TABLE OF CONTENTS

1 Introduction to Foxit Mobile PDF SDK

Have you ever worried about the complexity of the PDF specification? Or have you ever felt lost when asked to build a full-featured PDF app within a limited time-frame? If your answer is “Yes”, then congratulations! You have just found the best solution in the industry for rapidly integrating PDF functionality into your apps.

1.1 Why Foxit Mobile PDF SDK is your choice

Foxit is an Amazon-invested leading software provider of solutions for reading, editing, creating, organizing, and securing PDF documents. Foxit PDF SDK libraries have been used in many of today’s leading apps, and they are proven, robust, and battle-tested to provide the quality, performance, and features that the industry’s largest apps demand. Foxit Mobile PDF SDK is a new SDK product which is developed for providing quick PDF viewing and manipulation support for mobile platforms. Customers choose it for the following reasons:

- **Easy to integrate**

Developers can seamlessly integrate Foxit Mobile PDF SDK into their own apps with just a few lines of code.

- **Perfectly designed**

Foxit Mobile PDF SDK is designed with a simple, clean, and friendly style, which provides the best user experience.

- **Flexible customization**

Foxit Mobile PDF SDK provides the source code for the user interface which lets the developers have full control of the functionality and appearance of their apps.

- **Robust performance on mobile platforms**

Foxit Mobile PDF SDK provides an OOM (out-of-memory) recovery mechanism to ensure the app has high robust performance when running the app on a mobile device which offers limited memory.

- **Powered by Foxit’s high fidelity rendering PDF engine**

The core technology of Foxit Mobile PDF SDK is based on Foxit’s PDF engine, which is trusted by a large number of the world’s largest and well-known companies. Foxit’s powerful engine makes the app fast on parsing, rendering, and makes document viewing consistent on a variety of devices.

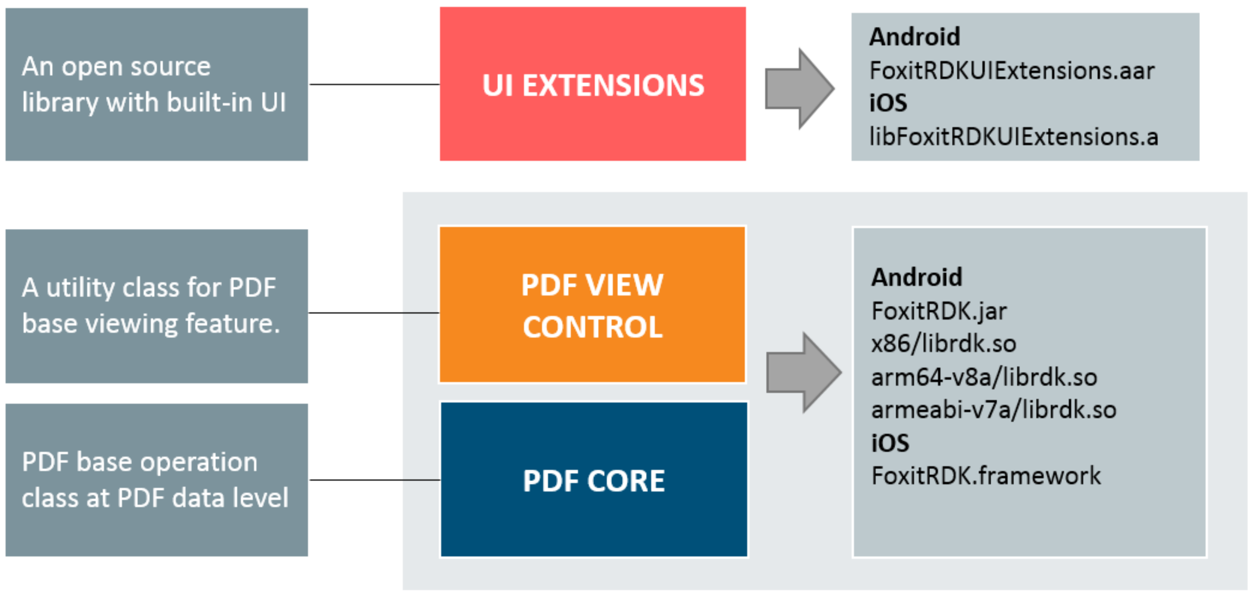
- **Premium World-side Support**

Foxit offers premium support for its developer products because when you are developing mission critical products you need the best support. Foxit has one of the PDF industry's largest team of support engineers. Updates are released on a regular basis to improve user experience by adding new features and enhancements.

1.2 Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF viewer with just a few lines of code. Now, it is available on iOS and Android platforms.

Foxit Mobile PDF SDK includes three Level APIs as the following picture.



Three Level APIs for Foxit Mobile PDF SDK

- **PDF Core API**

  The PDF Core API is the heart of this SDK and is built on Foxit's powerful underlying technology. It provides the functionality for basic PDF operation features, and is utilized by the PDF View Control API and UI Extensions API, which ensures the apps can achieve high performance and efficiency. The Core API can be used by used independently for document rendering, analysis, text extraction, text search, annotation creation and manipulation and much more.

- **PDF View Control API**

The PDF View Control API is a utility class that provides the functionality for developers to interact with rendering PDF documents according to their requirements. With Foxit's renowned and widely used PDF rendering technology at its core, the Viewer Control provides fast and high quality rendering, zooming, scrolling, page view modes and page navigation features. The View Control derives from platform related viewer classes (e.g. UIView on iOS and Android.View.ViewGroup on Android) and allows for extension to accommodate specific user needs.

- **UI Extensions API**

The UI Extensions API is an open source library that provides a customizable user interface with built-in UI implementations for text selection, interactive markup annotation creation and editing, night mode viewing, bookmarks and page thumbnail navigation and full-text searching. The features in the UI extensions library are implemented using the PDF Core API and View Control API. Developers can utilize these ready-to-use UI implementations to build a PDF viewer quickly, and also have complete flexibility and control to customize the UI design as desired.

1.3 Key Features

Foxit Mobile PDF SDK has several main features which help app developers quickly implement the functions that they really need and reduce the development cost.

Features	
PDF Document	Open and close files, set and get metadata
PDF Page	Parse, render, read and set the properties of a page
Render	Graphics engine created on a bitmap for platform graphics device
PDF Text Select	Select text in a PDF document
Bookmark	Directly locate and link to point of interest within a document
Annotation	Create, edit and remove annotations
Out of Memory	Recover from an OOM condition

1.4 Evaluation

Foxit Mobile PDF SDK allows users to download trial version to evaluate SDK. The trial version has no difference from the standard licensed version except for the free 28-day trial limitation and the trial watermarks in the generated pages. After the evaluation period expires, customers should contact the Foxit sales team and purchase licenses to continue using Foxit Mobile PDF SDK.

1.5 License

Developers should purchase licenses to use Foxit Mobile PDF SDK in their solutions. Licenses grant developers permission to release their apps which utilize Foxit Mobile PDF SDK. However, users are prohibited to distribute any documents, sample code, or source code in the released package of Foxit Mobile PDF SDK to any third party without written permission from Foxit Software Incorporated.

1.6 About this Guide

Foxit Mobile PDF SDK is currently available on iOS and Android platforms. This guide is intended for the developers who need to integrate Foxit Mobile PDF SDK for Android into their own apps. It aims at introducing the following sections:

- Section 1: gives an introduction of Foxit Mobile PDF SDK.
- Section 2: illustrates the package structure, running demo, and adding PDF SDK into app.
- Section 3: introduces how to customize the UI implementation.
- Section 4: shows how to create a custom tool.
- Section 5: provides support information.

2 Getting Started

It is very easy to setup Foxit Mobile PDF SDK and see it in action! It takes just a few minutes and we will show you how to use it on the Android platform. The following sections introduce the structure of the installation package, how to run a demo, and how to create your own project in Android Studio.

