

开发者指南

Foxit PDF SDK

For Android

Microsoft® Partner
Gold Independent Software Vendor (ISV)

TABLE OF CONTENTS

1 Foxit PDF SDK 简介.....	1
1.1 Foxit PDF SDK.....	1
1.2 Foxit PDF SDK for Android	1
1.2.1 为什么选择 Foxit PDF SDK for Android.....	1
1.2.2 Foxit PDF SDK for Android 的主要框架.....	2
1.2.3 UI Extensions 组件概述.....	3
1.2.4 Foxit PDF SDK for Android 的主要功能	5
1.3 评估.....	7
1.4 授权.....	7
1.5 关于此文档	7
2 入门指南.....	8
2.1 系统要求	8
2.2 包结构说明	8
2.3 运行 demo.....	10
2.3.1 Function demo	11
2.3.2 Viewer control demo	13
2.3.3 Complete PDF viewer demo	16
3 快速构建一个功能齐全的 PDF 阅读器.....	20
3.1 创建一个新的 Android 工程.....	20
3.2 集成 Foxit PDF SDK for Android 到您的应用程序	22
3.3 初始化 Foxit PDF SDK for Android	27
3.4 使用 PDFViewCtrl 显示 PDF 文档	28
3.5 打开一个 RMS 加密的文档	33
3.6 使用 UI Extensions 组件构建一个功能齐全的 PDF 阅读器.....	36
3.7 基于功能齐全的 PDF 阅读器添加扫描功能.....	43
3.8 分区存储 (Scoped Storage)处理	49

4 自定义 UI.....	51
4.1 通过配置文件自定义 UI.....	51
4.1.1 JSON 文件介绍.....	51
4.1.2 配置项描述	56
4.1.3 使用配置文件实例化一个 UIExtensionsManager 对象.....	61
4.1.4 通过配置文件自定义 UI 的示例	62
4.2 通过 APIs 自定义 UI 元素	64
4.2.1 自定义 top/bottom toolbar.....	64
4.2.2 自定义添加/移除一个特定的面板.....	74
4.2.3 自定义隐藏 View setting bar 上的 UI 元素	78
4.2.4 自定义添加/隐藏 More Menu 菜单上的 UI 元素.....	81
4.3 通过源代码自定义 UI 实现.....	87
5 使用 SDK API	93
5.1 Render.....	93
5.1.1 如何将指定的 PDF 页面渲染到 bitmap	93
5.2 Text Page	94
5.2.1 如何通过选择获取页面上的文本区域.....	95
5.3 Text Search.....	96
5.3.1 如何在 PDF 文档中搜索指定的文本模型.....	96
5.4 Bookmark (Outline)	97
5.4.1 如何使用深度优先顺序遍历 PDF 文档的 bookmarks	97
5.5 Reading Bookmark.....	99
5.5.1 如何添加自定义 reading bookmark 并枚举所有的 reading bookmarks	99
5.6 Attachment.....	100
5.6.1 如何将指定文件嵌入到 PDF 文档.....	100
5.6.2 如何从 PDF 文档中导出嵌入的附件文件，并将其另存为单个文件	101
5.7 Annotation.....	101
5.7.1 如何向 PDF 页面中添加注释	103
5.7.2 如何删除 PDF 页面中的注释	104

5.8	Form	105
5.8.1	如何通过 XML 文件导入表单数据或将表单数据导出到 XML 文件.....	105
5.9	Security	105
5.9.1	如何使用密码加密 PDF 文件	106
5.10	Signature.....	106
5.10.1	如何对 PDF 文档进行签名，并验证签名.....	107
6	创建自定义工具	109
7	使用 Cordova 实现 Foxit PDF SDK for Android	121
8	使用 React Native 实现 Foxit PDF SDK for Android.....	122
9	使用 Xamarin 实现 Foxit PDF SDK for Android	123
10	FAQ.....	124
10.1	从指定的 PDF 文件路径打开一个 PDF 文档	124
10.2	打开 PDF 文档时显示指定的页面	125
10.3	License key 和序列号无法正常工作	126
10.4	在 PDF 文档中添加 link 注释.....	126
10.5	向 PDF 文档中插入图片	127
10.6	高亮 PDF 文档中的 links 和设置高亮颜色	128
10.7	高亮 PDF 文档中的表单域和设置高亮颜色.....	129
10.8	支持全文索引搜索	130
10.9	打印 PDF 文档	132
10.10	夜间模式颜色设置	133
10.11	输出 exception/crash 日志信息	134
10.12	减小 APK 的大小	135
10.13	开启 shrink-code (设置 "minifyEnabled" 为 "true").....	135
10.14	本地化设置	136
10.15	迁移到 AndroidX.....	137

10.16 支持 Chromebook.....	139
11 技术支持.....	140

1 Foxit PDF SDK 简介

1.1 Foxit PDF SDK

Foxit PDF SDK 提供高性能的开发库，帮助软件开发人员使用最流行的开发语言和环境在不同平台(包括 Windows、Mac、Linux、Web、Android、iOS 和 UWP)的企业版、移动版和云应用程序中添加强大的 PDF 功能。

使用 Foxit PDF SDK 的应用开发人员可以利用 Foxit 强大、标准化的 PDF 技术安全地显示、创建、编辑、批注、格式化、管理、打印、共享，搜索 PDF 文档，以及填写 PDF 文档。此外，Foxit PDF SDK 包括一个内置可嵌入的 PDF Viewer，使得开发过程更加简单和快速。有关更多详细信息，可以访问网站 <https://developers.foxitsoftware.cn/pdf-sdk/>。

在本指南中，我们将重点介绍 Foxit PDF SDK for Android 平台。

1.2 Foxit PDF SDK for Android

您是否曾经为 PDF 规范的复杂性而感到不知所措？您是否曾经为被要求在有限的时间内构建一个功能齐全的 PDF 应用而感到迷茫。如果您的答案是"Yes"，那么恭喜您！您找到了在业界中快速将 PDF 功能集成到应用程序中的优选方案。

Foxit PDF SDK for Android 致力于帮助开发人员快速将强大的 Foxit PDF 技术集成到他们自己的移动端应用程序中。通过 Foxit 开发包，即使是对 PDF 了解有限的开发人员也可以在 Android 平台上用几行代码快速构建一个专业的 PDF 阅读器。

1.2.1 为什么选择 Foxit PDF SDK for Android

Foxit 是领先的 PDF 软件解决方案供应商，专注于 PDF 显示、编辑、创建、管理以及安全方面。Foxit PDF SDK 开发库已在当今许多知名的应用程序中使用，并且经过长期的测试证明 Foxit PDF SDK 的质量、性能和功能正是业界大部分应用程序所需要的。

Foxit PDF SDK for Android 提供了快速 PDF 阅读和 Android 设备的操作控制。选择 Foxit PDF SDK for Android 的几大理由：

- **易于集成**

开发人员可以通过几行代码将 SDK 无缝集成到他们自己的应用程序中。

- **设计完美**

Foxit PDF SDK for Android 拥有简单、干净和友好的风格，并且提供了最好的用户体验。

- **灵活定制**

Foxit PDF SDK for Android 提供了应用层用户界面的源代码，可以帮助开发人员对应用程序的功能和界面外观进行灵活定制。

- **移动平台上的鲁棒性**

Foxit PDF SDK for Android 提供了 OOM (内存溢出) 恢复机制，以确保应用程序在内存有限的移动设备上运行时仍然具备较高的鲁棒性。

- **基于福昕高保真的 PDF 渲染引擎**

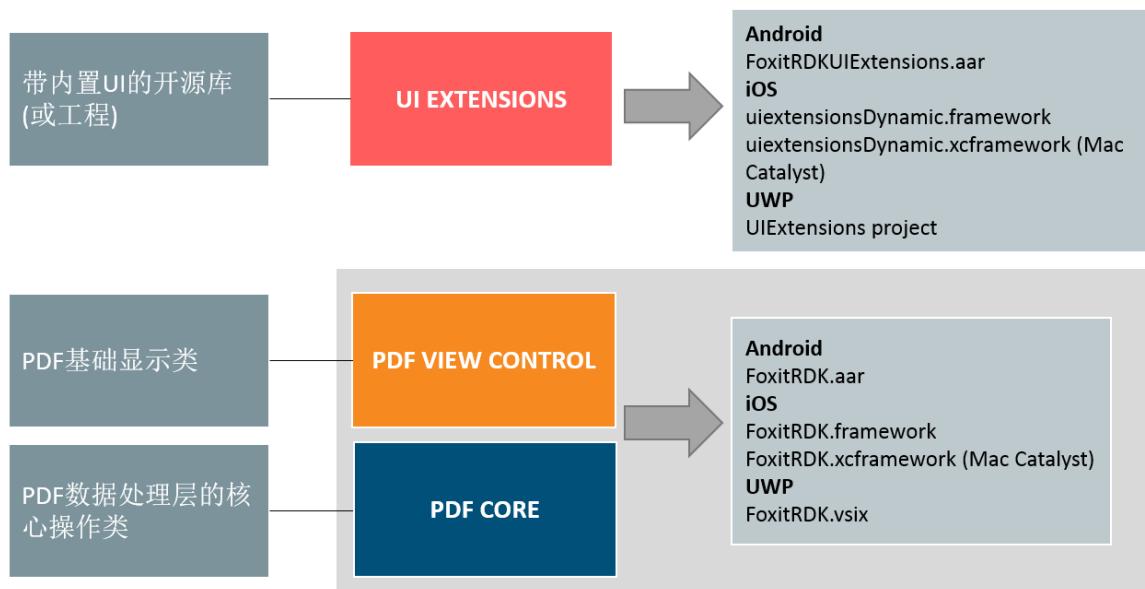
Foxit PDF SDK 的核心技术是基于世界众多知名企业所信赖的福昕 PDF 引擎。福昕强大的 PDF 引擎可快速解析和渲染文档，不受设备环境的约束。

- **优秀的技术支持**

福昕对自己的开发产品提供了优秀的技术支持，当您在开发关键重要的产品时，可以提供高效的帮助和支持。福昕拥有一支 PDF 行业优秀的技术支持工程师团队，同时将定期地进行版本更新发布，通过添加新的功能和增强已有的功能来提升用户体验。

1.2.2 Foxit PDF SDK for Android 的主要框架

Foxit PDF SDK for Android 由三个元素组成，如下图所示。Foxit PDF SDK 的所有移动平台版本共享此结构，这样便于在您的应用程序中集成，以及支持多种手机操作系统和框架。



Foxit PDF SDK for Android, iOS, UWP 的三种组成元素

- **PDF Core API**

PDF Core API 是 SDK 的核心部分，建立在福昕强大的底层 PDF 技术上。它提供了 PDF 基础功能操作相关的函数，包含了 PDF View 控件和 UI Extensions 组件中使用到的 PDF 核心处理功能，以确保应

用程序达到高的性能和效率。该 API 可单独用于文档的渲染、分析、文本提取、文本搜索、表单填写、数字签名、压感笔迹 (PSI)、证书和密码加密、注释的创建和管理等等。

- **PDF View Control**

PDF View 控件是一个工具类，根据开发人员的需求提供开发人员与渲染的 PDF 文档进行交互所需要的功能接口。以福昕享有盛誉且使用广泛的 PDF 渲染技术为核心，View Control 支持快速高质量的渲染、缩放、滚动和页面导览功能。该 View 控件继承于平台相关 viewer 的类，例如 `Android.View.ViewGroup`，并且允许进行扩展来满足特定用户的需求。

- **UI Extensions 组件**

UI Extensions 组件是一个带内置 UI 的开源库，支持对内置的文本选择、标记注释、大纲导航、阅读书签、全文检索、填表、文本重排、文档附件、数字/手写签名、文档编辑和密码加密等功能进行自定义。UI Extensions 组件中的这些功能是通过使用 PDF core API 和 PDF View Control 来实现的。开发人员可以利用这些已有的 UI 实现快速构建一个 PDF 阅读器，同时可以根据需要灵活自定义其 UI 界面。

从 4.0 版本开始，Foxit PDF SDK for Android 对 UI Extensions 组件做了一个重大的改变和优化。将基础的 UI 实现都封装到 `PDFReader` 类中，比如面板控件、工具栏设置、以及预警视图对话框等。因此，构建一个功能齐全的 PDF 阅读器变得越来越简单和容易。此外，用户可以通过一个配置文件灵活自定义他们需要的功能。

从 5.0 版本开始，内置 UI 中的任何元素都可以通过 API 来进行自定义。该版本为开发人员提供了更高级的 APIs 和更强大的配置文件来对 UI 元素进行自定义，比如向工具栏中添加新的功能按钮，或者从工具栏中移除已有的功能按钮，显示/隐藏特定的菜单或者功能面板等。

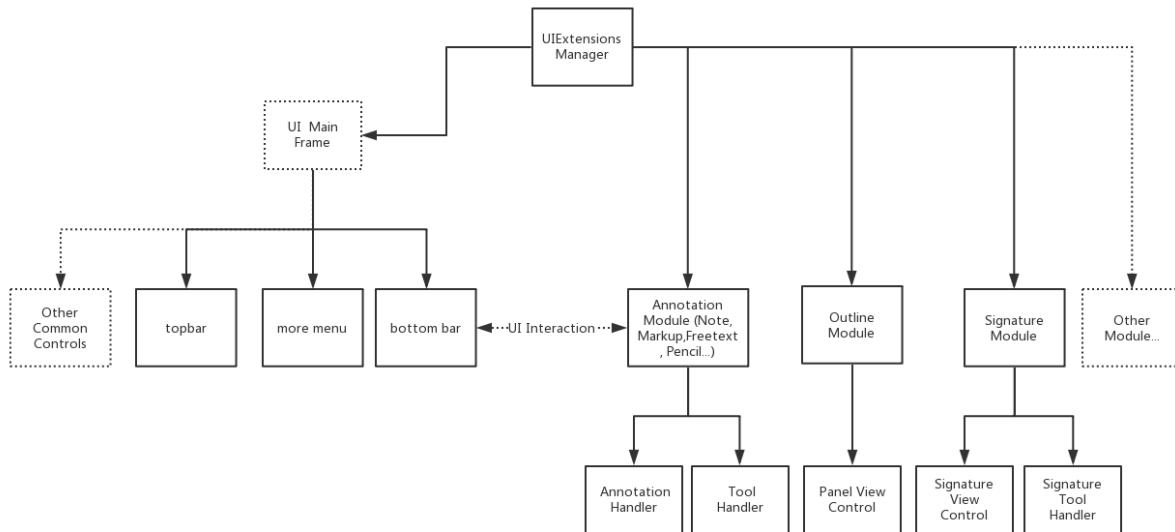
从 6.0 版本开始，Foxit PDF SDK for Android 移除了 `PDFReader` 类，将 `PDFReader` 类中封装的 APIs 移到了 UI Extensions 组件中。

1.2.3 UI Extensions 组件概述

UI Extensions 组件采用 module 机制，将每个功能细化成一个 module。当加入 UI Extensions 时，所有的 modules 除了 `LocalModule`(用于文件管理)会被默认自动加载。用户可以通过实现 Module 接口类来自定义 module，然后调用 **UIExtensionsManager#registerModule** 在当前 `UIExtensions Manager` 中进行注册。如果不需要使用时，可以调用 **UIExtensionsManager#unregisterModule** 进行反注册。

`UIExtensionsManager` 包含了主框架 UI，如 top/bottom toolbar，以及各个模块之间共享的 UI 组件。同时，各个功能模块也可以通过 `UIExtensionsManager` 来进行单独加载。功能模块在加载的时候会对主框架 UI 进行适配和调整，并且建立起消息事件响应的联系。各个功能模块可能包含了其模块特有的 UI 组件，同时也会有自己独立的消息事件处理逻辑。`UIExtensionsManager` 也会负责将从

View Control 组件接收到的消息和事件分发到各个功能模块中去。下面的图片讲述了 UIExtensionsManager 和 modules 之间的详细关系。

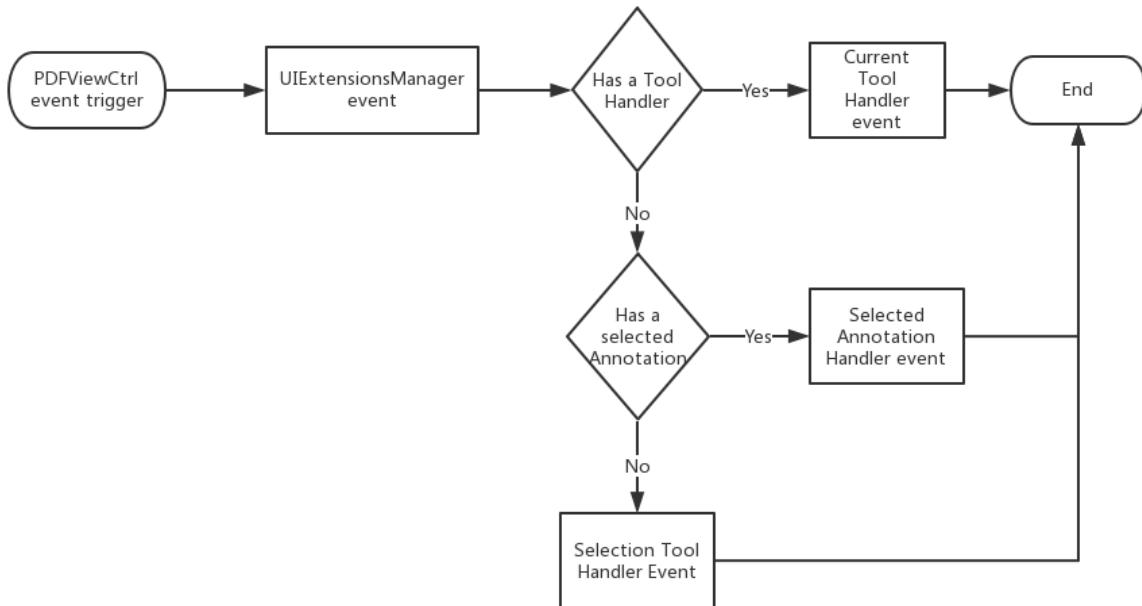


UIExtensionsManager 和 modules 之间的关系

Tool handler 与 annotation handler 处理来自 PDFViewCtrl 的触屏、手势等事件。当触屏和手势事件触发时，PDFViewCtrl 会将相应的事件传递给 UIExtensionsManager：

- 如果当前存在 tool handler, UIExtensionsManager 会将相应的事件传递给当前的 tool handle, 然后事件处理过程结束。
- 如果当前有选择 annotation, UIExtensionsManager 会将相应的事件传递给当前所选择的 annotation 对应的 annotation handler, 然后事件处理过程结束。
- 如果当前不存在 tool handler, 也没有选中的 annotation, 那么 UIExtensionsManager 会将相应的事件传递给 selection tool handler。Text Selection tool 用于文本选择相关事件的处理, 例如选择一段文本添加 highlight annotation。Blank Selection tool 用于空白处相关事件的处理, 例如在空白处添加 Note annotation。

备注: Tool Hander 和 Annotation Handler 不会同时响应事件。Tool Handler 主要用于 annotation 的创建(目前不支持 Link Annotation 的创建)、signature 的创建和文本选择。Annotation Handler 主要用于 annotation 的编辑以及表单填写。下图讲述了 Tool Handler 和 Annotation Handler 之间的事件响应流程。

*Tool Handler 和 Annotation Handler 之间的事件响应流程*

1.2.4 Foxit PDF SDK for Android 的主要功能

Foxit PDF SDK for Android 包括了一些主要的功能，用来帮助应用程序开发人员在快速实现他们所需的功能的同时减少开发成本。

功能	描述
PDF Document	打开和关闭文件，设置和获取 metadata。
PDF Page	解析、渲染、阅读、编辑文档页面。
Render	平台图像设备在 bitmap 上创建图像渲染引擎。
Reflow	重排页面内容。
Crop	裁剪 PDF 页面。
Text Select	文本选择。
Text Search	文本搜索，并且支持全文索引搜索。
Outline	定位和链接到文档中的兴趣点。
Reading Bookmark	标记文档中感兴趣的页面和段落位置。

Annotation	创建、编辑和移除 annotations。
Layers	添加、编辑和移除 PDF 层内容。
Attachments	添加、编辑和移除文档级的附件。
Form	支持 JavaScript 填表，通过 XFDF/FDF/XML 文件导入和导出表单数据。 支持创建文本域、复选框、单选按钮、组合框、列表框和签名域。
XFA	支持静态和动态 XFA。
Signature	签名 PDF 文档，验证签名，添加或删除签名域。 添加和验证第三方数字签名。 支持签名的长期验证 (LTV)。
Fill	用文本和符号填写扁平化表单（即非交互式表单）
Security	密码和证书加密 PDF 文档。
Pan and Zoom	调整视图中的放大倍数和位置以匹配 Pan&Zoom 缩略视图当中的矩形区域。
Print	打印 PDF 文档。
RMS	支持微软 IRMv1 和 IRMv2 标准的 RMS 解密。
Comparison	对比两个 PDF 文档，并且标记文档之间的差异。
Scanning	扫描纸质文档，并将其转换为 PDF 文档。
Speak	支持阅读 PDF 文档中的文本。
Split Screen	支持分屏。
Out of Memory	从内存不足中恢复运行。

备注：*Outline* 是 PDF 规范中的技术术语，在传统的桌面 PDF 阅读器中常叫做书签。*Reading bookmarks* 常用于移动端和平板的 PDF 阅读器中，用来标记阅读进度或者用户感兴趣的段落。*Reading bookmark* 在技术上并不是 *outline*，它存储在应用程序中而不是 PDF 本身。

Foxit PDF SDK for Android 支持鲁棒性的 PDF 应用程序

在有限内存的移动平台上开发鲁棒性的 PDF 应用程序是具有挑战性的。当内存分配失败，应用程序可能会 crash 或者意外退出。为了解决这个问题，Foxit PDF SDK for Android 提供了一种内存溢出 (OOM) 机制。

OOM 是 Foxit PDF SDK for Android 的一个高级功能，因为其本身的复杂性。OOM 机制的关键点是 Foxit PDF SDK for Android 会监视内存的使用情况，并在检测到 OOM 后自动执行恢复操作。在恢复的过程中，Foxit PDF SDK for Android 会自动重新加载文档和页面，将恢复到发生 OOM 之前的原始

状态。这意味着当前阅读的页面和位置，以及页面阅读模式(单页或者连续页面)都能够恢复，但是编辑相关的内容将会丢失。

Foxit PDF SDK for Android 在 PDFViewCtrl 类中提供一个属性"**shouldRecover**"。默认情况下，"**shouldRecover**"为 "**true**"。如果在检测到 OOM 时您不想开启自动恢复机制，您可以将"**shouldRecover**"设置为 "**false**"，如下所示：

```
PDFViewCtrl pdfViewerCtrl = new PDFViewCtrl(getActivity().getApplicationContext());
pdfViewerCtrl.shouldRecover = false;
```

此时，应用程序将会抛出异常，可能会crash或者意外退出。

1.3 评估

用户可申请下载 Foxit PDF SDK 的试用版本进行试用评估。试用版除了有试用期 10 天时间的限制以及生成的 PDF 页面上会有试用水印以外，其他都和标准认证版一样。当试用期到期后，用户需联系福昕销售团队并购买 licenses 以便继续使用 Foxit PDF SDK.

1.4 授权

程序开发人员需购买 licenses 授权才能在其解决方案中使用 Foxit PDF SDK。Licenses 授予用户发布基于 Foxit PDF SDK 开发的应用程序的权限。然而，在未经福昕软件公司授权下，用户不能将 Foxit PDF SDK 包中的任何文档、示例代码以及源代码分发给任何第三方机构。

1.5 关于此文档

此文档适用于需要将 Foxit PDF SDK for Android 集成到自己的应用程序中的开发人员。它旨在介绍以下章节：

- Section 1: 介绍 Foxit PDF SDK，特别是 Android 平台的 SDK。
- Section 2: 说明包的结构，以及运行 demos。
- Section 3: 介绍如何快速创建功能齐全的 PDF 阅读器。
- Section 4: 介绍如何自定义用户界面。
- Section 5: 介绍如何使用 Foxit PDF SDK core API。
- Section 6: 介绍如何创建自定义工具。
- Section 7: 介绍使用 Cordova 实现 Foxit PDF SDK。
- Section 8: 介绍使用 React Native 实现 Foxit PDF SDK。
- Section 9: 介绍使用 Xamarin 实现 Foxit PDF SDK。
- Section 10: 列出常见问题。
- Section 11: 提供技术支持信息。

2 入门指南

安装并集成 Foxit PDF SDK for Android 非常简单。您只需要几分钟就能见证其强大的功能。本指南主要介绍如何在 Android 平台使用 Foxit PDF SDK。本章的主要内容是包结构的介绍以及如何运行 demo。

2.1 系统要求

Android 设备要求

- Android 4.4 (API 19) 或更高版本
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32/64-bit Intel x86 CPU

Android Studio 3.2 or newer (支持 AndroidX)

包中 Demos 的运行环境：

- Android Studio 3.2
- JDK 1.8
- Gradle Version 4.6
- Gradle Build Tool 3.2

备注：从 7.2 版本开始，Foxit PDF SDK for Android 将只支持 AndroidX，而不再支持 Android support library。

2.2 包结构说明

下载 "foxitpdfsdk_8_1_android.zip" 包，解压到一个新的目录如 "foxitpdfsdk_8_1_android"，如 Figure 2-1 所示。其中解压包中包括如下的内容：

docs:	API 手册，开发文档和升级说明文档
icc_profile	输出预览 (output preview) 功能所使用的默认 icc profile 文件
libs:	License 文件，AAR，UI Extensions 组件源代码
samples:	Android 示例工程
getting_started_android.pdf:	Foxit PDF SDK for Android 快速入门
legal.txt:	法律和版权信息
release_notes.txt:	发布信息

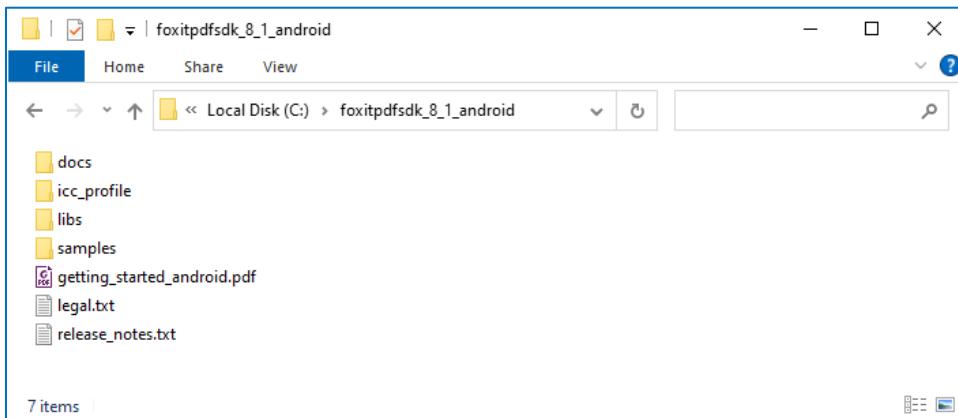


Figure 2-1

如 Figure 2-2 所示的"libs"文件夹下是 Foxit PDF SDK for Android 的核心组件，以及用于支持微软 RMS 的第三方库。

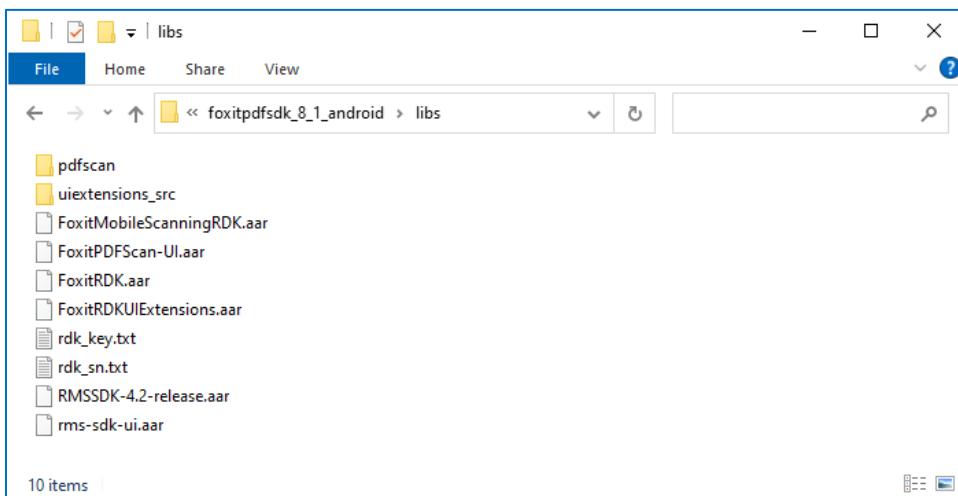


Figure 2-2

- **uiextensions_src** 工程 – 在"libs"文件夹下。它是一个开源库，包含了一些即用型的 UI 模块实现，可以帮助开发人员快速将功能齐全的 PDF 阅读器嵌入到他们的 Android 应用中。当然，开发人员也不是必须要使用默认的 UI，可以通过"uiextensions_src"工程为特定的应用灵活自定义和设计 UI。
- **FoxitRDK.aar** – 包含 JAR 包，其中包括 Foxit PDF SDK for Android 的所有 Java APIs，以及 ".so" 库。".so" 库是 SDK 的核心包含了 Foxit PDF SDK for Android 的核心函数。它针对每种架构单独编译，当期支持 armeabi-v7a, arm64-v8a, x86, 和 x86_64 架构。

- **FoxitRDKUIExtensions.aar** – 由"libs"目录下的 "**uiextensions_src**"工程 编译生成。包括 JAR 包，内置 UI 实现，以及 UI 所需要的资源文件，如图片，字符串、颜色值、布局文件以及其他 Android UI 资源。
- **pdfscan** 工程 - 是一个开源库，包含了扫描功能相关的 UI 实现，可以帮助开发人员快速将扫描功能集成到他们的 Android 应用中，或者根据需要自定义扫描功能的 UI。
- **FoxitMobileScanningRDK.aar** – 提供扫描功能所需要的库。
- **FoxitPDFScan-UI.aar** – 提供实现扫描功能所需 UI 的 Android Activities。
- **RMSDK-4.2-release.aar** – 微软权限管理系统的软件开发包。更多详细信息，请参考 <https://www.microsoft.com/en-ie/download/details.aspx?id=43673>.
- **rms-sdk-ui.aar** – 提供实现 RMS SDK 功能所需 UI 的 Android Activities。更多详细信息，请参考 <https://github.com/AzureAD/rms-sdk-ui-for-android>.

备注: 为了减小 *FoxitRDKUIExtensions.aar* 的文件大小, *Foxit PDF SDK for Android* 在 *uiextensions_src* 工程中使用 *shrink-code* 技术。如果您在编译 *Uiextensions_src* 工程时, 不需使用 *shrink-code*, 您可以在 *App* 下的 *build.gradle* 中通过设置 "minifyEnabled" 为 "**false**" 来进行禁用。关于 *shrink-code*, 请参考 <https://developer.android.com/studio/build/shrink-code.html>.

至此, 您应该初步了解了 *Foxit PDF SDK for Android* 包结构和内容, 接下来我们将进行详细介绍。

2.3 运行 demo

下载和安装 Android Studio IDE (<https://developer.android.com/studio/index.html>).

