditorType.editorImage,  
        CPDFContentEditorType.editorText  
    ], availableTools: [  
        CPDFConfigTool.undo,  
        CPDFConfigTool.redo,  
    ]));  
  
// CPDFReaderWidget 示例  
Scaffold(  
    resizeToAvoidBottomInset: false,  
    appBar: AppBar(),  
    body: CPDFReaderWidget(  
        document: documentPath,  
        configuration: configuration,  
        onCreated: (controller) {},  
    ));  
  
// ComPDFKit.openDocument 示例  
ComPDFKit.openDocument(documentPath, '', configuration)
```

自定义的工具栏将如下所示。



## 可用的内容编辑工具栏自定义选项

类型
CPDFContentEditorType.editorText
CPDFContentEditorType.editorImage

注意：请参考 `CPDFContentEditorType` 获取相关选项。

## 可用的内容编辑工具栏工具自定义选项

工具	描述
setting	设置按钮，对应打开所选注释、文本或图片属性面板。
undo	撤销注释、内容编辑、表单操作。
redo	重做被撤销的操作。

注意：请参考 `CPDFConfigTool` 获取相关选项。

## 显示或隐藏工具栏

可以通过配置 `CPDFConfiguration` 来控制内容编辑模式下底部工具栏的显示状态。

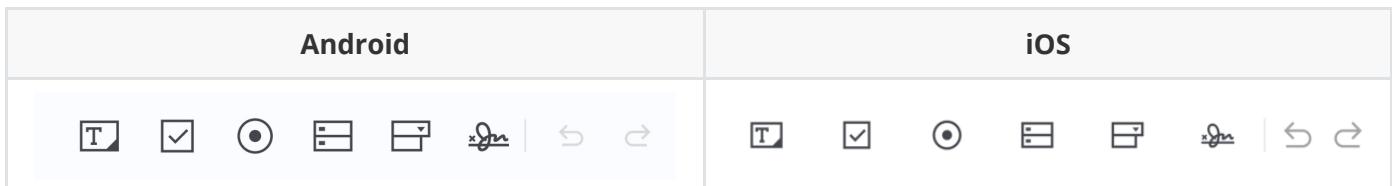
```
CPDFConfiguration configuration = CPDFConfiguration(  
    // 设置内容编辑模式底部工具栏是否可见  
    toolbarConfig: const CPDFToolbarConfig(contentEditorToolbarVisible: false),  
);
```

## 3.8.5 表单工具栏

在 ComPDFKit 中，表单工具栏可以灵活配置，使用户能够选择表单类型和工具。本节将介绍如何自定义表单工具栏。

### 默认表单工具栏

默认的表单工具栏包含以下表单类型和工具：



### 自定义表单工具栏按钮

您可以通过在 `formsConfig` 对象中设置 `availableTypes` 属性来启用或隐藏特定的表单类型。以下示例演示了如何调整表单类型，仅启用文本字段和列表框。

```
CPDFConfiguration configuration = CPDFConfiguration(  
    formsConfig: const CPDFFormsConfig(availableTypes: [  
        CPDFFormType.textField,  
        CPDFFormType.listBox  
    ], availableTools: [  
        CPDFFormConfigTool.undo,  
        CPDFFormConfigTool.redo  
    ]));  
  
// CPDFReaderWidget 示例  
Scaffold(
```

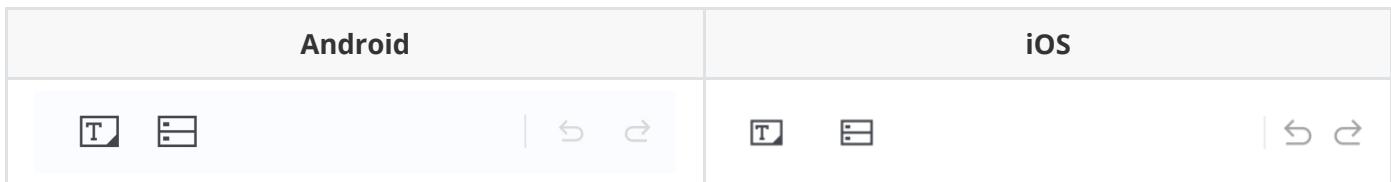
```

resizeToAvoidBottomInset: false,
appBar: AppBar(),
body: CPDFReaderWidget(
  document: documentPath,
  configuration: configuration,
  onCreated: (controller) {},
),
);

// ComPDFKit.openDocument 示例
ComPDFKit.openDocument(documentPath, '', configuration)

```

自定义的工具栏将如下所示。



#### 可用的表单工具栏自定义选项

类型
CPDFFormType.textField
CPDFFormType.checkBox
CPDFFormType.radioButton
CPDFFormType.listBox
CPDFFormType.comboBox
CPDFFormType.signaturesFields
CPDFFormType.pushButton

注意：请参考 `CPDFFormType` 获取相关选项。

#### 可用的表单工具栏工具自定义选项

工具	描述
SETTING	设置按钮，对应打开所选注释、文本或图片属性面板。
UNDO	撤销注释、内容编辑、表单操作。
REDO	重做被撤销的操作。

注意：请参考 `CPDFFormConfigTool` 获取相关选项。

#### 显示或隐藏工具栏

可以通过配置 `CPDFConfiguration` 来控制表单模式下底部工具栏的显示状态。

```
CPDFConfiguration configuration = CPDFConfiguration(  
    // 设置表单模式底部工具栏是否可见  
    toolbarConfig: const CPDFToolbarConfig(formToolbarVisible: false),  
);
```

## 3.8.6 签名工具栏

显示或隐藏工具栏

可以通过配置 `CPDFConfiguration` 来控制签名模式下底部工具栏的显示状态。

```
CPDFConfiguration configuration = CPDFConfiguration(  
    // 设置签名模式底部工具栏是否可见  
    toolbarConfig: const CPDFToolbarConfig(signatureToolbarVisible: false),  
);
```

## 3.8.7 国际化与本地化

ComPDFKit Flutter SDK 默认支持以下语言：

- 英语 (en)
- 简体中文 (zh-Hans)
- 西班牙语(es)

为 ComPDFKit 添加额外的本地化支持

- Android

您可以通过将翻译文件放入 `res/strings-XX` 目录中来为 ComPDFKit 添加额外的语言资源。Android 会在构建时自动合并所有字符串资源。您还可以通过将翻译文件放入相应的语言文件夹中来覆盖 ComPDFKit 已存在的字符串。

提示：要查看所有可用的 ComPDFKit 字符串资源，可以从以下链接将资源复制到您的项目中：

[strings.xml](#)

以下以添加法语支持为例：

1. 在您的项目中创建 `app/src/main/res/values-fr` 目录，并在该目录下创建 `strings.xml` 文件。
2. 将上述链接中的内容复制到 `strings.xml` 文件中。

The screenshot shows the Android Studio interface. On the left, the project structure is displayed under the 'app' module. On the right, the content of the 'fr/strings.xml' file is shown.

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools" tools:ignore="Ext">
    <string name="tools_document_info">Document Info</string>
    <string name="tools_view_setting">View Setting</string>
    <string name="tools_share">Share</string>
    <string name="tools_open_document">Open...</string>
    <string name="tools_flattened">Save as Flattened PDF</string>
    <string name="tools_display_mode">Display Mode</string>
    <string name="tools_scroll">Scroll</string>
    <string name="tools_vertical_scrolling">Vertical Scrolling</string>
    <string name="tools_horizontal_scrolling">Horizontal Scrolling</string>
    <string name="tools_single_page">Single Page</string>
    <string name="tools_two_page">Two Page</string>
    <string name="tools_book_mode">Cover Mode</string>
    <string name="tools_continuous_scroll">Continuous Scrolling</string>
    <string name="tools_crop">Crop</string>
    <string name="tools_themes">Themes</string>
    <string name="tools_light_mode">Light</string>
    <string name="tools_dark_mode">Dark</string>
    <string name="tools_sepia_mode">Sepia</string>
    <string name="tools_reseda_mode">Reseda</string>
```

1. 将文件中的字符串翻译成法语，例如：

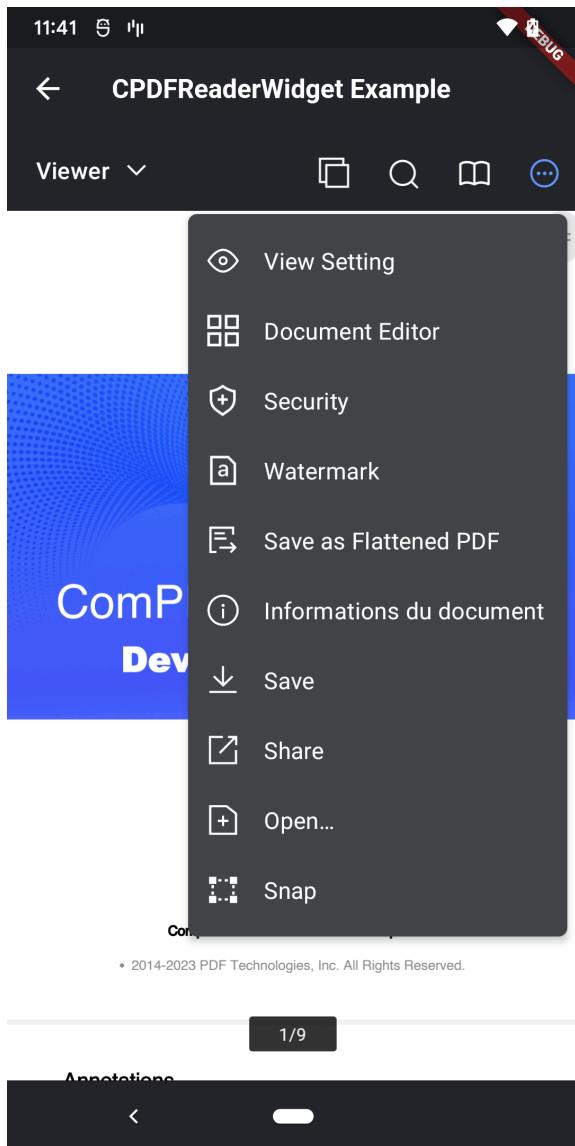
原始内容：

```
<string name="tools_document_info">Document Info</string>
```

翻译为法语并替换原内容：

```
<string name="tools_document_info">Informations du document</string>
```