2.1 System Requirements

Runtime requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a / arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

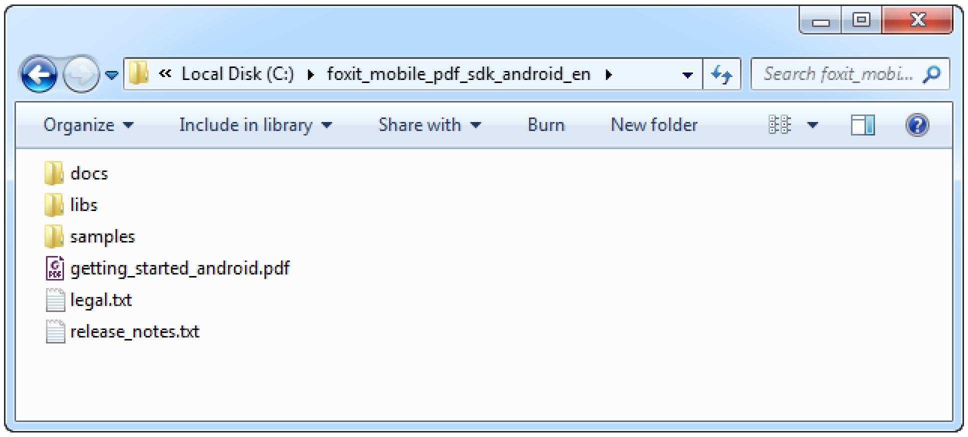
The IDE for the demos:

- Android Studio 2.1.1
- JDK: 1.7.0.79

2.2 What is in the Package

Download the "foxit_mobile_pdf_sdk_android_en.zip" package, and extract it to a new directory like "foxit_mobile_pdf_sdk_android_en" as shown in the Figure 2-1. The package contains:

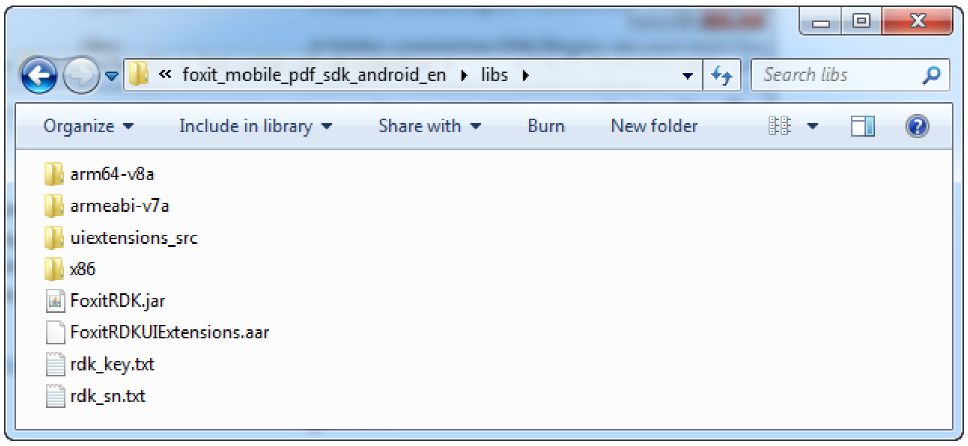
docs:	A folder containing API references, developer guide.
libs:	A folder containing license files, .so libraries, Jar, AAR files, and ui extensions library source code.
samples:	A folder containing Android sample projects.
getting_started_android.pdf:	A quick guide for Foxit Mobile PDF SDK for Android.
legal.txt:	Legal and copyright information.
release_notes.txt:	Release information.



docs
libs
samples
getting_started_android.pdf
legal.txt
release_notes.txt

Figure 2-1

In the "libs" folder as shown in the Figure 2-2, there are items that make up the core components of Foxit Mobile PDF SDK for Android.



arm64-v8a
armeabi-v7a
uiextensions_src
x86
FoxitRDK.jar
FoxitRDKUIExtensions.aar
rdk_key.txt
rdk_sn.txt

Figure 2-2

- ***librdk.so*** (libs/armeabi-v7a, libs/arm64-v8a, libs/x86) - called by the Java APIs in the FoxitRDK.jar. It is the heart of the SDK and contains the core functionalities of Foxit Mobile PDF SDK, and it is built separately for each architecture. Currently it is available for armeabi-v7a, arm64-v8a, and x86.

- ***FoxitRDK.jar***- used by the Java platform. It includes all of the Java APIs of Foxit Mobile PDF SDK.

- ***FoxitRDKUIExtensions.aar*** - generated by the "**uiextensions_src**" project found in the "libs" folder. It includes the FoxitRDK.jar, built-in UI implementation, and resource files that are needed for the built-in UI implementations, such as images, strings, color values, layout files, and other Android UI resources.

- ***uiextensions_src*** project - found in the "libs" folder. It is an open source library that contains some ready-to-use UI module implementations, which can help developers rapidly embed a fully functional PDF reader into their Android app. Of course, developers are not forced to use the default UI, they can freely customize and design the UI for their specific apps through the "uiextensions_src" project.

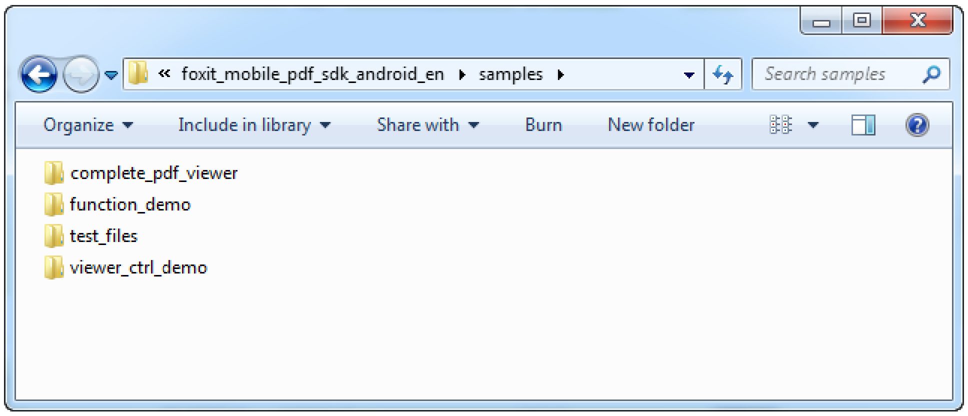
At this point you should just be getting a feel for what Foxit Mobile PDF SDK package looks like, we're going to cover everything in detail in a bit.

2.3 How to run a demo

Download and install Android Studio IDE (https://developer.android.com/studio/index.html).

Note: *In this guide, we do not cover the installation of Android Studio, Android SDK, and JDK. You can refer to Android Studio's developer site if you haven't installed it already.*

Foxit Mobile PDF SDK provides three useful demos for developers to learn how to call the SDK as shown in the Figure 2-3.



complete_pdf_viewer
function_demo
test_files
viewer_ctrl_demo

Figure 2-3

2.3.1 Function demo

The function demo is provided to show how to use Foxit Mobile PDF SDK to realize some specific features related to PDF with PDF core API. This demo includes the following features:

- **pdf2txt**: extract text from a PDF document to a TXT file.
- **bookmark**: edit bookmark appearances and titles.
- **annotation**: add annotations to a PDF page.
- **docinfo**: export document information of a PDF to a TXT file.

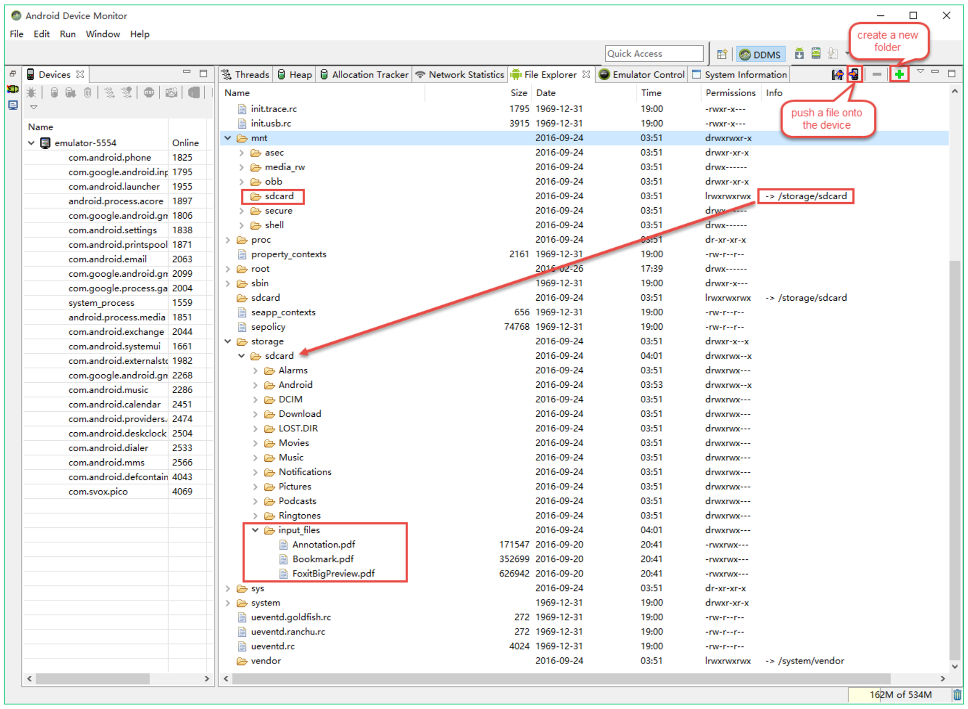
- **render**: render a specified page to Bitmap.