备注: 在本指南中, 不具体介绍 *Android Studio*、*Android SDK* 和 *JDK* 的安装步骤。如果您还没有安装, 请参考 *Android Studio* 的开发官网。

Foxit PDF SDK for Android 为开发人员提供了三种不同类型的 demos, 用来展示如何调用 SDK, 如 Figure 2-3 所示。

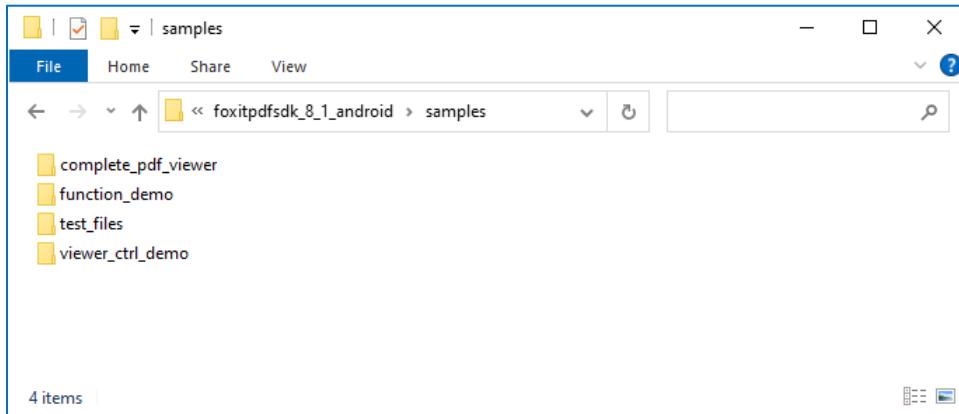


Figure 2-3

2.3.1 Function demo

Function demo 用来展示如何使用 Foxit PDF SDK for Android 的 core API 来实现一些 PDF 特定的功能。该 demo 包括如下的功能：

- **pdf2txt:** 从 PDF 文档中提取文本，并保存到一个 TXT 文件中。
- **outline:** 编辑大纲(也称为书签)的外观和主题。
- **annotation:** 向 PDF 页面添加 annotations 和以 JSON 文件格式导出 annotations。
- **docinfo:** 导出 PDF 文档信息到 TXT 文件中。
- **render:** 将指定的 PDF 页面渲染为位图。
- **signature:** 向 PDF 中添加签名，对 PDF 签名和验证签名。
- **image2pdf:** 将图像转化为 PDF 文件。
- **watermark:** 向 PDF 文件添加文本、图像和 PDF 页面水印。
- **search:** 搜索 PDF 文件。
- **graphics_objects:** 使用图形对象创建 PDF 文档。

在 Android Studio 中运行该 demo，请按如下的步骤：

- a) 在 Android Studio 中打开 demo，通过 "File -> New -> Import Project..." 或者 "File -> Open...", 然后找到 function_demo 所在的位置，选择 function_demo。点击"OK"。
- b) 开启一个 Android 设备或者模拟器(AVD)。在本章中，将使用模拟器 AVD 10.0 来运行 demo。Demo 所需要的测试文件在 demo 运行时会被自动拷贝到模拟器的存储卡中，测试文件在 "samples/test_files"文件夹下。
- c) 点击"Run -> Run 'app'" 来运行 demo。当在模拟器上安装完 APK 后，在弹出的窗口点击 "**Allow**" 允许 demo 访问设备上的文件。然后您就能看到如 Figure 2-4 所示的功能选项。

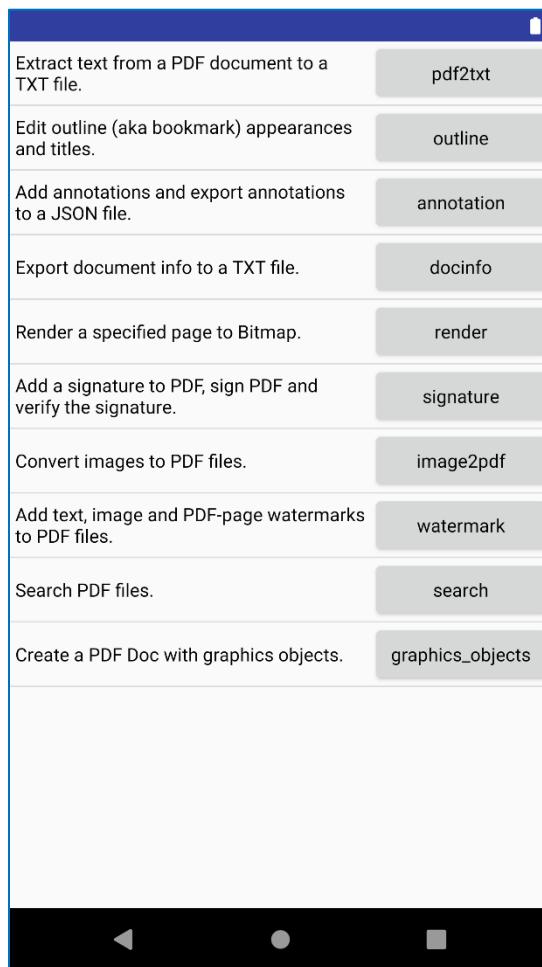


Figure 2-4

- d) 点击上图中的功能按钮去执行相应的操作。例如，点击 "pdf2txt"，则会弹出如 Figure 2-5 所示的消息框，提示转换后的文本文件保存的位置。请运行 demo 并自由体验其功能。

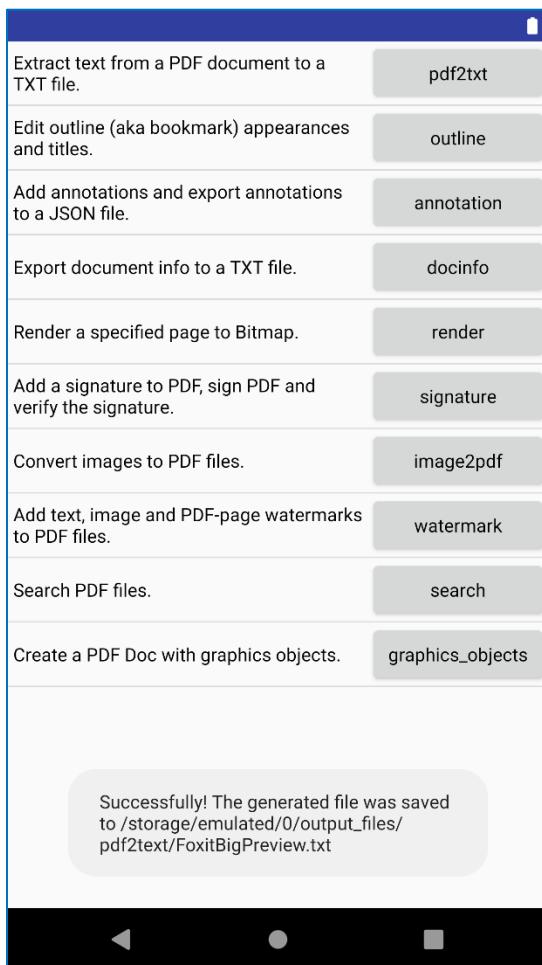


Figure 2-5

2.3.2 Viewer control demo

Viewer control demo 阐述如何使用 Foxit PDF SDK for Android 实现与 View Control 功能层相关的功能，比如操作 annotations (注释、高亮、下划线、删除线、波浪线等)，修改布局，文本搜索，大纲和页面缩略图。该 demo 的代码逻辑结构非常清晰和简单，开发人员可以快速定位 PDF 应用，比如 PDF 阅读器中某个功能的具体实现。并且通过这个 demo，开发人员可以更进一步的接触和了解 Foxit PDF SDK for Android 所提供的 APIs。

在 Android Studio 中运行该 demo，请参考 [Function demo](#) 中的步骤。

Viewer control demo 不会自动拷贝测试文件到 Android 设备或者模拟器中。该 demo 使用 "sample.pdf" (在"samples/test_files"文件夹下)作为测试文件，请确保在运行 demo 之前，您已经将该文件添加到 Android 设备或者模拟器中创建的"input_files" (或者"FoxitSDK"，取决于您在 demo 中的设置)。

Demo 成功运行后会如 Figure 2-6 所示。这里，使用 AVD 10.0 来运行 demo。



Figure 2-6

点击页面上的任意位置，会出现上下文操作栏，然后点击⋮(更多按钮)去查看更多的功能操作选项，如 Figure 2-7 所示。

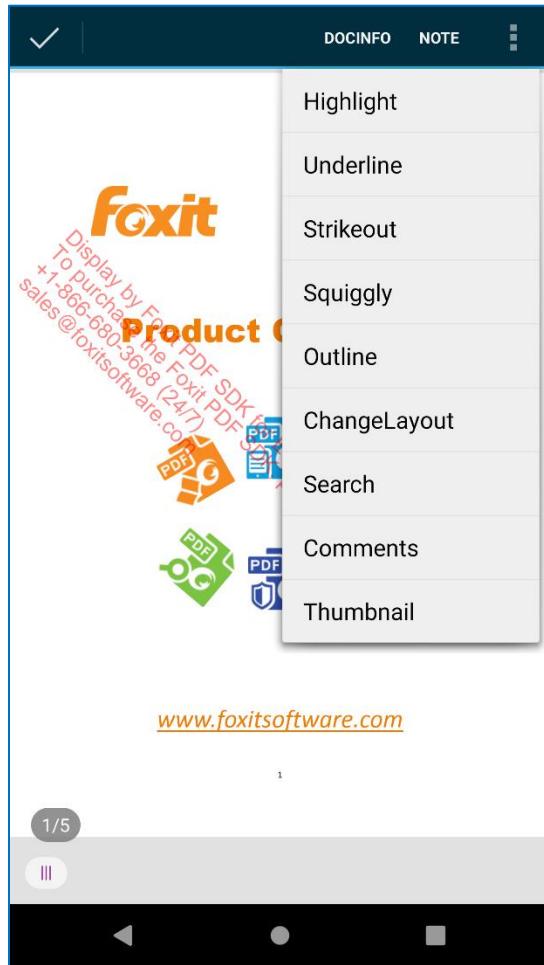


Figure 2-7

现在，可以选择其中一个选项执行，并且查看其结果。比如，点击"Outline"，您会看到该测试文档的大纲目录(outline 在 PDF 规范中指的是书签)，如 Figure 2-8 所示。您可以自行尝试其他的功能选项。

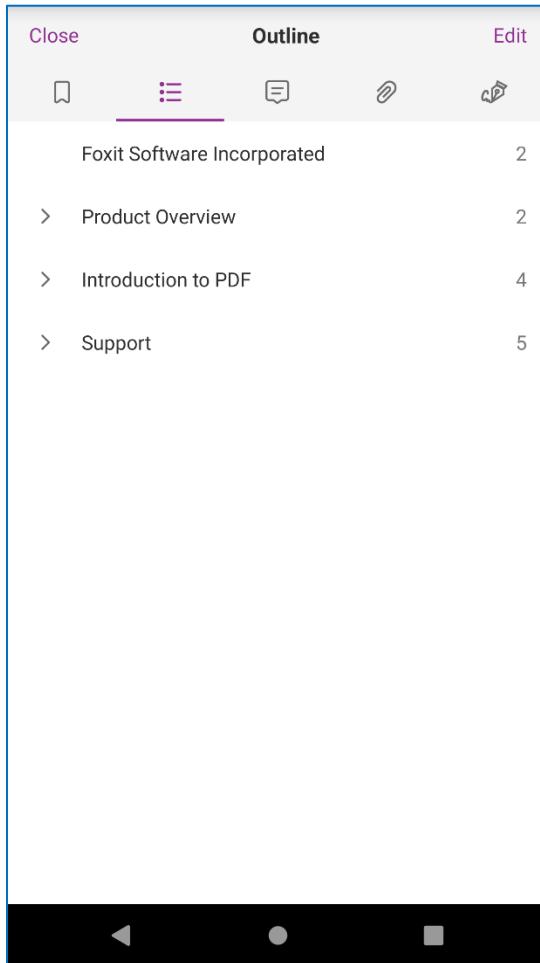


Figure 2-8

2.3.3 Complete PDF viewer demo

Complete PDF Viewer demo 阐述了如何通过使用 Foxit PDF SDK for Android 实现一个功能齐全的 PDF 阅读器，该阅读器几乎可以作为实际移动端的 PDF 阅读器使用。而且从 6.0 版本开始，支持多文件阅读模式。该 demo 使用了 Foxit PDF SDK for Android 所提供的所有功能和内置 UI 实现。

在 Android Studio 中运行该 demo，请参考 [Function demo](#) 中的步骤。

在运行该 demo 时，“samples\complete_pdf_viewer\app\src\main\assets”目录下的 “complete_pdf_viewer_guide_android.pdf” 和 “Sample.pdf” 文件将会被自动拷贝到模拟器的 “FoxitSDK”文件夹下。

这里，将使用 AVD 10.0 来运行 demo。当成功运行后，屏幕会列出 “complete_pdf_viewer_guide_android.pdf” 和 “Sample.pdf” 文档。如果您想要阅读多个文档，点击 切换到多文档阅读模式 (如 Figure 2-9 所示)。

备注: "complete_pdf_viewer_guide_android.pdf" 和 "Sample.pdf"会自动部署到您运行的设备中，因此您不需要手动将其添加到设备中。如果您想使用其他的 PDF 文档来测试该 demo，您需要手动将其添加到设备的 SD 卡中。

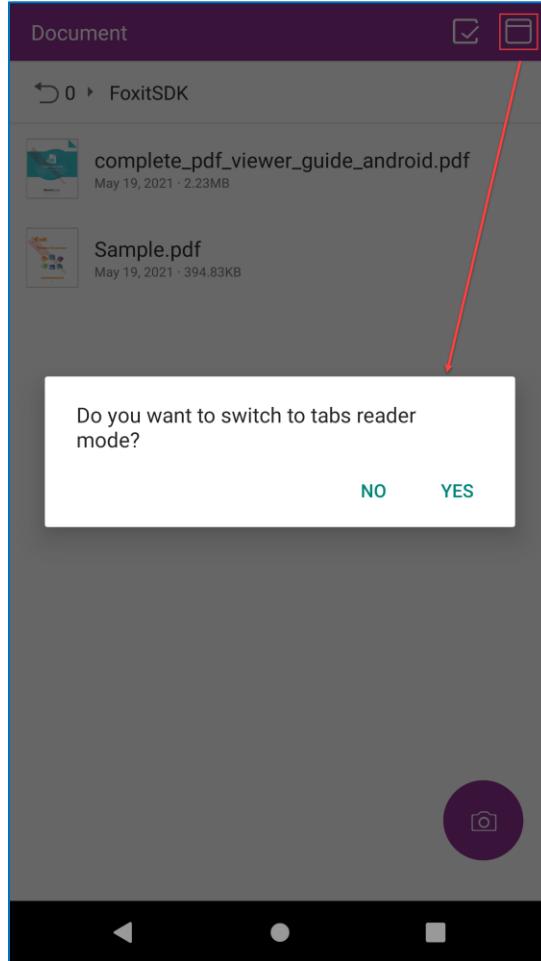


Figure 2-9

点击 **YES** 切换到多文档阅读模式。选择"complete_pdf_viewer_guide_android.pdf" 文档，点击 Back 按钮 ，再选择"Sample.pdf"，如 Figure 2-10 所示。现在，您可以通过切换选项卡浏览这两个文档。



Figure 2-10

该 demo 实现了一个功能齐全的 PDF 阅读器，请随意体验。

例如，它提供了页面缩略图功能。您可以点击底部工具栏上的缩略图菜单 ，然后可以看到如 Figure 2-11 所示的文档的缩略图。

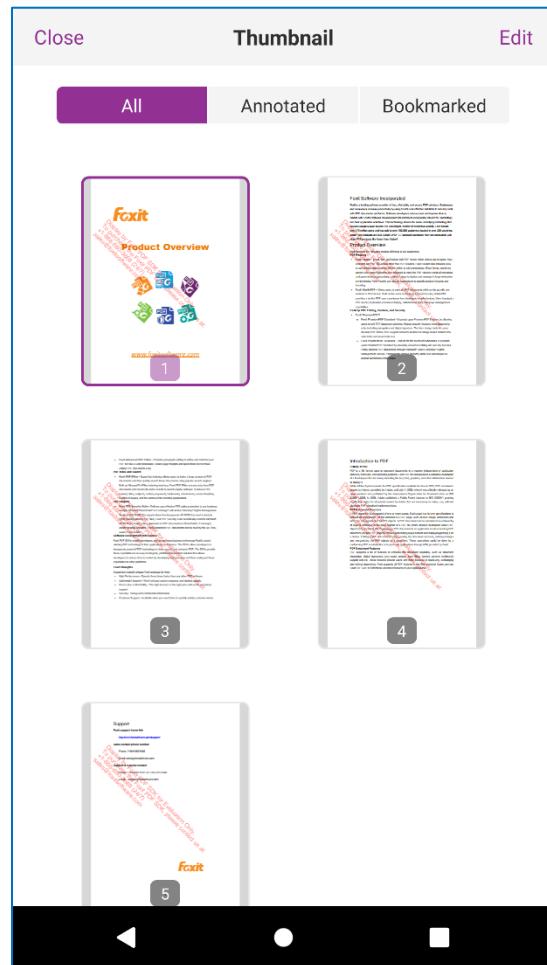


Figure 2-11

3 快速构建一个功能齐全的 PDF 阅读器

Foxit PDF SDK for Android 将所有的 UI 实现（包括应用程序的基本 UI 和即用型 UI 功能模块）封装在 UI Extensions 组件中，因此开发人员可以轻松快速通过几行代码构建一个功能齐全的 PDF 阅读器。本章将提供详细的教程来帮助您快速开始使用 Foxit PDF SDK for Android 在 Android 平台创建一个功能齐全的 PDF 阅读器。其主要包括以下的步骤：

- [创建一个新的 Android 工程](#)
- [集成 Foxit PDF SDK for Android 到您的应用程序](#)
- [初始化 Foxit PDF SDK for Android](#)
- [使用 PDFViewCtrl 显示 PDF 文档](#)
- [打开一个 RMS 加密的文档](#)
- [使用 UI Extensions 组件构建一个功能齐全的 PDF 阅读器](#)
- [基于功能齐全的 PDF 阅读器添加扫描功能](#)

3.1 创建一个新的 Android 工程

在本指南中，使用 Android Studio 4.1.2，以及 Android API 30.

打开 Android Studio，选择 **File -> New -> New Project...**，在 **Select a Project Template** 向导中，选择 "Empty Activity"，如 Figure 3-1 所示。然后点击 **Next**。

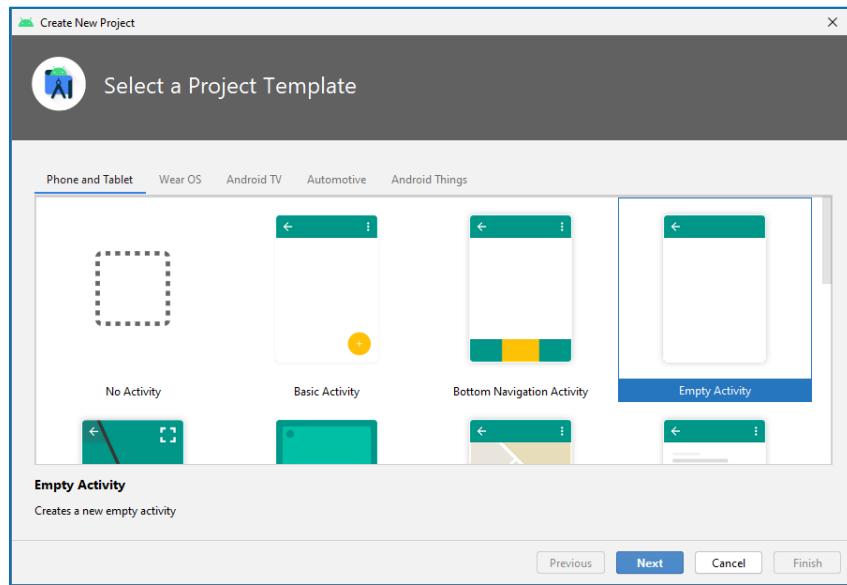


Figure 3-1

然后填写 **Configure your project** 对话框，如 Figure 3-2 所示。填写完后，点击 **Finish**。

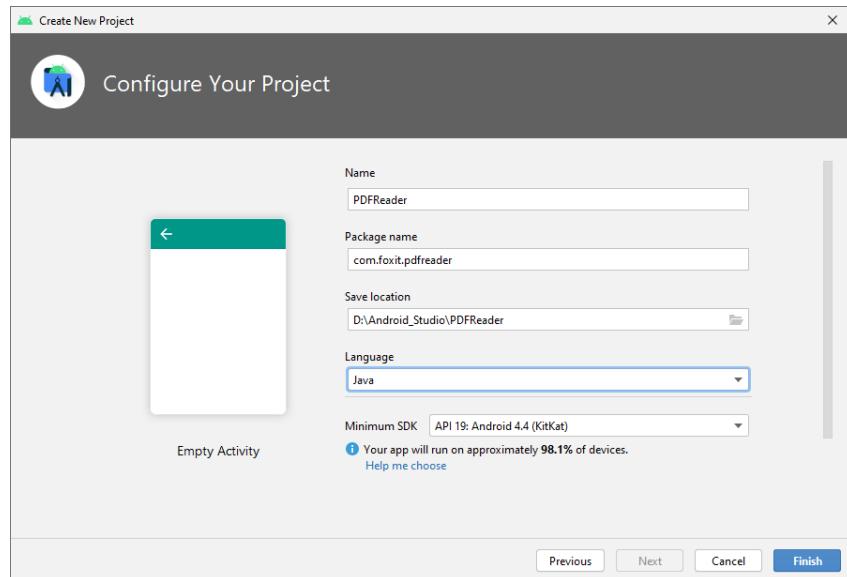


Figure 3-2

3.2 集成 Foxit PDF SDK for Android 到您的应用程序

备注: 在本章中，我们将使用默认的内置 UI 实现来开发该应用程序，为了简单和方便(直接使用 UI Extensions 组件，不需要源代码工程)，我们只需要添加以下的文件到 PDFReader 工程中。

- **FoxitRDK.aar** - 包含 JAR 包，其中包括 Foxit PDF SDK for Android 的所有 Java APIs，以及 ".so" 库。".so" 库是 SDK 的核心包含了 Foxit PDF SDK for Android 的核心函数。它针对每种架构单独编译，当期支持 armeabi-v7a, arm64-v8a, x86, 和 x86_64 架构。
- **FoxitRDKUIExtensions.aar** - 由 "libs" 目录下的 "**uiextensions_src**" 工程 编译生成。包括 JAR 包，内置 UI 实现，以及 UI 所需要的资源文件，如图片，字符串、颜色值、布局文件以及其他 Android UI 资源。

技巧: 在接下来的 "初始化 Foxit PDF SDK for Android"，"使用 PDFViewCtrl 显示 PDF 文档" 以及 "打开一个 RMS 加密的文档" 三小节中，不需要使用 UI Extensions 组件 (**FoxitRDKUIExtensions.aar**)，因此您可以先将 **FoxitRDK.aar** 添加到工程中。然后在您需要使用 UI Extensions 组件时再添加 **FoxitRDKUIExtensions.aar**，比如在 "使用 UI Extensions 组件构建一个功能齐全的 PDF 阅读器" 章节中。

添加上述的两个 AAR 文件到 PDFReader 工程，请切换到 "Project" 面板，然后按照如下的步骤：

- a) 从下载包的 "libs" 文件夹下拷贝 "**FoxitRDK.aar**" 和 "**FoxitRDKUIExtensions.aar**" 文件到 "PDFReader\app\libs" 文件夹下。

备注:

- 如果需要支持微软 RMS，您需要同时拷贝 "**RMS_SDK-4.2-release.aar**" 和 "**rms-sdk-ui.aar**" 文件。
- 如果需要支持 scanning 功能，您需要同时拷贝 "**FoxitMobileScanningRDK.aar**" 和 "**FoxitPDFScan-UI.aar**" 文件。

同步 PDFReader 工程，然后该工程将如 Figure 3-3 所示。

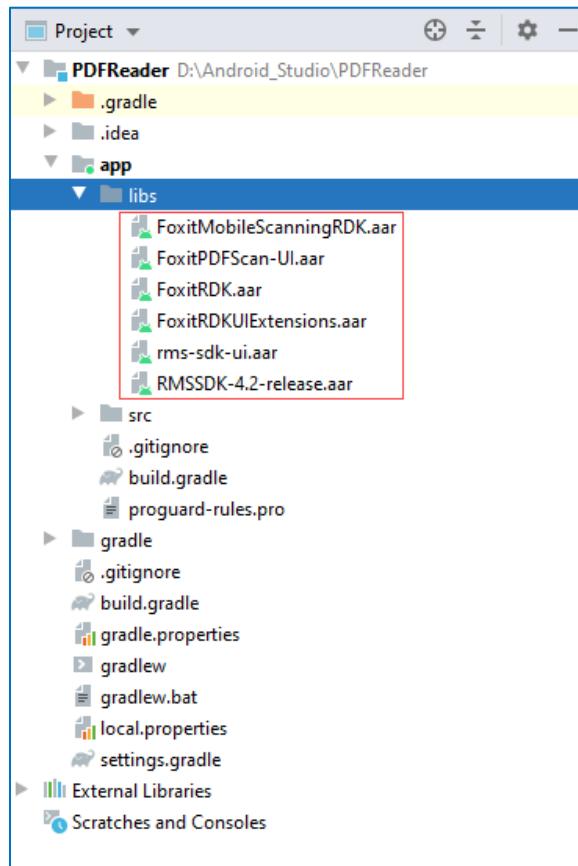


Figure 3-3

- b) 将"libs"目录定义为 repository。在 app 下面的 build.gradle 文件中，添加如下的配置：

build.gradle:

```
repositories {  
    flatDir {  
        dirs 'libs'  
    }  
}
```

- c) 启用 Multi-Dex。在 app 下面的 build.gradle 文件中，添加如下的代码：

```
...  
android {  
    compileSdkVersion 30  
    buildToolsVersion "30.0.3"  
    defaultConfig {  
        applicationId "com.foxit.pdfreader"  
        minSdkVersion 19  
        targetSdkVersion 30  
        versionCode 1
```

```
versionName "1.0"
testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
multiDexEnabled true
}

...
}

dependencies {
implementation 'androidx.multidex:multidex:2.0.1'
implementation 'androidx.appcompat:appcompat:1.2.0'
implementation 'com.google.android.material:material:1.3.0'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
}
...
```

- d) 将 Foxit PDF SDK for Android 作为工程的依赖项。在 app 下面的 "build.gradle" 文件中，添加 "**FoxitRDK.aar**", "**FoxitRDKUIExtensions.aar**" 以及相关支持的库到 dependencies。为简单起见，如下所示更新 dependencies：

```
dependencies {
implementation 'androidx.multidex:multidex:2.0.1'
implementation 'androidx.appcompat:appcompat:1.3.0'
implementation 'com.google.android.material:material:1.3.0'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
implementation(name: 'FoxitRDK', ext: 'aar')
implementation(name: 'FoxitRDKUIExtensions', ext: 'aar')
implementation 'com.edmodo:cropper:1.0.1'
// RMS
implementation 'com.microsoft.identity.client:msal:2.+'
implementation(name: 'RMSSDK-4.2-release', ext: 'aar')
implementation(name: 'rms-sdk-ui', ext: 'aar')
// Scanning
implementation(name: 'FoxitPDFScan-UI', ext: 'aar')
implementation(name: 'FoxitMobileScanningRDK', ext: 'aar')
implementation 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
// RxJava: Compare
implementation "io.reactivex.rxjava2:rxjava:2.2.16"
implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'
// Signature
implementation 'org.bouncycastle:bcpkix-jdk15on:1.64'
implementation 'org.bouncycastle:bcprov-jdk15on:1.64'
}
```

备注：

- (必选) Foxit PDF SDK for Android 依赖于 **com.google.android.material**, 如果 *dependencies* 中没有该依赖项, 则需要添加如下的条目:
implementation 'com.google.android.material:material:1.1.0'
- (可选) 如果您需要使用截图(Snapshot)功能(如在 Complete PDF viewer demo, 在右上方点击...，可以看到 Screen Capture 功能选项), 您需要添加如下的条目到 *dependencies* 中:
implementation 'com.edmodo:cropper:1.0.1'
- (可选) 如果您需要打开 RMS 加密的 PDF 文档, 您需要添加如下的条目到 *dependencies* 中。

implementation 'com.microsoft.identity.client:msal:2.+'

implementation(name: 'RMSSDK-4.2-release', ext: 'aar')

implementation(name: 'rms-sdk-ui', ext: 'aar')

备注: 您最好使用 RMS SDK 4.2 和 MSAL 2+ 版本, 否则可能会导致兼容性问题。

其次, 您需要在工程级的 "build.gradle" 文件中的 *repositories* 部分添加如下的配置:

```
allprojects {  
    repositories {  
        google()  
        jcenter()  
        maven {  
            url 'https://pkgs.dev.azure.com/MicrosoftDeviceSDK/DuoSDK-Public/\_packaging/Duo-SDK-Feed/maven/v1'  
        }  
    }  
}
```

有关更详细的说明, 请参阅 <https://github.com/AzureAD/microsoft-authentication-library-for-android>。

- (可选) 如果您需要使用扫描(scanning)功能(比如, 在 Complete PDF viewer demo 的启动页面, 可以看到扫描功能按钮选项 ), 则需要添加如下的条目到 *dependencies* 中。

implementation(name: 'FoxitMobileScanningRDK', ext: 'aar')

implementation(name: 'FoxitPDFScan-UI', ext: 'aar')

implementation 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'

- (可选) 如果您需要使用对比功能(比如, 在 Complete PDF viewer demo 的启动页面, 点击右上方的 , 可以看到对比功能选项), 则需要添加如下的条目到 *dependencies* 中。

```
implementation "io.reactivex.rxjava2:rxjava:2.2.16"
implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'
```

- (可选) 如果您需要使用签名功能，则需要添加如下的条目到 *dependencies* 中。

```
implementation 'org.bouncycastle:bcpkix-jdk15on:1.64'
implementation 'org.bouncycastle:bcprov-jdk15on:1.64'
```

此处，我们将如上所有的支持库都添加到 *dependencies* 中，因为稍后我们将构建一个功能齐全的 PDF 阅读器，该阅读器包含 Foxit PDF SDK for Android 所提供的所有的功能。在 APP 下的"build.gradle"中设置完成后，同步工程，然后您可以在 **External Libraries** 下看到 "**FoxitRDK.aar**", "**FoxitRDKUIExtensions.aar**", "**FoxitMobileScanningRDK.aar**", "**FoxitPDFScan-Ui.aar**", "universal-image-loader", "material", "cropper", "rxjava", "rxandroid", "bcpkix-jdk15on", "bcprov-jdk15on" 和 RMS 相关的包。

App 下的 "build.gradle" 的完整代码如下。

build.gradle:

```
plugins {
    id 'com.android.application'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.foxit.pdfreader"
        minSdkVersion 19
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        multiDexEnabled true
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

```
}

repositories {
    flatDir {
        dirs 'libs'
    }
}

dependencies {
    implementation 'androidx.multidex:multidex:2.0.1'
    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation(name: 'FoxitRDK', ext: 'aar')
    implementation(name: 'FoxitRDKUIExtensions', ext: 'aar')
    implementation 'com.edmodo:cropper:1.0.1'
    // RMS
    implementation 'com.microsoft.identity.client:msal:2.+'
    implementation(name: 'RMSSDK-4.2-release', ext: 'aar')
    implementation(name: 'rms-sdk-ui', ext: 'aar')
    // Scanning
    implementation(name: 'FoxitPDFScan-UI', ext: 'aar')
    implementation(name: 'FoxitMobileScanningRDK', ext: 'aar')
    implementation 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
    // RxJava: Compare
    implementation "io.reactivex.rxjava2:rxjava:2.2.16"
    implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'
    // Signature
    implementation 'org.bouncycastle:bcpkix-jdk15on:1.64'
    implementation 'org.bouncycastle:bcprov-jdk15on:1.64'
}
```

备注：使用AndroidX，`compileSdkVersion` 最低需要设置成28。在本工程中，我们将`compileSdkVersion` 和`targetSdkVersion` 设置为API 30。如果您也使用API 30，请确保您已经安装了Android 11.0，API 30 SDK 平台。如果没有，请首先打开Android SDK Manager 进行下载和安装。

3.3 初始话 Foxit PDF SDK for Android

在调用任何 API 之前，应用程序必须使用 license 初始化 Foxit PDF SDK for Android。

`Library.initializeApp(sn, key)`函数用于 SDK 库的初始化。试用 license 文件在下载包的"libs"文件夹下。当试用期结束后，您需要购买正式 license 以继续使用该 SDK。下面是 SDK 库初始化的示例代码。在下一节中将介绍该代码在 `PDFReader` 工程中的位置。

```
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.Library;
```

```
...  
  
int errorCode = Library.initialize("sn", "key");  
if (errorCode != Constants.e_ErrorSuccess)  
    return;
```

备注：参数 "sn" 的值在 "rdk_sn.txt" 中 ("SN=" 后面的字符串)， "key" 的值在 "rdk_key.txt" 中 ("Sign=" 后面的字符串)。

3.4 使用 PDFViewCtrl 显示 PDF 文档

到目前为止，我们已经在 *PDFReader* 工程中添加了 Foxit PDF SDK for Android 库，并且完成了 SDK 库的初始化。现在，我们将使用 PDFViewCtrl 通过几行代码来显示一个 PDF 文档。

备注：如果只需要显示一个 PDF 文档，则不需要 *UI Extensions* 组件。

显示一个 PDF 文档，请按照如下的步骤：

- 实例化一个 PDFViewCtrl 对象来显示一个 PDF 文档。

在 *MainActivity.java* 中，实例化一个 PDFViewCtrl 对象，调用 **PDFViewCtrl.openDoc** 函数打开和渲染 PDF 文档。

```
import com.foxit.sdk.PDFViewCtrl;  
...  
private PDFViewCtrl pdfViewCtrl = null;  
...  
pdfViewCtrl = new PDFViewCtrl(this);  
  
String path = "/mnt/sdcard/input_files/Sample.pdf";  
pdfViewCtrl.openDoc(path, null);  
  
SetContentView(pdfViewCtrl);
```

备注：请确保您已经将 "Sample.pdf" 文档添加到用于运行该工程的 *Android* 设备或者模拟器中已创建的 "input_files" 文件夹。

更新 *MainActivity.java*：

```
package com.foxit.pdfreader;  
  
import android.os.Bundle;  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.foxit.sdk.PDFViewCtrl;  
import com.foxit.sdk.common.Constants;
```

```
import com.foxit.sdk.common.Library;

public class MainActivity extends AppCompatActivity {

    private PDFViewCtrl pdfViewCtrl = null;

    // The value of "sn" can be found in the "rdk_sn.txt".
    // The value of "key" can be found in the "rdk_key.txt".
    private static String sn = "";
    private static String key = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // initialize the library.
        int errorCode = Library.initialize(sn, key);
        if (errorCode != Constants.e_ErrorSuccess)
            return;

        pdfViewCtrl = new PDFViewCtrl(this);
        String path = "/mnt/sdcard/input_files/Sample.pdf";
        pdfViewCtrl.openDoc(path, null);

        setContentView(pdfViewCtrl);
    }
}
```

- b) 设置 Android 设备或者模拟器的 SD 卡的读写权限。您需要添加额外的代码申请授权运行时权限。

在 **MainActivity.java** 中，添加如下的代码：

```
import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Build;

import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.annotation.NonNull;

...

private static final int REQUEST_EXTERNAL_STORAGE = 1;
private static final String[] PERMISSIONS_STORAGE = {
    Manifest.permission.READ_EXTERNAL_STORAGE,
    Manifest.permission.WRITE_EXTERNAL_STORAGE
};
```

```
// Require the authorization of runtime permissions.  
if (Build.VERSION.SDK_INT >= 23) {  
    int permission = ContextCompat.checkSelfPermission(this.getApplicationContext(),  
Manifest.permission.WRITE_EXTERNAL_STORAGE);  
    if (permission != PackageManager.PERMISSION_GRANTED) {  
        ActivityCompat.requestPermissions(this, PERMISSIONS_STORAGE, REQUEST_EXTERNAL_STORAGE);  
        return;  
    }  
}  
...  
  