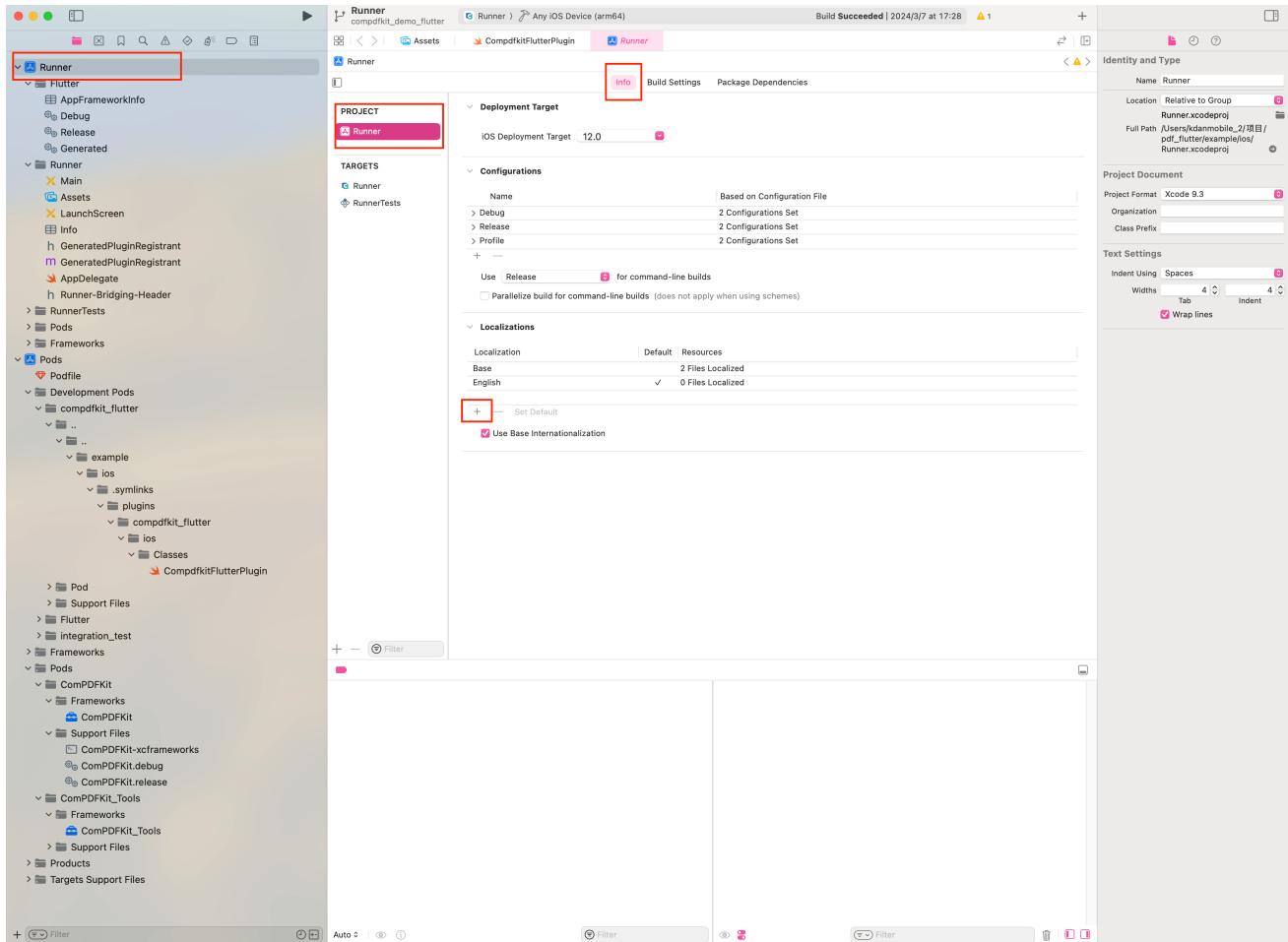
1. 将您的设备切换到法语，运行项目后，您将看到翻译后的內容：



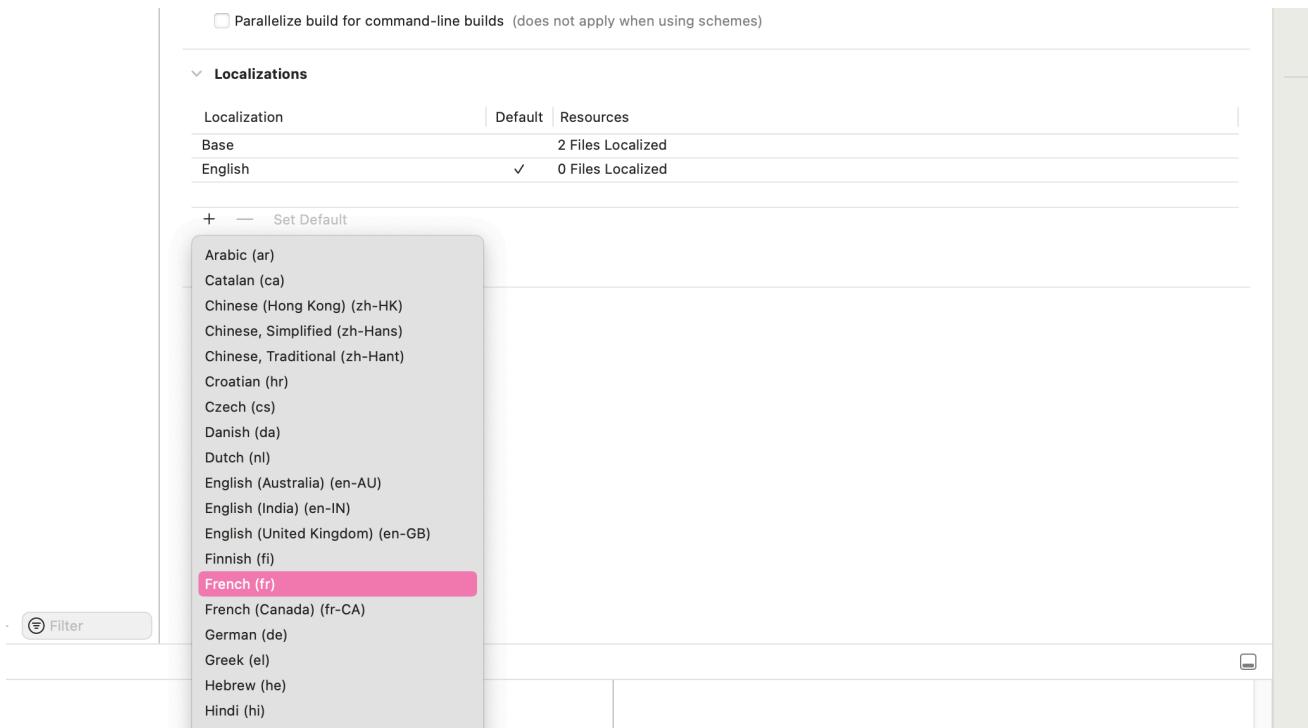
- iOS

### 配置需要国际化的语言

1. 使用 Xcode 打开你的 Flutter 项目的 iOS 部分。然后，选择项目 -> Info -> Localizations，点击 '+', 添加你想要国际化/本地化的语言，如下图所示（确保默认勾选 'Use Base Internationalization'）：

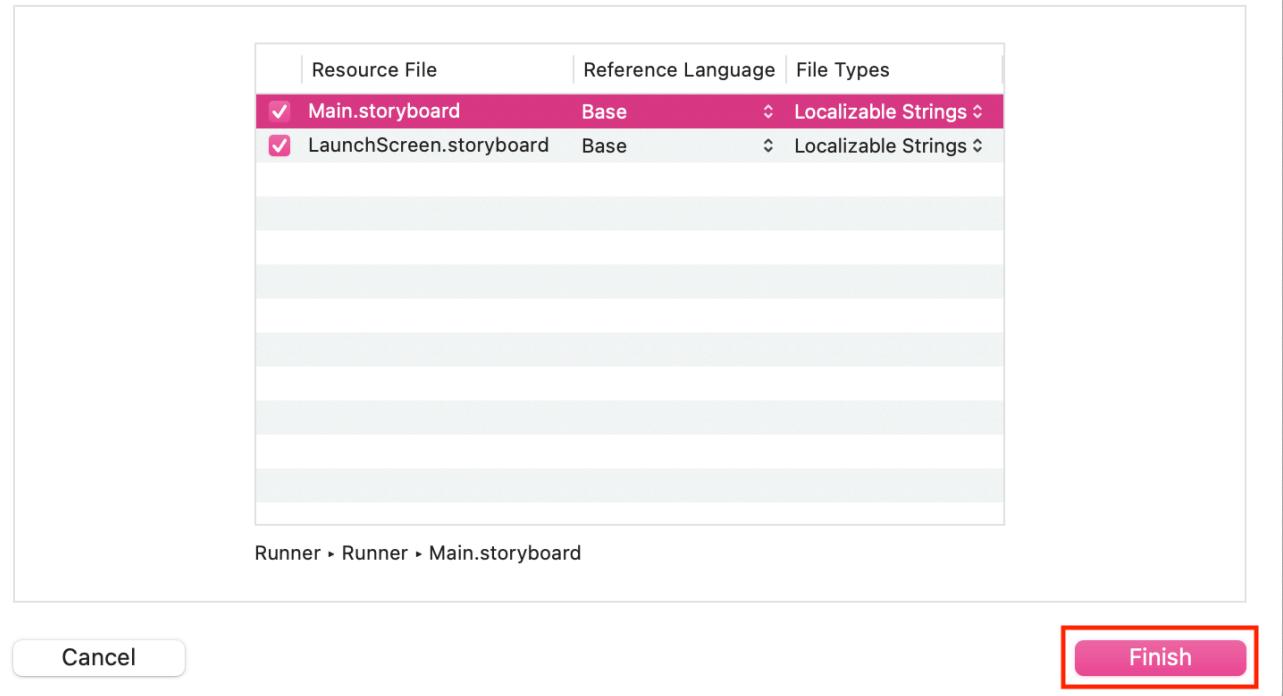


## 2. 这里我们以添加法语为例，如下图所示：



## 3. 弹出以下对话框，直接点击 'Finish'，如下图所示：

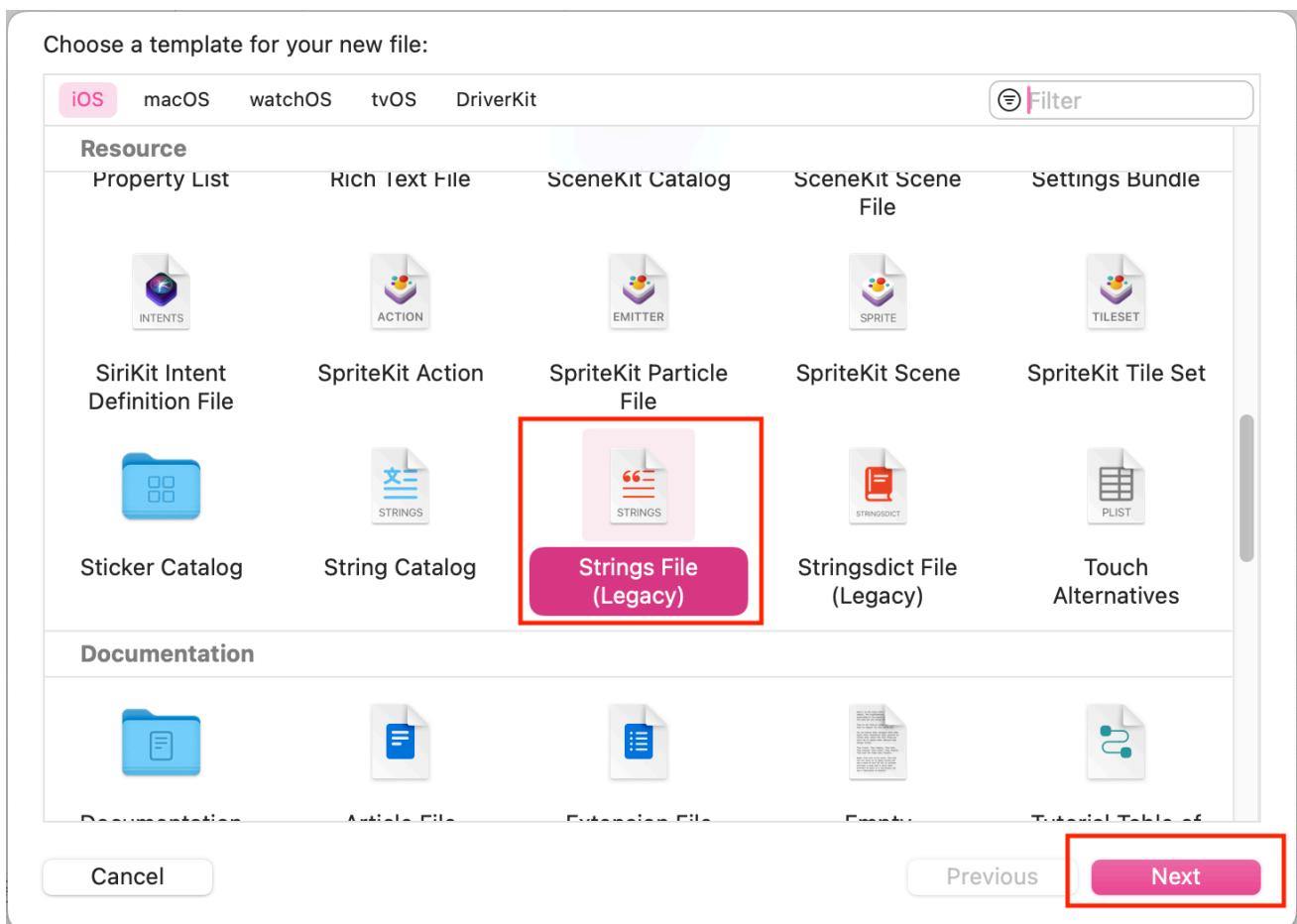
Choose files and reference language to create French localization