To run it in Android Studio, follow the steps below:

a) Load the demo in Android Studio following "File -> New -> Import Project…" to import the project, or "File -> Open…" to open the project directly, and then choose the directory where the demo was placed.

b) Launch an Android device or an emulator (AVD). Open the **Android Device Monitor**, select the device or emulator that has just been launched and create a new folder named "input_files" under the "mnt/sdcard"(a soft link), then you can find the created folder under the "storage/sdcard". Push the three test files found in the "samples/test_files" folder to this folder. In this guide, an AVD targeting 4.4.2 will be used as an example. Then, the added PDF files will be displayed in "storage/sdcard/input_files" as shown in the Figure 2-4.

   Note *For some Android device, maybe you can only find sdcard0, or sdcard1, such as these. The name of the sdcard is not important, just make sure you have created an "input_files" folder in your device's storage card and pushed the test files to this folder.*



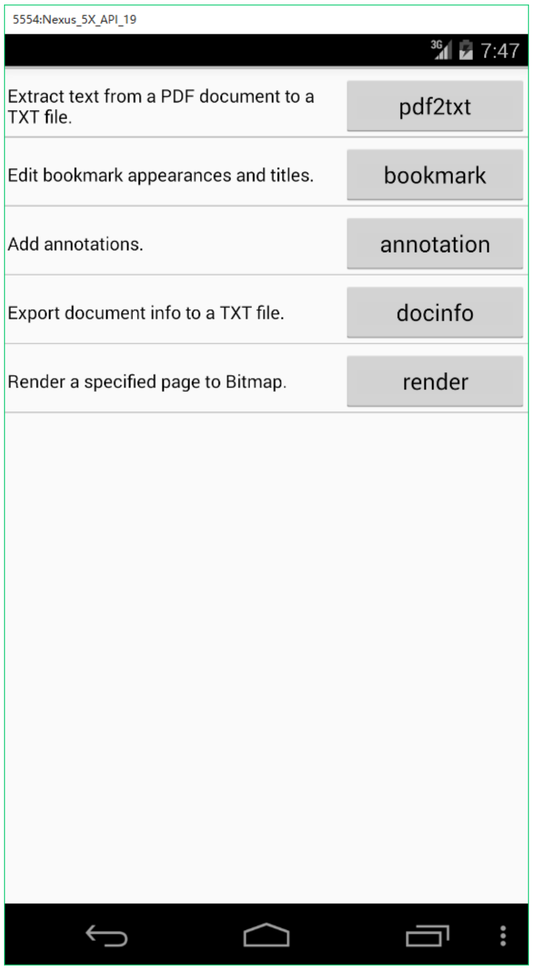
Name	Online
emulator-5554	Online
com.android.phone	1825
com.google.android.inp	1795
com.android.launcher	1955
android.process.acore	1897
com.google.android.gm	1806
com.android.settings	1838
com.android.printspool	1871
com.android.email	2063
com.google.android.gm	2099
com.google.process.ga	2004
system_process	1559
android.process.media	1851
com.android.exchange	2044
com.android.systemui	1661
com.android.externalstc	1982
com.google.android.gm	2268
com.android.music	2286
com.android.calendar	2451
com.android.providers.	2474
com.android.deskclock	2504
com.android.dialer	2533
com.android.mms	2566
com.android.defcontain	4043
com.svox.pico	4069

Name	Size	Date	Time	Permissions	Info
init.trace.rc	1795	1969-12-31	19:00	-rwxr-x---	
init.usb.rc	3915	1969-12-31	19:00	-rwxr-x---	
mnt		2016-09-24	03:51	drwxrwxr-x	
asec		2016-09-24	03:51	drwxr-xr-x	
media_rw		2016-09-24	03:51	drwx------	
obb		2016-09-24	03:51	drwxr-xr-x	
sdcard		2016-09-24	03:51	lrwxrwxrwx	-> /storage/sdcard
secure		2016-09-24	03:51	drwx------	
shell		2016-09-24	03:51	drwx------	
proc		2016-09-24	03:51	dr-xr-xr-x	
property_contexts	2161	1969-12-31	19:00	-rw-r--r--	
root		2016-02-26	17:39	drwx------	
sbin		1969-12-31	19:00	drwxr-x---	
sdcard		2016-09-24	03:51	lrwxrwxrwx	-> /storage/sdcard
seapp_contexts	656	1969-12-31	19:00	-rw-r--r--	
sepolicy	74768	1969-12-31	19:00	-rw-r--r--	
storage		2016-09-24	03:51	drwxr-x--x	
sdcard		2016-09-24	04:01	drwxrwx--x	
Alarms		2016-09-24	03:51	drwxrwx---	
Android		2016-09-24	03:53	drwxrwx--x	
DCIM		2016-09-24	03:51	drwxrwx---	
Download		2016-09-24	03:51	drwxrwx---	
LOST.DIR		2016-09-24	03:51	drwxrwx---	
Movies		2016-09-24	03:51	drwxrwx---	
Music		2016-09-24	03:51	drwxrwx---	
Notifications		2016-09-24	03:51	drwxrwx---	
Pictures		2016-09-24	03:51	drwxrwx---	
Podcasts		2016-09-24	03:51	drwxrwx---	
Ringtones		2016-09-24	03:51	drwxrwx---	
input_files		2016-09-24	04:01	drwxrwx---	
Annotation.pdf	171547	2016-09-20	20:41	-rwxrwx---	
Bookmark.pdf	352699	2016-09-20	20:41	-rwxrwx---	
FoxitBigPreview.pdf	626942	2016-09-20	20:41	-rwxrwx---	
sys		2016-09-24	03:51	dr-xr-xr-x	
system		1969-12-31	19:00	drwxr-xr-x	
ueventd.goldfish.rc	272	1969-12-31	19:00	-rw-r--r--	
ueventd.ranchu.rc	272	1969-12-31	19:00	-rw-r--r--	
ueventd.rc	4024	1969-12-31	19:00	-rw-r--r--	
vendor		2016-09-24	03:51	lrwxrwxrwx	-> /system/vendor

Figure 2-4

c) Click on "Run -> Run 'app'" to run the demo. After building the demo successfully, the features are listed like the Figure 2-5.

   Note: *when running the demo, if you encounter an error like "Error running app: Instant Run requires 'Tools|Android|Enable ADB integration' to be enabled", just follow the prompt to enable the ADB integration in "**Tools** -> **Android** -> **Enable ADB integration**".*

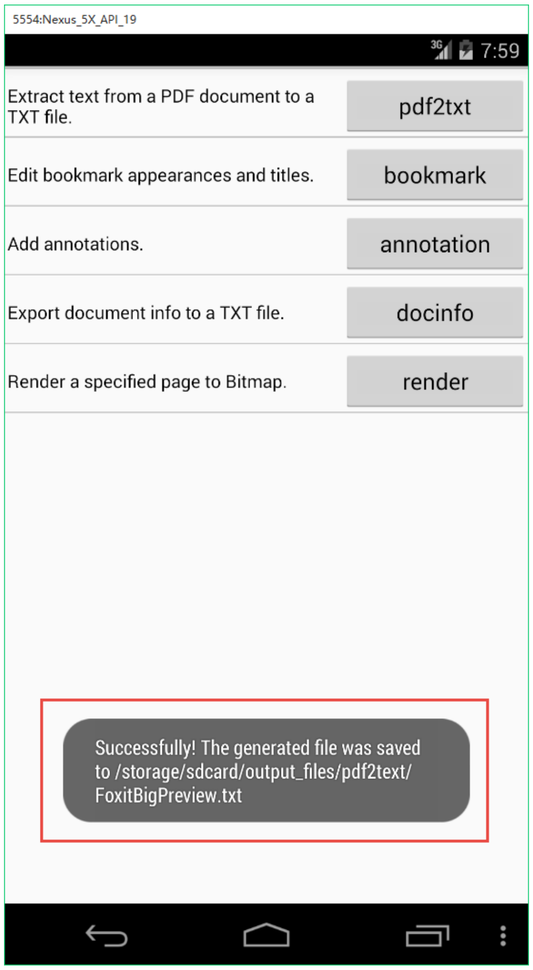

5554:Nexus_5X_API_19

7:47

Extract text from a PDF document to a TXT file.	pdf2txt
Edit bookmark appearances and titles.	bookmark
Add annotations.	annotation
Export document info to a TXT file.	docinfo
Render a specified page to Bitmap.	render

Figure 2-5

d) Click the feature buttons in the above picture to perform the corresponding actions. For example, click "pdf2txt", and then a message box will be popped up as shown in the Figure 2-6. It shows where the text file was saved to. Just run the demo and try the features.