@Override  
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
    if (requestCode == REQUEST_EXTERNAL_STORAGE && grantResults[0] ==  
PackageManager.PERMISSION_GRANTED) {  
        // Open and Reader a PDF document.  
        String path = "/mnt/sdcard/input_files/Sample.pdf";  
        pdfViewCtrl.openDoc(path, null);  
        setContentView(pdfViewCtrl);  
    } else {  
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
    }  
}
```

然后，**MainActivity.java** 的全部代码如下所示：

```
package com.foxit.pdfreader;  
  
import android.Manifest;  
import android.content.pm.PackageManager;  
import android.os.Build;  
import android.os.Bundle;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
import androidx.annotation.NonNull;  
  
import com.foxit.sdk.PDFViewCtrl;  
import com.foxit.sdk.common.Constants;  
import com.foxit.sdk.common.Library;  
  
public class MainActivity extends AppCompatActivity {  
  
    private PDFViewCtrl pdfViewCtrl = null;
```

```
private static final int REQUEST_EXTERNAL_STORAGE = 1;
private static final String[] PERMISSIONS_STORAGE = {
    Manifest.permission.READ_EXTERNAL_STORAGE,
    Manifest.permission.WRITE_EXTERNAL_STORAGE
};

// The value of "sn" can be found in the "rdk_sn.txt".
// The value of "key" can be found in the "rdk_key.txt".
private static String sn = "";
private static String key = "";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Initialize the library.
    int errorCode = Library.initialize(sn, key);
    if (errorCode != Constants.e_ErrSuccess)
        return;

    // Instantiate a PDFViewCtrl object.
    pdfViewCtrl = new PDFViewCtrl(this);

    // Require the authorization of runtime permissions.
    if (Build.VERSION.SDK_INT >= 23) {
        int permission = ContextCompat.checkSelfPermission(this.getApplicationContext(),
Manifest.permission.WRITE_EXTERNAL_STORAGE);
        if (permission != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, PERMISSIONS_STORAGE, REQUEST_EXTERNAL_STORAGE);
            return;
        }
    }

    // Open and Render a PDF document.
    String path = "/mnt/sdcard/input_files/Sample.pdf";
    pdfViewCtrl.openDoc(path, null);
    setContentView(pdfViewCtrl);
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
    if (requestCode == REQUEST_EXTERNAL_STORAGE && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
        // Open and Render a PDF document.
        String path = "/mnt/sdcard/input_files/Sample.pdf";
        pdfViewCtrl.openDoc(path, null);
    }
}
```

```
        setContentView(pdfViewCtrl);
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

- c) 更新 "PDFReader\app\src\main" 目录下的 AndroidManifest.xml 文件，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.foxit.pdfreader">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:requestLegacyExternalStorage="true"
        tools:replace="android:theme"
        android:theme="@style/Theme.PDFReader">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

备注：如果在 android 10 设备或者模拟器上运行该工程，则需要添加 "**android:requestLegacyExternalStorage="true"**" 来申请使用旧的存储模式。

在本节中，使用 AVD 10.0 (API 29) 来编译和运行该工程。

现在，我们已经使用 Foxit PDF SDK for Android 通过几行代码完成了一个显示 PDF 文档的简单的 Android 应用程序。下一步是运行该工程。

当编译完项目并在模拟器上安装 APK 后，在弹出的窗口点击"**Allow**" 允许工程访问设备上的文件。然后您将看到"Sample.pdf"文档显示如 Figure 3-4 所示。该示例应用程序具有一些基本的 PDF 功能，比如放大/缩小和翻页。您可以进行体验。



Figure 3-4

3.5 打开一个 RMS 加密的文档

从 6.2.1 版本开始，Foxit PDF SDK for Android 支持打开微软 RMS 加密的文档。

打开 RMS 加密的文档，您需要注意以下几点：

- 1) 在项目中添加 RMS 相关库依赖。请参考"[集成 Foxit PDF SDK for Android 到您的应用程序](#)"。
- 2) 在打开 RMS 加密的文档时需要有 UI 相关的操作，因此在打开文档之前需要设置关联的 activity。

```
pdfViewCtrl.setAttachedActivity(activity);
```

- 3) 处理来自 UI 操作的 activity 结果。在 **onActivityResult** 函数中调用 API "["pdfViewCtrl.onActivityResult\(\)](#)"。

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    pdfViewCtrl.onActivityResult(requestCode, resultCode, data);  
}
```

基于上一节"[使用 PDFViewCtrl 显示 PDF 文档](#)"，更新整个 **MainActivity.java**，如下所示：

```
package com.foxit.pdfreader;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.annotation.NonNull;

import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.Library;

public class MainActivity extends AppCompatActivity {

    private PDFViewCtrl pdfViewCtrl = null;

    private static final int REQUEST_EXTERNAL_STORAGE = 1;
    private static final String[] PERMISSIONS_STORAGE = {
        Manifest.permission.READ_EXTERNAL_STORAGE,
        Manifest.permission.WRITE_EXTERNAL_STORAGE
    };

    // The value of "sn" can be found in the "rdk_sn.txt".
    // The value of "key" can be found in the "rdk_key.txt".
    private static String sn = "";
    private static String key = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // initialize the library.
        int errorCode = Library.initialize(sn, key);
        if (errorCode != Constants.e_ErrSuccess)
            return;

        // Instantiate a PDFViewCtrl object.
        pdfViewCtrl = new PDFViewCtrl(this);

        // Set the associated activity for RMS UI operations.
        pdfViewCtrl.setAttachedActivity(this);
    }
}
```

```
// Require the authorization of runtime permissions.  
if (Build.VERSION.SDK_INT >= 23) {  
    int permission = ContextCompat.checkSelfPermission(this.getApplicationContext(),  
Manifest.permission.WRITE_EXTERNAL_STORAGE);  
    if (permission != PackageManager.PERMISSION_GRANTED) {  
        ActivityCompat.requestPermissions(this, PERMISSIONS_STORAGE, REQUEST_EXTERNAL_STORAGE);  
        return;  
    }  
}  
  
// Open and Render a PDF document.  
String path = "/mnt/sdcard/input_files/Sample_RMS.pdf";  
pdfViewCtrl.openDoc(path, null);  
setContentView(pdfViewCtrl);  
}  
  
@Override  
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull  
int[] grantResults) {  
    if (requestCode == REQUEST_EXTERNAL_STORAGE && grantResults[0] ==  
PackageManager.PERMISSION_GRANTED) {  
        // Open and Render a PDF document.  
        String path = "/mnt/sdcard/input_files/Sample_RMS.pdf";  
        pdfViewCtrl.openDoc(path, null);  
        setContentView(pdfViewCtrl);  
    } else {  
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
    }  
}  
  
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    pdfViewCtrl.onActivityResult(requestCode, resultCode, data);  
}
```

备注: 请确保您已经将一个 RMS 加密的文档比如 "Sample_RMS.pdf" 添加到用于运行该工程的 Android 设备或者模拟器中已创建的 "input_files" 文件夹。

当编译完项目并在模拟器上安装 APK 后，在弹出的窗口点击 "**Allow**" 允许工程访问设备上的文件。然后您将看到如 Figure 3-5 所示的界面，提示您输入组织电子邮件，密码，然后您就可以打开该 RMS 加密的文档。

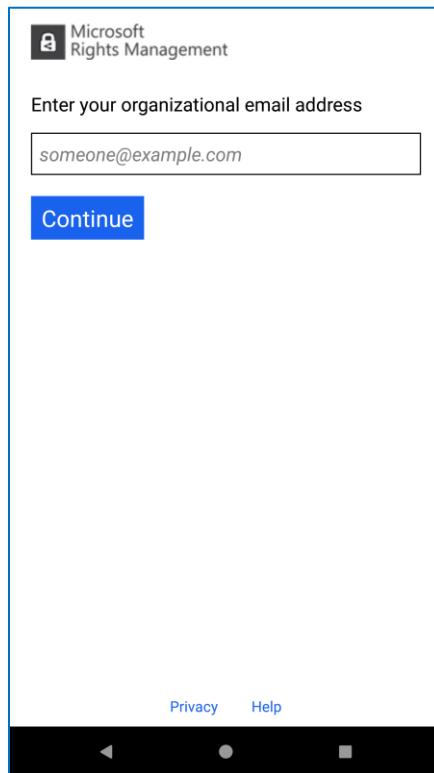


Figure 3-5

3.6 使用 UI Extensions 组件构建一个功能齐全的 PDF 阅读器

Foxit PDF SDK for Android 带有内置的 UI 设计，包括应用程序的基础 UI 和功能模块 UI。内置 UI 通过 Foxit PDF SDK for Android 实现，并且封装在 UI Extensions 组件中。因此，构建一个功能齐全的 PDF 阅读器变得越来越简单。您只需要实例化一个 UIExtensionsManager 对象，然后将其设置给 PDFViewCtrl。

实例化一个 UIExtensionsManager 对象，并且设置给 PDFViewCtrl

在"MainActivity.java"文件中，我们将添加包含 UIExtensionsManager 所需的代码。代码片段如下所示，后面您将看到"MainActivity.java"的完整代码。

- 将系统主题设置为 "No Title"模式，并且将窗口设置为全屏。

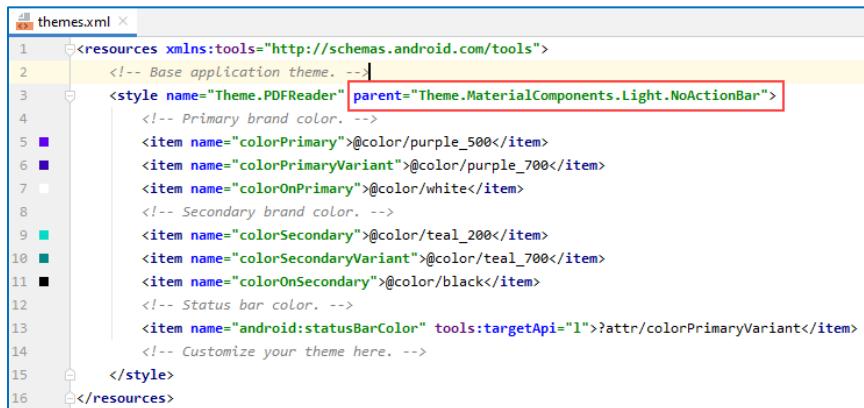
备注：UI Extensions 组件已自定义用户界面，因此您需要将系统主题设置为"No Title"模式，以及将窗口设置为全屏。否则，内置功能的布局可能会影响。

```
import android.view.Window;
import android.view.WindowManager;
...
```

```
// Turn off the title at the top of the screen.
this.requestWindowFeature(Window.FEATURE_NO_TITLE);

// Set the window to Fullscreen.
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
 WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

请确保将 "PDFReader\app\src\main\res\values\themes.xml" 文件中的 theme style 设置为 "Theme.MaterialComponents.Light.NoActionBar", 如下所示:



- b) 添加代码实例化一个 UIExtensionsManager 对象，并且将其设置给 PDFViewCtrl。

```
import com.foxit.uiextensions.UIExtensionsManager;
...
private UIExtensionsManager uiExtensionsManager = null;
...
uiExtensionsManager = new UIExtensionsManager(this.getApplicationContext(), pdfViewCtrl);
uiExtensionsManager.setAttachedActivity(this);
uiExtensionsManager.onCreate(this, pdfViewCtrl, savedInstanceState);
pdfViewCtrl.setUIExtensionsManager(uiExtensionsManager);
```

- c) 打开和渲染一个 PDF 文档，设置内容视图。

调用 UIExtensionsManager.openDocument() 函数打开和渲染 PDF 文档，而不是调用 PDFViewCtrl.openDoc() 函数。

```
import com.foxit.uiextensions.UIExtensionsManager;
...
String path = "/mnt/sdcard/input_files/Sample.pdf";
uiExtensionsManager.openDocument(path, null);
setContentView(uiExtensionsManager.getContentView());
```

更新 MainActivity.java :

备注：添加Activity 生命周期事件，否则某些功能可能无法正常使用。

```
package com.foxit.pdfreader;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.os.Build;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.Window;
import android.view.WindowManager;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.annotation.NonNull;

import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.Library;
import com.foxit.uiextensions.UIExtensionsManager;

public class MainActivity extends AppCompatActivity {

    private PDFViewCtrl pdfViewCtrl = null;
    private UIExtensionsManager uiExtensionsManager = null;

    private static final int REQUEST_EXTERNAL_STORAGE = 1;
    private static final String[] PERMISSIONS_STORAGE = {
        Manifest.permission.READ_EXTERNAL_STORAGE,
        Manifest.permission.WRITE_EXTERNAL_STORAGE
    };

    // The value of "sn" can be found in the "rdk_sn.txt".
    // The value of "key" can be found in the "rdk_key.txt".
    private static String sn = " ";
    private static String key = " ";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // initialize the library.
```

```
int errorCode = Library.initialize(sn, key);
if (errorCode != Constants.e_ErrSuccess)
    return;

// Turn off the title at the top of the screen.
this.requestWindowFeature(Window.FEATURE_NO_TITLE);

// Set the window to Fullscreen.
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

// Instantiate a PDFViewCtrl object.
pdfViewCtrl = new PDFViewCtrl(this);

// Set the associated activity for RMS UI operations.
pdfViewCtrl.setAttachedActivity(this);

// Initialize a UIExtensionsManager object and set it to PDFViewCtrl.
uiExtensionsManager = new UIExtensionsManager(this.getApplicationContext(), pdfViewCtrl);
uiExtensionsManager.setAttachedActivity(this);
uiExtensionsManager.onCreate(this, pdfViewCtrl, savedInstanceState);
pdfViewCtrl.setUIExtensionsManager(uiExtensionsManager);

// Require the authorization of runtime permissions.
if (Build.VERSION.SDK_INT >= 23) {
    int permission = ContextCompat.checkSelfPermission(this.getApplicationContext(),
Manifest.permission.WRITE_EXTERNAL_STORAGE);
    if (permission != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, PERMISSIONS_STORAGE, REQUEST_EXTERNAL_STORAGE);
        return;
    }
}

// Open and Render a PDF document.
String path = "/mnt/sdcard/input_files/Sample.pdf";
uiExtensionsManager.openDocument(path, null);
setContentView(uiExtensionsManager.getContentView());
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[]
grantResults) {
    if (requestCode == REQUEST_EXTERNAL_STORAGE && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
        // Open and Render a PDF document.
        String path = "/mnt/sdcard/input_files/Sample.pdf";
        uiExtensionsManager.openDocument(path, null);
        setContentView(uiExtensionsManager.getContentView());
    }
}
```

```
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    pdfViewCtrl.onActivityResult(requestCode, resultCode, data);
}

@Override
public void onStart() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onStart(this);
    }
    super.onStart();
}

@Override
public void onStop() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onStop(this);
    }
    super.onStop();
}

@Override
public void onPause() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onPause(this);
    }
    super.onPause();
}

@Override
public void onResume() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onResume(this);
    }
    super.onResume();
}

@Override
protected void onDestroy() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onDestroy(this);
    }
}
```

```
super.onDestroy();
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onConfigurationChanged(this, newConfig);
    }
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (uiExtensionsManager != null && uiExtensionsManager.onKeyDown(this, keyCode, event))
        return true;
    return super.onKeyDown(keyCode, event);
}
}
```

更新 AndroidManifest.xml

添加"<uses-permission android:name="android.permission.CAMERA"/>"授权工程访问摄像机的权限。

添加"<uses-permission android:name="android.permission.RECORD_AUDIO"/>"授权工程录制音频和视频的权限。如果不添加，音频和视频功能将无法正常使用。

添加"<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>"授权手机设备浮动窗口的权限。如果不添加，Pan and Zoom 功能可能无法正常使用。

添加

"**android:configChanges="keyboardHidden|orientation|locale|layoutDirection|screenSize"**"属性，以确保在旋转屏幕时只执行 onConfigurationChanged()函数，而不会重新调用 activity 生命周期。如果不添加，签名功能可能无法正常使用。

更新 AndroidManifest.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.foxit.pdfreader">

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />

```

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:requestLegacyExternalStorage="true"
    tools:replace="android:theme"
    android:theme="@style/Theme.PDFReader">
    <activity android:name=".MainActivity"
        android:configChanges="keyboardHidden|orientation|locale|layoutDirection|screenSize">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>

</manifest>
```

运行工程

在本节中，使用 AVD 10.0 (API 29) 来编译和运行该工程。当编译完项目并在模拟器上安装 APK 后，在弹出的窗口点击“Allow”允许工程访问设备上的文件。然后您将看到“Sample.pdf”文档显示如 Figure 3-6 所示。到目前为止，该工程是一个功能齐全的 PDF 阅读器，包含 Complete PDF viewer demo 中的所有功能，并且支持打开 RMS 加密的文档。请您随意体验。



Figure 3-6

3.7 基于功能齐全的 PDF 阅读器添加扫描功能

扫描功能是一个单独的模块，没有封装在 UI Extensions 组件中。因此，如果您需要在工程中使用该功能，那么请添加如下的核心代码来调用 scan 模块：

```
import com.foxit.pdfscan.PDFScanManager;
import androidx.fragment.app.DialogFragment;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
...
// Initialize the scan module.
long framework1 = 0;
long framework2 = 0;
PDFScanManager.initializeScanner(this.getApplication(), framework1, framework2);

long compression1 = 0;
long compression2 = 0;
PDFScanManager.initializeCompression(this.getApplication(), compression1, compression2);

if (PDFScanManager.isInitializeScanner() && PDFScanManager.isInitializeCompression()) {
    final PDFScanManager pdfScanManager = PDFScanManager.instance();
    pdfScanManager.showUI(MainActivity.this);
```

```
} else {
    Toast.makeText(getApplicationContext(), " ", Toast.LENGTH_SHORT).show();
}
```

对于 **PDFScanManager.initializeScanner** 和 **PDFScanManager.initializeCompression** 接口，如果您将第二和第三个参数设置为 0，则扫描后的图片会带有水印。如果您需要去掉水印，请联系 Foxit 销售或者技术支持团队来获取授权的 key。

基于上一节，添加一个新的 button 来调用 scan 模块。

更新 **MainActivity.java**，如下所示：

(假设您已经将 "samples\complete_pdf_viewer\app\src\main\res\drawable" 文件夹下的 "**fx_floatbutton_scan.xml**" 拷贝到 "PDFReader\app\src\main\res\drawable" 文件夹下。)

```
package com.foxit.pdfreader;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.os.Build;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.Toast;

import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.Library;
import com.foxit.uiextensions.UIExtensionsManager;
import com.foxit.pdfscan.PDFScanManager;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

public class MainActivity extends AppCompatActivity {

    private PDFViewCtrl pdfViewCtrl = null;
    private UIExtensionsManager uiExtensionsManager = null;

    private static final int REQUEST_EXTERNAL_STORAGE = 1;
    private static final String[] PERMISSIONS_STORAGE = {
```

```
Manifest.permission.READ_EXTERNAL_STORAGE,
Manifest.permission.WRITE_EXTERNAL_STORAGE
};

// The value of "sn" can be found in the "rdk_sn.txt".
// The value of "key" can be found in the "rdk_key.txt".
private static String sn = " ";
private static String key = " ";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Initialize the library.
    int errorCode = Library.initialize(sn, key);
    if (errorCode != Constants.e_ErrSuccess)
        return;

    // Turn off the title at the top of the screen.
    this.requestWindowFeature(Window.FEATURE_NO_TITLE);

    // Set the window to Fullscreen.
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
    WindowManager.LayoutParams.FLAG_FULLSCREEN);

    // Instantiate a PDFViewCtrl object.
    pdfViewCtrl = new PDFViewCtrl(this);

    // Set the associated activity for RMS UI operations.
    pdfViewCtrl.setAttachedActivity(this);

    uiExtensionsManager = new UIExtensionsManager(this.getApplicationContext(), pdfViewCtrl);
    uiExtensionsManager.setAttachedActivity(this);
    uiExtensionsManager.onCreate(this, pdfViewCtrl, savedInstanceState);
    pdfViewCtrl.setUIExtensionsManager(uiExtensionsManager);

    // Require the authorization of runtime permissions.
    if (Build.VERSION.SDK_INT >= 23) {
        int permission = ContextCompat.checkSelfPermission(this.getApplicationContext(),
Manifest.permission.WRITE_EXTERNAL_STORAGE);
        if (permission != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, PERMISSIONS_STORAGE, REQUEST_EXTERNAL_STORAGE);
            return;
        }
    }

    // Open and Render a PDF document.
    String path = "/mnt/sdcard/input_files/Sample.pdf";
    uiExtensionsManager.openDocument(path, null);

    RelativeLayout rootView = new RelativeLayout(getApplicationContext());
```

```
    RelativeLayout.LayoutParams layoutParams = new
RelativeLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup
    .LayoutParams.MATCH_PARENT);
rootView.addView(uiExtensionsManager.getContentView(), layoutParams);
rootView.addView(getScanButton());
setContentView(rootView);
}

// Get a scan button.
private View getScanButton() {
    ImageView ivScan = new ImageView(this);
    ivScan.setImageResource(R.drawable.fx_floatbutton_scan);
    ivScan.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            if (!PDFScanManager.isInitializeScanner()) {
                long framework1 = 0;
                long framework2 = 0;
                PDFScanManager.initializeScanner(MainActivity.this.getApplication(), framework1, framework2);
            }

            if (!PDFScanManager.isInitializeCompression()) {
                long compression1 = 0;
                long compression2 = 0;
                PDFScanManager.initializeCompression(MainActivity.this.getApplication(), compression1,
compression2);
            }

            if (PDFScanManager.isInitializeScanner() && PDFScanManager.isInitializeCompression()) {
                final PDFScanManager pdfScanManager = PDFScanManager.instance();
                pdfScanManager.showUI(MainActivity.this);
            } else {
                Toast.makeText(getApplicationContext(), "You are not authorized to use this add-on module,
please contact us for upgrading your license.", Toast.LENGTH_SHORT).show();
            }
        }
    });
}

    RelativeLayout.LayoutParams layoutParams = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,
RelativeLayout.LayoutParams.WRAP_CONTENT);

    layoutParams.bottomMargin = 120;
    layoutParams.rightMargin = 50;
    layoutParams.addRule(RelativeLayout.ALIGN_PARENT_BOTTOM);
    layoutParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
    ivScan.setLayoutParams(layoutParams);
    return ivScan;
}
```

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    if (requestCode == REQUEST_EXTERNAL_STORAGE && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
        // Open and Render a PDF document.
        String path = "/mnt/sdcard/input_files/Sample.pdf";
        uiExtensionsManager.openDocument(path, null);
        setContentView(uiExtensionsManager.getContentView());
    } else {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    pdfViewCtrl.onActivityResult(requestCode, resultCode, data);
}

@Override
public void onStart() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onStart(this);
    }
    super.onStart();
}

@Override
public void onStop() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onStop(this);
    }
    super.onStop();
}

@Override
public void onPause() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onPause(this);
    }
    super.onPause();
}

@Override
public void onResume() {
    if (uiExtensionsManager != null) {
        uiExtensionsManager.onResume(this);
    }
    super.onResume();
}
```

```
@Override  
protected void onDestroy() {  
    if (uiExtensionsManager != null) {  
        uiExtensionsManager.onDestroy(this);  
    }  
    super.onDestroy();  
}  
  
@Override  
public void onConfigurationChanged(Configuration newConfig) {  
    super.onConfigurationChanged(newConfig);  
    if (uiExtensionsManager != null) {  
        uiExtensionsManager.onConfigurationChanged(this, newConfig);  
    }  
}  
  
@Override  
public boolean onKeyDown(int keyCode, KeyEvent event) {  
    if (uiExtensionsManager != null && uiExtensionsManager.onKeyDown(this, keyCode, event))  
        return true;  
    return super.onKeyDown(keyCode, event);  
}
```

当编译完项目并在模拟器上安装 APK 后，在弹出的窗口点击“Allow”允许工程访问设备上的文件。然后您将看到如 Figure 3-7 所示的界面，点击 scan 按钮可以开始扫描文档。



Figure 3-7

3.8 分区存储 (Scoped Storage) 处理

Android 11 的发布对分区存储做了进一步的改进。使用分区存储，设备文件的访问会变得更加严格，同时也更加容易。分区存储可以更好地保护用户的隐私，以及提供更加安全的数据保护。兼容分区存储的应用程序为面向用户的数据提供了特定的文件夹，以及提供了一个专有的沙盒文件夹用于存储其所需的文件，该沙盒文件夹是其他任何应用程序都无法访问和使用的。分区存储能够为应用程序生成的文件创建第二个文件夹。

如果您需要在工程中禁用分区存储，可以选择如下任意一种的配置：

- 在 APP 下的 **build.gradle** 文件中，将 `targetSdkVersion` 设置为 `<=28`。
- 在 APP 下的 **build.gradle** 文件中，将 `targetSdkVersion` 设置为 `=29`，然后在 **AndroidManifest.xml** 文件中，在 `application` 节点中添加 `"android:requestLegacyExternalStorage="true""`。

如果 `targetSdkVersion >= 30`，您可以添加如下的配置来减少分区存储对您工程的影响：

在 **AndroidManifest.xml** 文件中：

- 在 manifest 节点中，添加 "<uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE"/>"。
- 在 application 节点中，添加 "android:requestLegacyExternalStorage="true"" 和 "android:preserveLegacyExternalStorage="true""。

如果在分区存储被启用时，您需要去适配分区存储，则可以参考 complete PDF Viewer demo 以获取更详细的信息。

4 自定义 UI

Foxit PDF SDK for Android 为开发人员提供了一个简单、干净和友好的用户界面，可以快速构建一个功能齐全的 PDF 应用程序而不需花费太多的时间在设计上。此外，自定义用户页面也非常简单。Foxit PDF SDK for Android 提供了 UI Extensions 组件的源代码（包含即用型 UI 模块的实现），这样开发人员可以根据需要灵活自定义界面的外观。

从 4.0 版本开始，开发人员可以通过一个配置文件对功能进行自定义。

从 5.0 版本开始，内置 UI 中的任何元素都是可配置的，并且为开发人员提供了更高级的 APIs 和更强大的配置文件用来进一步自定义 UI 元素，比如显示或隐藏一个特定的面板，top/bottom toolbar，top/bottom toolbar 中的菜单项，以及 View setting bar 和 More Menu View 中的菜单项。

从 6.3 版本开始，配置文件被进一步改进，提供了更多设置选项对 UI 进行自定义，包括权限管理和 UI 元素的属性。

从 8.0 版本开始，UI Extensions Component 的内置 UI 进行了全新的改版。

以下部分将介绍如何通过配置文件、APIs、源代码自定义功能模块、权限管理和 UI 元素。

4.1 通过配置文件自定义 UI

通过配置文件，开发人员可以轻松选择功能模块，设置权限管理和 UI 元素的属性，而无需编写任何额外的代码或者重新设计应用程序的 UI。

4.1.1 JSON 文件介绍

配置文件可以作为 JSON 文件提供，也可以直接在代码中编写。我们建议您使用 JSON 文件格式，因为其可以更直观，更清晰的查看和配置各个选项。

您可以参考 Foxit PDF SDK for Android 包中

"samples\complete_pdf_viewer\app\src\main\res\raw"文件夹下的 JSON 文件。其内容如下所示：

```
{  
  "modules": {  
    "readingbookmark": true,  
    "outline": true,  
    //    "annotations":true,  
    "annotations": {  
      "highlight": true,  
      "underline": true,  
    }  
  }  
}
```

```
"squiggly": true,
"strikeout": true,
"insert": true,
"replace": true,
"line": true,
"rectangle": true,
"oval": true,
"arrow": true,
"pencil": true,
"eraser": true,
"typewriter": true,
"textbox": true,
"callout": true,
"note": true,
"stamp": true,
"polygon": true,
"cloud": true,
"polyline": true,
"measure": true,
"image": true,
"audio": true,
"video": true,
"redaction": true
},
"thumbnail": true,
"attachment": true,
"signature": true,
"fillSign": true,
"search": true,
"navigation": true,
"form": true,
"selection": true,
"encryption": true
"multipleSelection": true
},
"permissions": {
    "runJavaScript": true,
    "copyText": true,
    "disableLink": false
},
"uiSettings": {
    "pageMode": "Single",
    "continuous": false,
    "reflowBackgroundColor": "#FFFFFF",
    "zoomMode": "FitWidth",
    "colorMode": "Normal",
    "mapForegroundColor": "#5d5b71",
```

```
"mapBackgroundColor": "#00001b",
"disableFormNavigationBar": false,
"highlightForm": true,
"highlightFormColor": "#200066cc",
"highlightLink": true,
"highlightLinkColor": "#16007fff",
"fullscreen": true,
"annotations": {
    "continuouslyAdd": true,
    "highlight": {
        "color": "#ffff00", "opacity": 1.0
    },
    "areaHighlight": {
        "color": "#ffff00", "opacity": 1.0
    },
    "underline": {
        "color": "#66cc33", "opacity": 1.0
    },
    "squiggly": {
        "color": "#993399", "opacity": 1.0
    },
    "strikeout": {
        "color": "#ff0000", "opacity": 1.0
    },
    "insert": {
        "color": "#993399", "opacity": 1.0
    },
    "replace": {
        "color": "#0000ff", "opacity": 1.0
    },
    "line": {
        "color": "#ff0000", "opacity": 1.0, "thickness": 2
    },
    "rectangle": {
        "color": "#ff0000", "opacity": 1.0, "thickness": 2
    },
    "oval": {
        "color": "#ff0000", "opacity": 1.0, "thickness": 2
    },
    "arrow": {
        "color": "#ff0000", "opacity": 1.0, "thickness": 2
    },
    "pencil": {
        "color": "#ff0000", "opacity": 1.0, "thickness": 2
    },
    "polygon": {
        "color": "#ff0000", "opacity": 1.0, "thickness": 2
    }
}
```