## 应用程序名称的国际化

应用程序名称的国际化是指在不同语言环境（即，移动设备上的语言设置）中显示相同应用程序的不同名称。

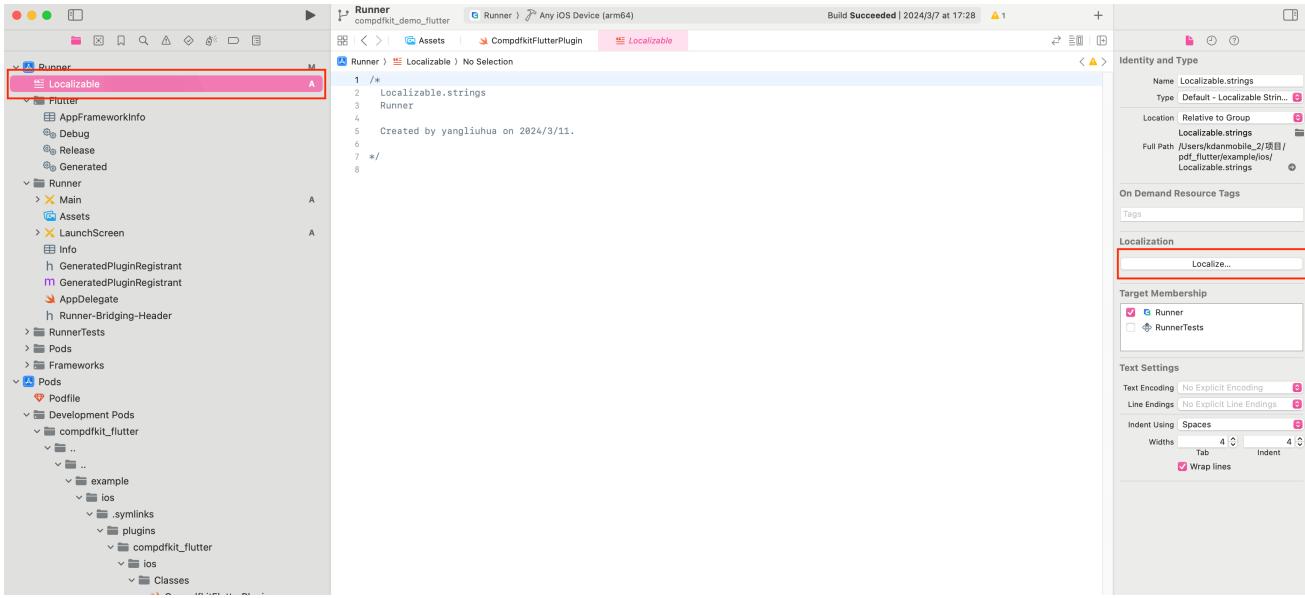
### 1. 创建 Localizable.strings 文件



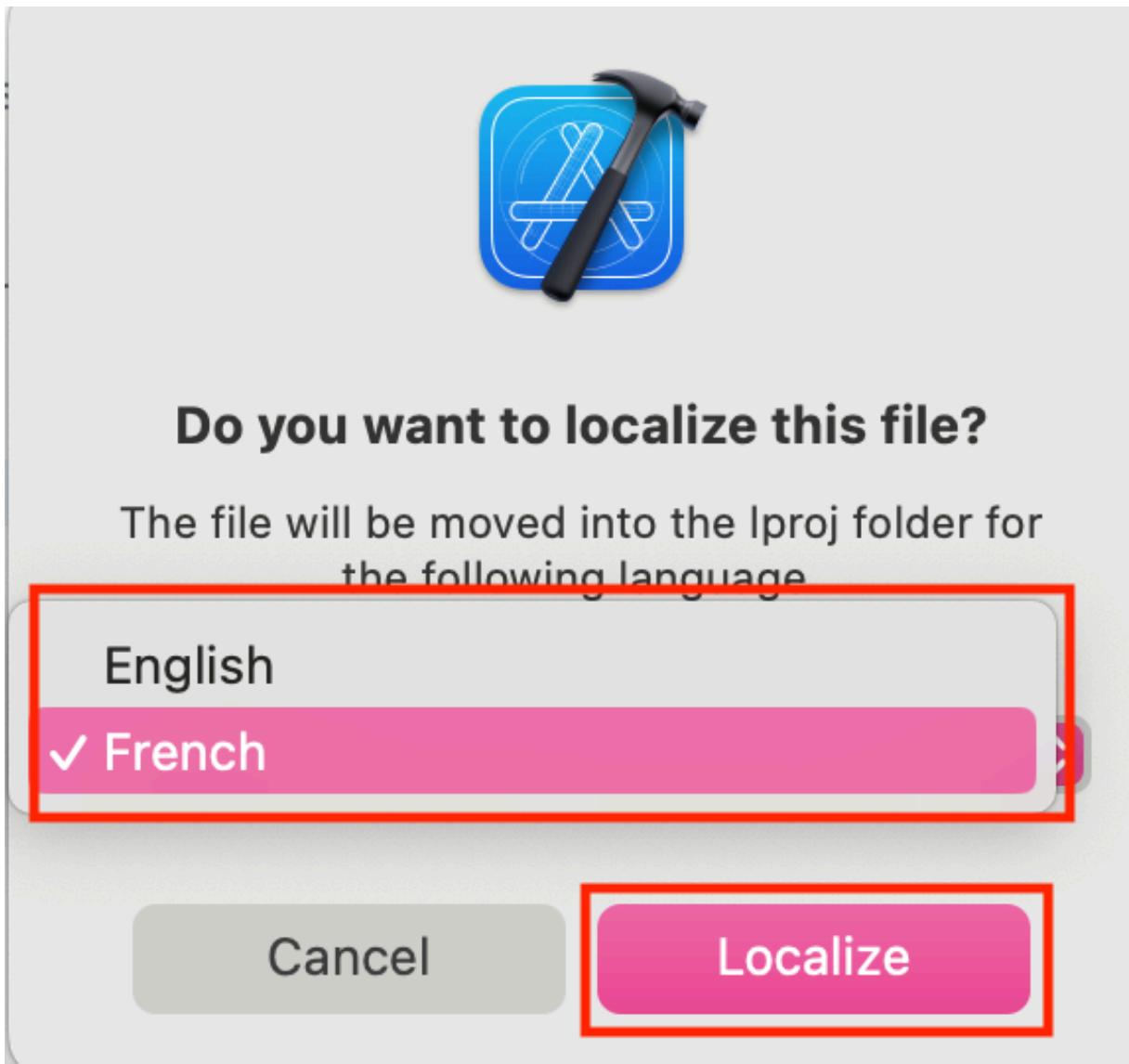
### 2. 选择 Localizable.strings，在 Xcode 的文件检查器（右侧文件检查器）中点击 Localize。目的是选择我

我们要本地化的语言，如下图所示：

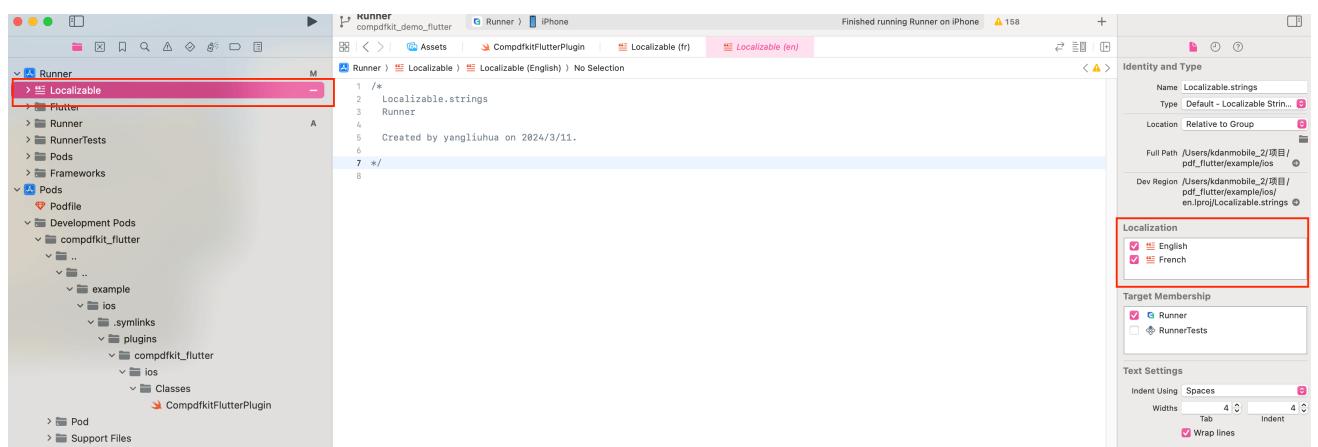
注意：在点击 Localize 之前，确保我们已经添加了本地化语言。这一步是我们配置国际化语言的步骤（步骤：项目->Info->Localizations，然后点击 '+', 添加你想要国际化/本地化的语言）。



3. 点击 Localize 后，会出现一个对话框。展开对话框列表，你会发现下拉列表显示了我们之前配置的国际化语言。选择我们要本地化的语言，然后点击对话框中的 'Localize' 按钮，如下图所示：



4. 接下来，勾选法语和英语，如下图所示：



5. 然后打开 ComPDFKit Flutter iOS Demo，找到 Localizable (English) 文件，全选文本，复制后粘贴到你新建的 Localizable (English) 文件中。

```

1 /*
2  Localizable.strings
3  PDFViewer-Swift
4 */
5 Localizable.strings
6 PDFViewer
7
8 // Copyright © 2014-2025 PDF Technologies, Inc. All Rights Reserved.
9 //
10 // THIS SOURCE CODE AND ANY ACCOMPANYING DOCUMENTATION ARE PROTECTED BY INTERNATIONAL COPYRIGHT LAW
11 // AND MAY NOT BE RESOLD OR REDISTRIBUTED. USAGE IS BOUND TO THE ComPDFKit LICENSE AGREEMENT.
12 // UNAUTHORIZED REPRODUCTION OR DISTRIBUTION IS SUBJECT TO CIVIL AND CRIMINAL PENALTIES.
13 // This notice may not be removed from this file.
14 //
15
16 */
17
18 "Close" = "Close";
19 "Settings" = "Settings";
20 "Highlight Links" = "Highlight Links";
21 "Highlight Form Fields" = "Highlight Form Fields";
22 "SDK Information" = "SDK Information";
23 "ComPDFKit" = "ComPDFKit";
24 "Annotator" = "Annotator";
25 "The fonts data is saved with the file" = "The fonts data is saved with the file";
26 "SDK Information" = "SDK Information";
27 "Version" = "Version";
28 "v 1.11.0" = "v 1.11.0";
29 "Company Information" = "Company Information";
30 "https://www.compdf.com/" = "https://www.compdf.com/";
31 "https://www.compdf.com/contact-sales" = "https://www.compdf.com/contact-sales";
32 "https://www.compdf.com/privacy-policy" = "https://www.compdf.com/privacy-policy";
33 "https://www.compdf.com/terms-of-service" = "https://www.compdf.com/terms-of-service";
34 "About ComPDFKit" = "About ComPDFKit";
35 "Contact Sales" = "Contact Sales";
36 "Technical Support" = "Technical Support";
37 "support@compdf.com" = "support@compdf.com";
38 "Technical Supports" = "Technical Support";
39 "Mail account is not set up!" = "Mail account is not set up!";
40 /* 2014-2025 PDF Technologies, Inc. All Rights Reserved. */ = "2014-2025 PDF Technologies, Inc. All Rights Reserved.";
41 "Privacy Policy" = "Privacy Policy";
42 "Service Terms" = "Service Terms";
43 "Click to Open & Process" = "Click to Open & Process";
44 "Features" = "Features";
45 "Closes" = "Closes";
46 "OKs" = "OKs";
47 "Update Config" = "Update Config";

```

6. 最后, 选择 Localizable (French) 文件, 并根据 Localizable (English) 文件配置相对应的法语翻译, 确保两个文件的文本段数量相同。例如, 如果 Localizable (English) 文件中有 "Viewer" = "Viewer";, 那么 Localizable (French) 文件中应该有相应的 "Viewer" = "Téléspectatrice";。

```

1 /*
2  Localizable.strings
3  Runner
4 */
5 Created by yangliuhua on 2024/3/11.
6
7 */
8
9 "Viewer" = "Téléspectatrice";