5554:Nexus_5X_API_19
7:59
Extract text from a PDF document to a TXT file.
pdf2txt
Edit bookmark appearances and titles.
bookmark
Add annotations.
annotation
Export document info to a TXT file.
docinfo
Render a specified page to Bitmap.
render

Successfully! The generated file was saved to /storage/sdcard/output_files/pdf2text/FoxitBigPreview.txt

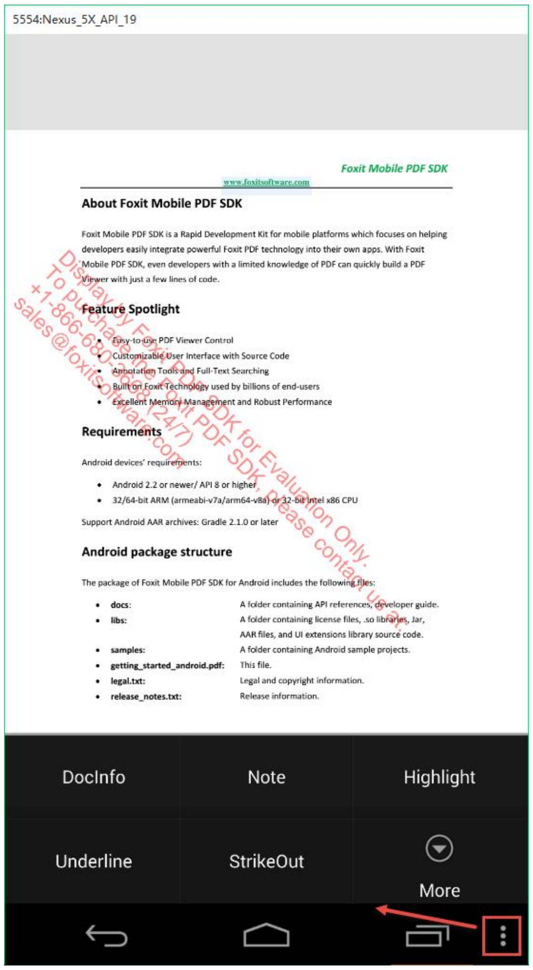
Figure 2-6

2.3.2 Viewer control demo

The viewer control demo demonstrates how to implement the features related to the View Control feature level, such as performing annotations (note, highlight, underline, strikeout, squiggly, etc.), changing layout, text search, outline, and page thumbnail. The logical structure of the code is quite clear and simple so that developers can quickly find the detailed implementation of features which are used widely in PDF apps, such as a PDF viewer. With this demo, developers can take a closer look at the APIs provided in Foxit Mobile PDF SDK.

To run the demo in Android Studio, please refer to the setup steps outlined in the Function demo. Please push the “getting_started_android.pdf” file found in the download package onto the Android device or emulator before running this demo.

Figure 2-7 shows what the demo looks like after it was built successfully. Here, an AVD targeting 4.4.2 will be used as an example to run the demo.



5554:Nexus_5X_API_19

Foxit Mobile PDF SDK

www.foxitsoftware.com

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

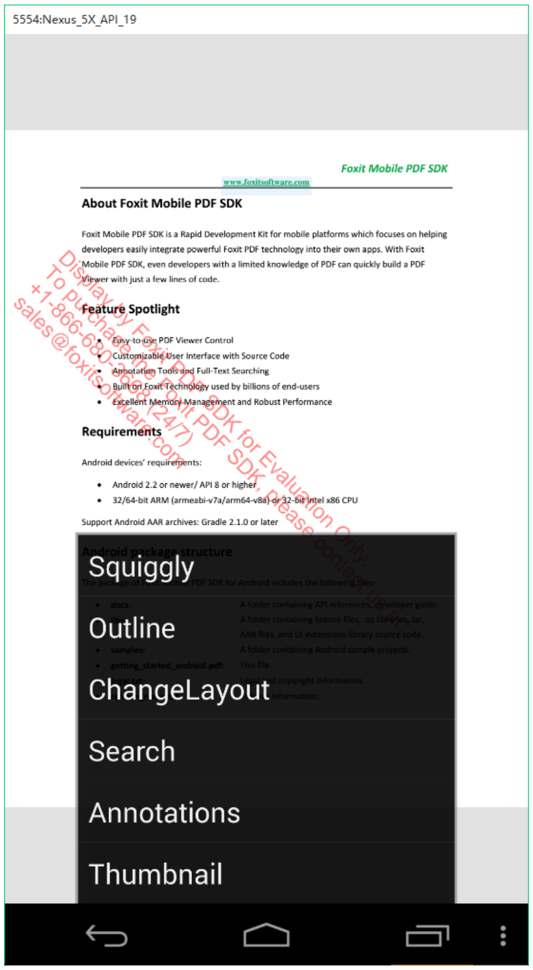
The package of Foxit Mobile PDF SDK for Android includes the following files:

• **docs**:	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, .so libraries, Jar, AAR files, and UI extensions library source code.
• **samples:**	A folder containing Android sample projects.
• **getting_started_android.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Displayed by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 2-7

Click **more** in the above picture to see more features as shown in the Figure 2-8.



5554:Nexus_5X_API_19

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

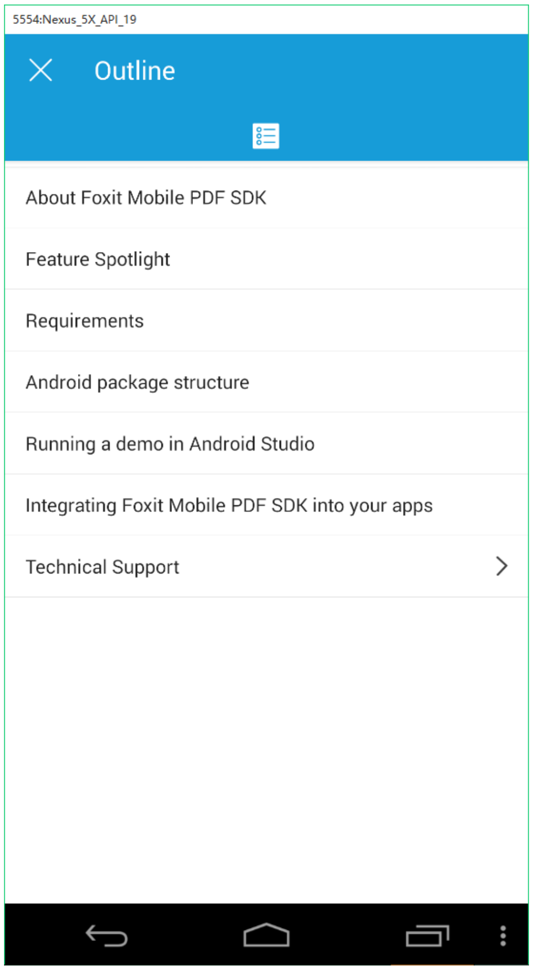
Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

Display by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 2-8

Now we can choose one button to perform the action and see the result. For example, click "Outline", then you will see the outline (outline is the technical terms for bookmark in the PDF specification) of the document as shown in the Figure 2-9. Try using the other features to see it in action.