```
},
"cloud": {
  "color": "#ff0000", "opacity": 1.0, "thickness": 2
},
"polyline": {
  "color": "#ff0000", "opacity": 1.0, "thickness": 2
},
"typewriter": {
  "textColor": "#0000ff",
  "opacity": 1.0,
  "textFace": "Courier",
  "textSize": 18
},
"textbox": {
  "color": "#ff0000",
  "textColor": "#0000ff",
  "opacity": 1.0,
  "textFace": "Courier",
  "textSize": 18
},
"callout": {
  "color": "#ff0000",
  "textColor": "#0000ff",
  "opacity": 1.0,
  "textFace": "Courier",
  "textSize": 18
},
"note": {
  "color": "#ff0000",
  "opacity": 1.0,
  "icon": "Comment"
},
"attachment": {
  "color": "#ff0000",
  "opacity": 1.0,
  "icon": "PushPin"
},
"image": {
  "rotation": 0,
  "opacity": 1.0
},
"measure": {
  "color": "#ff0000",
  "opacity": 1.0,
  "thickness": 2,
  "scaleFromUnit": "inch",
  "scaleToUnit": "inch",
  "x": 0.0,
  "y": 0.0
}
```

```
"scaleFromValue": 1,  
"scaleToValue": 1  
},  
"redaction": {  
    "fillColor": "#000000",  
    "textColor": "#ff0000",  
    "textFace": "Courier",  
    "textSize": 12  
}  
},  
"form": {  
    "textField": {  
        "textColor": "#000000",  
        "textFace": "Courier", // textFace:"Courier"/"Helvetica"/"Times"  
        "textSize": 0 // 0 means auto-adjust font size (textSize >= 0)  
    },  
    "checkBox": {  
        "textColor": "#000000"  
    },  
    "radioButton": {  
        "textColor": "#000000"  
    },  
    "comboBox": {  
        "textColor": "#000000",  
        "textFace": "Courier", // textFace:"Courier"/"Helvetica"/"Times"  
        "textSize": 0, // 0 means auto-adjust font size (textSize >= 0)  
        "customText": false  
    },  
    "listBox": {  
        "textColor": "#000000",  
        "textFace": "Courier", // textFace:"Courier"/"Helvetica"/"Times"  
        "textSize": 0, // 0 means auto-adjust font size (textSize >= 0)  
        "multipleSelection": false  
    },  
    "signature": {  
        "color": "#000000", "thickness": 8  
    }  
}
```

备注:

- 上述JSON文件中的值是配置项的默认值。如果某些配置项不在JSON文件中，则将使用其默认值。例如，如果您注释掉“highlight”: true，“，但高亮功能仍然是可用的。

- 只有附件 (attachment) 注释是不受 "annotations" 的子项控制的。点击顶部工具栏 **Home** 旁边的 图标，选择 **Comment**，可以看到附件注释，如 Figure 4-1 所示。

"**attachment**": **true**, " 控制了 attachments 面板和 attachment 注释。如果您将其设置为 "**false**", 则两者都将被禁用。如果您需要隐藏 **Comment** 中的所有工具，那么您需要将 "annotations" 和 "attachment" 同时设置为 "**false**"。



Figure 4-1

4.1.2 配置项描述

JSON 配置文件包括三个部分：功能模块，权限管理和 UI 设置（例如，UI 元素属性）。本节将详细介绍这些配置项。

配置功能模块

备注：功能模块项的值类型是 **bool**，其中 "**true**" 表示启用该功能模块，"**false**" 表示将禁用该功能模块。默认值为 "**true**"。

功能模块	描述
readingbookmark	用户定义书签
outline	PDF 文档书签
annotations	注释模块集合

(highlight, underline, squiggly, strikeout, insert, replace, line, rectangle, oval, arrow, pencil, eraser, typewriter, textbox, callout, note, stamp, polygon, cloud, polyline, measure, image, audio, video, redaction)	
thumbnail	PDF 页面缩略图显示和页面管理
attachment	PDF 文档附件和附件注释
signature	电子签名和手写签名
fillSign	用文本和符号填写扁平化表单（即非交互式表单）
search	文本搜索
navigation	PDF 页面导航
form	表单填写和表单数据导入导出
selection	文本选择
encryption	PDF 加密
multipleSelection	选择多个 annotations

配置权限管理

备注：配置项的值类型为 **bool**，其中 "**true**" 表示将启用该权限，"**false**" 表示将禁用该权限。
runJavaScript 和 **copyText** 的默认值为 "**true**"，**disableLink** 的默认值为 "**false**"。

权限管理	描述
runJavaScript	是否允许执行 JavaScript
copyText	是否允许复制文本
disableLink	是否禁用超链接

配置 UI 项及其属性

UI 配置子项	描述/属性	值类型	可选值	默认值	备注
pageMode	页面显示模式	String	Single/ Facing/ CoverLeft/ CoverMiddle/ CoverRight/ Reflow	Single	动态 XFA 文件不支持 Reflow 模式。
continuous	是否连续的显示单页页面	Bool	true/false	false	True 表示连续显示，false 表示不连续显示。该配置项在"Reflow"模式下无效。
reflowBackgroundColor	Reflow (重排) 页面的背景	RGB	---	#FFFFFF	
zoomMode	页面缩放模式	String	FitWidth/FitPage	FitWidth	
colorMode	页面颜色显示模式	String	Normal/Night/Map	Normal	"Night" 是一种特殊的"Map"模式。

UI 配置子项	描述/属性	值类型	可选值	默认值	备注	
mapForegroundColor	页面显示的前景颜色	RGB	---	#5d5b71	只有在 "colorMode" 设置为"Map"时，该配置项才有效。	
mapBackgroundColor	页面显示的背景颜色	RGB	---	#00001b	只有在 "colorMode" 设置为"Map"时，该配置项才有效。	
disableFormNavigationBar	是否禁用表单的辅助导航栏	Bool	true/false	false		
highlightForm	是否高亮表单域	Bool	true/false	true		
highlightFormColor	表单高亮颜色	ARGB		#200066cc	包括 alpha 通道，并且对动态 xfa 文件无效。	
highlightLink	是否高亮超链接	Bool	true/false	true		
highlightLinkColor	超链接高亮颜色	ARGB		#16007fff	包括 alpha 通道。	
fullscreen	是否全屏显示	Bool	true/false	true	当"fullscreen" 设置为"true"时，文档将以全屏方式显示。如果用户点击页面，工具栏将会出现。如果 5 秒内无任何动作，工具栏和其他辅助工具按钮将自动隐藏。	
annotations	continuousl yAdd		Bool	true/false	true	是否连续添加某个注释
	highlight	color	RGB		#ffff00	
		opacity	numeric	[0.0-1.0]	1.0	
	areaHighlig ht	color	RGB		#ffff00	
		opacity	numeric	[0.0-1.0]	1.0	如果区域超出页面，则使用默认的配置。
	underline	color	RGB		#66cc33	
		opacity	numeric	[0.0-1.0]	1.0	
	squiggly	color	RGB		#993399	
		opacity	numeric	[0.0-1.0]	1.0	
	strikeout	color	RGB		#ff0000	
		opacity	numeric	[0.0-1.0]	1.0	
	insert	color	RGB		#993399	
		opacity	numeric	[0.0-1.0]	1.0	

UI 配置子项		描述/属性	值类型	可选值	默认值	备注
replace	color	RGB			#0000ff	
	opacity	numeric	[0.0-1.0]	1.0		
line	color	RGB			#ff0000	
	opacity	numeric	[0.0-1.0]	1.0		
	thickness	numeric	[1-12]	2		
rectangle	color	RGB			#ff0000	
	opacity	numeric	[0.0-1.0]	1.0		
	thickness	numeric	[1-12]	2		
oval	color	RGB			#ff0000	
	opacity	numeric	[0.0-1.0]	1.0		
	thickness	numeric	[1-12]	2		
arrow	color	RGB			#ff0000	
	opacity	numeric	[0.0-1.0]	1.0		
	thickness	numeric	[1-12]	2		
pencil	color	RGB			#ff0000	
	opacity	numeric	[0.0-1.0]	1.0		
	thickness	numeric	[1-12]	2		
polygon	color	RGB			#ff0000	
	opacity	numeric	[0.0-1.0]	1.0		
	thickness	numeric	[1-12]	2		
cloud	color	RGB			#ff0000	
	opacity	numeric	[0.0-1.0]	1.0		
	thickness	numeric	[1-12]	2		
polyline	color	RGB			#ff0000	
	opacity	numeric	[0.0-1.0]	1.0		
	thickness	numeric	[1-12]	2		
typewriter	textColor	RGB			#0000ff	
	opacity	numeric	[0.0-1.0]	1.0		
	textFace	String	Courier/ Helvetica/ Times	Courier	文本字体名称。 如果设置为非法 值，则使用默认字 体。	
	textSize	Integer	>=1	18		
textbox	color	RGB			#ff0000	
	textColor	RGB			#0000ff	
	opacity	numeric	[0.0-1.0]	1.0		
	textFace	String	Courier/ Helvetica/ Times	Courier	文本字体名称。 如果设置为非法 值，则使用默认字 体。	
	textSize	Integer	>=1	18		
callout	color	RGB			#ff0000	
	textColor	RGB			#0000ff	
	opacity	numeric	[0.0-1.0]	1.0		
	textFace	String	Courier/ Helvetica/	Courier	文本字体名称。	

UI 配置子项		描述/属性	值类型	可选值	默认值	备注
note				Times		如果设置为非法值，则使用默认字体。
		textSize	Integer	>=1	18	
		color	RGB		#ff0000	
		opacity	numeric	[0.0-1.0]	1.0	
		icon	String	Comment/ Key/ Note/ Help/ NewParagraph/ Paragraph/ Insert	Comment	如果设置为非法值，则使用默认值。
		attachment	color	RGB	#ff0000	
			opacity	numeric	[0.0-1.0]	1.0
			icon	String	Graph/ PushPin/ Paperclip/ Tag	PushPin
		image	rotation	numeric	0/90/180/270	0
			opacity	numeric	[0.0-1.0]	1.0
measure		color	RGB		#ff0000	
		opacity	numeric	[0.0-1.0]	1.0	
		thickness	numeric	[1-12]	2	
		scaleFromUnit	String	pt/m/cm/mm/in ch/p/ft/yd	inch	缩放的基准单位。
		scaleToUnit	String	pt/m/cm/mm/in ch/p/ft/yd	inch	缩放的目标单位。
		scaleFromValue	numeric		1	缩放的基准数值。
		scaleToValue	numeric		1	缩放的目标数值。
redaction		fillColor	RGB		#000000	
		textColor	RGB		#ff0000	
		textFace	String	Courier/ Helvetica/ Times	Courier	文本字体名称。 如果设置为非法值，则使用默认字体。
		textSize	Integer	>=1	12	
		form	textFace	Courier/ Helvetica/ Times	Courier	文本字体名称。 如果设置为非法值，则使用默认字体。
form	textField	textColor	RGB		#000000	
		textFace	String	Courier/ Helvetica/ Times	Courier	文本字体名称。 如果设置为非法值，则使用默认字体。
		textSize	Integer	>=0	0	0 表示自动调整字体大小。
	checkbox	textColor	RGB		#000000	
	radioButton	textColor	RGB		#000000	

UI 配置子项		描述/属性	值类型	可选值	默认值	备注	
comboBox		textColor	RGB		#000000		
		textFace	String	Courier/ Helvetica/ Times	Courier	文本字体名称。 如果设置为非法 值，则使用默认字 体。	
		textSize	Integer	>=0	0	0 表示自动调整字 体大小。	
		customText			false	True 表示允许自定 义文本。 False 表示不允许自 定义文本。	
listBox		textColor	RGB		#000000		
		textFace	String	Courier/ Helvetica/ Times	Courier	文本字体名称。 如果设置为非法 值，则使用默认字 体。	
		textSize	Integer	>=0	0	0 表示自动调整字 体大小。	
		multipleSelection			false	True 表示支持多 选。 False 表示不支持多 选。	
signature		color	RGB		#000000		
		thickness	numeric	[1-12]	8		

4.1.3 使用配置文件实例化一个 **UIExtensionsManager** 对象

在 3.6 小节 "使用 UI Extensions 组件构建一个功能齐全的 PDF 阅读器"，我们已经介绍了如何实例化 **UIExtensionsManager**，而且使用这种方式，所有的内置 UI 框架将会被默认加载。在本节中，我们将提供另外一种使用配置文件来实例化一个 **UIExtensionsManager**，以便开发人员可以根据需要轻松自定义 UI。

请参阅以下代码使用配置文件实例化 **UIExtensionsManager** 对象。

备注：在这里，我们假设您已经将名为"uiextensions_config.json"的JSON 文件放到 "PDFReader\app\src\main\res\raw"文件夹下(请注意，您需要自己创建"raw" 文件夹)。

在 "MainActivity.java" 中：

```
import com.foxit.uiextensions.config.Config;
...
private PDFViewCtrl pdfViewCtrl = null;
private UIExtensionsManager uiExtensionsManager = null;
```

```
// Initialize a PDFViewCtrl object.  
pdfViewCtrl = new PDFViewCtrl(this);  
  
// Get the config file, and set it to UIExtensionsManager.  
InputStream stream =  
this.getApplicationContext().getResources().openRawResource(R.raw.uiextensions_config);  
Config config = new Config(stream);  
  
// Initialize a UIExtensionManager object with Configuration file, and set it to PDFViewCtrl.  
uiExtensionsManager = new UIExtensionsManager(this.getApplicationContext(), pdfViewCtrl, config);  
pdfViewCtrl.setUIExtensionsManager(uiExtensionsManager);  
uiExtensionsManager.setAttachedActivity(this);  
uiExtensionsManager.onCreate(this, pdfViewCtrl, savedInstanceState);
```

备注：在上述代码中，我们使用一个配置文件来实例化 `UIExtensionsManager`。如果您不想使用配置文件，可参考 3.6 小节 "[使用 UI Extensions 组件构建一个功能齐全的 PDF 阅读器](#)"。

4.1.4 通过配置文件自定义 UI 的示例

在本节中，我们将向您展示如何在您的项目中自定义功能模块、权限管理和 UI 设置（例如，UI 元素属性）。您会发现这些自定义都非常容易的实现，您只需要修改配置文件。下面列出了一些操作示例。

备注：为了方便起见，我们将在 "samples" 文件夹下的 "**complete_pdf_viewer**" demo 中进行演示。

在 Android Studio 中打开 "**complete_pdf_viewer**" demo。在 "complete_pdf_viewer\app\src\main\res\raw" 文件夹下找到配置文件 "uiextensions_config.json"。

示例 1：禁用"readingbookmark" 和 "navigation" 功能模块。

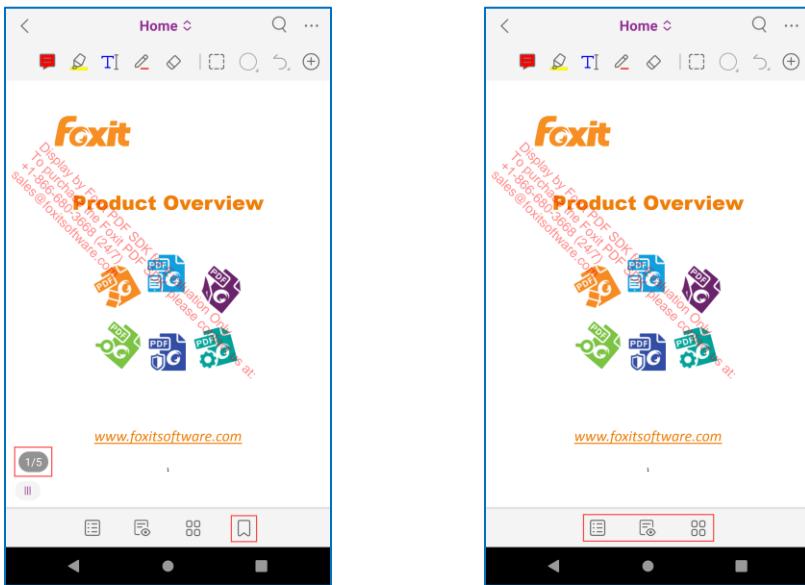
在 JSON 文件中，将 "readingbookmark" 和 "navigation" 的值设置为 "false"，如下所示：

```
"readingbookmark": false,  
"navigation": false,
```

然后，重新编译和运行该 demo。如下列出了前后对比图：

修改前：

修改后：



"readingbookmark" 和 "navigation" 功能模块被移除了。

示例 2：禁用超链接。

在 JSON 文件中，将 "disableLink" 的值设置为 "true"，如下所示：

```
"permissions": {  
    "runJavaScript": true,  
    "copyText": true,  
    "disableLink": true  
},
```

然后，重新编译和运行该 demo，您会发现当您单击超链接时没有任何响应。

示例 3：将高亮颜色从黄色设置为红色。

在 JSON 文件中，将 "highlight" 的 color 属性设置为 "#ff0000"，如下所示：

```
"highlight": {  
    "color": "#ff0000", "opacity": 1.0 },
```

然后，重新编译和运行该 demo。如下列出了前后对比图：

修改前：

修改后：



高亮颜色变为红色了。

4.2 通过 APIs 自定义 UI 元素

在 4.0 版本中，Foxit PDF SDK for Android 支持自定义显示或者隐藏整个 top toolbar 或者 bottom toolbar，从 5.0 版本开始，提供了 APIs 去自定义显示或者隐藏一个特定的面板，top/bottom toolbar、View setting bar 和 More Menu view 上面的菜单项，方便开发人员在内置 UI 框架下对 UI 元素进行修改。

从 8.0 版本开始，UI Extensions Component 的内置 UI 进行了全新的改版。

备注：为了方便起见，我们将在 "samples" 文件夹下的 "**complete_pdf_viewer**" demo 中向您展示如何通过 APIs 对 UI 元素进行自定义。我们假设您没有修改过 demo 中的 "uiextensions_config.json" 文件，也就是说 UI Extensions 组件中的所有内置 UI 都是启用的。

4.2.1 自定义 top/bottom toolbar

对于 top/bottom toolbar，您可以执行以下操作：

1. 显示或者隐藏 top/bottom toolbar。
2. 添加自定义菜单项。
3. 移除某个特定的菜单项。
4. 移除 toolbar 上所有的菜单项。
5. 显示或者隐藏某个特定的菜单项。
6. 添加自定义 toolbar。

7. 移除某个特定的 toolbar。
8. 设置 toolbar 的背景色。
9. 获取 toolbar 上特定位置的菜单项个数。
10. 移除在 top toolbar 中心位置的特定 tab。

Table 4-1 列出了用于自定义 top/bottom toolbar 相关的 APIs。

Table 4-1

<code>void enableTopToolbar(boolean isEnabled)</code>	启用或者禁用 top toolbar。
<code>void enableBottomToolbar(boolean isEnabled)</code>	启用或者禁用 bottom toolbar。
<code>boolean addItem(BarName barName, BaseBar.TB_Position gravity, Baseltem item, int index);</code>	在 toolbar 上添加自定义菜单项。
<code>boolean addItem(BarName barName, BaseBar.TB_Position gravity, int textId, int resId, int index, IItemClickListener clickListener);</code>	在 toolbar 上添加默认菜单项。
<code>boolean addItem(BarName barName, BaseBar.TB_Position gravity, CharSequence text, int index, IItemClickListener clickListener);</code>	在 toolbar 上添加默认的文本菜单项。
<code>boolean addItem(BarName barName, BaseBar.TB_Position gravity, Drawable drawable, int index, IItemClickListener clickListener);</code>	在 toolbar 上添加默认的图片菜单项。
<code>IBaseltem getItemByIndex(BarName barName, BaseBar.TB_Position gravity, int index);</code>	通过索引获取菜单项。
<code>void setItemVisibility(BarName barName, BaseBar.TB_Position gravity, int index, int visibility);</code>	设置菜单项视图的状态：可见或者不可见。
<code>int getItemVisibility(BarName barName, BaseBar.TB_Position gravity, int index);</code>	返回菜单项视图的状态：可见或者不可见。
<code>int getItemCount(BarName barName, BaseBar.TB_Position gravity);</code>	通过 IBarsHandler.BarName 和 BaseBar.TB_Position 获取菜单项个数。
<code>boolean removeItem(BarName barName, BaseBar.TB_Position gravity, int index);</code>	移除 toolbar 上指定索引位置的菜单项。
<code>boolean removeItem(BarName barName, BaseBar.TB_Position gravity, Baseltem item);</code>	移除 toolbar 上指定索引位置的菜单项。
<code>void removeAllItems(BarName barName);</code>	移除 toolbar 上所有的菜单项。
<code>boolean addCustomToolBar(BarName barName, View view);</code>	通过 BarName 添加自定义 toolbar。
<code>boolean removeToolBar(BarName barName);</code>	通过 BarName 移除 toolbar。
<code>void setBackgroundColor(BarName barName, int color);</code>	设置 toolbar 的背景色。
<code>void setBackgroundResource(BarName barName, int resid);</code>	将从指定的资源文件中加载背景样式。
<code>Baseltem getItem(BarName barName, BaseBar.TB_Position gravity, int tag);</code>	通过 tag 获取菜单项，如果 tag 不存在，则返回 null。
<code>void removeTab (int type);</code>	移除 top toolbar 中心位置的特定 tab 以及其子工具栏。

为了更好的定位需要添加新菜单或者移除已有菜单的位置，定义了两个比较重要的枚举类型，如下所示：

```
enum BarName {  
    TOP_BAR,  
    BOTTOM_BAR;  
}  
  
enum TB_Position {  
    Position_LT,  
    Position_CENTER,  
    Position_RB;  
}
```

备注：

1. 对于平板设备，其 UI 界面上移除了 bottom toolbar。
2. 在 top toolbar 上添加菜单项，需要将 **BaseBar.TB_Position** 设置为 **Position_LT** 或者 **Position_RB**。在 bottom toolbar 上添加菜单项，需要将 **BaseBar.TB_Position** 设置为 **Position_CENTER**。否则，菜单项之间可能会重叠。
3. 对于手机设备，Bottom toolbar 是一个部分，而 top toolbar 分成两个部分，因此 toolbar 一共有三个部分，每个部分都有单独的 index。对于平板设备，其没有 bottom toolbar。(见 Figure 4-2)
4. 为了获得最佳的 UI 显示效果，建议 top toolbar 的文本字符数不超过 15，bottom toolbar 不超过 8. 如果超出建议的字符数，可能会导致 view 排版混乱。

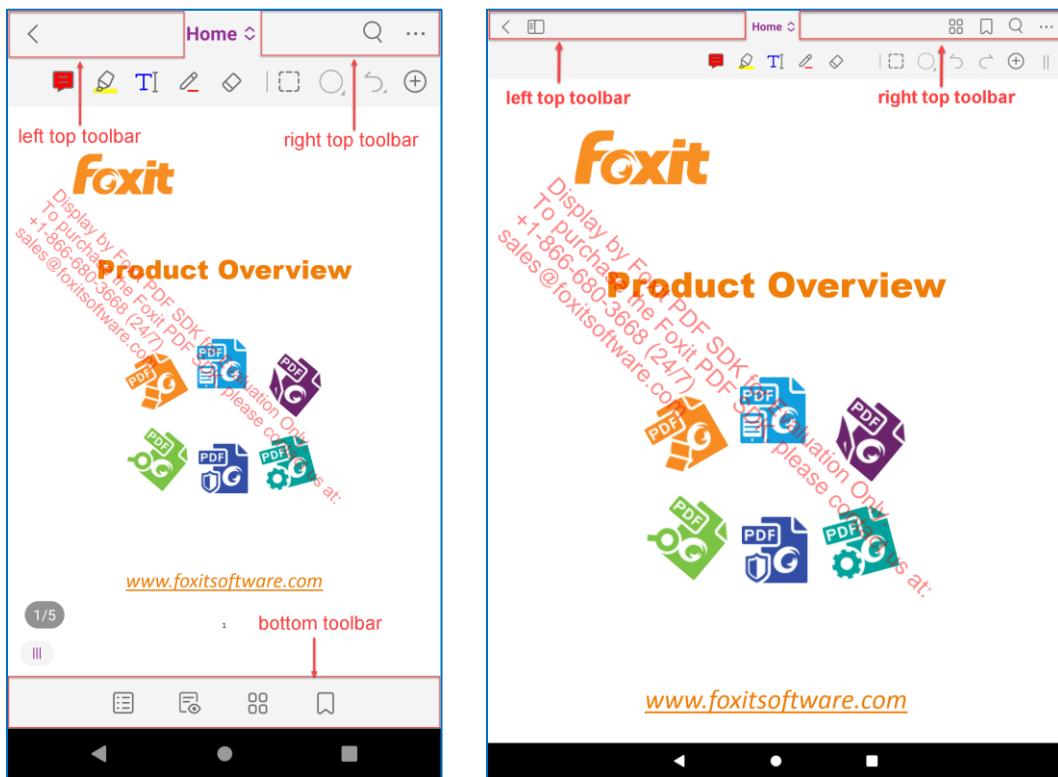


Figure 4-2

在下面的示例中，我们将在"samples"文件夹下的 "**complete_pdf_viewer**" demo 中向您展示如何通过 APIs 对 top/bottom toolbar 进行自定义。

在 Android Studio 中打开 "**complete_pdf_viewer**" demo。将示例代码加入到 "PDFReaderFragment.java" 文件中 (加在 "**mUiExtensionsManager = new UIExtensionsManager(getActivity().getApplicationContext(), pdfViewerCtrl, config);**" 之后)。

备注：手机和平板上面的内置 UI 有一些不同。下面列举的示例有的适用于手机和平板，也有一些示例仅仅适用于手机端。在本手册中，如果在手机和平板上的自定义结果类似，则只列出在手机上的效果图。

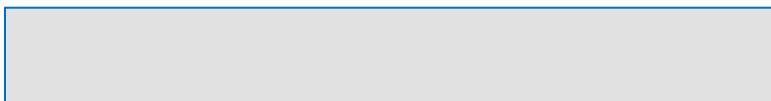
示例 1：隐藏整个 top toolbar. (适用于手机和平板)

```
mUiExtensionsManager.enableTopToolbar(false);
```

修改前：



修改后：



示例 2：隐藏整个 bottom toolbar. (仅适用于手机)

```
mUiExtensionsManager.enableBottomToolbar(false);
```

修改前：



修改后：



示例 3：在 top toolbar 左侧的第二个位置加入一个菜单项。 (适用于手机和平板)

```
BaseItemImpl mTopItem1 = new BaseItemImpl(getContext(), R.drawable.rd_delete_menu);
mTopItem1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        UI.Toast.getInstance(getActivity()).show("Add an item in the left top toolbar at the second position.");
    }
});
mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_LT, mTopItem1, 1);
```

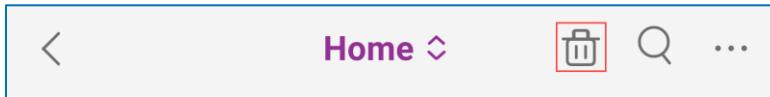
重新运行 demo 后：



示例 4：在 top toolbar 右侧的第一个位置加入一个菜单项。 (适用于手机和平板)

```
BaseItemImpl mTopItem2 = new BaseItemImpl(getContext(), R.drawable.rd_delete_menu);
mTopItem2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        UI.Toast.getInstance(getActivity()).show("Add an item in the right top toolbar at the first position.");
    }
});
mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_RB, mTopItem2, 0);
```

重新运行 demo 后：



示例 5：在 bottom toolbar 的最左边加入一个菜单项。(仅适用于手机)

```
mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.BOTTOM_BAR,
BaseBar.TB_Position.Position_CENTER,
"", R.drawable.ic_bar_item_more, 0, new IBarsHandler.IItemClickListener() {
    @Override
    public void onClick(View v) {
        UIToast.getInstance(getActivity()).show("Add an item to the bottom toolbar at the first position.");
    }
});
```

重新运行 demo 后：



示例 6：在 bottom toolbar 的第二个位置加入一个自定义样式的菜单项。(仅适用于手机)

```
BaselItemImpl mSettingBtn = new BaselItemImpl(this.getContext(), R.drawable.rubber_duck_create_ok_selector);
mSettingBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        UIToast.getInstance(getActivity()).show("Add an item with custom style to the bottom toolbar at the second position.");
    }
});
mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.BOTTOM_BAR,
BaseBar.TB_Position.Position_CENTER, mSettingBtn, 1);
```

重新运行 demo 后：



示例 7：通过索引值移除一个菜单项(移除 bottom toolbar 上的第一个菜单项)。(仅适用于手机)

```
mUiExtensionsManager.getBarManager().removeItem(IBarsHandler.BarName.BOTTOM_BAR,
BaseBar.TB_Position.Position_CENTER,0);
```

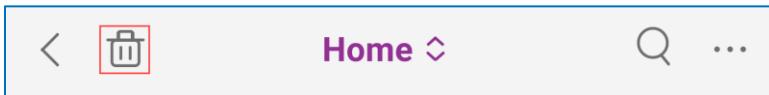
重新运行 demo 后：



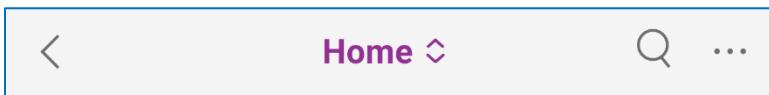
示例 8: 通过 **BaseItem** 对象移除一个菜单项 (从 **top toolbar** 上移除您刚添加的自定义菜单项)。
(适用于手机和平板)

```
mUiExtensionsManager.getBarManager().removeItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_LT, mTopItem1);
```

修改前: (见示例 3)



修改后:



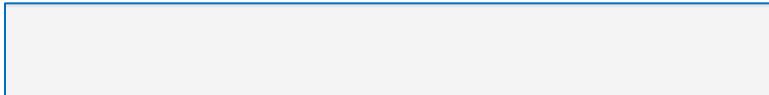
示例 9: 移除 **bottom toolbar** 上所有的菜单项。 (仅适用于手机)

```
mUiExtensionsManager.getBarManager().removeAllItems(IBarsHandler.BarName.BOTTOM_BAR);
```

修改前:



修改后:



示例 10: 在 **top toolbar** 的左侧添加 2 个菜单项去控制"more menu"菜单项的显示和隐藏。 (适用于手机和平板)

```
// Get and save the item that you want to show or hide.
BaseBarManager baseBarManager = (BaseBarManager) mUiExtensionsManager.getBarManager();
final BaseItemImpl moreItem = (BaseItemImpl) baseBarManager.getItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_RB, ToolbarItemConfig.ITEM_TOPBAR_MORE);

// Add a button in the left top toolbar to hide the "moreItem" item.
BaseItemImpl mTopItem = new BaseItemImpl(getContext(), R.drawable.rd_delete_menu);
mTopItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```

// Hide the "moreItem" item.
mUiExtensionsManager.getBarManager().removeItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_RB, moreItem);
}

});

mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_LT, mTopItem, 1);

// Add a button in the left top toolbar to show the "moreItem" item.
BaseItemImpl mTopItem2 = new BaseItemImpl(getContext(), R.drawable.common_add_icon);
mTopItem2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Show the "moreItem" item.
        if (AppDisplay.isPad())
            mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_RB, moreItem, 3);
        else
            mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_RB, moreItem, 1);
    }
});

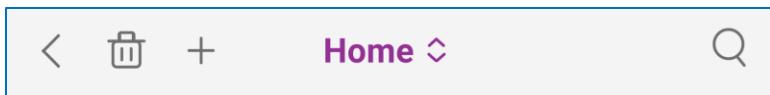
mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_LT, mTopItem2, 2);

```

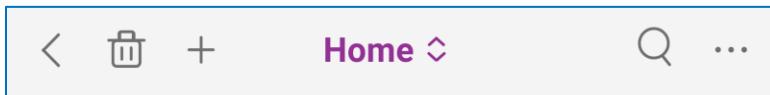
重新运行 demo 后，top toolbar 如下图所示：



点击 ，则 "more menu" 会被隐藏，如下图所示：



点击 ，则 "more menu" 会显示，如下图所示：



示例 11: 移除整个 bottom toolbar。(仅适用于手机)

```
mUiExtensionsManager.getBarManager().removeToolBar(IBarsHandler.BarName.BOTTOM_BAR);
```

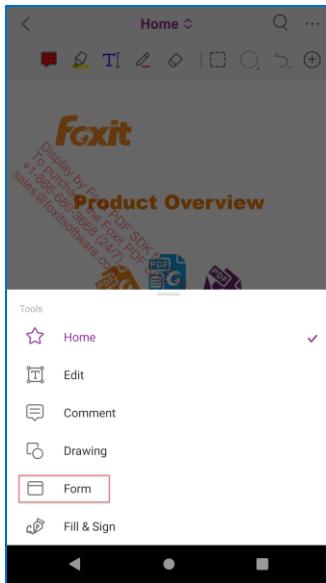
示例 12: 添加一个自定义 toolbar。(添加一个自定义布局文件 "test_top_layout") (适用于手机和平板)