```

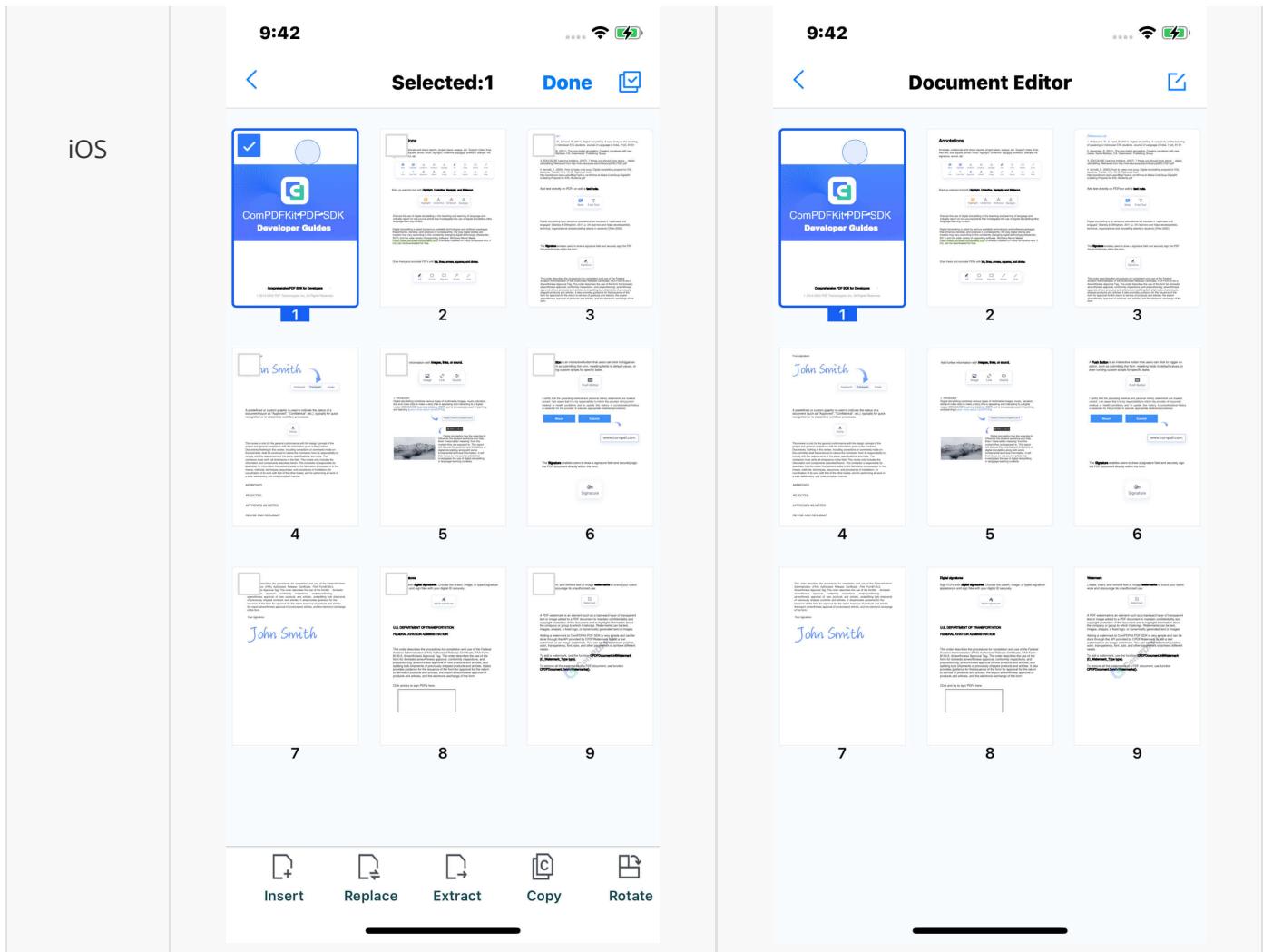
## 3.8.8 工具

本部分介绍如何使用 `CPDFReaderWidgetController` 提供的 API, 直接访问页面编辑视图、水印添加视图、安全设置视图等功能, 同时在 Flutter 中通过 `CPDFReaderWidget` 显示 PDF 文档, 为用户提供更灵活的使用体验。

### 打开页面编辑视图

```
// 控制是否进入页面编辑模式; 为 false 时, 显示缩略图列表  
bool editMode = true;  
controller.showThumbnailView(editMode);
```

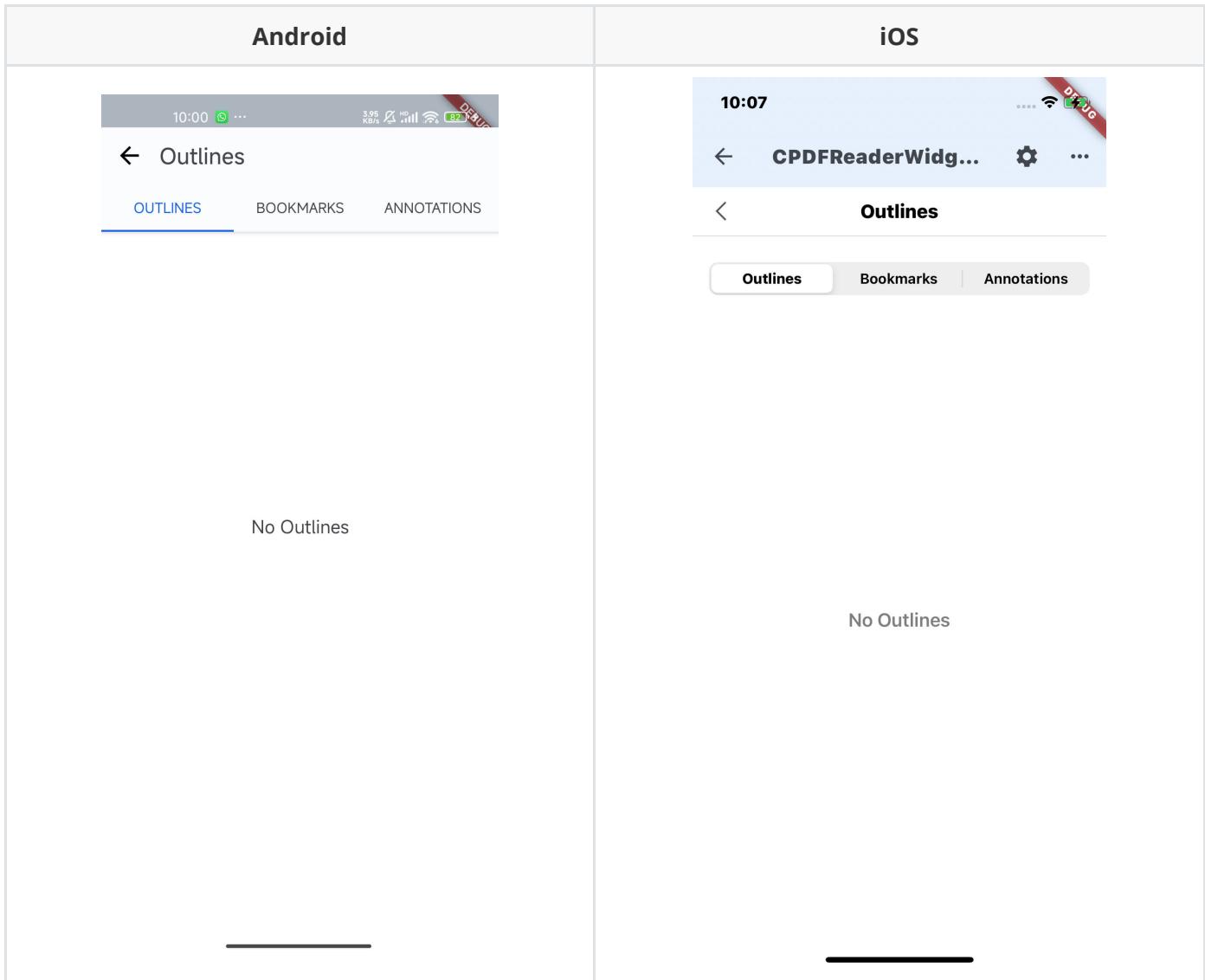
	editMode:true	editMode:false
Android	 <p>The screenshot shows the Document Editor app's interface in editMode:true. At the top, there is a toolbar with a back arrow, the title "Document Editor", a checkbox, and a "Done" button. Below the toolbar is a grid of nine thumbnail previews of PDF pages. Each thumbnail has a small number (1, 2, 3, 4, 5, 6, 7, 8, 9) at its bottom center. The thumbnails are arranged in three rows of three. The first thumbnail (1) shows a document titled "ComPDFKit+PDF-SDK Developer Guides". The other thumbnails show various other PDF documents, some with annotations like "Annotations" and "Signature".</p>	 <p>The screenshot shows the Document Editor app's interface in editMode:false. The layout is identical to the one above, but instead of thumbnail previews, it displays full-page versions of the same nine PDF documents. The first page (1) is the "ComPDFKit+PDF-SDK Developer Guides" document.</p>
	 <p>A horizontal toolbar with six icons: "Insert" (document with plus), "Replace" (document with minus), "Extract" (document with arrow), "Copy" (copy/paste), "Rotate" (document with arrow), and "Delete" (trash bin).</p>	 <p>A horizontal toolbar with the same six icons as the one above, positioned below the second column of thumbnails.</p>



## 打开 BOTA 视图

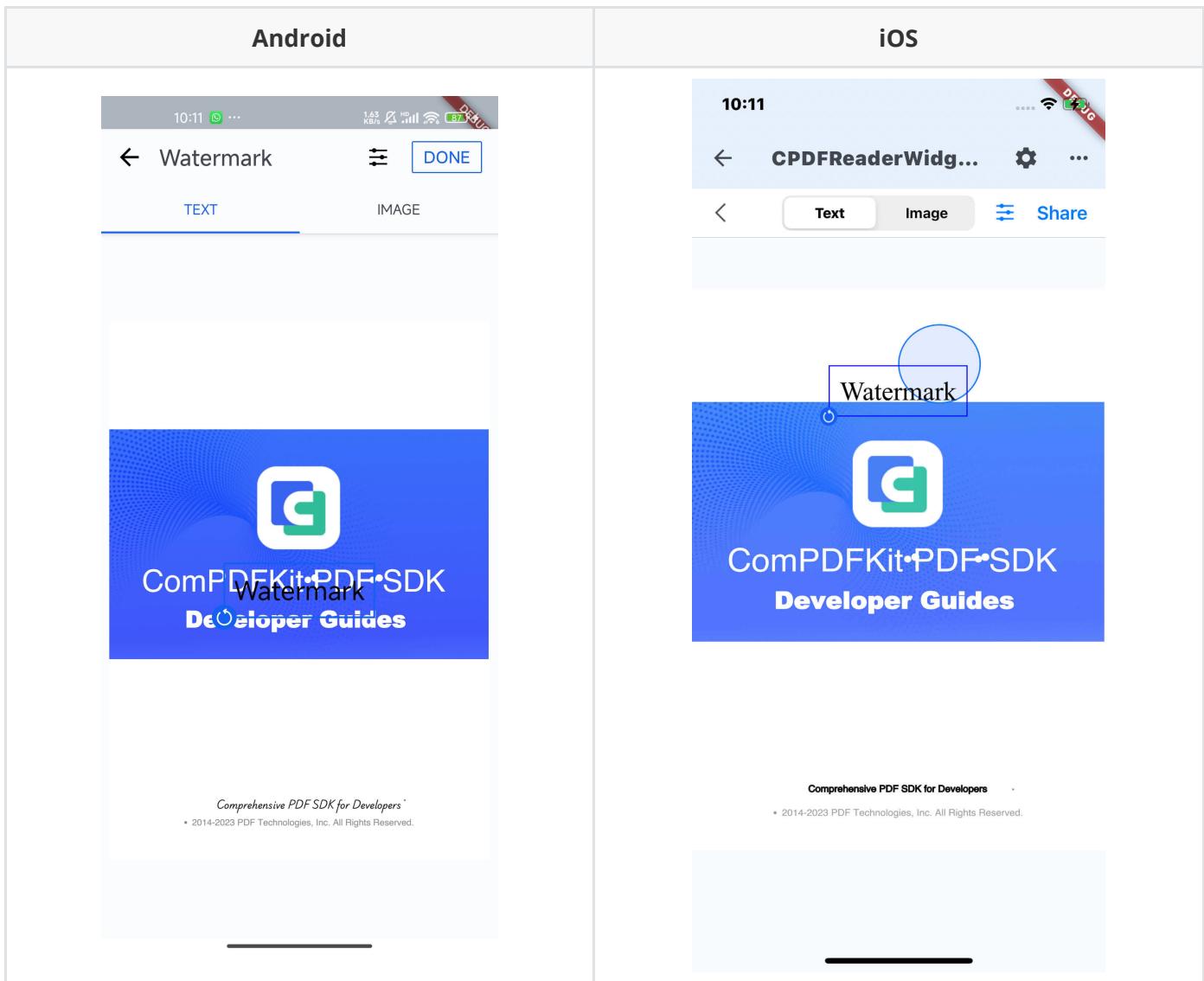
此视图显示文档大纲、书签和注释列表。

```
await controller.showBotaView();
```