5554:Nexus_5X_API_19

Outline

About Foxit Mobile PDF SDK

Feature Spotlight

Requirements

Android package structure

Running a demo in Android Studio

Integrating Foxit Mobile PDF SDK into your apps

Technical Support

Figure 2-9

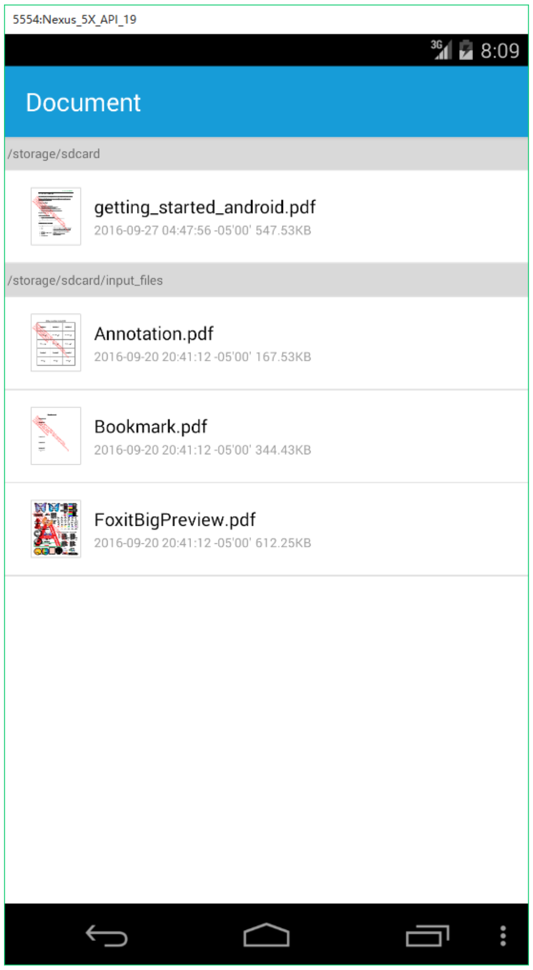
2.3.3 Complete PDF viewer demo

The complete PDF viewer demo demonstrates how to use Foxit Mobile PDF SDK to realize a completely full-featured PDF viewer which is almost ready-to-use as a real world mobile PDF reader. This demo utilizes all of the features and built-in UI implementations which are provided in Foxit Mobile PDF SDK.

To run the demo in Android Studio, please refer to the setup steps outlined in the Function demo.

Here, an AVD targeting 4.4.2 will also be used as an example to run the demo. After building the demo successfully, on the start screen, it will list all the PDF files stored in the device's SD card as shown in the Figure 2-10.

Note *If you want to use other PDF files to test this demo, you need to push them onto the* device's SD card.


5554:Nexus_5X_API_19

Document

/storage/sdcard

getting_started_android.pdf
2016-09-27 04:47:56 -05'00' 547.53KB

/storage/sdcard/input_files

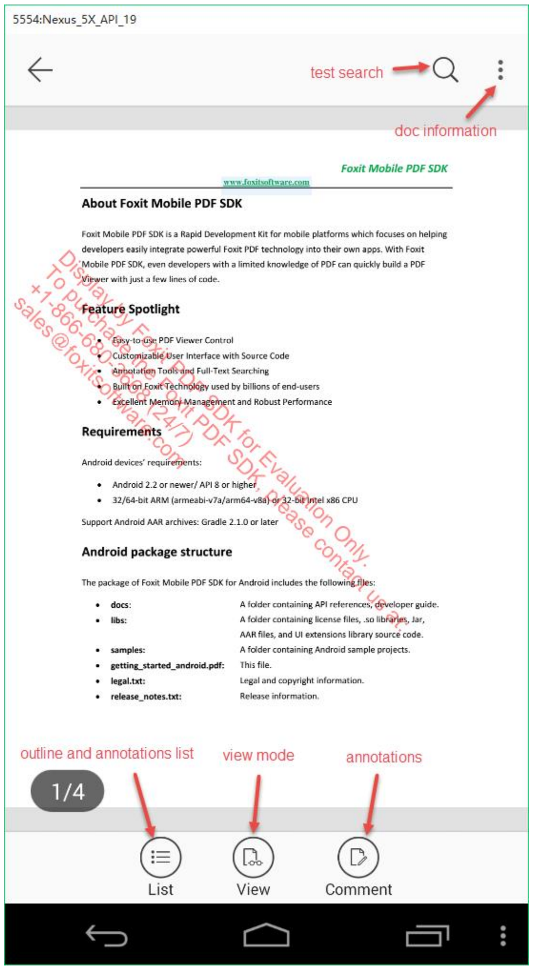
Annotation.pdf
2016-09-20 20:41:12 -05'00' 167.53KB

Bookmark.pdf
2016-09-20 20:41:12 -05'00' 344.43KB

FoxitBigPreview.pdf
2016-09-20 20:41:12 -05'00' 612.25KB

Figure 2-10

Here, we choose the "getting_started_android.pdf" document, click it, and then it will be opened and displayed as shown in the Figure 2-11.



Foxit Mobile PDF SDK

www.foxitsoftware.com

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

The package of Foxit Mobile PDF SDK for Android includes the following files:

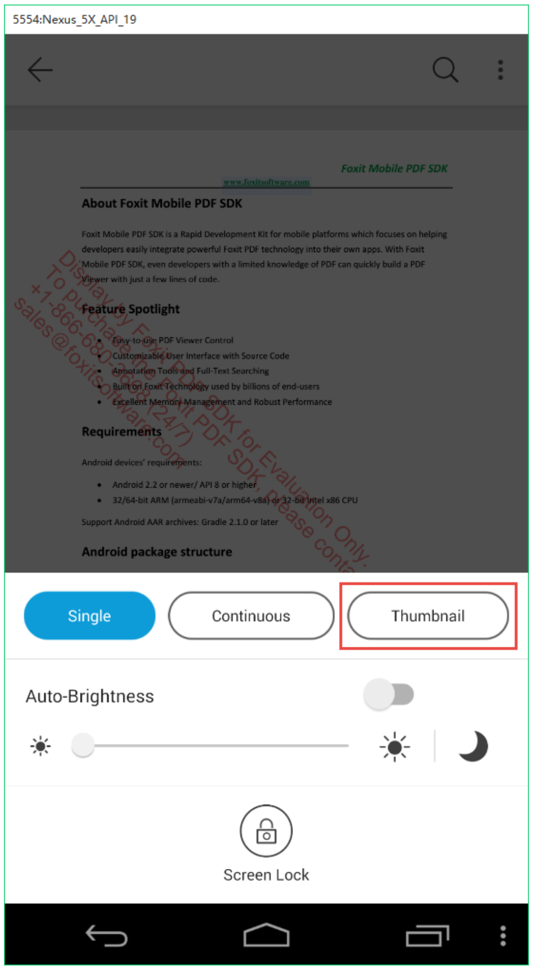
• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, .so libraries, Jar, AAR files, and UI extensions library source code.
• **samples:**	A folder containing Android sample projects.
• **getting_started_android.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Displayed by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 2-11

This demo realizes a completely full-featured PDF viewer, please feel free to run it and try it.

For example, it provides the page thumbnail feature. You can click the **View** menu, choose the **Thumbnail** as shown in the Figure 2-12, and then the thumbnail of the document will be displayed as shown in the Figure 2-13.