```
View topView = View.inflate(getContext(), R.layout.test_top_layout, null);
mUiExtensionsManager.getBarManager().addCustomToolBar(IBarsHandler.BarName.TOP_BAR, topView);
```

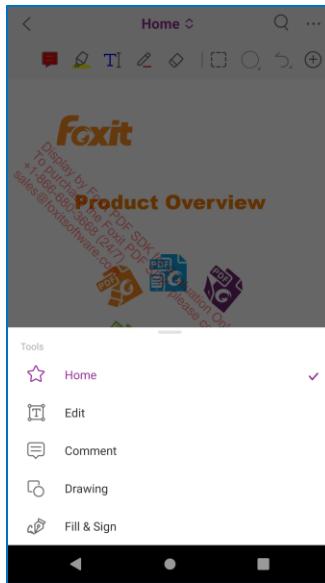
示例 13: 移除 top toolbar 中间位置列表中的 "From" tab。 (适用于手机和平板)

```
mUiExtensionsManager.getMainFrame().removeTab(ToolbarItemConfig.ITEM_FORM_TAB);
```

修改前:



修改后:



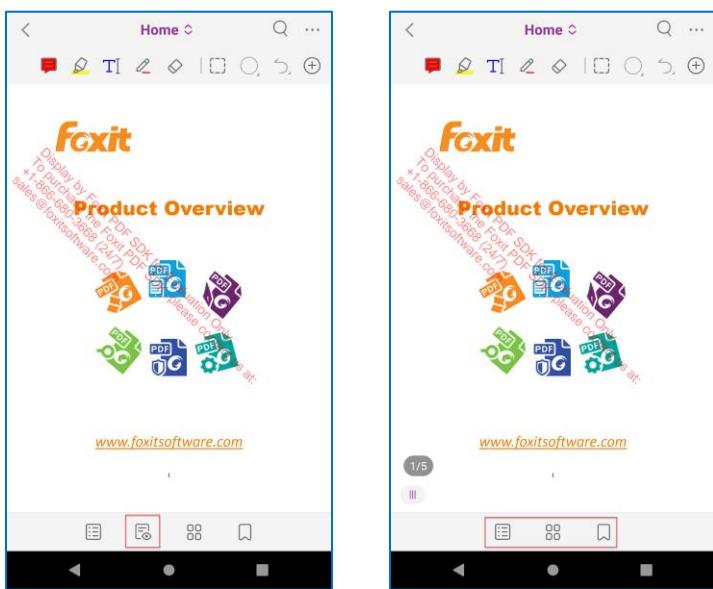
示例 14: 移除手机端在 bottom toolbar 上的 "view" 菜单，或者移除平板端在 top toolbar 中间位置列表中的 "view" tab。 (适用于手机和平板)

```
if (AppDisplay.isPad())
    mUiExtensionsManager.getMainFrame().removeTab(ToolbarItemConfig.ITEM_VIEW_TAB);
else
    mUiExtensionsManager.getBarManager().removeItem(IBarsHandler.BarName.BOTTOM_BAR,
BaseBar.TB_Position.Position_CENTER, 1);
```

对于手机设备:

修改前:

修改后:



对于平板设备：

修改前：



修改后：



4.2.2 自定义添加/移除一个特定的面板

通过 Table 4-2 中列出的 APIs，您可以添加自定义的面板或者移除一个特定的面板。Complete PDF Viewer demo 包含了 "Bookmarks"、"Outline"、"Annotations"、"Attachments" 和 "Digital Signatures" 面板，手机端点击底部工具栏上的  图标查看，平板端点击左上工具栏上的  图标来查看，如 Figure 4-3 所示。

Table 4-2

Public void addPanel (PanelSpec panelSpec)	添加一个子面板，如果使用 PanelSpec#getPanelType() 获取到的面板已经存在，则不会再被添加。
public void addPanel (int index, PanelSpec panelSpec)	在指定的位置添加一个子面板，如果使用 PanelSpec#getPanelType() 获取到的面板已经存在，则不会再被添加。
public void removePanel (int panelType)	移除指定位置的子面板。
public void removePanel (PanelSpec panelSpec)	移除子面板。

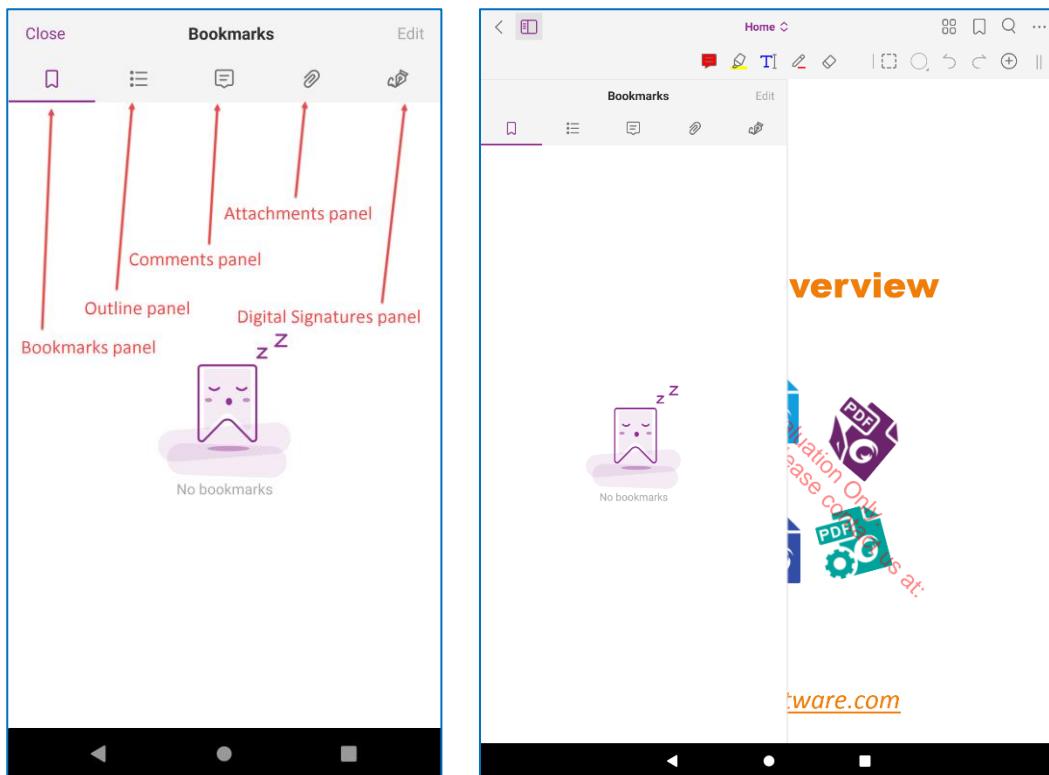


Figure 4-3

在本节中，我们提供 2 个示例：

- **Example1** 用来展示如何通过 APIs 来移除一个特定的面板。我们在 "samples" 文件夹下的 "**complete_pdf_viewer**" demo 中进行示例展示。以 "Outline" 面板为例，对于其他的面板，您只需要修改 **PanelType** 的值。Panels 和 **PanelType** 之间的对应关系如下表所示：

Panel	PanelSpec	integer
Bookmarks	PanelSpec.BOOKMARKS	0
Outline	PanelSpec.OUTLINE	1
Comments	PanelSpec.ANNOTATIONS	2
Attachments	PanelSpec.ATTACHMENTS	3
Digital Signatures	PanelSpec.SIGNATURES	4

- **Example2** 用来展示如何通过 APIs 来添加一个自定义的面板。首先需要创建一个 Java 类(比如，命名为 **CustomPanel**)，该类实现 "**PanelSpec.java**" 接口。当实现了相关方法和事件后，即可添加面板。

在 Android Studio 中打开 "**complete_pdf_viewer**" demo。将示例代码加入到 "PDFReaderFragment.java"文件中 (加在 "**mUiExtensionsManager = new UIExtensionsManager(getActivity().getApplicationContext(), pdfViewerCtrl, config);**"之后)。

示例 1：在 top toolbar 左侧的第二个位置添加一个菜单项用来移除 "Outline" 面板。(适用于手机和平板)

```
BaselItemImpl removePanel = new BaselItemImpl(getActivity(), R.drawable.rd_delete_menu);
removePanel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mUiExtensionsManager.getPanelManager().removePanel(PanelSpec.OUTLINE);
    }
});
mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_LT, removePanel, 1);
```

运行 demo 后，点击顶部工具栏上的 ，然后手机端点击底部工具栏上的 ，平板端点击左上工具栏上的 ，则您可以看到 "Outline" 面板被移除了(见 Figure 4-4)。

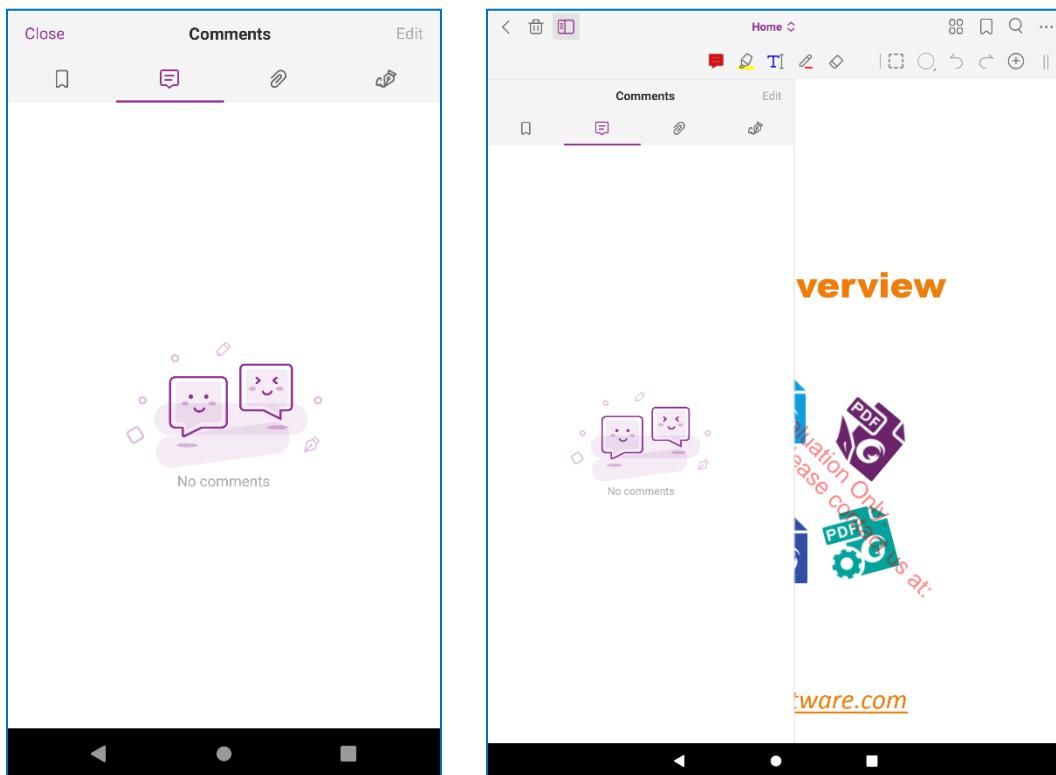


Figure 4-4

示例 2：在 top toolbar 左侧的第二个位置添加一个菜单项用来添加自定义面板。(适用于手机和平板)

```
BaseItemImpl customPanel = new BaseItemImpl(getActivity(), R.drawable.common_add_icon);
customPanel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mUiExtensionsManager.getPanelManager().addPanel(new CustomPanel(getApplicationContext()));
    }
});
mUiExtensionsManager.getBarManager().addItem(IBarsHandler.BarName.TOP_BAR,
BaseBar.TB_Position.Position_LT, customPanel, 1);
```

假设您已经创建了一个自定义的面板。创建一个名为 **CustomPanel** 的 Java 类，该类实现 "**PanelSpec.java**" 接口：

```
package com.foxit.home;

import android.content.Context;
import android.content.res.ColorStateList;
import android.view.View;
import android.widget.TextView;
```

```
import com.foxit.uiextensions.controls.panel.PanelSpec;
import com.foxit.uiextensions.controls.toolbar.BaseBar;
import com.foxit.uiextensions.controls.toolbar.IBaseItem;
import com.foxit.uiextensions.controls.toolbar.impl.BaseItemImpl;
import com.foxit.uiextensions.controls.toolbar.impl.TopBarImpl;

public class CustomPanel implements PanelSpec {

    private Context mContext;

    public CustomPanel(Context context){
        mContext= context;
    }

    @Override
    public int getIcon() {
        return R.drawable.toolbar_thumbnail_icon;
    }

    @Override
    public ColorStateList getIconTint() {
        return null;
    }

    @Override
    public int getPanelType() {
        return 100;
    }

    @Override
    public View getTopToolbar() {
        TopBarImpl topBar = new TopBarImpl(mContext);
        IBaseItem baselItem = new BaseItemImpl(mContext);
        baselItem.setText("Custom Panel");
        topBar.addView(baselItem, BaseBar.TB_Position.Position_CENTER);
        return topBar.getContentView();
    }

    @Override
    public View getContentView() {
        TextView textView = new TextView(mContext);
        textView.setText("This is empty");
        return textView;
    }

    @Override
    public void onActivated() {

    }

    @Override
```

```
public void onDeactivated() {  
}  
}
```

运行 demo 后，点击顶部工具栏上的 ，然后手机端点击底部工具栏上的 ，平板端点击左上工具栏上的 ，则您可以看到如 Figure 4-5 所示的自定义面板。

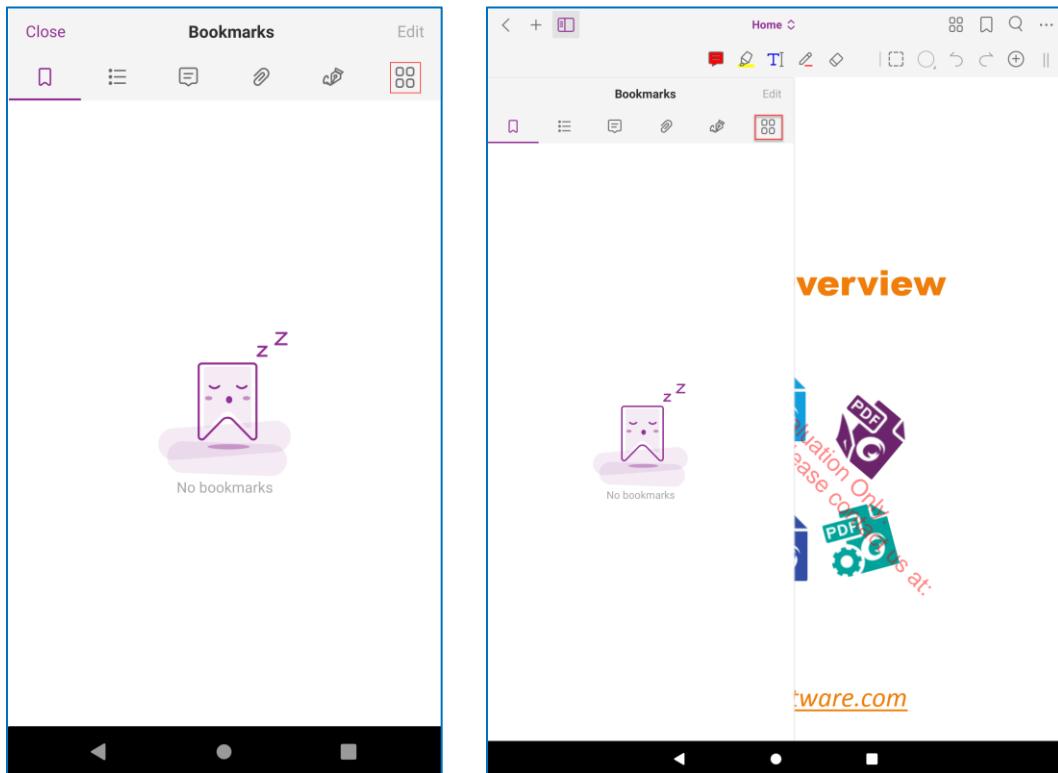


Figure 4-5

4.2.3 自定义隐藏 View setting bar 上的 UI 元素

隐藏 View setting bar 上的 UI 元素 (见 Figure 4-6，手机端点击底部工具栏上的 查看，平板端点击顶部工具栏中靠近 Home 的 图标找到 view)，您只需要使用下面的 API：

在 `IViewSettingsWindow` 类中，

```
public void setVisible (int type, boolean visible);
```

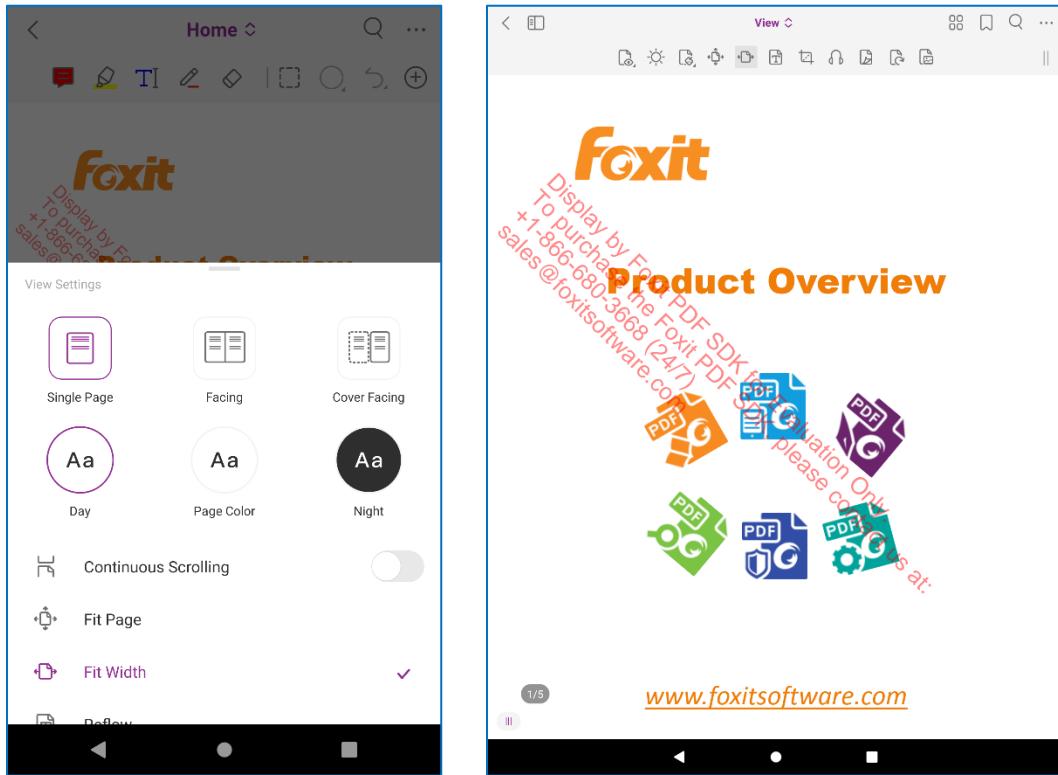


Figure 4-6

参数 "type" 的值可以参考如下的表格，其对应 View setting bar 上的菜单项。

type	integer
IViewSettingsWindow.TYPE_SINGLE_PAGE	1
IViewSettingsWindow.TYPE_FACING_PAGE	2
IViewSettingsWindow.TYPE_COVER_PAGE	4
IViewSettingsWindow.TYPE_DAY	8
IViewSettingsWindow.TYPE_PAGE_COLOR	16
IViewSettingsWindow.TYPE_NIGHT	32
IViewSettingsWindow.TYPE_CONTINUOUS_PAGE	64
IViewSettingsWindow.TYPE_FIT_PAGE	128
IViewSettingsWindow.TYPE_FIT_WIDTH	256
IViewSettingsWindow.TYPE_REFLOW	288
IViewSettingsWindow.TYPE_CROP	320
IViewSettingsWindow.TYPE_TTS	384
IViewSettingsWindow.TYPE_AUTO_FLIP	512
IViewSettingsWindow.TYPE_ROTATE_VIEW	544
IViewSettingsWindow.TYPE_PAN_ZOOM	576

在本节中，我们在 "samples" 文件夹下的 "complete_pdf_viewer" demo 中以 "Fit Width" 菜单为例展示如何通过 APIs 来隐藏 View setting bar 上的 UI 元素。对于其他的 UI 元素，您只需要修改参数 "type"。

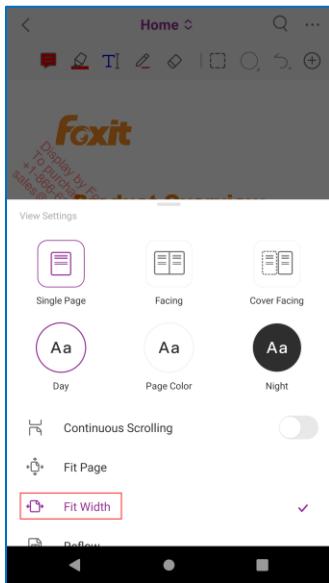
在 Android Studio 中打开 "complete_pdf_viewer" demo。将示例代码加入到 "PDFReaderFragment.java" 文件中 (加在 "mUiExtensionsManager.onCreate(getActivity(), pdfViewerCtrl, savedInstanceState);" 之后)。

示例 1：隐藏 View setting bar 上面的 Fit Width 菜单。(适用于手机和平板)

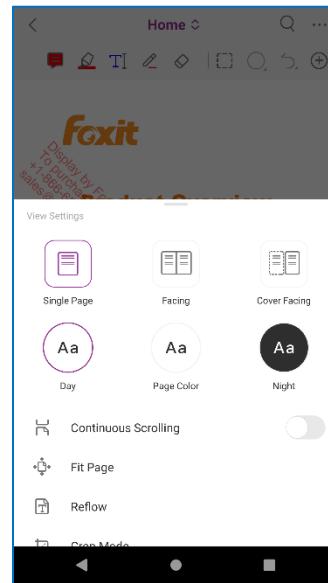
```
mUiExtensionsManager.getSettingBar().setVisibility(IMultiLineBar.TYPE_REFLOW, View.INVISIBLE);
```

手机端：

修改前：



修改后：



平板端：

修改前：



修改后：



4.2.4 自定义添加/隐藏 More Menu 菜单上的 UI 元素

显示或者隐藏 More Menu 菜单, 请参考 4.2.1 小节 "自定义 top/bottom toolbar" (见示例 10)

您可以使用 Table 4-3 中列出的 APIs 来添加一个自定义的 group 或者子菜单项, 以及隐藏一个特定的 group 或者子菜单项。Table 4-3 只列出了本节示例所使用的 APIs, 有关其他的 APIs, 请参阅 API Reference。

点击右上角工具栏上的 ···, 您可以看到 More Menu view 上的 UI 元素, 如 Figure 4-7 所示。

Table 4-3

IMenuGroup.addItem(IMenuItem item)	添加 IMenuItem 类型的菜单项。
IMenuGroup.addItem(CharSequence title)	添加带有指定标题的菜单项。
IMenuGroup.addItem(Drawable icon, CharSequence title)	添加带有指定图标和标题的菜单项。
IMenuGroup.setVisible(boolean visible);	设置 menu group 的可见状态。
IMenuItem.setVisible(boolean visible)	设置菜单项的可见状态。

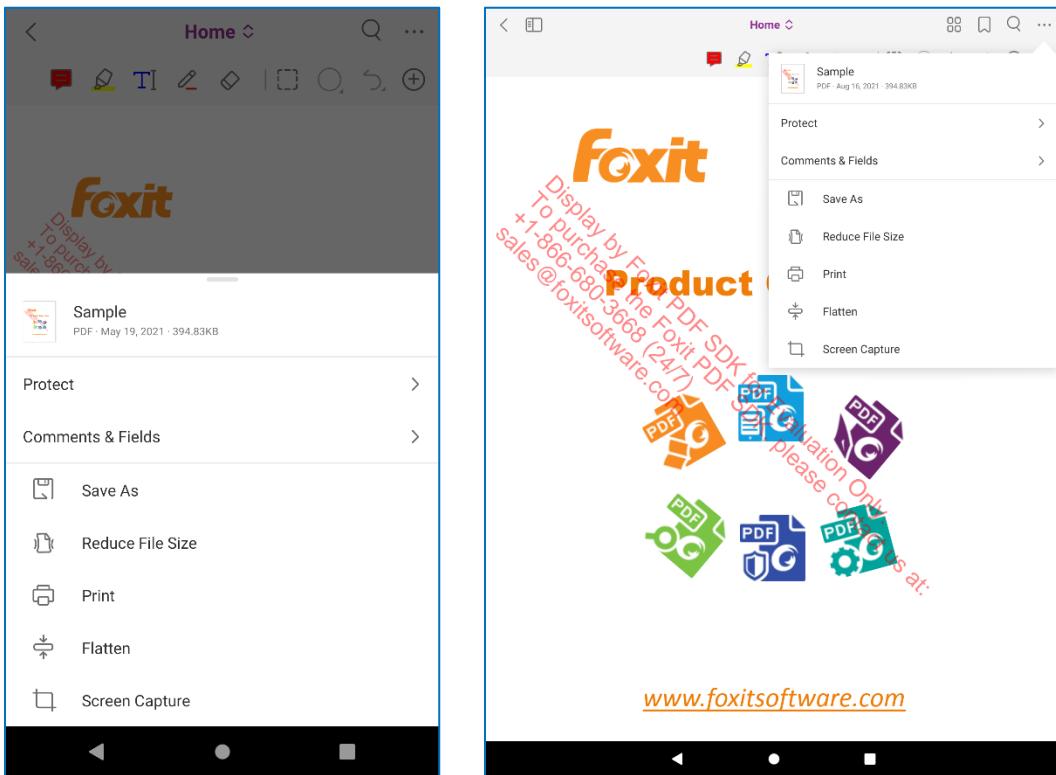


Figure 4-7

More Menu view 被划分为如下的 2 个组:

Group	integer
GROUP_ACTION_MENU_PRIMARY	1000
GROUP_ACTION_MENU_SECONDARY	1001

每个组有如下的菜单项:

Group	Item	integer
GROUP_ACTION_MENU_PRIMARY	ITEM_PRIMARY_PROTECT	1
	ITEM_PRIMARY_COMMENT_FIELDS	2
GROUP_ACTION_MENU_SECONDARY	ITEM_SECONDARY_SAVE_AS	1
	ITEM_SECONDARY_REDUCE_FILE_SIZE	2
	ITEM_SECONDARY_PRINT	3
	ITEM_SECONDARY_FLATTEN	4
	ITEM_SECONDARY_SCREEN	5

其中, ITEM_PRIMARY_PROTECT 和 ITEM_PRIMARY_COMMENT_FIELDS 菜单有如下的子菜单项:

Item	sub-Item	integer
ITEM_PRIMARY_PROTECT	ITEM_PROTECT_REDACTION	1
	ITEM_PROTECT_REMOVE_PASSWORD	2

	ITEM_PROTECT_FILE_ENCRYPTION	3
	ITEM_PROTECT_TRUSTED_CERTIFICATES	4
ITEM_PRIMARY_COMMENT_FIELDS	ITEM_COMMENTS_FIELDS_IMPORT_COMMENTS	1
	ITEM_COMMENTS_FIELDS_EXPORT_COMMENTS	2
	ITEM_COMMENTS_FIELDS_SUMMARIZE_COMMENTS	3
	ITEM_COMMENTS_FIELDS_RESET_FORM_FIELDS	4
	ITEM_COMMENTS_FIELDS_IMPORT_FORM_DATA	5
	ITEM_COMMENTS_FIELDS_EXPORT_FORM_DATA	6

在本节中，我们提供 3 个示例：

- **Example1** 和 **Example2** 用来展示如何通过 APIs 来隐藏 More Menu view 中特定的组或者菜单。我们在 "samples" 文件夹下的 "**complete_pdf_viewer**" demo 中进行示例展示。以 "**GROUP_ACTION_MENU_PRIMARY**" 组和 "**ITEM_SECONDARY_PRINT**" 菜单为例，对于其他的组和菜单，请参考该示例。
- **Example3** 用来展示如何通过 APIs 来添加带有菜单项的自定义组。

在 Android Studio 中打开 "**complete_pdf_viewer**" demo。将示例代码加入到 "PDFReaderFragment.java" 文件中 (加在 "**mUiExtensionsManager = new UIExtensionsManager(getActivity().getApplicationContext(), pdfViewerCtrl, config);**" 之后)。

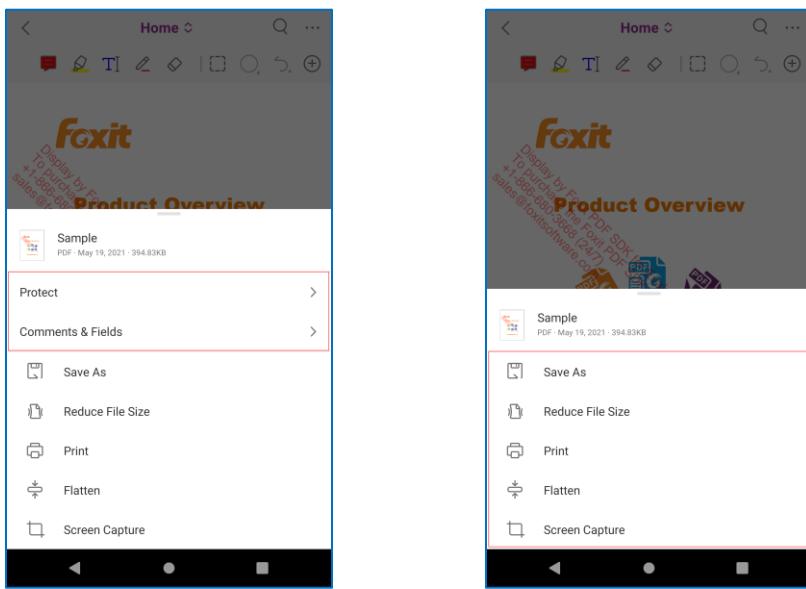
备注：手机和平板上面的内置 UI 有一些不同。下面列举的示例适用于手机和平板，在本手册中，如果在手机和平板上的自定义结果类似，则只列出在手机上的效果图。

示例 1：隐藏 More Menu view 上面的 "GROUP_ACTION_MENU_PRIMARY**" 菜单组。(适用于手机和平板)**

```
IMenuView menuView = mUiExtensionsManager.getMenuView();
IMenuGroup menuGroup = menuView.getGroup(MoreMenuConstants.GROUP_ACTION_MENU_PRIMARY);
menuGroup.setVisible(false);
```

修改前:

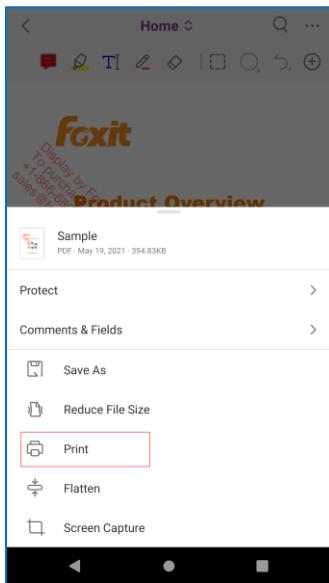
修改后:



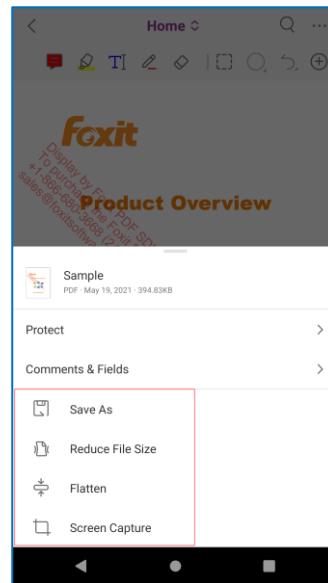
示例 2：隐藏 More Menu view 上面的 "ITEM_SECONDARY_PRINT" 菜单项。(适用于手机和平板)

```
IMenuView menuView = mUiExtensionsManager.getMenuView();
IMenuGroup menuGroup = menuView.getGroup(MoreMenuConstants.GROUP_ACTION_MENU_SECONDARY);
IMenuItem menuItem = menuGroup.getItem(MoreMenuConstants.ITEM_SECONDARY_PRINT);
menuItem.setVisible(false);
```

修改前:



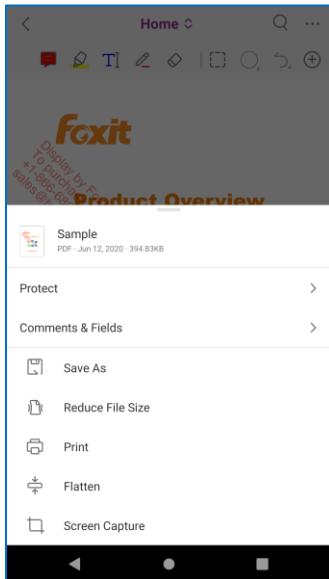
修改后:



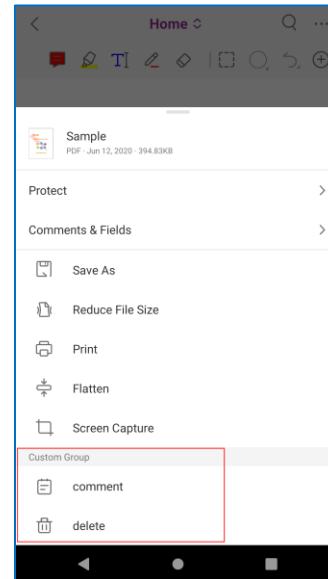
示例 3：在 More Menu view 上添加一个带有 2 个菜单项的菜单组。(适用于手机和平板)