打开添加水印对话框

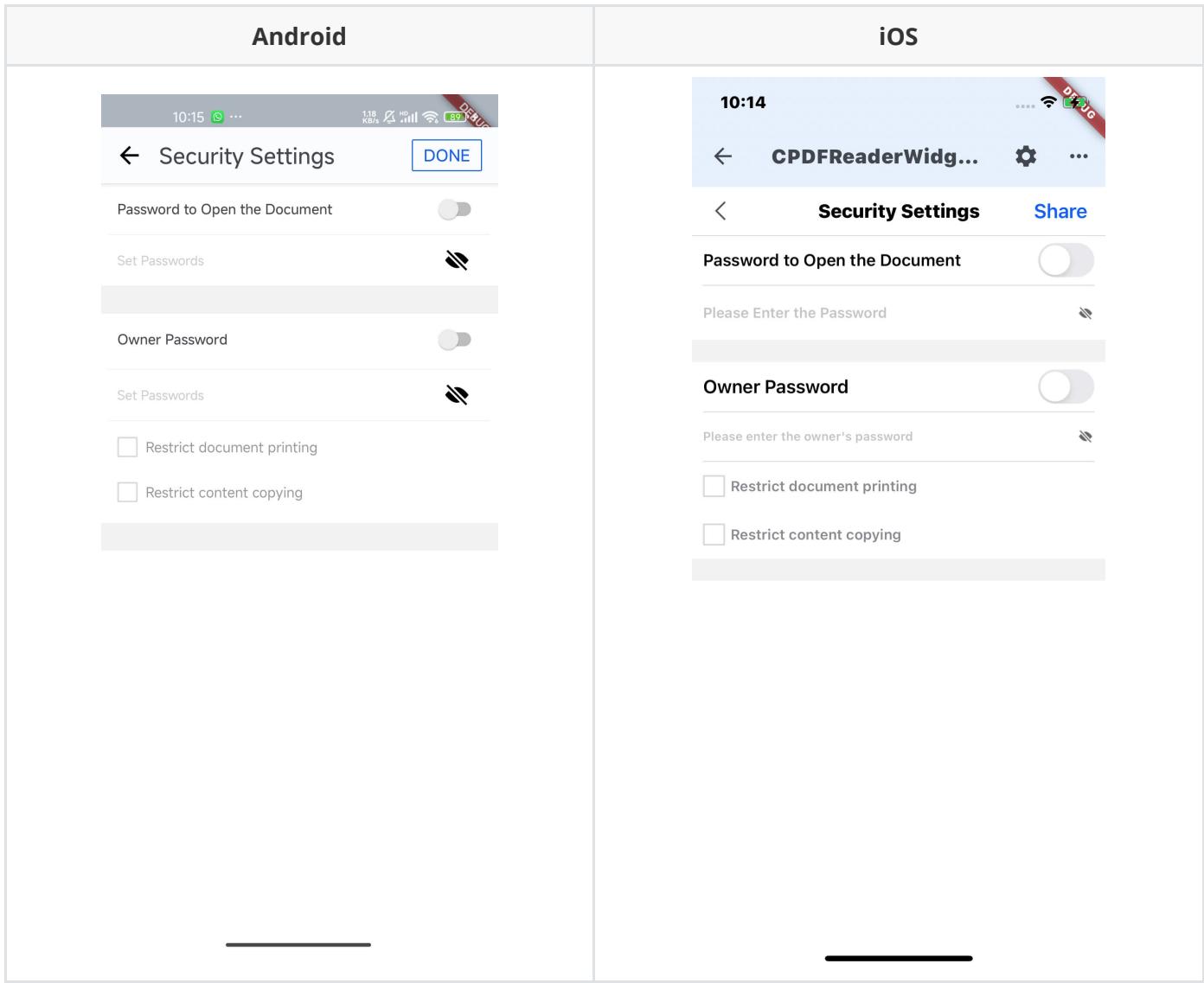
```
await controller.showAddWatermarkView();
```



## 打开安全设置视图

在安全设置视图中，您可以配置文档密码、拥有者权限密码和加密方式。

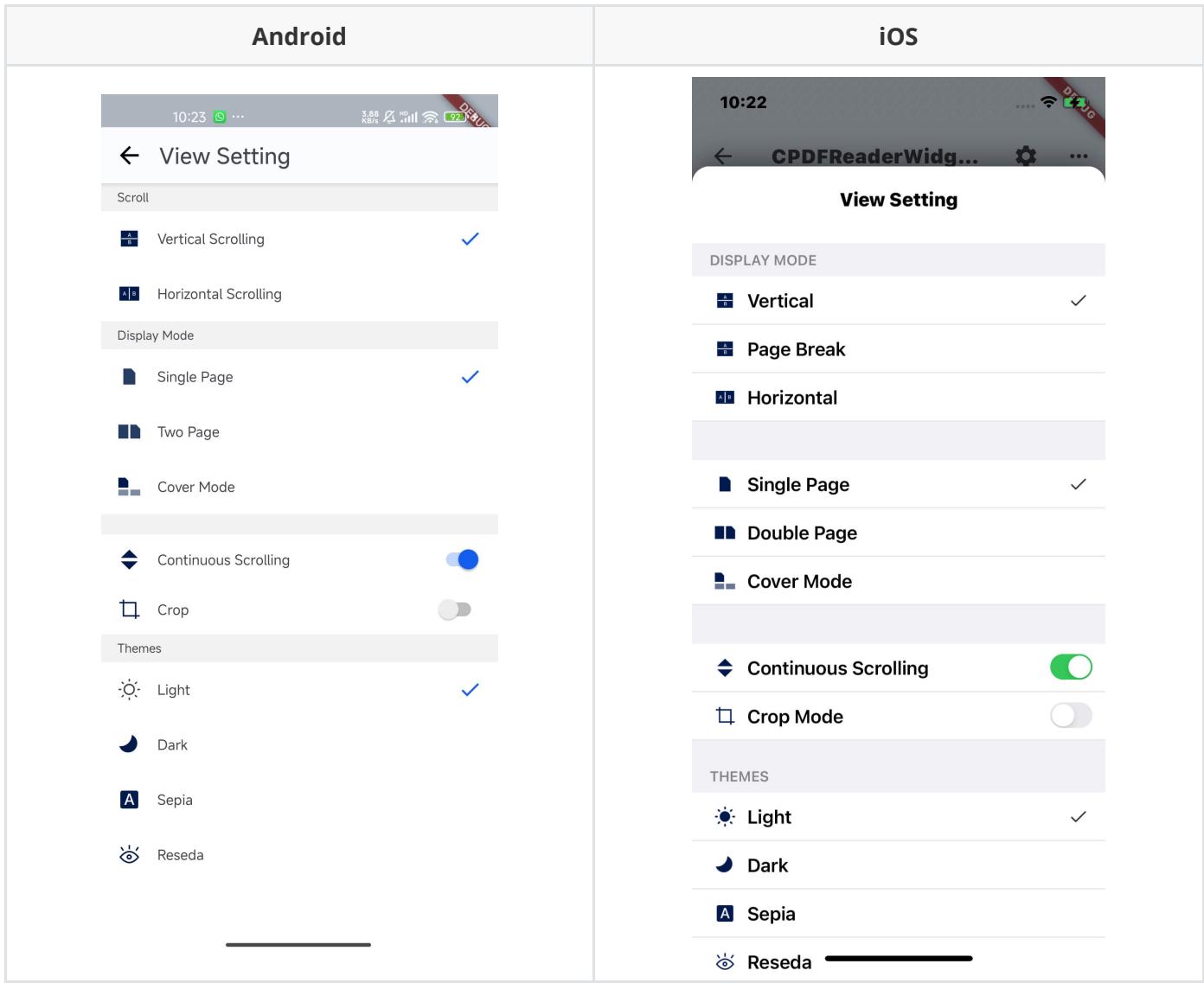
```
await controller.showSecurityView();
```



## 打开显示设置视图

显示设置视图允许您配置滚动方向、滚动模式、主题颜色等选项。

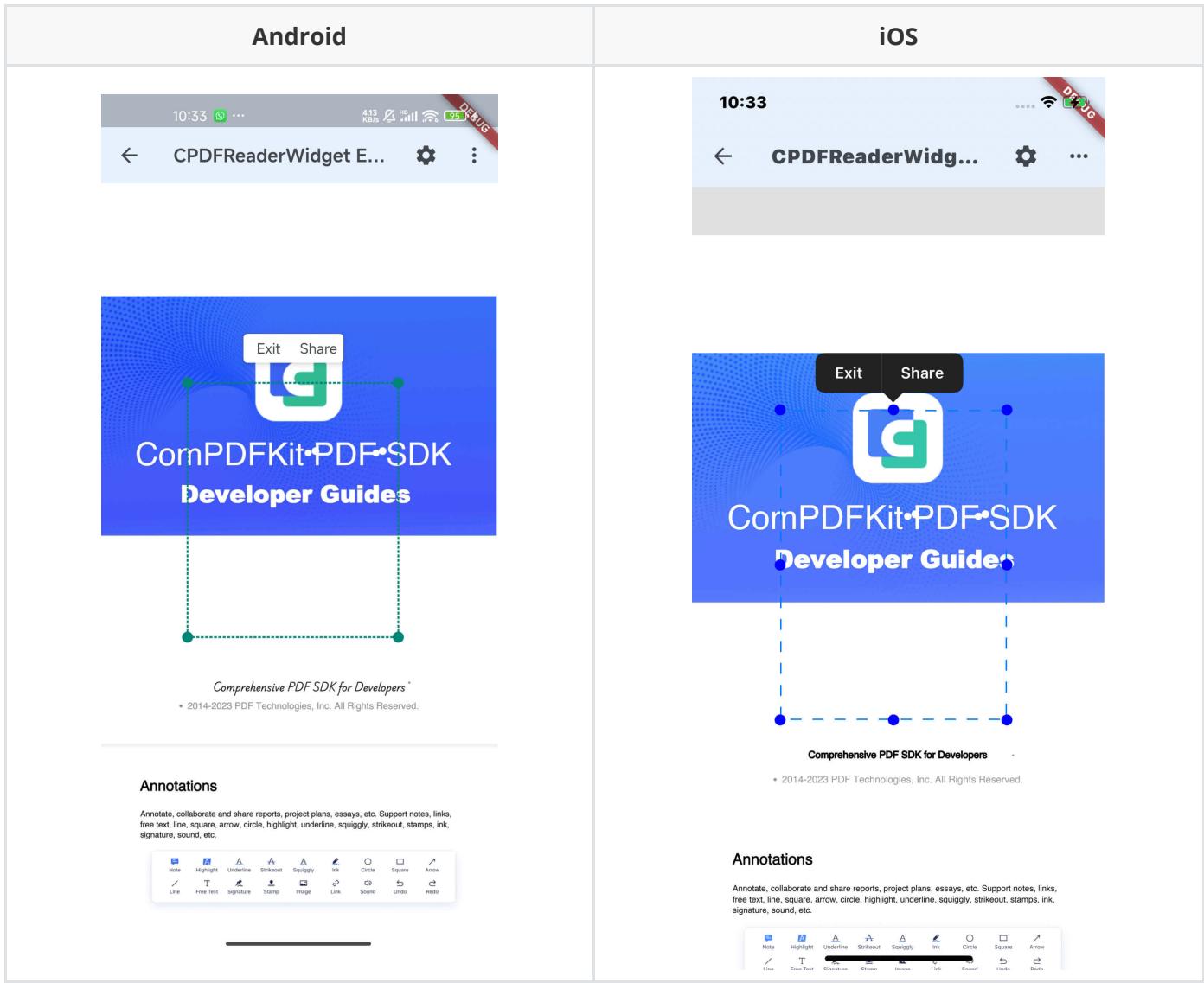
```
await controller.showDisplaySettingView();
```



## 截图功能

您可以在使用 API 显示 PDF 文档时进入截图模式，以捕获特定区域。

```
// 进入截图模式
await controller.enterSnipMode();
// 退出截图模式
await controller.exitSnipMode();
```