Foxit Mobile PDF SDK

www.foxitsoftware.com

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

Display by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please conta
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 2-12

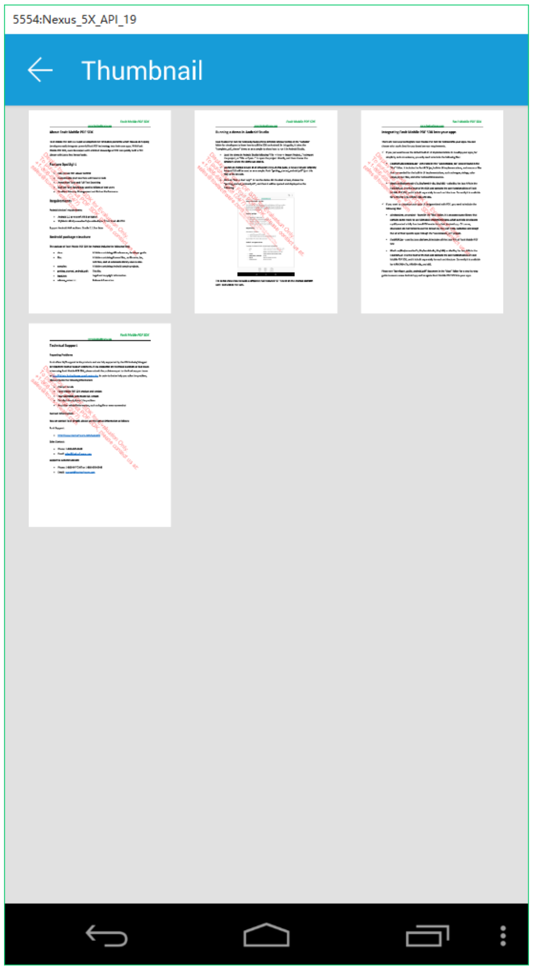


Figure 2-13

2.4 How to make an Android app with Foxit Mobile PDF SDK

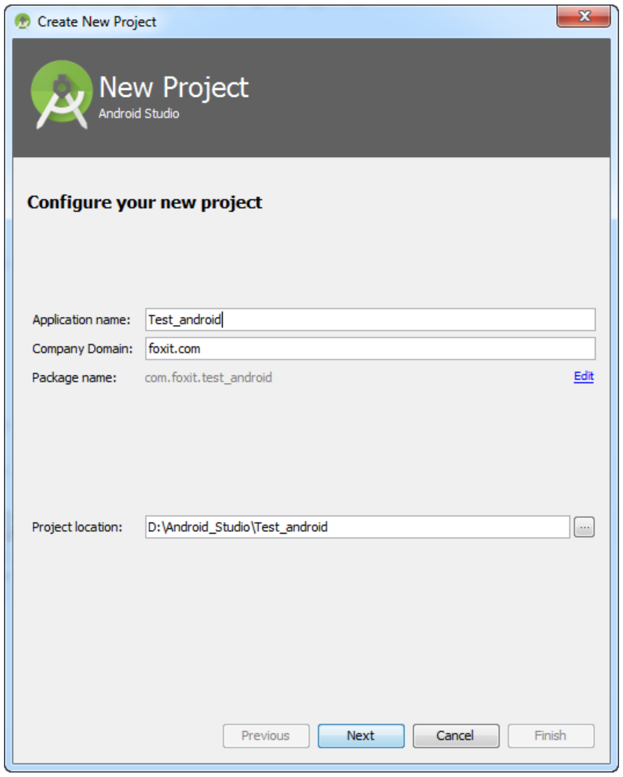
This section will help you to quickly get started with using Foxit Mobile PDF SDK to make an Android app with step-by-step instructions provided. From now, you can get familiar with Foxit Mobile PDF SDK and create your first PDF Android app in Android Studio. This section includes the following steps:

1) Create a new Android project
2) Integrate Foxit Mobile PDF SDK into your apps
3) Apply the license key
4) Display a PDF document
5) Add support for Text Search, Bookmarks, and Annotations

2.4.1 Create a new Android project

In this guide, we use Android Studio 2.1.1, along with Android API revision 23.

Open Android Studio, choose **File** -> **New** -> **New Project**... to start the **Create New Project** wizard, and then fill the **New Project** dialog as shown in the Figure 2-14. After filling, click **Next**.

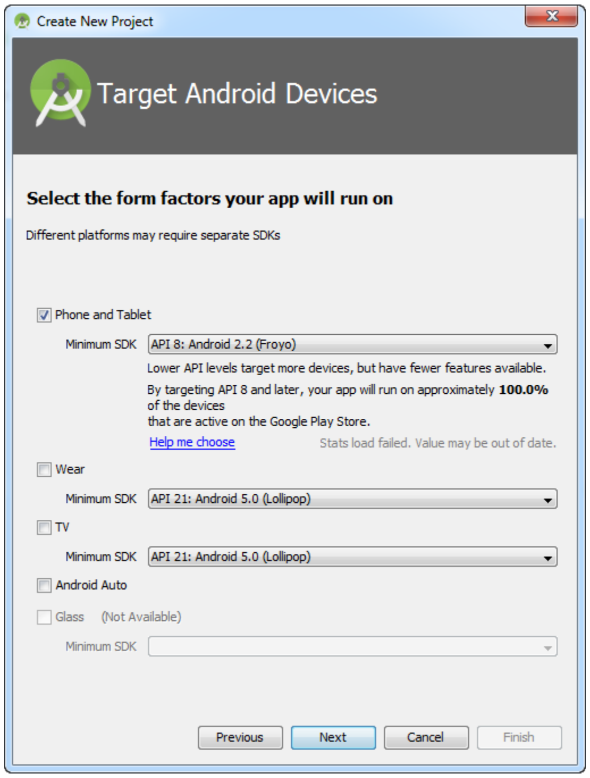


Configure your new project

Application name: Test_android
Company Domain: foxit.com
Package name: com.foxit.test_android
Project location: D:\Android_Studio\Test_android

Figure 2-14

In the **Target Android Devices** dialog, select "Phone and Tablet" to run your app, and set the Minimum SDK to API 8, which is shown in the Figure 2-15. Then, click **Next**.



Select the form factors your app will run on

Different platforms may require separate SDKs

☑ Phone and Tablet

Minimum SDK: API 8: Android 2.2 (Froyo)

Lower API levels target more devices, but have fewer features available.

By targeting API 8 and later, your app will run on approximately **100.0%** of the devices that are active on the Google Play Store.

Help me choose — Stats load failed. Value may be out of date.

☐ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Glass (Not Available)

Minimum SDK

Figure 2-15

In the **Add an activity to Mobile** dialog, select “Empty Activity” (for some other Android Studio versions, it might be “Blank Activity”) as shown in the Figure 2-16, and then click **Next**.

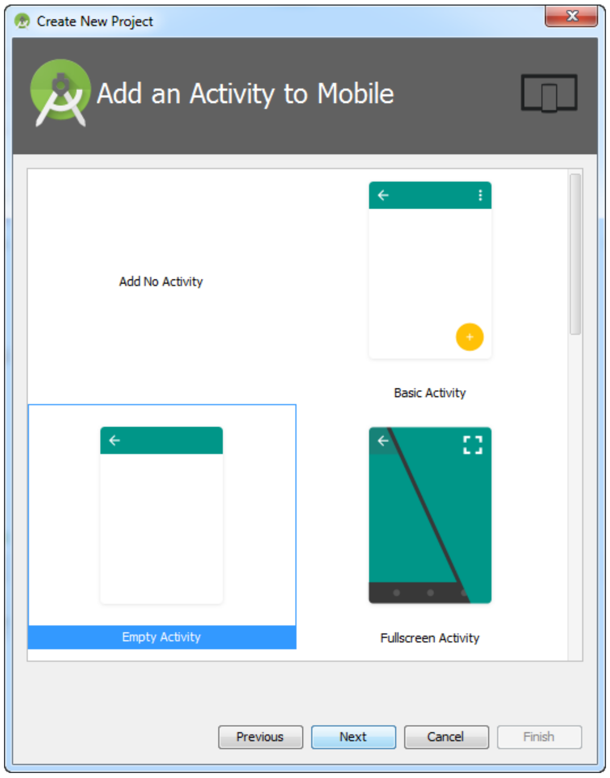
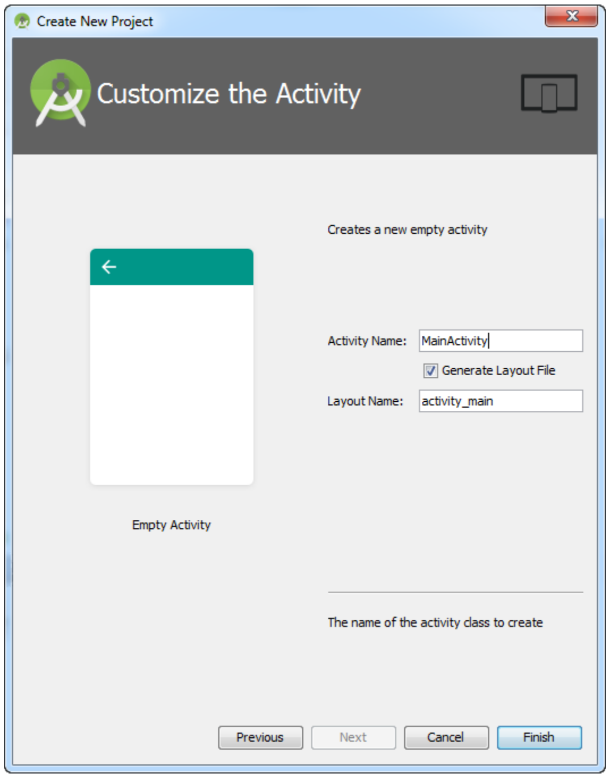


Figure 2-16

In the **Customize the Activity** dialog, customize your activity as desired, here, we use the default setting as shown in the Figure 2-17, and then click **Finish**.