```
IMenuView menuView = mUiExtensionsManager.getMenuView();
IMenuGroup customGroup = menuView.addGroup("Custom Group");
IMenuItem item1 =
customGroup.addItem(AppResource.getDrawable(getActivity(),R.drawable.rd_comment_menu), "comment");
item1.setOnMenuItemClickListener(new IMenuItem.OnMenuItemClickListener() {
    @Override
    public void onClick(IMenuItem item) {
        UIToast.getInstance(getActivity()).show("I am item1");
    }
});
IMenuItem item2 = customGroup.addItem(AppResource.getDrawable(getActivity(),R.drawable.rd_delete_menu),
"delete");
item2.setOnMenuItemClickListener(new IMenuItem.OnMenuItemClickListener() {
    @Override
    public void onClick(IMenuItem item) {
        UIToast.getInstance(getActivity()).show("I am item2");
    }
});
```

修改前：



修改后：



4.3 通过源代码自定义 UI 实现

在前面的章节中，我们详细地介绍了如何通过配置文件或者 APIs 对 UI 进行自定义。这些修改是基于 Foxit PDF SDK for Android 的内置 UI 框架的。如果您不想使用当前现成的 UI 框架，您可以通过修改 UI Extensions 组件中的源代码来重新设计 UI。

为了自定义 UI 实现，您可以按照下面的步骤：

备注：在本节中，我们只介绍如何自定义 UI Extensions 组件中的 UI，有关自定义扫描功能的 UI，您可以参考本节的教程。

首先，在您的工程中添加如下的文件，这些文件都在包中的"libs"文件夹下。

- **uiextensions_src** 工程 - 一个开源库，包含了一些即用型的 UI 模块实现，可以帮助开发人员快速将功能齐全的 PDF 阅读器嵌入到他们的 Android 应用中。当然，开发人员也不是必须要使用默认的 UI，可以通过"uiextensions_src"工程为特定的应用灵活自定义和设计 UI。
- **FoxitRDK.aar** - 包含 JAR 包，其中包括 Foxit PDF SDK for Android 的所有 Java APIs，以及 ".so"库。".so"库是 SDK 的核心包含了 Foxit PDF SDK for Android 的核心函数。它针对每种架构单独编译，当期支持 armeabi-v7a, arm64-v8a, x86, 和 x86_64 架构。

备注：**uiextensions_src** 工程依赖于 **FoxitRDK.aar**，因此最好将它们放在一个目录下。如果不是，您需要在 **uiextensions_src** 工程的 "**build.gradle**" 文件中手动修改 **FoxitRDK.aar** 的引用路径。

其次，在 **uiextensions_src** 工程中定位到您需要自定义的 XML 布局文件，然后根据您的需求进行修改。

为方便起见，我们将在"sample"文件夹下的"**viewer_ctrl_demo**" 中向您展示如何自定义 UI 实现。

UI Customization Example UI 自定义示例

步骤 1：向 "**viewer_ctrl_demo**" 中添加 **uiextensions_src** 工程，请确保其与 **FoxitRDK.aar** 文件在一个目录下。如果您没有修改默认文件夹的层结构，那么它们就是在正确的位置。

备注："**viewer_ctrl_demo**"已经添加了 **FoxitRDK.aar** 的引用，因此我们只需要通过配置 "**settings.gradle**" 文件来添加 **uiextensions_src** 工程。当添加 **uiextensions_src** 作为依赖后，需要将 **FoxitRDKUIExtensions.aar** 的引用移除。

在 Android Studio 中加载 "**viewer_ctrl_demo**"。然后按照下面的步骤：

- a) 在 "settings.gradle" 文件中，添加 **uiextensions_src** 工程，代码如下。

settings.gradle:

```
include ':app'
include ':uiextensions_src'
project(':uiextensions_src').projectDir = new File('../.. libs/uiextensions_src/')
```

重新编译 gradle，则 **uiextensions_src** 工程会被添加到 demo 中，如 Figure 4-8 所示。

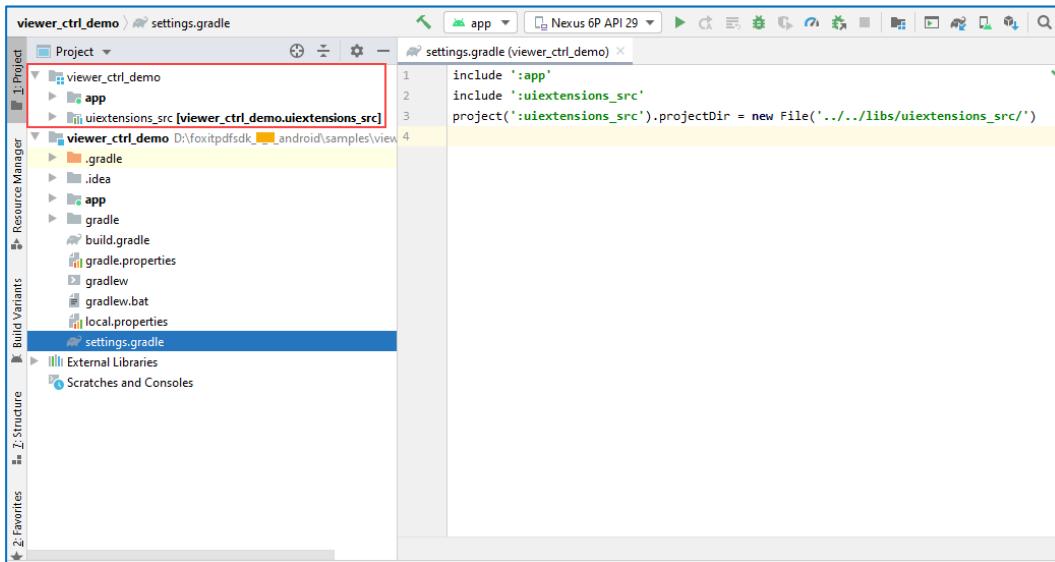


Figure 4-8

- b) 在 demo 中将 **uiextensions_src** 工程作为 demo 的依赖项。在 app 的 "build.gradle" 文件中，添加 `implementation project(":uiextensions_src")`，然后注释掉 `implementation(name:'FoxitRDKUIExtensions', ext:'aar')`，如下所示：

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.0'
    implementation 'androidx.multidex:multidex:2.0.1'
    implementation (name: 'FoxitRDK', ext: 'aar')
    // implementation(name:'FoxitRDKUIExtensions', ext:'aar')
    implementation project(":uiextensions_src")
    implementation 'com.edmodo:cropper:1.0.1'
    implementation 'com.microsoft.identity.client:msal:2.+'
    implementation(name: 'RMSSDK-4.2-release', ext: 'aar')
    implementation(name: 'rms-sdk-ui', ext: 'aar')
    implementation 'org.bouncycastle:bcpkix-jdk15on:1.60'
    implementation 'org.bouncycastle:bcprov-jdk15on:1.60'
    // RxJava
    implementation "io.reactivex.rxjava2:rxjava:2.2.16"
    implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'
}
```

当进行更改后，重新编译 gradle。然后，选择 "File -> Project Structure..." 打开 **Project Structure** 对话框。在对话框中点击 "**Dependencies -> app**"，您可以看到该 demo 依赖于 **uiextensions_src** 工程，如 Figure 4-9。

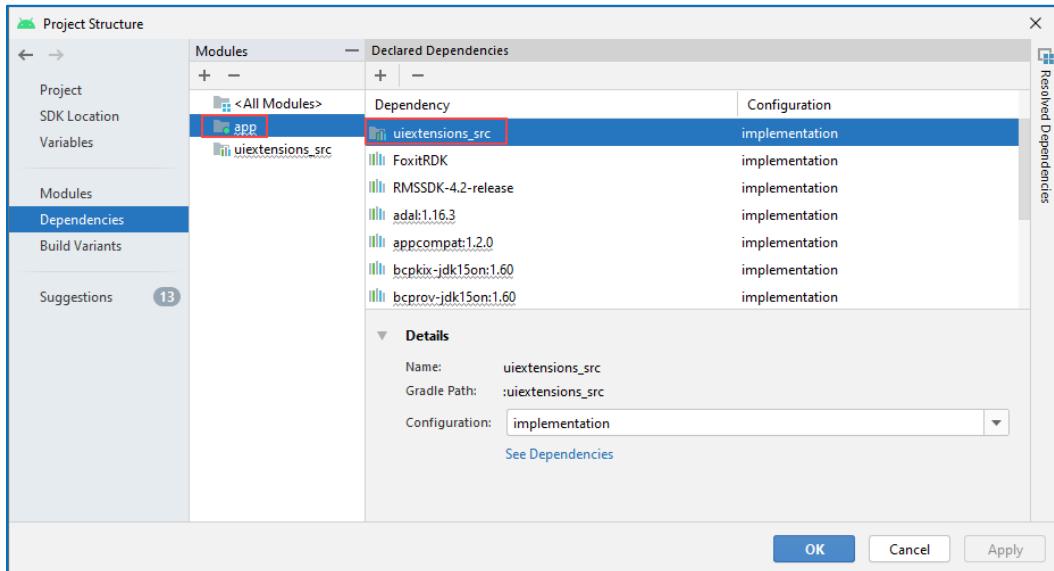


Figure 4-9

恭喜您！您已经完成了第一步。

步骤 2：查找和修改与您需要自定义的 UI 相关的布局文件。

现在，我们将向您展示一个简单的示例，在搜索面板中修改 button 的图标，如 Figure 4-10 所示。

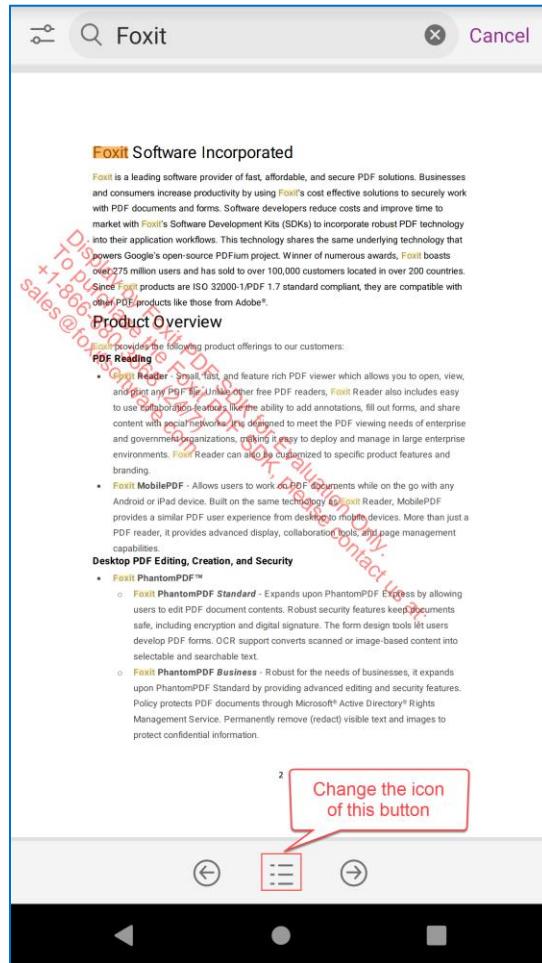


Figure 4-10

要替换图标，我们只需要找到存储该 button 图标的位罝，然后使用另一个具有相同名称的图标来替换它。

在工程中，点击 "**uiextensions_src -> src -> main -> res -> layout**"，如 Figure 4-11 所示。

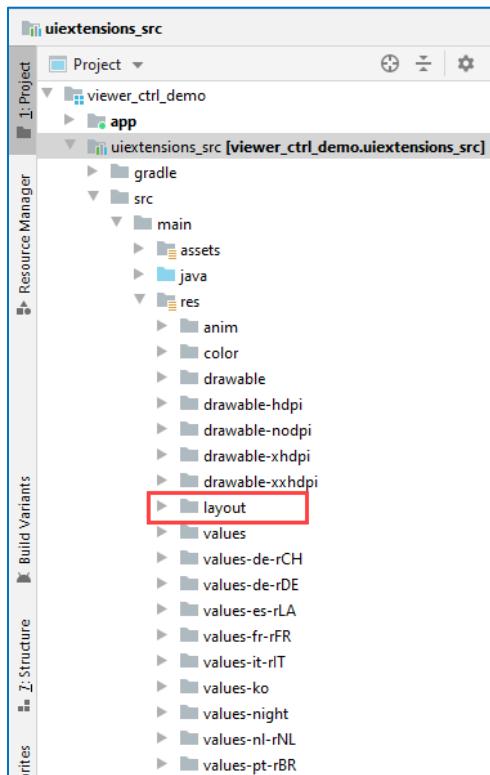


Figure 4-11

在布局文件列表中，找到 "**search_layout.xml**" 文件，然后双击它。在 Preview 窗口中找到该按钮，然后通过单击它来定位到其相关代码，如 Figure 4-12 所示。

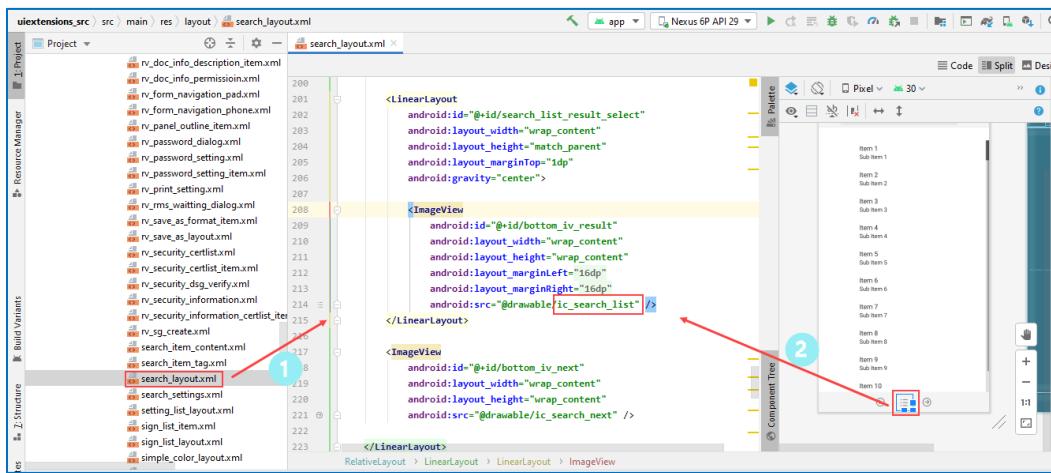


Figure 4-12

此时，可以看到该 button 的图标存储在 "drawable" 文件夹中，图标名称为 "**ic_search_list**"。只需将其替换为您自己的图标即可。

例如，我们使用 search next 的图标 ("ic_search_next.xml" 与 "ic_search_list.xml" 在相同的文件夹下) 来替换它。然后重新运行 demo，使用搜索功能，可以看到底部搜索 button 的图标已更改，如 Figure 4-13 所示。

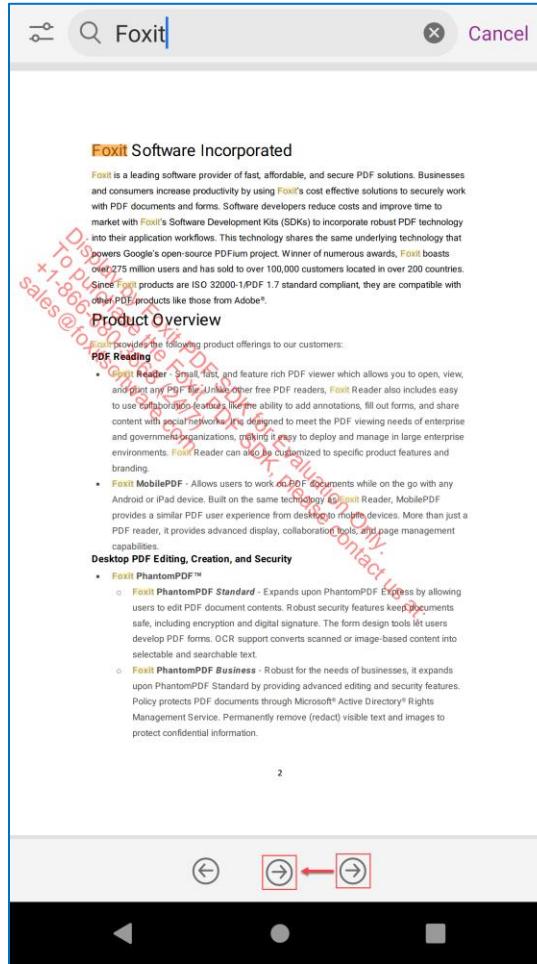


Figure 4-13

这只是一个简单的示例用来展示如何自定义 UI 实现。您可以作为参考，通过 **uiextensions_src** 工程您可以自由的对特定的应用程序进行 UI 自定义和设计。

5 使用 SDK API

Foxit PDF SDK for Android 将所有功能实现封装在 UI Extensions 组件中。如果您对功能实现的详细过程感兴趣，请参考本节内容。

在本节中，我们将介绍 Foxit PDF SDK for Android 的主要功能，并列举相关示例来展示如何使用 Foxit PDF SDK Core API 实现这些功能。

5.1 Render

PDF 渲染是通过 Foxit 渲染引擎实现的，Foxit 渲染引擎是一个图形引擎，用于将页面渲染到位图或平台设备上下文。Foxit PDF SDK 提供了 APIs 用来设置渲染选项 flags，例如设置 flag 来决定是否渲染表单域和签名，是否绘制图像反锯齿 (anti-aliasing) 和路径反锯齿。可以使用以下 APIs 进行渲染：

- 渲染页面和注释时，首先使用 `Renderer.setRenderContentFlags` 接口来决定是否同时渲染页面和注释，然后使用 `Renderer.startRender` 接口进行渲染。`Renderer.startQuickRender` 接口也可以用来渲染页面，但仅用于缩略图。
- 渲染单个 annotation 注释，使用 `Renderer.renderAnnot` 接口。
- 在位图上渲染，使用 `Renderer.startRenderBitmap` 接口。
- 渲染一个重排的页面，使用 `Renderer.startRenderReflowPage` 接口。

在 Foxit PDF SDK 中，Widget 注释常与表单域和表单控件相关联。渲染 widget 注释，推荐使用如下的流程：

- 加载 PDF 页面后，首先渲染页面以及该页面上所有的注释 (包括 widget 注释)。
- 然后，如果使用 `pdf.interform.Filler` 对象来填表，则应使用 `pdf.interform.Filler.render` 接口来渲染当前获取到焦点的表单控件，而不是使用 `Renderer.renderAnnot` 接口。

Example:

5.1.1 如何将指定的 PDF 页面渲染到 bitmap

```
import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.Progressive;
import com.foxit.sdk.common.Renderer;
import com.foxit.sdk.common.fxcrt.Matrix2D;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.PDFPage;
```

```
...  
public Bitmap renderPageToBitmap(PDFPage pdfPage, int drawPageWidth, int drawPageHeight) {  
    try {  
        // If the page hasn't been parsed yet, throw an exception.  
        if (pdfPage.isParsed()) {  
            throw new PDFException(Constants.e_ErrNotParsed, "PDF Page should be parsed first");  
        }  
  
        // Peparate matrix to render on the bitmap.  
        Matrix2D matrix2D = pdfPage.getDisplayMatrix(0, 0, drawPageWidth, drawPageHeight,  
Constants.e_Rotation0);  
        // Create a bitmap according to the required drawPageWidth and drawPageHeight.  
        Bitmap bitmap = Bitmap.createBitmap(drawPageWidth, drawPageHeight, Bitmap.Config.RGB_565);  
        // Fill the bitmap with white color.  
        bitmap.eraseColor(Color.WHITE);  
        Renderer renderer = new Renderer(bitmap, true);  
        // Set the render flag, both page content and annotation will be rendered.  
        renderer.setRenderContentFlags(Renderer.e_RenderPage | Renderer.e_RenderAnnot);  
        // Start to render the page progressively.  
        Progressive progressive = renderer.startRender(pdfPage, matrix2D, null);  
        int state = Progressive.e_ToBeContinued;  
        while (state == Progressive.e_ToBeContinued) {  
            state = progressive.resume();  
        }  
  
        if (state != Progressive.e_Finished) return null;  
        return bitmap;  
    } catch (PDFException e) {  
        e.printStackTrace();  
    }  
    return null;  
}
```

5.2 Text Page

Foxit PDF SDK 提供 APIs 来提取，选择，搜索和检索 PDF 文档中的文本。PDF 文本内容存储在与特定页面相关的 [TextPage](#) 对象中。[TextPage](#) 类可用于检索 PDF 页面中文本的信息，例如单个字符，单个单词，指定字符范围或矩形内的文本内容等。它还可用于构造其他文本相关类的对象，用来对文本内容执行更多操作或从文本内容访问指定信息：

- 在 PDF 页面的文本内容中搜索文本，使用 `TextPage` 对象来构建 `TextSearch` 对象。
- 访问类似超文本链接的文本，使用 `TextPage` 对象来构建 `PageTextLinks` 对象。
- 高亮 PDF 页面上的选中文本，构建一个 `TextPage` 对象来计算选中文本区域。

Example:

5.2.1 如何通过选择获取页面上的文本区域

```
import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.fxcrt.RectF;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.PDFPage;
import com.foxit.sdk.pdf.TextPage;
import com.foxit.sdk.common.fxcrt.PointF;
...
// Get the text area on page by selection. The starting selection position and ending selection position are specified by startPos and endPos.
public ArrayList<RectF> getTextRectsBySelection(PDFPage page, PointF startPos, PointF endPos) {
    try {
        // If the page hasn't been parsed yet, throw an exception.
        if (page.isParsed()) {
            throw new PDFException(Constants.e_ErrNotParsed, "PDF Page should be parsed first");
        }

        // Create a text page from the parsed PDF page.
        TextPage textPage = new TextPage(page, TextPage.e_ParseTextNormal);
        if (textPage == null || textPage.isEmpty()) return null;
        int startCharIndex = textPage.getIndexAtPos(startPos.getX(), startPos.getY(), 5);
        int endCharIndex = textPage.getIndexAtPos(endPos.getX(), endPos.getY(), 5);
        // API getTextRectCount requires that start character index must be lower than or equal to end character index.
        startCharIndex = startCharIndex < endCharIndex ? startCharIndex : endCharIndex;
        endCharIndex = endCharIndex > startCharIndex ? endCharIndex : startCharIndex;
        int count = textPage.getTextRectCount(startCharIndex, endCharIndex - startCharIndex);

        if (count > 0) {
            ArrayList<RectF> array = new ArrayList<RectF>();
            for (int i = 0; i < count; i++) {
                RectF rectF = textPage.getTextRect(i);
                if (rectF == null || rectF.isEmpty()) continue;
                array.add(rectF);
            }
            // The return rects are in PDF unit, if caller need to highlight the text rects on the screen, then these rects should be converted in device unit first.
            return array;
        }
    }
}
```

```
    }
} catch (PDFException e) {
    e.printStackTrace();
}
return null;
}
...
```

5.3 Text Search

Foxit PDF SDK 提供 APIs 来搜索 PDF 文档、XFA 文档、文本页面或者 PDF 注释中的文本。它提供了文本搜索和获取搜索结果的函数：

- 指定搜索模式和选项，使用 `TextSearch.setPattern`、`TextSearch.setStartPage` (仅对 PDF 文档中的文本搜索有用)、`TextSearch.setEndPage` (仅对 PDF 文档中的文本搜索有用)、和 `TextSearch.setSearchFlags` 接口。
- 进行搜索，使用 `TextSearch.findNext` 和 `TextSearch.findPrev` 接口。
- 获取搜索结果，使用 `TextSearch.getMatchXXX()` 接口。

Example:

5.3.1 如何在 PDF 文档中搜索指定的文本模型

```
import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.fxcrt.RectF;
import com.foxit.sdk.common.fxcrt.RectFArray;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.PDFPage;
import com.foxit.sdk.pdf.TextPage;
import com.foxit.sdk.common.fxcrt.PointF;
import com.foxit.sdk.pdf.TextSearch;
...
try {
    String pdfpath = "XXX/Sample.pdf";
    PDFDoc doc = new PDFDoc(pdfpath);
    doc.load(null);

    // Create a text search handler for searching in PDF document.
    TextSearch textSearch = new TextSearch(doc, null, TextPage.e_ParseTextNormal);
    // Set the start page index which searching will begin. By default, end page will be the last page.
    textSearch.setStartPage(0);
    // Set the text to be searched.
    textSearch.setPattern("foxit");
    // Set the search flags to be matching case and matching whole word.
```

```

textSearch.setSearchFlags(TextSearch.e_SearchMatchCase | TextSearch.e_SearchMatchWholeWord);
while (textSearch.findNext()) {
    // If true, then we found a matched result.
    // Get the found page index.
    int pageIdx = textSearch.getMatchPageIndex();
    // Get the start character index of the matched text on the found page.
    int startCharIndex = textSearch.getMatchStartCharIndex();
    // Get the end character index of the matched text on the found page.
    int endCharIndex = textSearch.getMatchEndCharIndex();
    // Get the rectangular region of the matched text on the found page.
    RectFArray matchRects = textSearch.getMatchRects();
}
catch (Exception e) {}
...

```

5.4 Bookmark (Outline)

Foxit PDF SDK 提供了名为 Bookmarks 的导航工具，允许用户在 PDF 文档中快速定位和链接他们感兴趣的部分。PDF 书签也称为 outline，每个书签包含一个目标位置或动作来描述它链接到的位置。它是一个树形的层次结构，因此在访问 bookmark 树之前，必须首先调用接口 [PDFDoc.getRootBookmark](#) 以获取整个 bookmark 树的 root。这里，“root bookmark”是一个抽象对象，它只有一些 child bookmarks，没有 next sibling bookmarks，也没有任何数据（包括 bookmark 数据，目标位置数据和动作数据）。因为它没有任何数据，因此无法在应用程序界面上显示，能够调用的接口只有 [Bookmark.getFirstChild](#)。

在检索 root bookmark 后，就可以调用以下的接口去访问其他的 bookmarks：

- 访问 parent bookmark，使用 [Bookmark.getParent](#) 接口。
- 访问第一个 child bookmark，使用 [Bookmark.getFirstChild](#) 接口。
- 访问 next sibling bookmark，使用 [Bookmark.getNextSibling](#) 接口。
- 插入一个新的 bookmark，使用 [Bookmark.insert](#) 接口。
- 移动一个 bookmark，使用 [Bookmark.moveTo](#) 接口。

Example:

5.4.1 如何使用深度优先顺序遍历 PDF 文档的 bookmarks

```

import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.pdf.Bookmark;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.actions.Destination;
...

```

```
private void DepthFistTravelBookmarkTree(Bookmark bookmark, PDFDoc doc) {
    if(bookmark == null || bookmark.isEmpty())
        return;
    try {
        DepthFistTravelBookmarkTree(bookmark.getFirstChild(), doc);
        while (true) {
            // Get bookmark title.
            String title = bookmark.getTitle();
            Destination dest = bookmark.getDestination();
            if (dest != null && !dest.isEmpty())
            {
                float left, right, top, bottom;
                float zoom;
                int pageIndex = dest.getPageIndex(doc);
                // left,right,top,bottom,zoom are only meaningful with some special zoom modes.
                int mode = dest.getZoomMode();
                switch (mode) {
                    case Destination.e_ZoomXYZ:
                        left = dest.getLeft();
                        top = dest.getTop();
                        zoom = dest.getZoomFactor();
                        break;
                    case Destination.e_ZoomFitPage:
                        break;
                    case Destination.e_ZoomFitHorz:
                        top = dest.getTop();
                        break;
                    case Destination.e_ZoomFitVert:
                        left = dest.getLeft();
                        break;
                    case Destination.e_ZoomFitRect:
                        left = dest.getLeft();
                        bottom = dest.getBottom();
                        right = dest.getRight();
                        top = dest.getTop();
                        break;
                    case Destination.e_ZoomFitBBox:
                        break;
                    case Destination.e_ZoomFitBHorz:
                        top = dest.getTop();
                        break;
                    case Destination.e_ZoomFitBVert:
                        left = dest.getLeft();
                        break;
                    default:
                        break;
                }
            }
        }
    }
}
```

```
        }
        bookmark = bookmark.getNextSibling();
        if(bookmark == null || bookmark.isEmpty())
            break;
        DepthFirstTravelBookmarkTree(bookmark.getFirstChild(), doc);
    }
}
catch (Exception e) {
}
}
```

5.5 Reading Bookmark

Reading bookmark 不是 PDF bookmark，换句话说，它不是 PDF outlines。Reading bookmark 是应用层的书签。它存储在目录的元数据（XML 格式）中，允许用户根据他们的阅读偏好添加或删除 reading bookmark，并通过选择 reading bookmark 可以轻松导航到一个 PDF 页面。

为了检索 reading bookmark，可以调用 [PDFDoc.getReadingBookmarkCount](#) 接口来计算其个数，并且可以调用 [PDFDoc.getReadingBookmark](#) 接口以索引方式获取相应的 reading bookmark。

此类提供了接口用来获取/设置 reading bookmarks 属性，比如标题，目标页面索引，以及创建/修改日期时间。

5.5.1 如何添加自定义 reading bookmark 并枚举所有的 reading bookmarks

```
import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.DateTime;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.ReadingBookmark;
...
// Add a new reading bookmark to pdf document, the returned bookmark stores the title and the page index.
ReadingBookmark addReadingBookmark(PDFDoc pdfDoc, String title, int pageIndex) {
    int count = pdfDoc.getReadingBookmarkCount();
    return pdfDoc.insertReadingBookmark(count,title,pageIndex);
}

// Enumerate all the reading bookmarks from the pdf document.
void getReadingBookmark(PDFDoc pdfDoc) {
    try {
        int count = pdfDoc.getReadingBookmarkCount();
        for (int i = 0; i < count; i++) {
            ReadingBookmark readingBookmark = pdfDoc.getReadingBookmark(i);
            if(readingBookmark.isEmpty()) continue;
            // Get bookmark title.
    }
}
```

```
String title = readingBookmark.getTitle();
// Get the page index which associated with the bookmark.
int pageIndex = readingBookmark.getPageIndex();
// Get the creation date of the bookmark.
DateTime creationTime = readingBookmark.getDateTime(true);
// Get the modification date of the bookmark.
DateTime modificationTime = readingBookmark.getDateTime(false);
}
} catch (PDFException e) {
    e.printStackTrace();
}
}
```

5.6 Attachment

在 Foxit PDF SDK 中，attachments 指的是文档附件而不是文件附件注释。它允许将整个文件封装在文档中，就像电子邮件附件一样。Foxit PDF SDK 提供应用程序 APIs 来访问附件，例如加载附件，获取附件，插入/删除附件，以及访问附件的属性。

Example:

5.6.1 如何将指定文件嵌入到 PDF 文档

```
import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.pdf.Attachments;
import com.foxit.sdk.pdf.FileSpec;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.objects.PDFNameTree;
...
try {
    String pdfpath = "XXX/Sample.pdf";
    PDFDoc doc = new PDFDoc(pdfpath);
    doc.load(null);

    // Embed the specified file to PDF document.
    String filePath = "/xxx/fileToBeEmbedded.xxx";
    PDFNameTree nameTree = new PDFNameTree(doc, PDFNameTree.e_EMBEDDEDFILES);
    Attachments attachments = new Attachments(doc, nameTree);
    FileSpec fileSpec = new FileSpec(doc);
    fileSpec.setFileName(filePath);
    if (!fileSpec.embed(filePath)) return;
    attachments.addEmbeddedFile(filePath, fileSpec);
} catch (PDFException e) {
    e.printStackTrace();
}
```

...