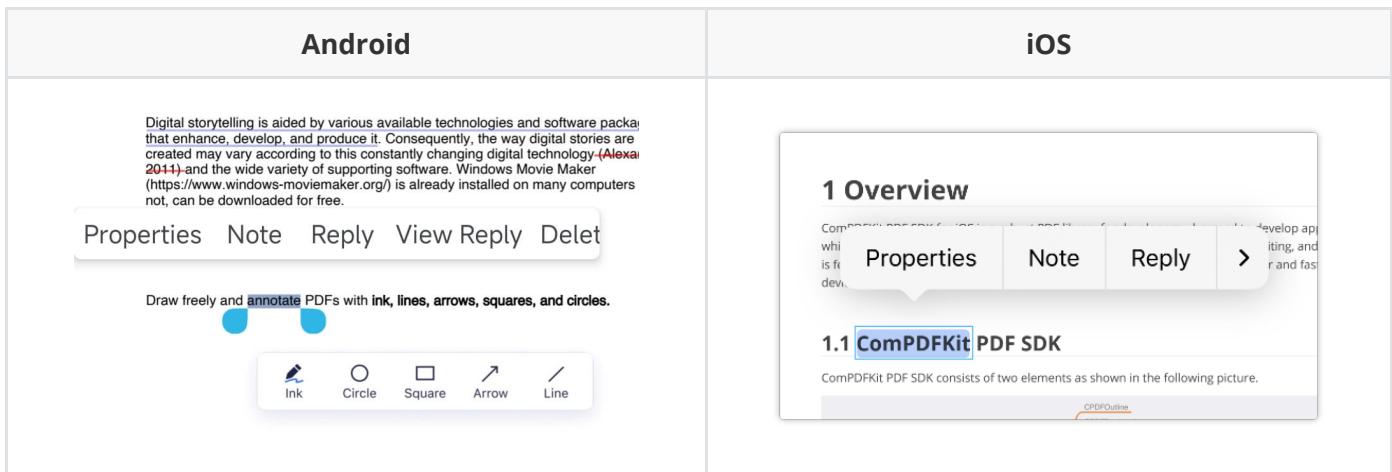


### 3.8.9 上下文菜单

在创建 `CPDFReaderWidget` 时，你可以通过 `CPDFConfiguration` 对象中的 `contextMenuConfig` 字段，来自定义当注释、文本、图片或表单字段被选中时弹出的上下文菜单。

#### 默认上下文菜单

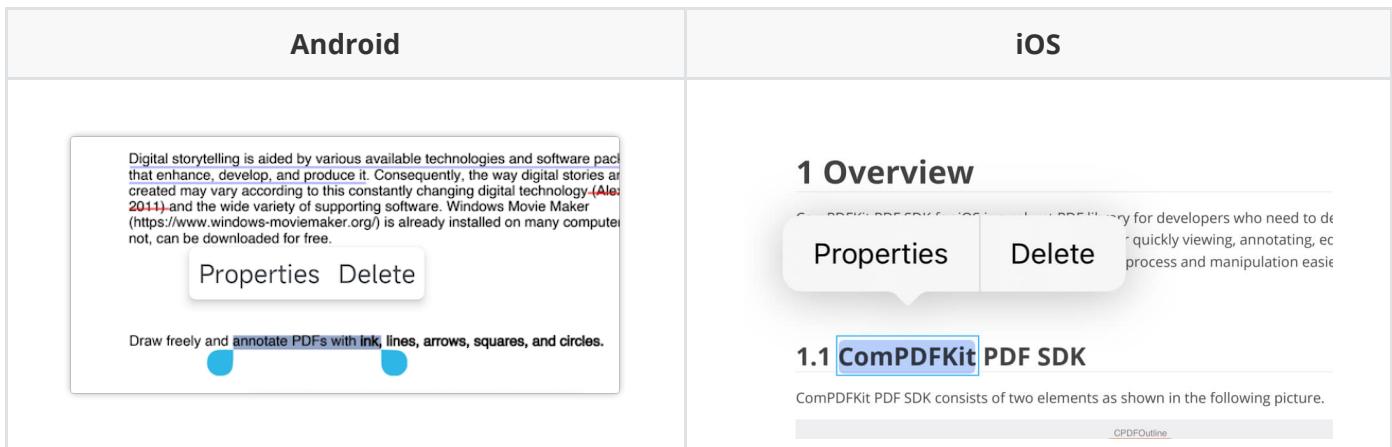
默认情况下，当用户选中一个高亮注释时，ComPDFKit 会显示包含常用选项的上下文菜单，例如“备注”、“删除”和“属性”：



## 自定义菜单项

你可以通过指定要包含的菜单项来自定义上下文菜单。以下示例仅为高亮注释保留 属性 和 删除 两个选项：

```
CPDFReaderWidget(
    document: widget.documentPath,
    password: widget.password,
    configuration: CPDFConfiguration(
        contextMenuConfig: const CPDFContextMenuConfig(
            annotationMode: CPDFAnnotationModeContextMenu(
                markup: [
                    CPDFContextMenuItem(CPDFAnnotationMarkupMenuKey.properties),
                    CPDFContextMenuItem(CPDFAnnotationMarkupMenuKey.delete),
                ]
            )
        ),
        onCreated: (controller) {
            },
    );
)
```



更多自定义配置说明，请参阅官方文档：[CONFIGURATION.md - contextMenuConfig](#)

## 3.8.10 UI可见性

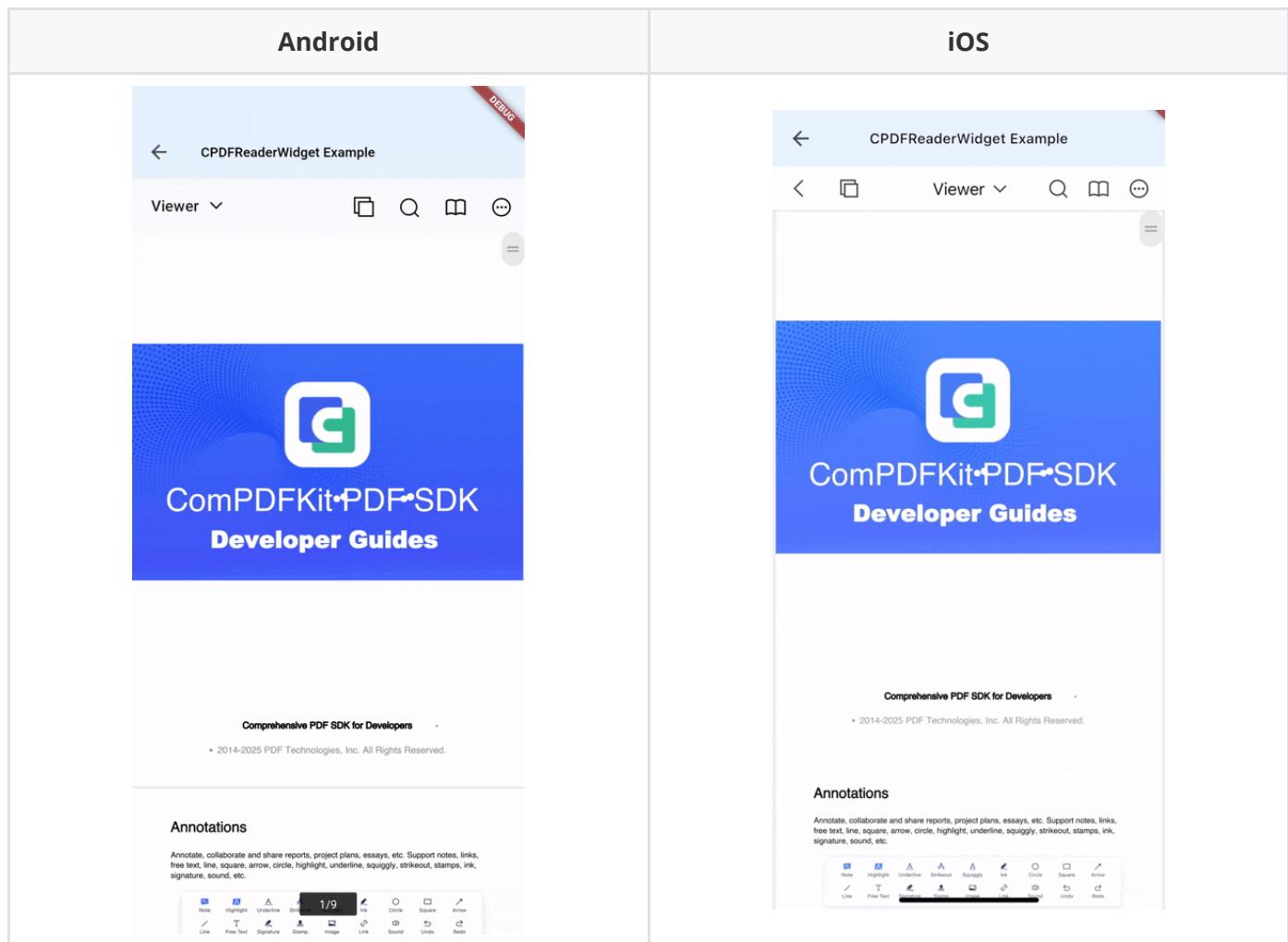
ComPDFKit Flutter SDK提供了几种显示或隐藏用户界面(UI)的方法。下表概述了支持的选项

选项	描述
automatic	点击页面时，工具栏和其他 UI 元素会自动显示或隐藏
always	用户界面始终可见
never	用户界面将始终隐藏

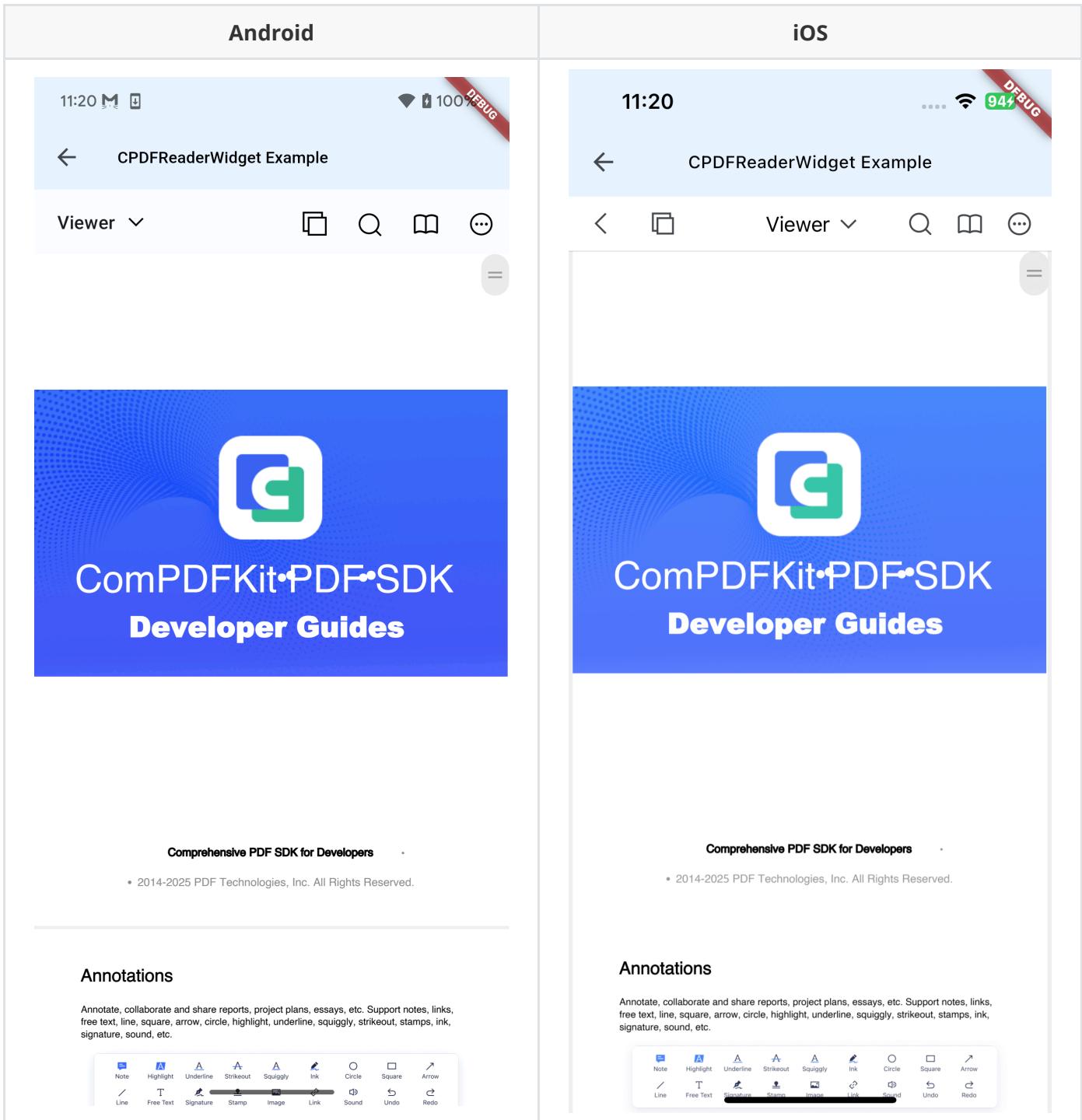
可以使用 `uiVisibilityMode` 配置选项更改用户界面视图模式，以下示例显示如何使用自动视图模式：

```
CPDFConfiguration(  
    modeConfig: const CPDFModeConfig(  
        uiVisibilityMode: CPDFUIVisibilityMode.automatic,  
    ),  
);
```