Create New Project

Customize the Activity

Creates a new empty activity

Activity Name: MainActivity

[✓] Generate Layout File

Layout Name: activity_main

Empty Activity

The name of the activity class to create

Figure 2-17

Note: *If your Android Studio has installed a high Android SDK version, such as API 24, after building the project, it may encounter a problem as follows:*

```
Error:Execution failed for task ':app:processDebugManifest'.

> Manifest merger failed: uses-sdk:minSdkVersion 8 cannot be smaller than version 9 declared in library
[com.android.support:appcompat-v7:24.2.0] D:\Android_Studio\Test_android\app\build\intermediates\exploded-
aar\com.android.support\appcompat-v7\24.2.0\AndroidManifest.xml

Suggestion: use tools:overrideLibrary="android.support.v7.appcompat" to force usage
```

To solve this problem, you should modify the settings in the app's "build.gradle" file (if you are using the "Android" view panel then you might see two "build.gradle" files, use the "Module:app" one):

- *If you want to support the "minSdkVersion 8", then please change the "CompileSdkVersion" and "targetSdkVersion" entries to a lower SDK version, such as 23, and change the "buildToolsVersion" entry to a lower version such as 23.0.2, and then change the "com.android.support:appcompat-v7:24.2.0" entry to "com.android.support:appcompat-v7:23.4.0".*

- *If you want to use "com.android.support:appcompat-v7:24.2.0" library, please change the "minSdkVersion" entry to version higher than 8.*

2.4.2 Integrate Foxit Mobile PDF SDK into your apps

***Note:** In this section, we will use the default built-in UI implementation to develop the app, for simplicity and convenience, we only need to add the following files to the Test_android project.*

- ***FoxitRDKUIExtensions.aar*** - generated by the "**uiextensions_src**" project found in the "libs" folder. It includes the FoxitRDK.jar, built-in UI implementations, and resource files that are needed for the built-in UI implementations, such as images, strings, color values, layout files, and other Android UI resources.

- ***librdk.so*** (libs/armeabi-v7a, libs/arm64-v8a, libs/x86) - called by Java APIs in the FoxitRDK.jar. It is the heart of the SDK containing the core functionalities of Foxit Mobile PDF SDK, and built separately for each architecture. Currently it is available for armeabi-v7a, arm64-v8a, and x86.

  ***Note:** For this project, we will run the project on an emulator with x86 architecture, so we just add the "**x86/librdk.so**" library into the project. If you are not clear about the architecture of the device you want to use, you can add all of the ".so" libraries, because the linker can distinguish the device's architecture and then choose the proper library.*

To add the above two files into *Test_android* project, please switch to the "Project" view panel and then follow the steps below:

a) Copy and paste the "***FoxitRDKUIExtensions.aar***" file from the "libs" folder of the download package to "Test_android\app\libs" folder.

b) Copy and paste the "***librdk.so***" file from the "libs\x86" folder of the download package to "Test_android\app\libs\x86" (*note that you might need to create this folder by yourself*).

   Then, the project should look like the Figure 2-18.

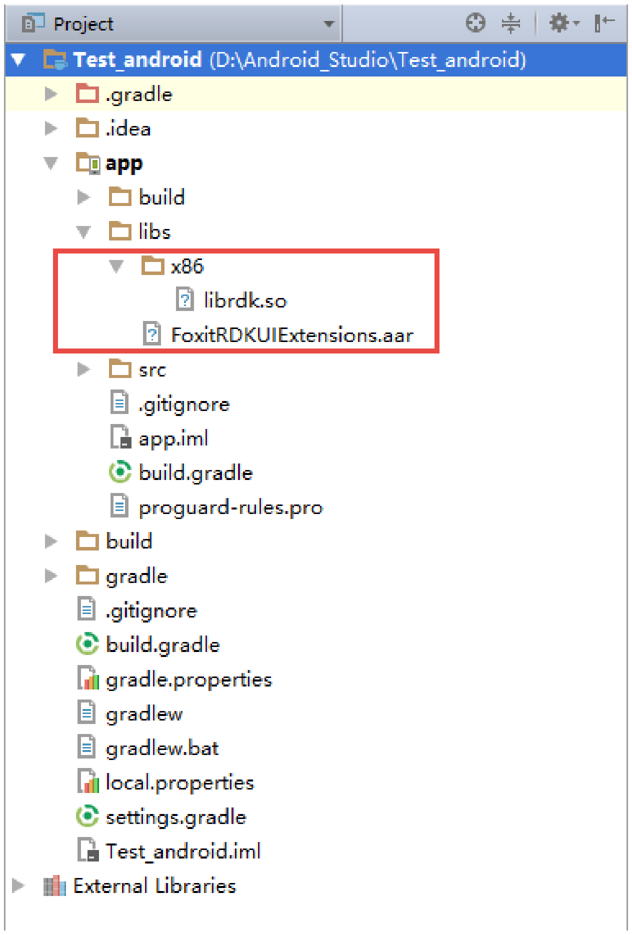


Figure 2-18

c) Add a reference to the "***librdk.so***". Inside the app's "build.gradle" file, add the following configuration:

build.gradle:

```
android {

    sourceSets {
        main {
            jniLibs.srcDirs = ['libs']
        }
    }
}
```

d) Include Foxit Mobile PDF SDK as a dependency in the project by setting up the App module in the "build.gradle" file as follows:

First, define the "libs" directory as a repository in the app's "build.gradle" repositories section.

build.gradle:

```
repositories {

    flatDir {
        dirs 'libs'
    }
}
```

And then, add "***FoxitRDKUIExtensions.aar***" to the dependencies.

```
dependencies {

    compile (name:'FoxitRDKUIExtensions', ext:'aar')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.4.0'
}
```

After setting, rebuild the project following "**Build** -> **Rebuild project**", you can see the detailed information inside the "*FoxitRDKUIExtensions.aar*" in the "\app\build\intermediates\exploded-aar\FoxitRDKUIExtensions" folder as shown in the Figure 2-19.

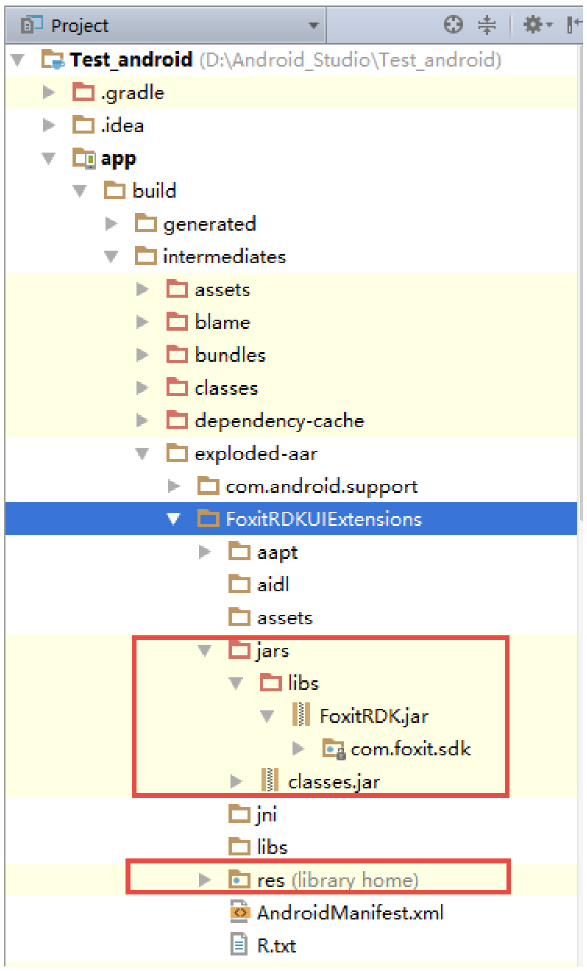


Figure 2-19

The following code shows "build.gradle" in its entirety.