5.6.2 如何从 PDF 文档中导出嵌入的附件文件，并将其另存为单个文件

```
import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.pdf.Attachments;
import com.foxit.sdk.pdf.FileSpec;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.objects.PDFNameTree;
...
try {
    String pdfpath = "XXX/Sample.pdf";
    PDFDoc doc = new PDFDoc(pdfpath);
    doc.load(null);

    PDFNameTree nameTree = new PDFNameTree(doc, PDFNameTree.e_EMBEDDEDFILES);
    Attachments attachments = new Attachments(doc, nameTree);
    // Extract the embedded attachment file.
    int count = attachments.getCount();
    for (int i = 0; i < count; i++) {
        String key = attachments.getKey(i);
        if (key != null) {
            FileSpec fileSpec1 = attachments.getEmbeddedFile(key);
            String exportedFile = "/somewhere/" + fileSpec1.getFileName();
            if (fileSpec1 != null && !fileSpec1.isEmpty()) {
                fileSpec1.exportToFile(exportedFile);
            }
        }
    }
} catch (PDFException e) {
    e.printStackTrace();
}
...
}
```

5.7 Annotation

一个 annotation 注释将对象（如注释，线条和高亮）与 PDF 文档页面上的位置相关联。PDF 包括如 Table 5-1 中列出的各种标准注释类型。在这些注释类型中，许多被定义为标记注释，因为它们主要用于标记 PDF 文档。Table 5-1 中的 "Markup" 列用以说明是否为标记注释。

Foxit PDF SDK 支持 PDF Reference 中定义的大多数注释类型。Foxit PDF SDK 提供了注释创建，属性访问和修改，外观设置和绘制的 APIs。

Table 5-1

Annotation type	Description	Markup	Supported by SDK
Text(Note)	Text annotation	Yes	Yes
Link	Link Annotation	No	Yes
FreeText (TypeWriter/TextBox/Callout)	Free text annotation	Yes	Yes
Line	Line annotation	Yes	Yes
Square	Square annotation	Yes	Yes
Circle	Circle annotation	Yes	Yes
Polygon	Polygon annotation	Yes	Yes
PolyLine	PolyLine annotation	Yes	Yes
Highlight	Highlight annotation	Yes	Yes
Underline	Underline annotation	Yes	Yes
Squiggly	Squiggly annotation	Yes	Yes
StrikeOut	StrikeOut annotation	Yes	Yes
Stamp	Stamp annotation	Yes	Yes
Caret	Caret annotation	Yes	Yes
Ink(pencil)	Ink annotation	Yes	Yes
Popup	Popup annotation	No	Yes
File Attachment	FileAttachment annotation	Yes	Yes
Sound	Sound annotation	Yes	No
Movie	Movie annotation	No	No
Widget*	Widget annotation	No	Yes
Screen	Screen annotation	No	Yes
PrinterMark	PrinterMark annotation	No	No
TrapNet	Trap network annotation	No	No
Watermark*	Watermark annotation	No	No
3D	3D annotation	No	No
Redact	Redact annotation	Yes	Yes

备注：Foxit SDK 支持名为 PSI (pressure sensitive ink, 压感笔迹) 的自定义注释类型。在 PDF 规范中没有对该注释进行描述。通常，PSI 用于手写功能，Foxit SDK 将其视为 PSI 注释，以便其他 PDF 产品可以对其进行相关处理。

Example:

5.7.1 如何向 PDF 页面中添加注释

```
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.fxcrt.RectF;
import com.foxit.sdk.common.fxcrt.RectFArray;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.PDFPage;
import com.foxit.sdk.pdf.TextSearch;
import com.foxit.sdk.pdf.annots.Annot;
import com.foxit.sdk.pdf.annots.Note;
import com.foxit.sdk.pdf.annots.QuadPoints;
import com.foxit.sdk.pdf.annots.QuadPointsArray;
import com.foxit.sdk.pdf.annots.TextMarkup;
import com.foxit.sdk.common.fxcrt.PointF;

...
// Add text annot.
try {
    String pdfpath = "xxx/Sample.pdf";
    PDFDoc doc = new PDFDoc(pdfpath);
    doc.load(null);
    PDFPage pdfPage = doc.getPage(1);
    RectF rect = new RectF(100, 100, 120, 120);
    Note note = new Note(pdfPage.addAnnot(Annot.e_Note, rect));
    if (note == null || note.isEmpty()){
        return;
    }
    note.setIconName("Comment");
    // Set color to blue.
    note.setBorderColor(0xff0000ff);
    note.setContent("This is the note comment, write any content here.");
    note.resetAppearanceStream();

    // The following code demonstrates how to add hightlight annotation on the searched text.
    TextSearch textSearch = new TextSearch(pdfPage.getDocument(), null, TextPage.e_ParseTextNormal);
    if (textSearch == null || textSearch.isEmpty()){
        return;
    }
    // Suppose that the text for highlighting is "foxit".
    textSearch.setPattern("foxit");
    boolean bMatched = textSearch.findNext();
    if (bMatched) {
        RectFArray rects = textSearch.getMatchRects();
        int rectCount = rects.getSize();
        // Fill the quadpoints array according to the text rects of matched result.
        QuadPointsArray arrayOfQuadPoints = new QuadPointsArray();
        for (int i = 0; i < rectCount; i++) {
```

```
rect = rects.getAt(i);
QuadPoints quadPoints = new QuadPoints();
PointF point = new PointF();
point.set(rect.getLeft(), rect.getTop());
quadPoints.setFirst(point);
point.set(rect.getRight(), rect.getTop());
quadPoints.setSecond(point);
point.set(rect.getLeft(), rect.getBottom());
quadPoints.setThird(point);
point.set(rect.getRight(), rect.getBottom());
quadPoints.setFourth(point);
arrayOfQuadPoints.add(quadPoints);
}

// Just set an empty rect to markup annotation, the annotation rect will be calculated according to the quadpoints that set to it later.

rect = new RectF(0, 0, 0, 0);
TextMarkup textMarkup = new TextMarkup(pdfPage.addAnnot(Annot.e_HIGHLIGHT, rect));
// Set the quadpoints to this markup annot.
textMarkup.setQuadPoints(arrayOfQuadPoints);
// set to red.
textMarkup.setBorderColor(0xffff0000);
// set to thirty-percent opacity.
textMarkup.setOpacity(0.3f);
// Generate the appearance.
textMarkup.resetAppearanceStream();
}

} catch (Exception e) {}
```

5.7.2 如何删除 PDF 页面中的注释

```
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.PDFPage;
import com.foxit.sdk.pdf.annots.Annot;
...

try {
    String pdfpath = "xxx/Sample.pdf";
    PDFDoc doc = new PDFDoc(pdfpath);
    doc.load(null);
    PDFPage pdfPage = doc.getPage(1);
    Annot annot = pdfPage.getAnnot(0);
    if (annot == null || annot.isEmpty())
        return;
    // Remove the first annot, so the second annot will become first.
    pdfPage.removeAnnot(annot);
} catch (Exception e) {}
```

5.8 Form

Form (AcroForm) 是用于收集用户交互信息的表单域的集合。Foxit PDF SDK 提供了以编程方式查看和编辑表单域的 APIs。在 PDF 文档中，表单域通常用于收集数据。 Form 类提供了 APIs 用来检索表单域或表单控件，导入/导出表单数据，以及其他功能，例如：

- 检索表单域，使用 [Form.getFieldCount](#) 和 [Form.getField](#) 接口。
- 检索 PDF 页面中的表单控件，使用 [Form.getControlCount](#) 和 [Form.getControl](#) 接口。
- 从 XML 文件导入表单数据，使用 [Form.importFromXML](#) 接口；导出表单数据到 XML 文件，使用 [Form.exportToXML](#) 接口。
- 检索 form filler 对象，使用 [Form.getFormFiller](#) 接口。

从FDF/XFDF文件中导入表单数据，或者导出数据到FDF/XFDF文件，请参考[PDFDoc.importFromFDF](#) 和 [PDFDoc.exportToFDF](#) 接口。

Example:

5.8.1 如何通过 XML 文件导入表单数据或将表单数据导出到 XML 文件

```
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.interform.Form;
...
// Check if the document has a form.
try {
    String pdfpath = "xxx/Sample.pdf";
    PDFDoc doc = new PDFDoc(pdfpath);
    doc.load(null);
    boolean hasForm = doc.hasForm();
    if(hasForm) {
        // Create a form object from document.
        Form form = new Form(doc);
        // Export the form data to a XML file.
        form.exportToXML("/somewhere/export.xml");
        // Or import the form data from a XML file.
        form.importFromXML("/somewhere/export.xml");
    }
} catch (Exception e) {}
```

5.9 Security

Foxit PDF SDK 提供了一系列加密和解密功能，以满足不同级别的文档安全保护。用户可以使用常规密码加密和证书驱动加密，或使用自己的安全处理机制来自定义安全实现。

Example:

5.9.1 如何使用密码加密 PDF 文件

```
import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.SecurityHandler;
import com.foxit.sdk.pdf.StdEncryptData;
import com.foxit.sdk.pdf.StdSecurityHandler;

...
// Encrypt the source pdf document with specified owner password and user password, the encrypted PDF will be
// saved to the path specified by parameter savePath.
public boolean encryPDF(PDFDoc pdfDoc, byte[] ownerPassword, byte[] userPassword, String savePth) {
    if (pdfDoc == null || (ownerPassword == null && userPassword == null) || savePth == null)
        return false;
    // The encryption setting data. Whether to encrypt meta data:true, User permission: modify,assemble,fill form.
    // Cipher algorithm:AES 128.
    StdEncryptData encryptData = new StdEncryptData(true,
        PDFDoc.e_PermModify | PDFDoc.e_PermAssemble | PDFDoc.e_PermFillForm,
        SecurityHandler.e_CipherAES, 16);

    StdSecurityHandler securityHandler = new StdSecurityHandler();
    try {
        if (!securityHandler.initialize(encryptData, userPassword, ownerPassword)) return false;
        pdfDoc.setSecurityHandler(securityHandler);
        if (!pdfDoc.saveAs(savePth, PDFDoc.e_SaveFlagNormal)) return false;
        return true;
    } catch (PDFException e) {
        e.printStackTrace();
    }
    return false;
}
```

5.10 Signature

PDF 签名可用于创建和签署 PDF 文档的数字签名，从而保护文档内容的安全性并避免文档被恶意篡改。它可以让接收者确保其收到的文档是由签名者发送的，并且文档内容是完整和未被篡改的。Foxit PDF SDK 提供 APIs 用来创建数字签名，验证签名的有效性，删除现有的数字签名，获取和设置数字签名的属性，显示签名和自定义签名表单域的外观。

备注： Foxit PDF SDK 提供了默认签名回调函数，支持如下两种类型的 *signature filter* 和 *subfilter*:

(1) *filter*: Adobe.PPKLite *subfilter*: adbe.pkcs7.detached

(2) filter: Adobe.PPKLite subfilter: adbe.pkcs7.sha1

如果您使用以上任意一种的 *signature filter* 和 *subfilter*, 您可以直接签名 PDF 文档和验证签名的有效性, 而不需要注册自定义回调函数。

Example:

5.10.1 如何对 PDF 文档进行签名，并验证签名

```
import com.foxit.sdk.PDFException;
import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.Constants;
import com.foxit.sdk.common.Progressive;
import com.foxit.sdk.common.fxcrt.RectF;
import com.foxit.sdk.pdf.PDFDoc;
import com.foxit.sdk.pdf.PDFPage;
import com.foxit.sdk.pdf.Signature;
...
// Sample code demonstrate signing and verifying of PDF signature.
public void addNewSignatureAndSign(PDFPage page, RectF rect) {
    try {
        // Add a new signature on the specified page rect.
        Signature signature = page.addSignature(rect);
        // Set the appearance flags, if the specified flag is on, then the associated key will be displayed on the signature
        // appearance.
        signature.setAppearanceFlags(Signature.e_APFlagLabel | Signature.e_APFlagDN | Signature.e_APFlagText
            | Signature.e_APFlagLocation | Signature.e_APFlagReason | Signature.e_APFlagSigner);
        // Set signer.
        signature.setKeyValue(Signature.e_KeyNameSigner, "Foxit");
        // Set location.
        signature.setKeyValue(Signature.e_KeyNameLocation, "AnyWhere");
        // Set reason.
        signature.setKeyValue(Signature.e_KeyNameReason, "ANyReason");
        // Set contact info.
        signature.setKeyValue(Signature.e_KeyNameContactInfo, "AnyInfo");
        // Set domain name.
        signature.setKeyValue(Signature.e_KeyNameDN, "AnyDN");
        // Set description.
        signature.setKeyValue(Signature.e_KeyNameText, "AnyContent");
        // Filter "Adobe.PPKLite" is supported by default.
        signature.setFilter("Adobe.PPKLite");
        // SubFilter "adbe.pkcs7.sha1" or "adbe.pkcs7.detached" are supported by default.
        signature.setSubFilter("adbe.pkcs7.detached");

        // The input PKCS#12 format certificate, which contains the public and private keys.
        String certPath = "/somewhere/cert.pfx";
        // Password for that certificate.
    }
}
```

```
byte[] certPassword = "123".getBytes();
String signedPDFPath = "/somewhere/signed.pdf";
// Start to sign the signature, if everything goes well, the signed PDF will be saved to the path specified by
"save_path".
Progressive progressive = signature.startSign(certPath, certPassword, Signature.e_DigestSHA1,
signedPDFPath, null, null);
if (progressive != null) {
    int state = Progressive.e_ToBeContinued;
    while (state == Progressive.e_ToBeContinued) {
        state = progressive.resume();
    }
    if (state != Progressive.e_Finished) return;
}

// Get the signatures from the signed PDF document, then verify them all.
PDFDoc pdfDoc = new PDFDoc(signedPDFPath);
int err = pdfDoc.load(null);
if (err != Constants.e_ErSuccess) return;
int count = pdfDoc.getSignatureCount();
for (int i = 0; i < count; i++) {
    Signature sign = pdfDoc.getSignature(i);
    if (sign != null && !sign.isEmpty()) {
        Progressive progressive_1 = sign.startVerify(null, null);
        if (progressive_1 != null) {
            int state = Progressive.e_ToBeContinued;
            while (state == Progressive.e_ToBeContinued) {
                state = progressive_1.resume();
            }
            if (state != Progressive.e_Finished) continue;
        }
        int verifiedState = sign.getState();
        if ((verifiedState & sign.e_StateVerifyValid) == sign.e_StateVerifyValid) {
            Log.d("Signature", "addNewSignatureAndSign: Signature" + i + "is valid.");
        }
    }
} catch (PDFException e) {
    e.printStackTrace();
}
}
```

6 创建自定义工具

使用 Foxit PDF SDK for Android 创建自定义工具非常简单。UI Extensions Component 中已经实现了一些工具，开发人员可以在这些工具的基础上进行二次开发，或者参考这些工具来创建新的工具。为了快速创建自己的工具，我们建议您参阅 "libs" 文件夹下的 **uiextensions_src** 工程。

创建一个新的自定义工具，最主要的是创建一个 Java 类，然后实现 "**ToolHandler.java**" 接口。

在本节中，我们通过创建一个区域屏幕截图工具来展示如何使用 Foxit PDF SDK for Android 创建一个自定义工具。该工具帮助用户在 PDF 页面中选择一个区域进行截图，并将其保存为图片。现在，让我们开始吧。

为方便起见，我们将基于 "samples" 文件夹下的 "**viewer_ctrl_demo**" 工程来构建该工具。实现该工具的步骤如下：

- 创建名为 **ScreenCaptureToolHandler** 的 Java 类，该类实现 "**ToolHandler.java**" 接口。
- 处理 **onTouchEvent** 和 **onDraw** 事件。
- 实例化 **ScreenCaptureToolHandler** 对象，然后将其注册到 **UIExtensionsManager**。
- 将 **ScreenCaptureToolHandler** 对象设置为当前的 tool handler。

步骤 1：创建一个名为 **ScreenCaptureToolHandler** 的 Java 类，该类实现 "**ToolHandler.java**" 接口。

- a) 在 Android Studio 中加载 "**viewer_ctrl_demo**" 工程。在 "com.foxit.pdf.viewctrl" 包中创建名为 "**ScreenCaptureToolHandler**" 的 Java 类。
- b) **ScreenCaptureToolHandler.java** 类实现 **ToolHandler** 接口，如 Figure 6-1 所示。

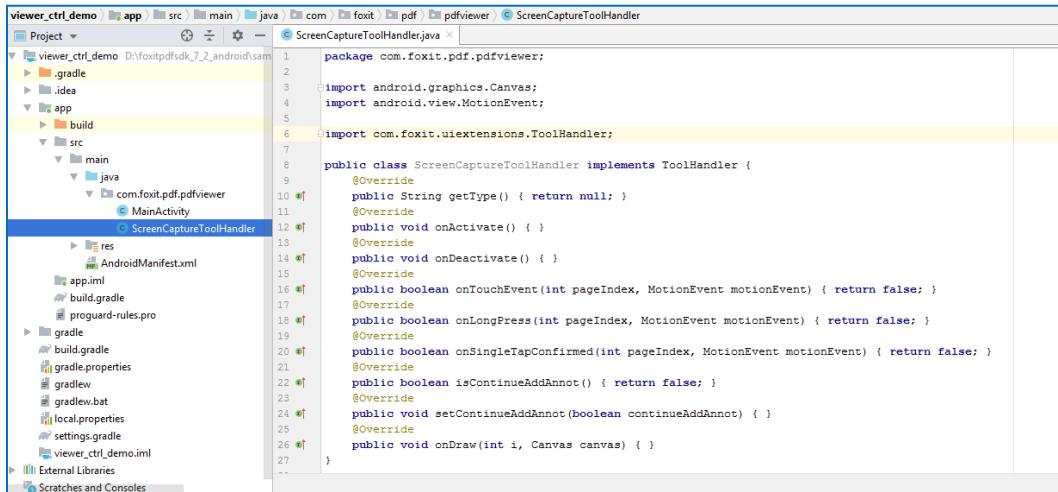


Figure 6-1

步骤 2: 处理 onTouchEvent 和 onDraw 事件。

更新 **ScreenCaptureToolHandler.java**, 如下所示:

```

package com.foxit.pdf.pdffviewer;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PointF;
import android.graphics.Rect;
import android.graphics.RectF;
import android.view.MotionEvent;
import android.widget.Toast;

import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.PDFException;
import com.foxit.sdk.common.Progressive;
import com.foxit.sdk.common.fxcrt.Matrix2D;
import com.foxit.sdk.pdf.PDFPage;
import com.foxit.sdk.common.Renderer;
import com.foxit.uiextensions.ToolHandler;
import com.foxit.uiextensions.UIExtensionsManager;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
  
```

```
public class ScreenCaptureToolHandler implements ToolHandler {

    private Context mContext;
    private PDFViewCtrl mPdfViewCtrl;

    public ScreenCaptureToolHandler(Context context, PDFViewCtrl pdfViewCtrl) {
        mPdfViewCtrl = pdfViewCtrl;
        mContext = context;
    }

    @Override
    public String getType() {
        return "";
    }

    @Override
    public void onActivate() {

    }

    @Override
    public void onDeactivate() {

    }

    private PointF mStartPoint = new PointF(0, 0);
    private PointF mEndPoint = new PointF(0, 0);
    private PointF mDownPoint = new PointF(0, 0);
    private Rect mRect = new Rect(0, 0, 0, 0);
    private RectF mNowRect = new RectF(0, 0, 0, 0);
    private int mLastPageIndex = -1;

    // Handle OnTouch event
    @Override
    public boolean onTouchEvent(int pageIndex, MotionEvent motionEvent) {
        // Get the display view point in device coordinate system
        PointF devPt = new PointF(motionEvent.getX(), motionEvent.getY());
        PointF point = new PointF();
        // Convert display view point to page view point.
        mPdfViewCtrl.convertDisplayPtToPageViewPt(devPt, point, pageIndex);
        float x = point.x;
        float y = point.y;

        switch (motionEvent.getAction()) {
            // Handle ACTION_DOWN event: get the coordinates of the StartPoint.
            case MotionEvent.ACTION_DOWN:
                if (mLastPageIndex == -1 || mLastPageIndex == pageIndex) {

```

```
mStartPoint.x = x;
mStartPoint.y = y;
mEndPoint.x = x;
mEndPoint.y = y;
mDownPoint.set(x, y);
if (mLastPageIndex == -1) {
    mLastPageIndex = pageIndex;
}
}
return true;

// Handle ACTION_Move event.
case MotionEvent.ACTION_MOVE:
if (mLastPageIndex != pageIndex)
    break;
if (!mDownPoint.equals(x, y)) {
    mEndPoint.x = x;
    mEndPoint.y = y;

    // Get the coordinates of the Rect.
    getDrawRect(mStartPoint.x, mStartPoint.y, mEndPoint.x, mEndPoint.y);

    // Convert the coordinates of the Rect from float to integer.
    mRect.set((int) mNowRect.left, (int) mNowRect.top, (int) mNowRect.right, (int)
mNowRect.bottom);

    // Refresh the PdfViewCtrl, then the onDraw event will be triggered.
    mPdfViewCtrl.refresh(pageIndex, mRect);
    mDownPoint.set(x, y);
}
return true;

// Save the selected area as a bitmap.
case MotionEvent.ACTION_UP:
if (mLastPageIndex != pageIndex)
    break;
if (!mStartPoint.equals(mEndPoint.x, mEndPoint.y)) {
    renderToBmp(pageIndex, "/mnt/sdcard/ScreenCapture.bmp");
    Toast.makeText(mContext, "The selected area was saved as a bitmap stored in the
/mnt/sdcard/ScreenCapture.bmp", Toast.LENGTH_LONG).show();
}
mDownPoint.set(0, 0);
mLastPageIndex = -1;
return true;
default:
return true;
}
```

```
    return true;
}

// Save a bimap to a specified path.
public static void saveBitmap(Bitmap bm, String outPath) throws IOException {
    File file = new File(outPath);
    file.createNewFile();

    FileOutputStream fileout = null;
    try {
        fileout = new FileOutputStream(file);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }

    bm.compress(Bitmap.CompressFormat.JPEG, 100, fileout);
    try {
        fileout.flush();
        fileout.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

// Render the selected area to a bitmap.
private void renderToBmp(int pageIndex, String filePath) {
    try {
        PDFPage page = mPdfViewCtrl.getDoc().getPage(pageIndex);

        mPdfViewCtrl.convertPageViewRectToPdfRect(mNowRect, mNowRect, pageIndex);
        int width = (int) page.getWidth();
        int height = (int) page.getHeight();

        Bitmap bmp = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        bmp.eraseColor(Color.WHITE);

        // Create a Renderer object
        Renderer renderer = new Renderer(bmp, true);

        // Get the display matrix.
        Matrix2D matrix = page.getDisplayMatrix(0, 0, width, height, 0);
        Progressive progress = renderer.startRender(page, matrix, null);
        int state = Progressive.e_ToBeContinued;
        while (state == Progressive.e_ToBeContinued) {
            state = progress.resume();
        }
    }
}
```

```
// Create a bitmap with the size of the selected area.  
        bmp = Bitmap.createBitmap(bmp, (int) mNowRect.left, (int) (height - mNowRect.top), (int)  
mNowRect.width(), (int) Math.abs(mNowRect.height()));  
        try {  
            saveBitmap(bmp, filePath);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
        } catch (PDFException e) {  
            e.printStackTrace();  
        }  
    }  
  
    // Get the coordinates of a Rect.  
    private void getDrawRect(float x1, float y1, float x2, float y2) {  
        float minx = Math.min(x1, x2);  
        float miny = Math.min(y1, y2);  
        float maxx = Math.max(x1, x2);  
        float maxy = Math.max(y1, y2);  
  
        mNowRect.left = minx;  
        mNowRect.top = miny;  
        mNowRect.right = maxx;  
        mNowRect.bottom = maxy;  
    }  
  
    @Override  
    public boolean onLongPress(int pageIndex, MotionEvent motionEvent) {  
        return false;  
    }  
  
    @Override  
    public boolean onSingleTapConfirmed(int pageIndex, MotionEvent motionEvent) {  
        return false;  
    }  
  
    @Override  
    public boolean isContinueAddAnnot() {  
        return false;  
    }  
  
    @Override  
    public void setContinueAddAnnot(boolean continueAddAnnot) {  
    }  
  
    // Handle the drawing event.
```

```

@Override
public void onDraw(int i, Canvas canvas) {
    if (((UIExtensionsManager) mPdfViewCtrl).getUIExtensionsManager().getCurrentToolHandler() != this)
        return;
    if (mLastPageIndex != i) {
        return;
    }
    canvas.save();
    Paint mPaint = new Paint();
    mPaint.setStyle(Paint.Style.STROKE);
    mPaint.setAntiAlias(true);
    mPaint.setDither(true);
    mPaint.setColor(Color.BLUE);
    mPaint.setAlpha(200);
    mPaint.setStrokeWidth(3);
    canvas.drawRect(mNowRect, mPaint);
    canvas.restore();
}
}

```

步骤 3: 实例化 **ScreenCaptureToolHandler** 对象，然后将其注册到 **UIExtensionsManager**。

```

private ScreenCaptureToolHandler screenCapture = null;
...
screenCapture = new ScreenCaptureToolHandler(mContext, pdfViewCtrl);
uiExtensionsManager.registerToolHandler(screenCapture);

```

步骤 4: 将 **ScreenCaptureToolHandler** 对象设置为当前的 tool handler。

```
uiExtensionsManager.setCurrentToolHandler(screenCapture);
```

现在，我们已经完成了自定义工具的创建。为了验证该工具，我们需要在 **MainActivity.java** 中添加一个 action 菜单，以及步骤 3 和步骤 4 中的代码。

首先，在 "app/src/main/res/menu" 目录下的 **Main.xml** 文件中添加一个 action 菜单，如下所示。

```

<item
    android:id="@+id/ScreenCapture"
    android:title="@string/screencapture"/>

```

在 "app/src/main/res/values/strings.xml" 中，添加以下字符串：

```
<string name="screencapture">ScreenCapture</string>
```

然后，将以下代码添加到 **MainActivity.java** 中的 **onActionItemClicked()** 函数中。

```

if (itemId == R.id.ScreenCapture) {
    if (screenCapture == null) {
        screenCapture = new ScreenCaptureToolHandler(mContext, pdfViewCtrl);
    }
}

```

```
    uiExtensionsManager.registerToolHandler(screenCapture);
}
uiExtensionsManager.setCurrentToolHandler(screenCapture);
}
```

请首先实例化一个 **ScreenCaptureToolHandler** 对象，如 (**private ScreenCaptureToolHandler screenCapture = null;**)。

完成上述所有工作后，可以编译和运行 demo 了。

备注: 在这里，我们使用 AVD 10.0 来运行该 demo。请确保您已经将 "Sample.pdf" 文档添加到模拟器 SD 卡中的正确位置(与 demo 中的文件路径匹配)。

当编译完 demo 并在模拟器上安装 APK 后，在弹出的窗口点击 "**Allow**" 允许 demo 访问设备上的文件。点击页面上的任意位置，会出现上下文操作栏，然后点击 **:**(更多按钮)，找到 **ScreenCapture** 菜单项，如 Figure 6-2 所示。

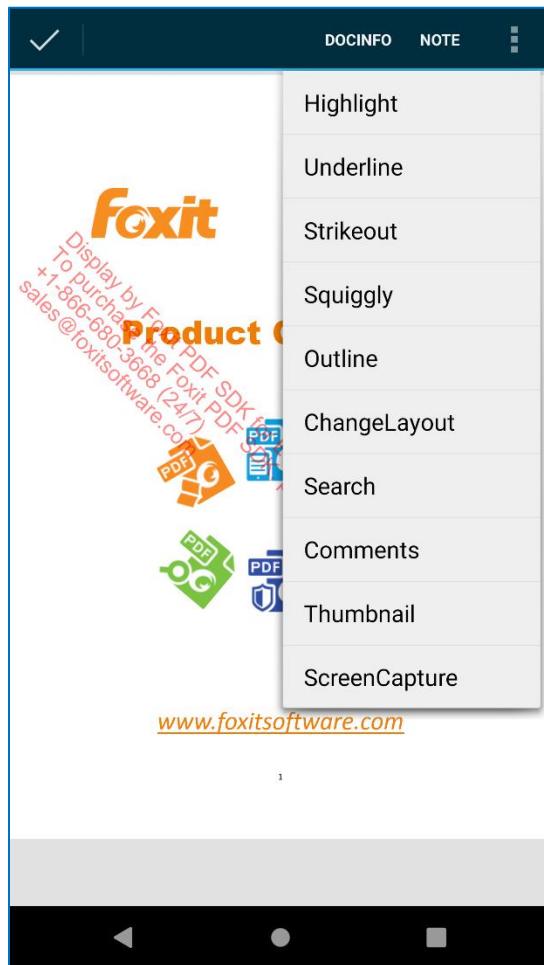


Figure 6-2

单击 **ScreenCapture**，在 PDF 页面长按并选择一个矩形区域，然后将弹出一个如 Figure 6-3 所示的消息框。该消息框指示 bitmap (选定区域) 保存的位置。



Figure 6-3

为了验证该工具是否成功截取和保存了所选区域，我们需要找到保存的屏幕截图。单击 IDE 右下角的 **Device File Explorer**，可以在 SD 卡中看到名为 "ScreenCapture.bmp" 的屏幕截图，如 Figure 6-4 所示。

Name	Permissions	Date	Size
proc	dr-xr-xr-x	2021-06-06 15:00	0 B
product	lrw-r--r--	2020-07-21 08:56	15 B
res	drwxr-xr-x	2020-07-21 08:41	4 KB
sbin	drwxr-x---	2020-07-21 08:18	4 KB
sdcard	lrw-r--r--	2020-07-21 08:56	21 B
Alarms	drwxrwx--x	2021-06-06 15:00	4 KB
Android	drwxrwx--x	2021-06-06 15:00	4 KB
DCIM	drwxrwx--x	2021-06-06 15:00	4 KB
Download	drwxrwx--x	2021-06-06 15:00	4 KB
FoxitSDK	drwxrwx--x	2021-06-06 15:27	4 KB
Movies	drwxrwx--x	2021-06-06 15:00	4 KB
Music	drwxrwx--x	2021-06-06 15:00	4 KB
Notifications	drwxrwx--x	2021-06-06 15:00	4 KB
Pictures	drwxrwx--x	2021-06-06 15:00	4 KB
Podcasts	drwxrwx--x	2021-06-06 15:00	4 KB
Ringtones	drwxrwx--x	2021-06-06 15:00	4 KB
ScreenCapture.bmp	-rw-rw----	2021-06-06 17:56	101 KB
storage	drwxr-xr-x	2021-06-06 15:00	100 B
sys	dr-xr-xr-x	2021-06-06 15:00	0 B
system	drwxr-xr-x	2020-07-21 08:56	4 KB
vendor	drwxr-xr-x	2020-07-21 08:42	4 KB
adb_keys	-rw-r--r--	2020-07-21 08:41	723 B
bugreports	lrw-r--r--	2020-07-21 08:56	50 B
charger	lrw-r--r--	2020-07-21 08:56	19 B
default.prop	lrw-----	2020-07-21 08:56	23 B

Figure 6-4

右键单击 "ScreenCapture.bmp" 图片，选择 "**Save AS...**" 将其保存到所需的位置。打开图片，可以看到其如 Figure 6-5 所示。



Figure 6-5

如您所见，我们已成功创建了区域屏幕截图工具。这只是一个示例，用来说明如何使用 Foxit PDF SDK for Android 创建自定义工具。您可以参考该示例或者我们的 demos 来开发您需要的工具。

7 使用 Cordova 实现 Foxit PDF SDK for Android

在开发跨平台移动应用程序时，Apache Cordova 是一个理想的开源框架。'cordova-plugin-**foxitpdf**'是我们提供的使用 Foxit PDF SDK for Android 的移动框架插件之一。该插件帮助您可以使
用 Cordova 框架实现强大的 PDF viewing 功能。通过此插件，您可以预览任何 PDF 文件，包括 PDF
2.0 标准的文件，XFA 文档和受 RMS 加密的文档，您还可以注释和编辑 PDF 文档。

对于 'cordova-plugin-foxitpdf' 插件的用法，请参阅网站
<https://github.com/foxitsoftware/cordova-plugin-foxitpdf>。

8 使用 React Native 实现 Foxit PDF SDK for Android

React Native 是一个使用 JavaScript 和 React 构建 native 应用程序的开源移动开发框架。**'react-native-foxitpdf'**是我们提供的使用 Foxit PDF SDK for Android 的移动框架插件之一。该插件帮助您可以使用 React Native 框架实现强大的 PDF viewing 功能。通过此插件，您可以预览任何 PDF 文件，包括 PDF 2.0 标准的文件，XFA 文档和受 RMS 加密的文档，您还可以注释和编辑 PDF 文档。

对于 '`react-native-foxitpdf`' 插件的用法，请参阅网站 <https://github.com/foxitsoftware/react-native-foxitpdf>。

9 使用 Xamarin 实现 Foxit PDF SDK for Android

Xamarin 是一个使用共享 C# 代码库构建 native 应用程序的跨平台开发框架。我们为 Android 和 iOS 平台分别提供了对应的 [bindings for Android and iOS](#) ('foxit_xamarin_android' 和 'foxit_xamarin_ios')，以便开发人员可以将 Foxit PDF SDK 强大的 PDF 功能无缝地集成到他们的 Xamarin 应用程序中。

对于 'foxit_xamarin_android' 插件的用法，请参阅网站 <https://github.com/foxitsoftware/xamarin-foxitpdf>。

10 FAQ

10.1 从指定的 PDF 文件路径打开一个 PDF 文档

如何从指定的 PDF 文件路径打开一个 PDF 文档？

Foxit PDF SDK for Android 提供了多个接口用来打开 PDF 文档。您可以从指定的 PDF 文件路径或从内存缓冲区打开一个 PDF 文档。对于指定的 PDF 文件路径，有两种方法可以使用。

第一种是使用 **openDoc** 接口，该接口包括以下的操作：创建 PDF 文档对象(**PDFDoc(String path)**)，加载文档内容(**load**)，以及将 PDF 文档对象设置给视图控件(**setDoc**)。以下是示例代码：

备注：**openDoc** 接口仅可用于从文件路径打开 PDF 文档。如果需要自定义加载 PDF 文档，可以在回调函数(**FileRead**) 中实现，然后使用带有回调函数 **FileRead** 的 **PDFDoc (FileRead fileRead)** 接口创建文档对象。接下来，使用 **load** 加载文档内容，并使用 **setDoc** 将 PDF 文档对象设置给视图控件。

```
// Assuming A PDFViewCtrl has been created.

// Open an unencrypted PDF document from a specified PDF file path.
String path = "/mnt/sdcard/input_files/Sample.pdf";
pdfViewCtrl.openDoc(path, null);
```

第二种是使用 **PDFDoc(String path)** 接口创建 PDF 文档对象，使用 **load** 接口加载文档内容，然后使用 **setDoc** 将 PDF 文档对象设置给视图控件。以下是示例代码：

```
// Assuming A PDFViewCtrl has been created.

String path = "/mnt/sdcard/input_files/Sample.pdf";
try {
    // Initialize a PDFDoc object with the path to the PDF file.
    PDFDoc document = new PDFDoc(path);

    // Load the unencrypted document content.
    document.load(null);

    // Set the document to view control.
    pdfViewCtrl.setDoc(document);
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

10.2 打开 PDF 文档时显示指定的页面

如何在打开 PDF 文档时，显示指定的页面？

为了在打开 PDF 文档时显示指定的页面，您需要使用接口 **gotoPage (int pageIndex)**。Foxit PDF SDK for Android 使用多线程来提高渲染速度，因此您需要确保在使用 **gotoPage** 接口之前，文档已经被成功加载。

请在 **IDocEventListener** 中实现回调接口，然后在 **onDocOpened** 事件中调用 **gotoPage** 接口。以下是示例代码：

```
// Assuming A PDFViewCtrl has been created.