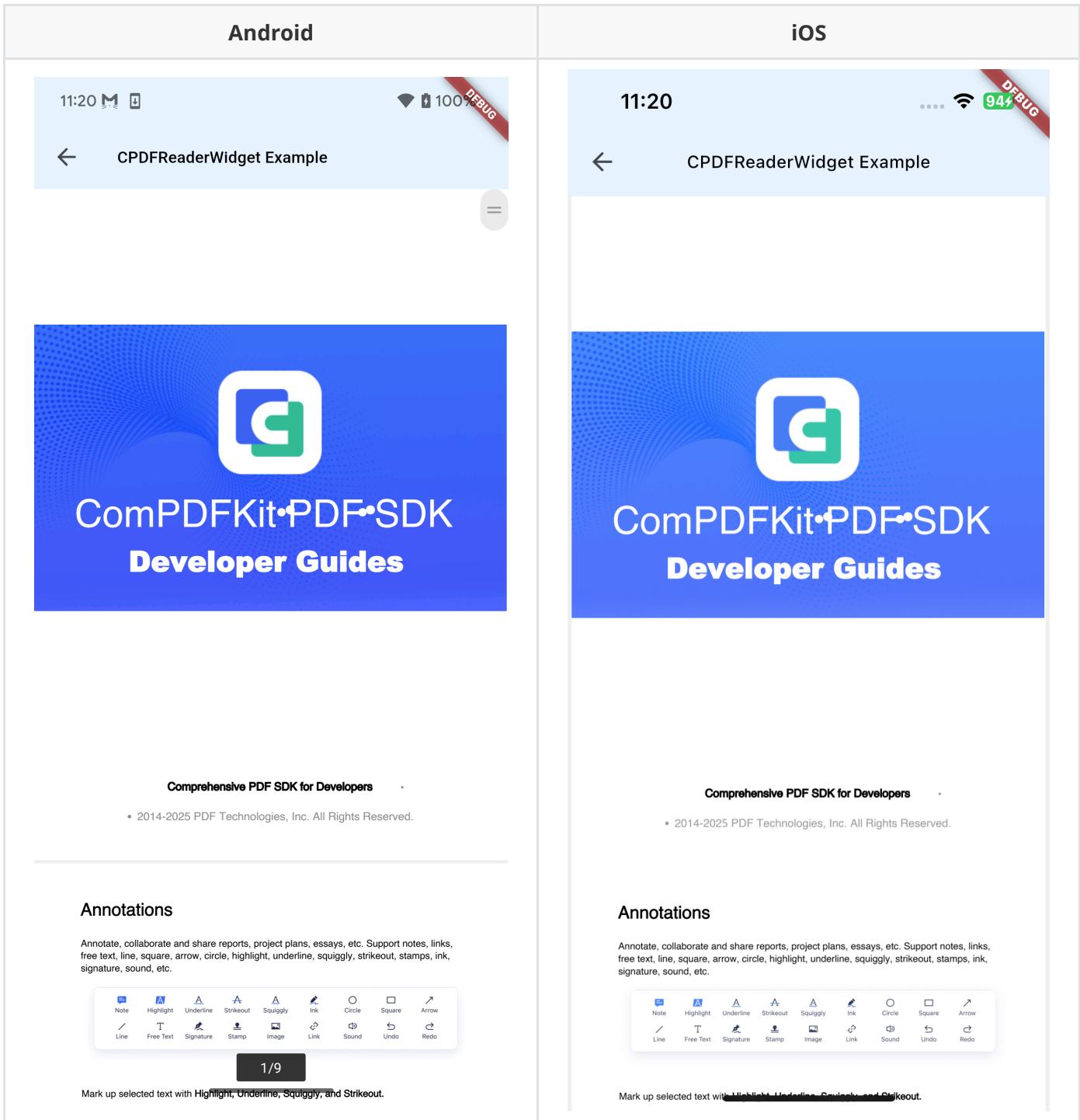
`automatic` 工作原理如下：



`always` 工作原理如下：



never 工作原理如下：



## 3.8.11 BOTA界面配置

### 功能说明

BOTA界面用于展示 PDF 文档的书签列表、大纲列表和注释列表。

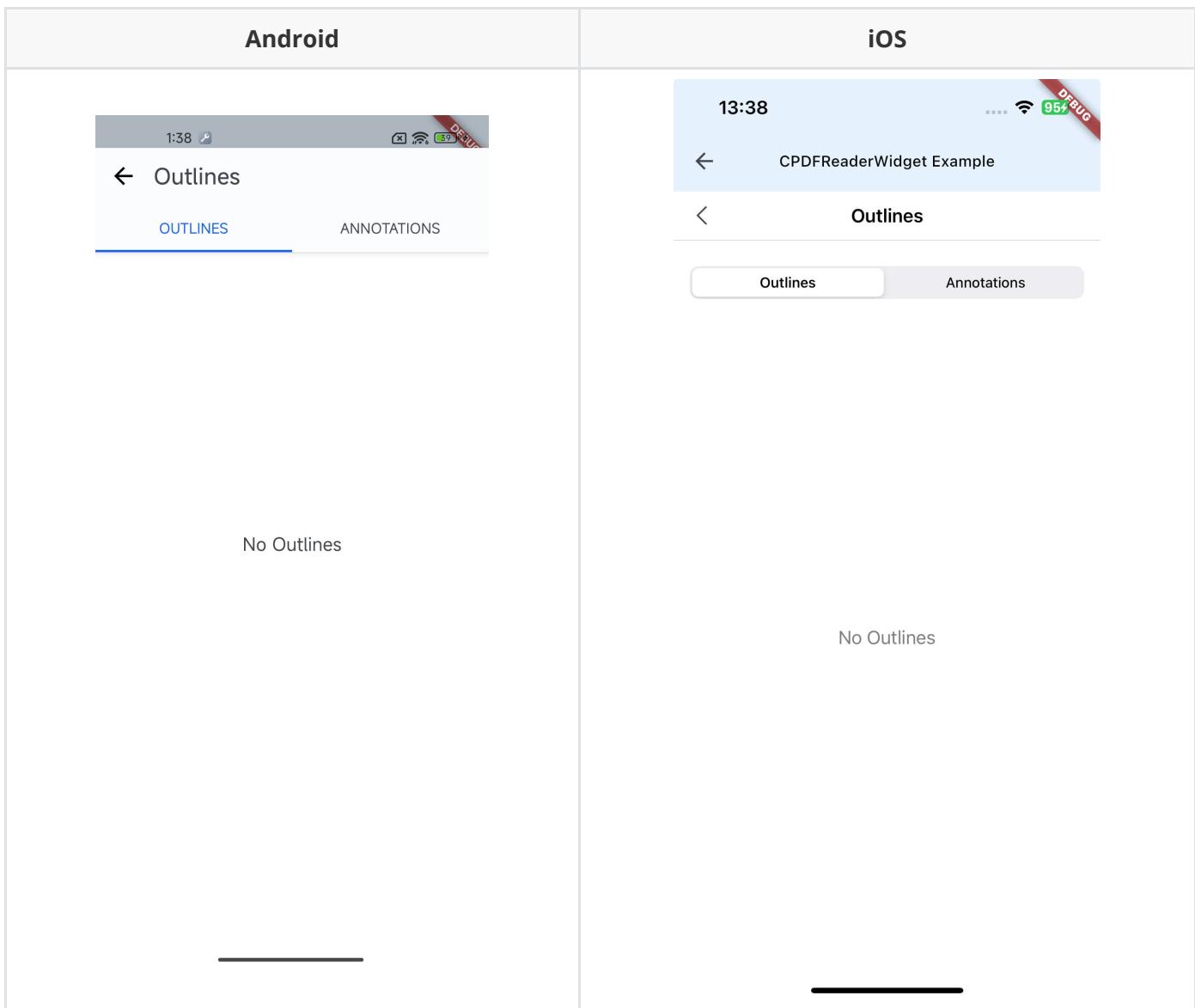
通过配置 BOTA，可以控制显示哪些类型的界面，以及每个界面中的菜单选项。

### 设置启用的标签页

使用 `CPDFBotaConfig.tabs` 配置启用的标签页：

```
CPDFConfiguration config = CPDFConfiguration(  
    globalConfig: const CPDFGlobalConfig(  
        bota: CPDFBotaConfig(  
            tabs: [  
                CPDFBotaTabs.outline,  
                CPDFBotaTabs.annotations  
            ]  
        )  
    )  
)  
;
```

效果示例：



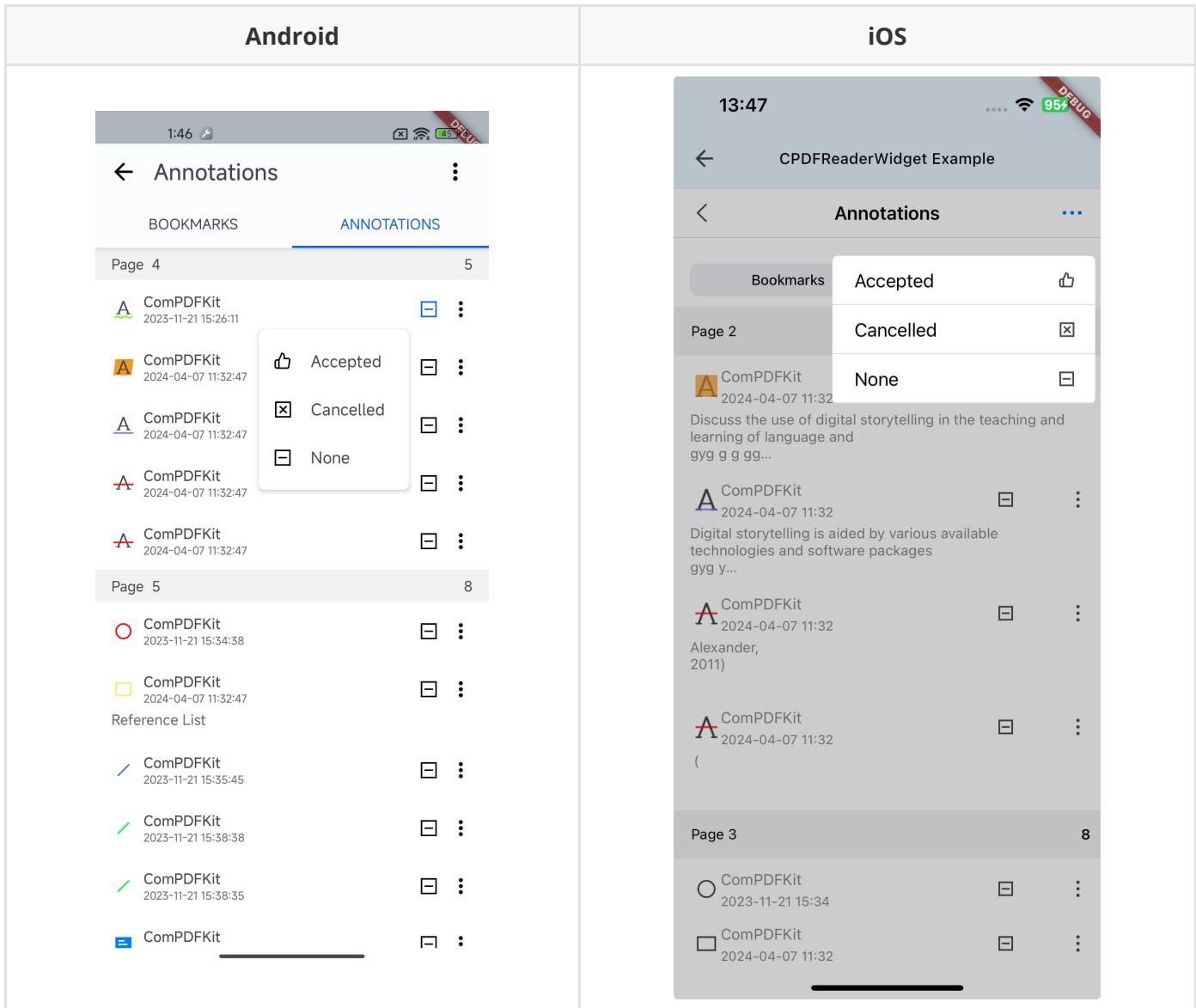
### 设置注释菜单选项

BOTA 支持为注释列表界面设置 全局菜单 和 单条注释菜单项：

```
CPDFConfiguration(  
    globalConfig: const CPDFGlobalConfig(  
        bota: CPDFBotaConfig(  
            menuItems: [
```

```
tabs: [
    CPDFBotaTabs.bookmark,
    CPDFBotaTabs.annotations,
],
menus: CPDFBotaMenuConfig(
    annotations: CPDFBotaAnnotationMenuConfig(
        global: [
            CPDFBotaMenuItem(id: CPDFBotaAnnotGlobalMenu.importAnnotation),
            CPDFBotaMenuItem(id: CPDFBotaAnnotGlobalMenu.exportAnnotation),
            CPDFBotaMenuItem(id: CPDFBotaAnnotGlobalMenu.removeAllAnnotation),
        ],
        item: [
            CPDFBotaMenuItem(id: CPDFBotaAnnotItemMenu.reviewStatus, subMenus: [
                CPDFReviewState.accepted,
                CPDFReviewState.cancelled,
                CPDFReviewState.none,
            ]),
            CPDFBotaMenuItem(id: CPDFBotaAnnotItemMenu.more, subMenus: [
                CPDFBotaAnnotMoreMenu.delete
            ]),
        ]
    )
)
)
```