// Register the PDF document event listener.
pdfViewCtrl.registerDocEventListener(docListener);

// Open an unencrypted PDF document from a specified PDF file path.
String path = "/mnt/sdcard/input_files/Sample.pdf";
pdfViewCtrl.openDoc(path, null);

...

PDFViewCtrl.IDocEventListener docListener = new PDFViewCtrl.IDocEventListener() {

    @Override
    public void onDocWillOpen() {}

    @Override
    public void onDocOpened(PDFDoc pdfDoc, int errCode) {
        pdfViewCtrl.gotoPage(2);
    }

    @Override
    public void onDocWillClose(PDFDoc pdfDoc) {}

    @Override
    public void onDocClosed(PDFDoc pdfDoc, int i) {}

    @Override
    public void onDocWillSave(PDFDoc pdfDoc) {}

    @Override
    public void onDocSaved(PDFDoc pdfDoc, int i) {}

};
```

10.3 License key 和序列号无法正常工作

从网站下载的 SDK 包，未进行任何更改，为什么 license key 和序列号无法正常工作？

通常，上传到网站的包，里面的 license key 和序列号是可以正常工作的。在上传到网站之前是经过测试的。因此，如果您发现 license key 和序列号无法使用，则可能是由设备的日期引起的。如果您设备的时间在下载包 "libs" 文件夹下 **rdk_key.txt** 文件中的 StartDate 之前，则 "librdk.so" 库将无法解锁。请检查您设备的日期。

10.4 在 PDF 文档中添加 link 注释

如何在 PDF 文档中添加 link 注释？

为了将 link 注释添加到 PDF 文档，首先需要调用 **PDFPage.addAnnot** 将一个 link 注释添加到指定页面，然后调用 **Action.Create** 创建一个 action，并将该 action 设置给刚添加的 link 注释。以下是在 PDF 首页添加一个 URI link 注释的示例代码：

```
private Link linkAnnot = null;  
...  
  
String path = "mnt/sdcard/input_files/sample.pdf";  
try {  
  
    // Initialize a PDFDoc object with the path to the PDF file.  
    PDFDoc document = new PDFDoc(path);  
  
    // Load the unencrypted document content.  
    document.load(null);  
  
    // Get the first page of the PDF file.  
    PDFPage page = document.getPage(0);  
  
    // Add a link annotation to the first page.  
    linkAnnot = new Link (page.addAnnot(Annot.e_Link, new RectF(250, 650, 400, 750)));  
  
    // Create a URI action and set the URI.  
    URIAction uriAction = new URIAction(Action.create(document, Action.e_TypeURI));  
    uriAction.setURI("www.foxitsoftware.com");  
  
    // Set the action to link annotation.  
    linkAnnot.setAction(uriAction);  
  
    // Reset appearance stream.  
    linkAnnot.resetAppearanceStream();  
  
    // Save the document that has added the link annotation.  
    document.saveAs("mnt/sdcard/input_files/sample_annot.pdf", PDFDoc.e_SaveFlagNormal);
```

```
} catch (Exception e) {
    e.printStackTrace();
}
```

10.5 向 PDF 文档中插入图片

如何向 PDF 文档中插入图片?

有两种方法可以帮助您将图片插入到 PDF 文档中。第一种是调用 **PDFPage.addImageFromFilePath** 接口。您可以参考如下示例代码，该代码将图片插入到 PDF 文档的首页：

备注：在调用 **PDFPage.addImageFromFilePath** 接口之前，您需要获取并解析将要添加图片的页面。

```
String path = "mnt/sdcard/input_files/sample.pdf";
try {

    // Initialize a PDFDoc object with the path to the PDF file.
    PDFDoc document = new PDFDoc(path);

    // Load the unencrypted document content.
    document.load(null);

    // Get the first page of the PDF file.
    PDFPage page = document.getPage(0);

    // Parse the page.
    if (!page.isParsed()) {
        Progressive parse = page.startParse(e_ParsePageNormal, null, false);
        int state = Progressive.e_ToBeContinued;
        while (state == Progressive.e_ToBeContinued) {
            state = parse.resume();
        }
    }

    // Add an image to the first page.
    page.addImageFromFilePath("mnt/sdcard/input_files/2.png", new PointF(20, 30), 60, 50, true);

    // Save the document that has added the image.
    document.saveAs("mnt/sdcard/input_files/sample_image.pdf", PDFDoc.e_SaveFlagNormal);

} catch (Exception e) {
    e.printStackTrace();
}
```

第二种是使用 **PDFPage.addAnnot** 接口在指定的页面添加一个 stamp，然后将图片转换为位图，并将位图设置给刚添加的 stamp 注释。您可以参考以下的示例代码，该代码将图片作为 stamp 注释插入到 PDF 文件的首页：

```
String path = "mnt/sdcard/input_files/sample.pdf";
try {

    // Initialize a PDFDoc object with the path to the PDF file.
    PDFDoc document = new PDFDoc(path);

    // Load the unencrypted document content.
    document.load(null);

    // Get the first page of the PDF file.
    PDFPage page = document.getPage(0);

    // Add a stamp annotation to the first page.
    Stamp stamp = new Stamp(page.addAnnot(Annot.e_Stamp, new RectF(100, 350, 250, 150)));

    // Load a local image and convert it to a Bitmap.
    Bitmap bitmap = BitmapFactory.decodeFile("mnt/sdcard/input_files/2.png");

    // Set the bitmap to the added stamp annotation.
    stamp.setImageBitmap(bitmap);

    // Reset appearance stream.
    stamp.resetAppearanceStream();

    // Save the document that has added the stamp annotation.
    document.saveAs("mnt/sdcard/input_files/sample_image.pdf", PDFDoc.e_SaveFlagNormal);

} catch (Exception e) {
    e.printStackTrace();
}
```

10.6 高亮 PDF 文档中的 links 和设置高亮颜色

如何设置是否高亮 PDF 文档中的 links？以及如何设置高亮的颜色？

默认情况下，高亮 PDF 文档中的 links 功能是启用的。如果您想要禁用它或者设置高亮颜色，可以通过 JSON 文件 (仅支持 6.3 及以上版本) 或调用 API 进行设置。

备注：如果您需要设置高亮的颜色，请确保 links 高亮的功能是启用的。

通过 JSON 文件

设置 **"highlightLink": false**，在 PDF 文档中禁用 links 高亮功能。

设置 **"highlightLinkColor": "#16007000"**，设置高亮颜色 (输入您需要的颜色值)。

通过调用 API

UIExtensionsManager.enableLinkHighlight() 接口用来设置是否在 PDF 文档中启用 links 高亮的功能。如果您不需要启用该功能，请将其参数设置为 "false"，如下所示：

```
// Assume you have already Initialized a UIExtensionsManager object  
uiExtensionsManager.enableLinkHighlight(false);
```

UIExtensionsManager.setLinkHighlightColor() 接口用来设置高亮的颜色。以下是调用此 API 的示例代码：

```
// Assume you have already Initialized a UIExtensionsManager object  
uiExtensionsManager.setLinkHighlightColor(0x4b0000ff);
```

10.7 高亮 PDF 文档中的表单域和设置高亮颜色

如何设置是否高亮 PDF 文档中的表单域？以及如何设置高亮的颜色？

默认情况下，高亮 PDF 文档中的表单域功能是启用的。如果您想要禁用它或者设置高亮颜色，可以通过 JSON 文件 (仅支持 6.3 及以上版本) 或调用 API 进行设置。

备注：如果您需要设置高亮的颜色，请确保表单域高亮的功能是启用的。

通过 JSON 文件

设置 "**"highlightForm": false**" 在 PDF 文档中禁用表单域高亮功能。

设置 "**"highlightFormColor": "#2000ffcc"**" 设置高亮颜色 (输入您需要的颜色值)。

通过调用 API

UIExtensionsManager.enableFormHighlight() 接口用来设置是否在 PDF 文档中启用表单域高亮的功能。如果您不需要启用该功能，请将其参数设置为 "false"，如下所示：

```
// Assume you have already Initialized a UIExtensionsManager object  
uiExtensionsManager.enableFormHighlight(false);
```

UIExtensionsManager.setFormHighlightColor() 接口用来设置高亮的颜色。以下是调用此 API 的示例代码：

```
// Set the highlight color to blue.  
uiExtensionsManager.setFormHighlightColor(0x4b0000ff);
```

10.8 支持全文索引搜索

Foxit PDF SDK for Android 是否支持全文索引搜索？如果支持，如何搜索在我的移动设备上离线存储的 PDF 文件？

是的。Foxit PDF SDK for Android 从 5.0 版本开始就支持全文索引搜索。

要使用此功能，请按照如下的步骤：

- 根据目录来创建一个文档来源，该目录为文档的搜索目录。

```
public DocumentsSource(String directory)
```

- 创建一个全文文本搜索对象，以及设置用于存储索引数据的数据库路径。

```
public FullTextSearch()  
public void set DataBasePath(String path DataBase)
```

- 开始索引文档来源中的 PDF 文档。

```
public Progressive startUpdateIndex(DocumentsSource source,  
PauseCallback pause, boolean reUpdate)
```

备注：您可以索引指定的PDF文件。例如，如果某个PDF文档的内容发生了更改，您可以使用以下的API对其重新进行索引：

```
public boolean updateIndexWithFilePath(java.lang.String filePath)
```

- 从索引数据源中搜索指定的内容。搜索的结果将通过指定的回调函数来返回给外部，每找到一个匹配结果，则调用一次回调函数。

```
public boolean searchOf(java.lang.String matchString,  
RankMode rankMode,  
SearchCallback searchCallback)
```

如下是使用全文索引搜索的示例代码：

```
String directory = "A search directory...";  
FullTextSearch search = new FullTextSearch();  
try {  
    String dbPath = "The path of data base to store the indexed data...";  
    search.set DataBasePath(dbPath);  
    // Get document source information.  
    DocumentsSource source = new DocumentsSource(directory);  
  
    // Create a Pause callback object implemented by users to pause the updating process.  
    PauseUtil pause = new PauseUtil(30);  
  
    // Start to update the index of PDF files which receive from the source.  
    Progressive progressive = search.startUpdateIndex(source, pause, false);
```

```
int state = Progressive.e_ToBeContinued;
while (state == Progressive.e_ToBeContinued) {
    state = progressive.resume();
}

// Create a callback object which will be invoked when a matched one is found.
MySearchCallback searchCallback = new MySearchCallback();

// Search the specified keyword from the indexed data source.
search.searchOf("looking for this text", RankMode.e_RankHitCountASC, searchCallback);
} catch (PDFException e) {
    e.printStackTrace();
}
```

PauseUtil 回调的示例代码如下所示：

```
public class PauseUtil extends PauseCallback{
    private long m_milliseconds = 0;
    private long m_startTime = 0;

    public PauseUtil(long milliseconds) {
        Date date = new Date();
        m_milliseconds = milliseconds;
        m_startTime = date.getTime();
    }

    @Override
    public boolean needToPauseNow() {
        // TODO Auto-generated method stub
        if (this.m_milliseconds < 1)
            return false;
        Date date = new Date();
        long diff = date.getTime() - m_startTime;
        if (diff > this.m_milliseconds) {
            m_startTime = date.getTime();
            return true;
        } else
            return false;
    }
}
```

MySearchCallback 回调的示例如下所示：

```
public class MySearchCallback extends SearchCallback {
    private static final String TAG = MySearchCallback.class.getCanonicalName();

    @Override
    public void release() {
    }

    @Override
```

```

public int retrieveSearchResult(String filePath, int pageIndex, String matchResult, int
matchStartTextIndex, int matchEndTextIndex) {
    String s = String.format("Found file is :%s \n Page index is :%d Start text index :%d End text
index :%d \n Match is :%s \n\n", filePath, pageIndex, matchStartTextIndex, matchEndTextIndex,
matchResult);
    Log.v(TAG, "retrieveSearchResult: " + s);
    return 0;
}
}

```

备注:

- Foxit PDF SDK for Android 提供的全文索引搜索将以递归的方式遍历整个目录，以便搜索目录下的文件和文件夹都会被索引。
- 如果需要中止索引进程，可以将 `pause` 回调参数传递给 `startUpdateIndex` 接口。其回调函数 `needToPauseNow` 都会被调用，一旦完成一个 PDF 文档的索引。因此，调用者可以在回调函数 `needToPauseNow` 返回 "true" 时中止索引进程。
- 索引数据库的位置由 `set DataBasePath` 接口设置。如果要清除索引数据库，请手动清除。目前，不支持从索引函数中删除指定文件相关的索引内容。
- `searchOf` 接口的每个搜索结果都由指定的回调返回到外部。一旦 `searchOf` 接口返回 "true" 或 "false"，则表示搜索已完成。

10.9 打印 PDF 文档

Foxit PDF SDK for Android 是否支持打印 PDF 文档？如果支持，如何使用？

是的。Foxit PDF SDK for Android 从 5.1 版本开始支持打印 PDF 文档的功能。您可以在 Complete PDF viewer demo 的 More Menu View 菜单中点击 Wireless Print 按钮来打印 PDF 文档。此外，您可以调用以下 API 来打印 PDF 文档：

```
// for iPhone and iTouch
```

```
public void startPrintJob(Context context, PDFDoc doc, String printJobName, String outputFileName,
IPrintResultCallback callback)
```

使用 PDF 打印功能的示例代码：

```
// Assume you have already Initialized a UIExtensionsManager object
```

```
PDFDoc doc = null;
IPrintResultCallback print_callback = new IPrintResultCallback() {
    @Override
    public void printFinished() {
    }

    @Override
```

```

public void printFailed() {
}

@Override
public void printCancelled() {
}
};

try {
    doc = new PDFDoc("/mnt/sdcard/input_files/Sample.pdf");
    doc.load(null);
} catch (PDFException e) {
    Assert.fail("unexpect a PDF Exception!!errCode = " + e.getLastErrorCode());
}
uiExtensionsManager.startPrintJob(getActivity(), doc, "print with name", "print_withAPI", print_callback);
}

```

10.10 夜间模式颜色设置

如何设置夜间模式颜色？

设置夜间模式颜色，需要首先调用 **PDFViewCtrl.setMappingModeBackgroundColor(int)** 和 **PDFViewCtrl.setMappingModeForegroundColor(int)** 接口根据您的需要设置颜色，然后使用 **PDFViewCtrl.setColorMode(int)** 设置颜色模式。

备注：如果颜色模式已经设置为
Renderer.e_ColorModeMapping/Renderer.e_ColorModeMappingGray，在调用
PDFViewCtrl.setMappingModeBackgroundColor(int) 和
PDFViewCtrl.setMappingModeForegroundColor(int) 接口之后，您仍然需要再次设置。否则，颜色
 设置可能不会起作用。

上述接口需要在 UI Extensions 组件的源代码中调用，请参考 4.3 小节 "[通过源代码自定义 UI 实现](#)" 将 "libs" 文件夹下的 "uiextensions_src" 工程添加到您的工程中。然后在 "com.foxit.uiextensions.pdfreader.impl.MainFrame.java" 文件中定位到 **onValueChanged** 函数，根据您的需要设置颜色。

设置夜间模式颜色的示例代码：

```

case IViewSettingsWindow.TYPE_DAY:
    mPageColorMode = (Integer) value;
    if (mPageColorMode == IViewSettingsWindow.DAY) {
        mUiExtensionsManager.getPDFViewCtrl().setBackgroundColor(AppResource.getColor(mContext,
R.color.ux_bg_color_docviewer));
        mUiExtensionsManager.getPDFViewCtrl().setNightMode(false);
    } else if (mPageColorMode == IViewSettingsWindow.NIGHT) {
        mUiExtensionsManager.getPDFViewCtrl().setBackgroundColor(Color.parseColor("#36404A"));
    }
}

```

```

mUiExtensionsManager.getPDFViewCtrl().setNightMode(true);
mUiExtensionsManager.setNightColorMode(UIExtensionsManager.NIGHTCOLORMODE_MAPPINGGRAY);
if (mUiExtensionsManager.getNightColorMode() ==
UIExtensionsManager.NIGHTCOLORMODE_MAPPINGGRAY) {
    mUiExtensionsManager.getPDFViewCtrl().setMappingModeForegroundColor(Color.argb(0xff, 0x00, 0xff,
0x00));
    mUiExtensionsManager.getPDFViewCtrl().setMappingModeBackgroundColor(Color.argb(0xff, 0xff, 0x00,
0x00));
    mUiExtensionsManager.getPDFViewCtrl().setColorMode(Renderer.e_ColorModeMappingGray);
}
}

```

10.11 输出 exception/crash 日志信息

如何在应用程序抛出异常或者 crash 时，输出 exception/crash 日志信息？

setExceptionLogger 接口用来输出 exception/crash 日志信息，该接口引用了第三方 xCrash 库。其使用方法如下：

- 1) 添加 XCrash 依赖。

```

dependencies {
    implementation 'com.iqiyi.xcrash:xcrash-android-lib:2.1.4'
}

```

- 2) 指定一个或多个 ABI。

```

android {
    defaultConfig {
        ndk {
            abiFilters 'armeabi', 'armeabi-v7a', 'arm64-v8a', 'x86', 'x86_64'
        }
    }
}

```

- 3) 在代码中调用 **setExceptionLogger**。

```

public class MainApplication extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        PDFViewCtrl.setExceptionLogger(this, "/mnt/sdcard/FoxitSDK/crash", new
PDFViewCtrl.IExceptionLogger() {
            @Override
            public void onExceptionLogger(String filePath)
                { Log.d("", "onExceptionLogger: " + filePath); }
        });
    }
}

```

10.12 减小 APK 的大小

如何减小 APK 的大小?

要减少 APK 的大小，您可以编译多个包含特定屏幕密度或者 ABI 文件的 APK。有关多 APK 更详细的介绍，请参阅 [Build Multiple APKs](#)。如下的代码片段是基于屏幕密度和 ABI 来配置多个 APK，该代码将添加到模块级的 **build.gradle** 文件中。

```
android {  
    ...  
    splits {  
        // Configures multiple APKs based on screen density.  
        density {  
            // Configures multiple APKs based on screen density.  
            enable true  
            // Specifies a list of screen densities Gradle should not create multiple APKs for.  
            exclude "ldpi", "xxhdpi", "xxxhdpi"  
            // Specifies a list of compatible screen size settings for the manifest.  
            compatibleScreens 'small', 'normal', 'large', 'xlarge'  
        }  
        // Configures multiple APKs based on ABI.  
        abi {  
            // Enables building multiple APKs per ABI.  
            enable true  
            // By default all ABIs are included, so use reset() and include to specify that we only  
            // want APKs for x86 and x86_64.  
            // Resets the list of ABIs that Gradle should create APKs for to none.  
            reset()  
            // Specifies a list of ABIs that Gradle should create APKs for.  
            include "x86", "x86_64", "armeabi-v7a", "arm64-v8a"  
            // Specifies that we do not want to also generate a universal APK that includes all ABIs.  
            universalApk false  
        }  
    }  
}
```

10.13 开启 shrink-code (设置 "minifyEnabled" 为 "true")

当在 App's build.gradle 中将 "minifyEnabled" 设置为 "true" 时，为什么在运行时会遇到一些异常？

当在 App's build.gradle 中将 "minifyEnabled" 设置为 "true" 来开启 shrink-code 时，请注意您需要在 **proguard-rules.pro** 文件中添加如下的内容，否则在运行时会抛出异常。

在 **proguard-rules.pro** 文件中，添加如下的内容：

```
-dontwarn com.foxit.sdk.**  
-keep class com.foxit.sdk.**{ *;}
```

```
-dontwarn com.microsoft.rightsmanagement.**
-keep class com.microsoft.rightsmanagement.** {*;}

-dontwarn com.microsoft.aad.adal.**
-keep class com.microsoft.aad.adal.** {*;}

-dontwarn com.edmodo.cropper.**
-keep class com.edmodo.cropper.** {*;}

-dontwarn org.bouncycastle.**
-keep class org.bouncycastle.** {*;}
```

10.14 本地化设置

如何进行本地化设置?

默认情况下，Foxit PDF SDK for Android 会根据您系统的当前语言自动切换 UI 语言，前提是 Foxit PDF SDK for Android 支持该语言。

当前，Foxit PDF SDK for Android 支持如下的语言：英语(English)，德语(German, de-CH, de-DE)，拉丁语(Latin, es-LA)，法语(Frence, fr-FR)，意大利语(Italian, it-IT)，韩语(Korean, ko)，荷兰语(Dutch, ni-NL)，葡萄牙语(Portuguese, pt-BR)，俄语(Russian, ru-Ru) 和 中文(Chinese, zh-CN, zh-TW)。这些语言的资源文件位于 "libs\uiextensions_src\src\main\res" 文件夹下。

如果您需要使用自己本地的语言(而该语言是 Foxit PDF SDK for Android 当前不支持的语言)，您需要首先将 UI 上面所有的字段都翻译成本地的语言，其次将翻译后的资源文件放入您工程中其他语言资源文件所在的同一目录中。然后，修改您系统的当前语言为您本地的语言，或者调用

Localization.setCurrentLanguage 接口使其生效。有关 **Localization.setCurrentLanguage** 接口更详细的说明，请参阅 "doc" 目录下的 API Reference。

例如，假如您需要将"complete_pdf_viewer" demo 的 UI 语言改为，您可以按照如下的步骤：

- a) 将"libs\uiextensions_src\src\main\res"目录下的 "**values**" (以此举例) 拷贝到 demo 的资源目录 "samples\complete_pdf_viewer\app\src\main\res"，并将其重命名为"**values-ja-rJA**"。
- b) 翻译(本地化) "**values-ja-rJA**" 文件夹下 XML 文件的所有字段。
- c) 然后，采用如下两种方法中的任意一种使本地化语言生效：
 - 将系统的当前语言修改为日文。
 - 调用 **Localization.setCurrentLanguage** 接口将当前语言设置为日文。

```
Locale locale = new Locale("ja","JA");
Localization.setCurrentLanguage(this.getContext(), locale);
```

10.15 迁移到 AndroidX

如何将 Foxit PDF SDK for Android 迁移到 AndroidX?

从 7.2 版本开始，Foxit PDF SDK for Android 只支持 AndroidX，不再支持 Android support 库。因此，如果您当前使用的是 7.2 版本之前的 SDK，并且您需要升级 SDK 以便使用最新发布中的相关新功能，那么您需要将 Foxit PDF SDK for Android 迁移到 AndroidX。

迁移要求

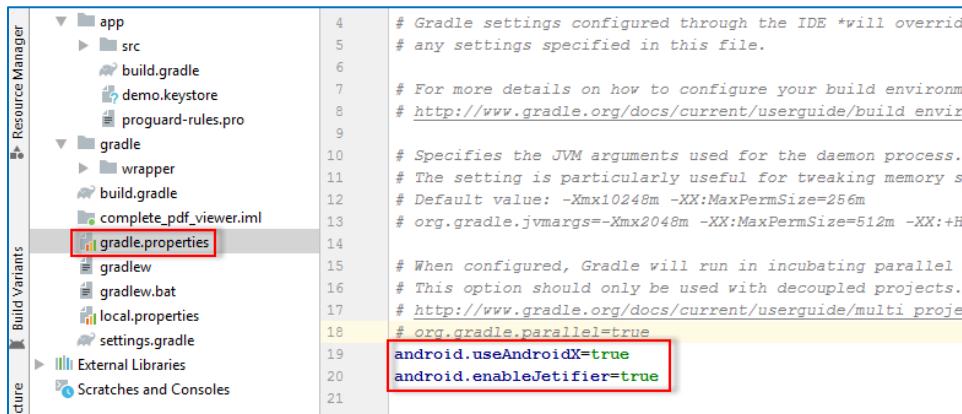
- Android Studio version >= 3.2.0
- Gradle Build Tool >= 3.2.0
- Gradle Version >= 4.6
- compileSdkVersion >= 28

迁移

关于迁移，您可以参阅[官方迁移文档](#)。在本教程中，我们将以 "samples" 文件夹下的 **complete_pdf_viewer** demo 为例来说明如何迁移到 AndroidX。

1) 修改工程级的 "gradle.properties"。在 "gradle.properties" 文件中添加如下的内容：

```
android.useAndroidX=true
android.enableJetifier=true
```



```

Resource Manager
Build Variants
Picture

app
src
build.gradle
demo.keystore
proguard-rules.pro
gradle
wrapper
build.gradle
complete_pdf_viewer.iml
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
External Libraries
Scratches and Consoles

4 # Gradle settings configured through the IDE *will override
5 # any settings specified in this file.
6
7 # For more details on how to configure your build environment
8 # http://www.gradle.org/docs/current/userguide/build\_environment.html
9
10 # Specifies the JVM arguments used for the daemon process.
11 # The setting is particularly useful for tweaking memory settings.
12 # Default value: -Xmx1024m -XX:MaxPermSize=256m
13 # org.gradle.jvmargs=-Xmx2048m -XX:MaxPermSize=512m -XX:+HeapDumpOnOutOfMemoryError -Dfile.encoding=UTF-8
14
15 # When configured, Gradle will run in incubating parallel mode.
16 # This option should only be used with decoupled projects.
17 # http://www.gradle.org/docs/current/userguide/multi\_project\_builds.html#parallel
18 # org.gradle.parallel=true
19 android.useAndroidX=true
20 android.enableJetifier=true
21

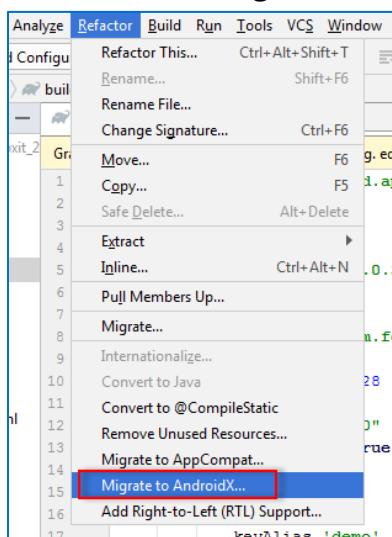
```

2) 在 app's "build.gradle" 中修改相关的依赖。

a) 将 **compileSdkVersion** 设置为 28 或以上：



- b) 将工程级 "build.gradle" 中的 '**com.android.tools.build:gradle**' 升级到 3.2.0 或更高版本。
- c) 点击 **Refactor > Migrate to AndroidX** 开始迁移。



- d) 升级第三方库。如果工程中的第三方库引用了 Android support 相关的库，那么第三方库也需要迁移到 AndroidX。

升级前，app's "build.gradle" 中的 dependencies 如下所示：

```

dependencies {
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support:design:27.1.1'
    implementation 'com.android.support:multidex:1.0.+'
    implementation(name: 'FoxitRDK', ext: 'aar')
    implementation(name: 'FoxitRDKUIExtensions', ext: 'aar')
    implementation 'com.edmodo:cropper:1.0.1'
    implementation('com.microsoft.aad:adal:1.1.16') {}
    implementation(name: 'RMSSDK-4.2-release', ext: 'aar')
    implementation(name: 'rms-sdk-ui', ext: 'aar')
    // RxJava
    implementation "io.reactivex.rxjava2:rxjava:2.2.2"
    implementation 'io.reactivex.rxjava2:rxandroid:2.1.0'
}

```

```
implementation 'org.bouncycastle:bcpkix-jdk15on:1.60'  
implementation 'org.bouncycastle:bcprov-jdk15on:1.60'  
}
```

升级后，app's "build.gradle" 中的 dependencies 如下所示：

```
dependencies {  
    implementation 'androidx.appcompat:appcompat:1.1.0'  
    implementation 'com.google.android.material:material:1.1.0'  
    implementation 'androidx.multidex:multidex:2.0.1'  
    implementation(name: 'FoxitRDK', ext: 'aar')  
    implementation(name: 'FoxitRDKUIExtensions', ext: 'aar')  
    implementation 'com.edmodo:cropper:1.0.1'  
    implementation('com.microsoft.aad:adal:1.16.3') {}  
    implementation(name: 'RMSSDK-4.2-release', ext: 'aar')  
    implementation(name: 'rms-sdk-ui', ext: 'aar')  
    // RxJava  
    implementation "io.reactivex.rxjava2:rxjava:2.2.16"  
    implementation 'io.reactivex.rxjava2:rxandroid:2.1.1'  
    implementation 'org.bouncycastle:bcpkix-jdk15on:1.64'  
    implementation 'org.bouncycastle:bcprov-jdk15on:1.64'  
}
```

迁移后，您可以需要手动修改某些包名。因为，在自动迁移的过程中，有些包可能无法正确导入。另外，您可能遇到一些其他的问题，在这种情况下，您需要根据具体的情况来解决相关的问题。

10.16 支持 Chromebook

Foxit PDF SDK for Android 是否支持 Chromebook?

是的。Foxit PDF SDK for Android 支持 Chromebook。但是您需要在 `AndroidManifest.xml` 文件中加入 "`android:name="com.foxit.uiextensions.FoxitApplication"`"。另外，如果您需要自定义 application，则需要继承 `FoxitApplication` 类。

11 技术支持

问题报告

Foxit 为其产品提供全天候 24 小时支持，并拥有 PDF 行业优秀的技术支持工程师开发团队。如果您在使用 Foxit PDF SDK for Android 时遇到任何技术问题或 bug，请在 <http://tickets.foxitsoftware.com/create.php> 网页上将问题报告提交给 Foxit 技术支持团队。为了更好地帮助您解决问题，请提供以下信息：

- 联系方式
- Foxit PDF SDK 产品和版本
- 您使用的操作系统和 IDE 版本
- 问题的详细说明
- 任何其他相关信息，例如日志文件或错误信息截图

联系方式

您可以直接联系 Foxit，请使用以下的联系方式：

线上支持：

- <http://www.foxitsoftware.com/support/>

联系销售：

- 电话: 1-866-680-3668
- 邮箱: sales@foxitsoftware.com

联系技术支持团队：

- 电话: 1-866-MYFOXIT or 1-866-693-6948
- 邮箱: support@foxitsoftware.com