效果示例：



### 注释全局菜单选项

选项	描述
CPDFBotaAnnotGlobalMenu.importAnnotation	导入注释
CPDFBotaAnnotGlobalMenu.exportAnnotation	导出注释
CPDFBotaAnnotGlobalMenu.removeAllAnnotation	删除所有注释
CPDFBotaAnnotGlobalMenu.removeAllReply	删除所有注释回复

### 注释回复状态子菜单选项

选项	描述
accepted	已接受
rejected	已拒绝
cancelled	已取消
completed	已完成
none	无状态

## 更多菜单选项

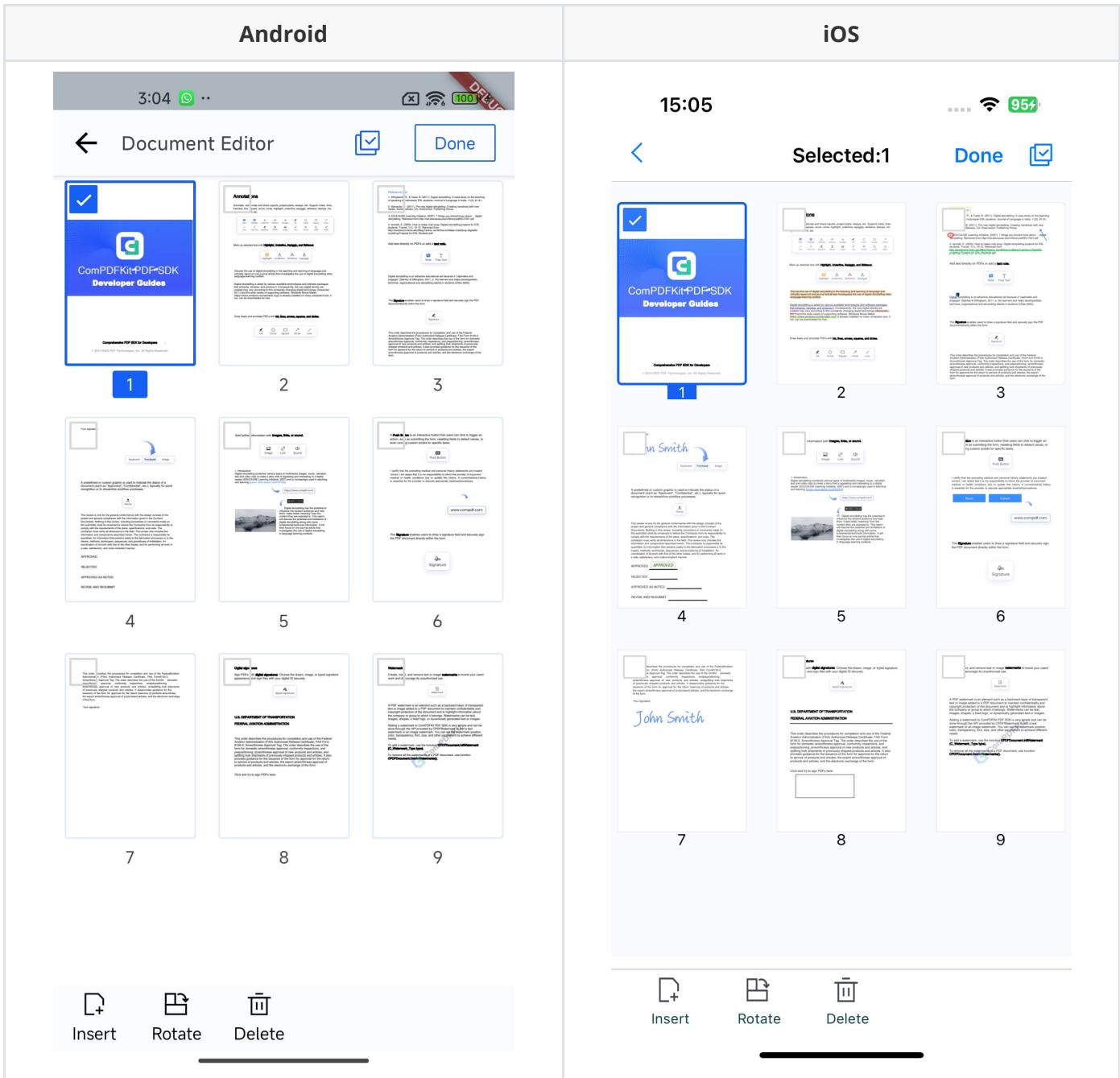
选项	描述
addReply	添加注释回复
viewReply	查看注释回复
delete	删除注释

## 3.8.12 页面编辑菜单

ComPDFKit Flutter SDK可通过 `CPDFConfiguration` 配置页面编辑界面底部启用的菜单选项。

```
CPDFConfiguration(  
  globalConfig: const CPDFGlobalConfig(  
    pageEditor: CPDFPageEditorConfig(  
      menus: [  
        CPDFPageEditorMenus.insertPage,  
        CPDFPageEditorMenus.rotatePage,  
        CPDFPageEditorMenus.deletePage,  
      ]  
    )  
  )  
)
```

## 效果示例



## 3.9 内容编辑

### 3.9.1 概述

内容编辑提供了文字、图片和路径编辑能力，让用户轻松插入、调整、删除文本、图片和路径，使 PDF 程序像富文本编辑器一样可以直接用于创作和修改。

#### 指南

##### 内容编辑模式

了解如何切换到内容编辑模式。

##### 设置编辑类型

通过 API 或工具栏切换要启用的编辑类型。

### 撤销和重做

撤销或恢复修改内容，支持工具栏操作及 API 调用。

## 3.9.2 内容编辑模式

在 CPDFReaderWidget 中进行内容编辑前，需要先切换到 **CPDFViewMode.contentEditor** 模式。

切换方式包括：

1. **UI 切换**: 通过 `CPDFReaderWidget` 右上角的模式切换菜单选择。
2. **初始化时设置**: 在 `CPDFConfiguration` 中设置初始模式。
3. **运行时切换**: 调用 `CPDFReaderWidgetController` 的 API。

```
// 创建时设置内容编辑模式
CPDFReaderWidget(
  document: documentPath,
  configuration: CPDFConfiguration(
    modeConfig: const CPDFModeConfig(
      initialViewMode: CPDFViewMode.contentEditor
    )
  ),
  onCreated: (controller) {},
);
// 使用控制器切换内容编辑模式
await controller.setPreviewMode(CPDFViewMode.contentEditor);
```

## 3.9.3 设置编辑类型

进入内容编辑模式后，开发者可以控制用户可编辑的元素类型：

- **文本编辑**: 增删改 PDF 中的文字内容。
- **图片编辑**: 插入、替换或删除 PDF 内的图片。
- **路径编辑**: 修改 PDF 矢量路径（如线条、形状）。

### 工具栏切换

用户可通过 **底部工具栏** 切换编辑类型。

### API 切换

如果你隐藏了底部工具栏，可以使用 API：

```
// 仅启用文本编辑
controller.editManager.changeEditType([CPDFEditType.text]);

// 启用文本和图片编辑
controller.editManager.changeEditType([CPDFEditType.text, CPDFEditType.image]);

// 禁用所有编辑（只读）
controller.editManager.changeEditType([CPDFEditType.none]);
```

## 3.9.4 撤销和重做

内容编辑支持 **撤销（Undo）** 与 **重做（Redo）**，帮助用户在编辑过程中快速回退或恢复操作。

### 工具栏操作

用户可直接点击底部工具栏的 **撤销/重做** 按钮。

### API 调用

开发者也可通过 `historyManager` 来实现自定义逻辑：

### 设置回调

```
// 设置回调（监听页面状态）
controller.editManager.historyManager.setOnHistoryChangedListener(
    (pageIndex, canUndo, canRedo) {
        debugPrint(
            'History changed: page=$pageIndex, canUndo=$canUndo, canRedo=$canRedo',
        );
    },
);

// 撤销
if (await controller.editManager.historyManager.canUndo()) {
    await controller.editManager.historyManager.undo();
}

// 重做
if (await controller.editManager.historyManager.canRedo()) {
    await controller.editManager.historyManager.redo();
}
```

## 4 故障排除

1.当cocopods下载iOS第三方库时，下载 `ComPDFKit` 库的SSL网络请求失败

如果运行 `pod install` 时SSL网络请求下载 `ComPDFKit` 库失败，请替换 `ComPDFKit` 库的第三方平台下载地址链接并执行 `pod install`。

```

require_relative '../node_modules/react-native/scripts/react_native_pods'
require_relative '../node_modules/@react-native-community/cli-platform-ios/native_modules'

- platform :ios, '10.0'
+ platform :ios, '12.0'
install! 'cocoapods', :deterministic_uuids => false

target 'PDFView_RN' do
  config = use_native_modules!

  # Flags change depending on the env values.
  flags = get_default_flags()

  use_react_native!(
    :path => config[:reactNativePath],
    # to enable hermes on iOS, change `false` to `true` and then install pods
    :hermes_enabled => flags[:hermes_enabled],
    :fabric_enabled => flags[:fabric_enabled],
    # An absolute path to your application root.
    :app_path => "#{Pod::Config.instance.installation_root}/.."
  )
end

target 'PDFView_RNTests' do
  inherit! :complete
  # Pods for testing
end

+ pod 'ComPDFKit', :git => 'https://github.com/ComPDFKit/compdfkit-pdf-sdk-ios-swift.git', :tag => '2.5.0'
+ pod 'ComPDFKit_Tools', :git => 'https://github.com/ComPDFKit/compdfkit-pdf-sdk-ios-swift.git', :tag => '2.5.0'

# Enables Flipper.
#
# Note that if you have use_frameworks! enabled, Flipper will not work and
# you should disable the next line.
use_flipper!()

post_install do |installer|
  react_native_post_install(installer)
  __apply_Xcode_12_5_M1_post_install_workaround(installer)
end
end

```

## 5 支持

### 5.1 报告问题

感谢您对 ComPDFKit 的感兴趣，它是一款易于使用且功能强大的开发解决方案，可将高质量 PDF 渲染功能集成到您的应用程序中。如果您在使用 ComPDFKit for Flutter 时遇到任何技术问题或错误问题，请将问题报告提交给 ComPDFKit 团队。以下更多信息将帮助我们解决您的问题：

- ComPDFKit 产品和版本。
- 您的操作系统和 IDE 版本。
- 问题的详细描述。
- 任何其他相关信息，例如错误屏幕截图。

## 5.2 联系信息

---

网站链接：

- 英文主页：<https://www.compdf.com>
- 中文主页：<https://www.compdf.com/zh-cn>
- 开发者文档：<https://www.compdf.com/guides/pdf-sdk/flutter/overview>

联系ComPDFKit：

- 联系销售：<https://www.compdf.com/contact-sales>
- 技术问题反馈至：<https://www.compdf.com/support>
- 电子邮件（直接联系）：[support@compdf.com](mailto:support@compdf.com)

谢谢您，

ComPDFKit 团队