build.gradle:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        applicationId "com.foxit.test_android"
        minSdkVersion 8
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    sourceSets {
        main {
            jniLibs.srcDirs = ['libs']
        }
    }
}

repositories {

    flatDir {
        dirs 'libs'
    }
}

dependencies {

    compile (name:'FoxitRDKUIExtensions', ext:'aar')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.4.0'
}
```

Note *So far, we set the compileSdkVersion and targetSdkVersion to API 23. If you also want to use API 23, please make sure you have already installed the SDK Platform Android 6.0, API 23. If you have not already done this, open the Android SDK Manager to download and install it first. (The targetSdkVersion setting will be changed based on the project's requirements in the following sections.)*

2.4.3 Apply the license key

It is necessary for apps to initialize and unlock Foxit Mobile PDF SDK using a license before calling any APIs. The function ***Library.init*** *(sn, key)* is provided to initialize Foxit Mobile PDF SDK. The trial license files can be found in the "libs" folder of the download package. After the evaluation period expires, you should purchase an official license to continue using it. Below you can see an example of how to unock the SDK library. The next section will show where to include this code in the *Test_android* project.

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

System.loadLibrary("rdk");
try {
    Library.init("sn", "key");

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

***Note** The parameter "sn" can be found in the "**rdk_sn.txt**" (the string after "SN=") and the "key" can be found in the "**rdk_key.txt**" (the string after "Sign=").*

2.4.4 Display a PDF document

So far, we have just created an Android app with a blank activity. Now, let's start building a simple PDF viewer with just a few lines of code.

***Note:** The UI extensions library is not required if you only want to display a PDF document.*

To display a PDF document, you should add the PDF View Control to show the PDF in the "activity_main.xml" found in the "app/src/main/res/layout".

Update activity_main.xml as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.foxit.test_android.MainActivity">

    <com.foxit.sdk.PDFViewCtrl
        android:id="@+id/pdfviewer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical|horizontal"/>
</RelativeLayout>
```

In the "MainActivity.java", instantiate a PDFDoc object to load an existing PDF document (**"/mnt/sdcard/getting_started_android.pdf"**); and instantiate a PDFViewCtrl object to show the document.

***Note:** By default, the MainActivity extends AppCompatActivity. We change it to extend FragmentActivity, because the UI extensions library which will be utilized in the next section has used the features in the*

FragmentActivity. In addition, please make sure you have pushed the "getting_started_android.pdf" document onto the Android device or emulator that will be used to run this project.

Update MainActivity.java as follows:

```
package com.foxit.test_android;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

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

import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.pdf.PDFDoc;

public class MainActivity extends FragmentActivity {

    private PDFViewCtrl pdfViewCtrl = null;

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

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

        // Load "librdk.so" library.
        System.loadLibrary("rdk");
        try {
            // initialize the library
            Library.init(sn, key);
        } catch (PDFException e) {
            e.printStackTrace();
            return;
        }

        // Inflate the view and get a reference to PDFViewCtrl
        setContentView(R.layout.activity_main);
        pdfViewCtrl = (PDFViewCtrl) findViewById(R.id.pdfviewer);

        // Load a document
        String path = "/mnt/sdcard/getting_started_android.pdf";
        try {
            PDFDoc document = PDFDoc.createFromFilePath(path);
            document.load(null);
            pdfViewCtrl.setDoc(document);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        try {
            Library.release();
        } catch (PDFException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void onResume() {
    super.onResume();
    if(pdfViewCtrl != null)
        pdfViewCtrl.requestLayout();
    }
}
```

Set the "users-permission" in the "AndroidManifest.xml" found in the "app/src/main" to give the project permission to write and read the SD card of the Android devices or emulators.

Update the AndroidManifest.xml as follows:

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

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <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>
```

Note: *If you want to run this project on an Android 6.0 (API 23) or higher devices/emulators, you need to write some additional code to require the authorization of runtime permissions, or you can change the targetSdkVersion in app's "build.gradle" from 23 to the SDK version that is just less than 23, such as 10 (this version will be used in the next section).*

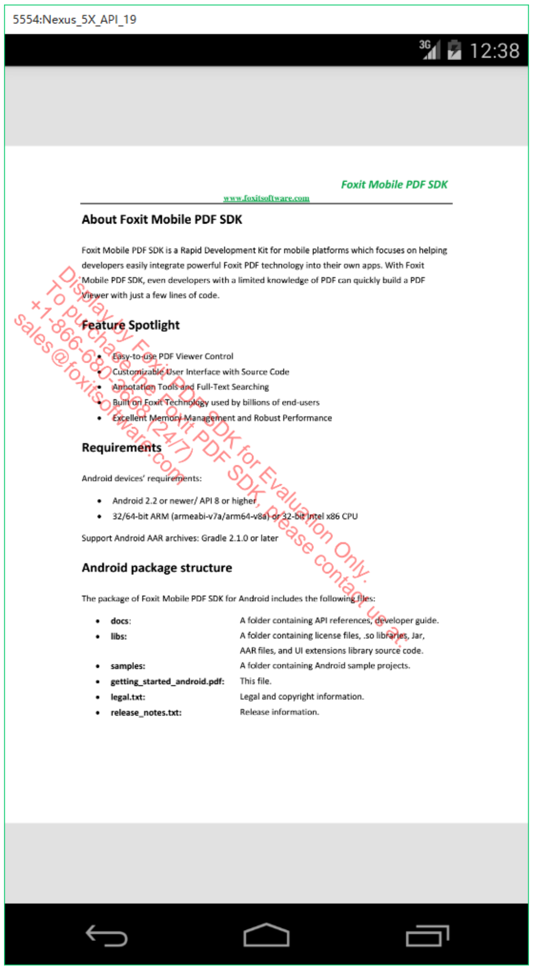
The following code snippets can be used as a reference for you to require the authorization of runtime permissions:

```
    // Check whether the device or simulator has the permission to access the external storage.
    if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.READ_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {
```

```
        // If has no permission, then request the permission.
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
                REQUEST_CODE);
    } else {
        viewDoc();
    }
}

// Define a request code.
private static final int REQUEST_CODE = 1;

// Handle the request after user responds to the request in the dialog.
@Override
public void onRequestPermissionsResult(int requestCode,
                                       String permissions[], int[] grantResults) {
    switch (requestCode) {
        case REQUEST_CODE: {
            // 1.If the request is cancelled, the result array is empty.

            // 2.The permission was granted.
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                //  Do the task you need to do.

            } else {

                // permission was denied. Disable the functionality that depends on this
permission.
            }
            return;
        }
    }
}
```

Fantastic! We have now finished building a simple Android app which uses Foxit Mobile PDF SDK to display a PDF document with just a few lines of code. The next step is to run the project on a device or emulator.

In this guide, we build and run the project on an AVD targeting Android 4.4.2 (API 19), and you will see that the “getting_started_android.pdf” document is displayed as shown in the Figure 2-20. Now, this sample app has some basic PDF features, such as zooming in/out and page turning. Just have a try!


5554:Nexus_5X_API_19

12:38

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

The package of Foxit Mobile PDF SDK for Android includes the following files:

• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, .so libraries, Jar, AAR files, and UI extensions library source code.
• **samples:**	A folder containing Android sample projects.
• **getting_started_android.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Displayed by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 2-20

2.4.5 Add support for Text Search, Bookmarks, and Annotations

Foxit Mobile PDF SDK comes with built-in support for features such as annotations, text search and bookmarks which require a UI implementation. These visual features are implemented using Foxit Mobile PDF SDK API and are shipped in UI extensions library.

Note *The top toolbar and bottom toolbar are customized in the UI extensions library, so the system theme needs to be set to "No Title" mode when using Foxit Mobile PDF SDK. You can refer to the "uiextensions_src" project to customize the toolbar as you wish.*

In the previous sections, we have introduced how to add Foxit Mobile PDF SDK to a project and how to build a simple Android app for displaying a PDF document. Now, let's extend the simple app (*Test_android*) further to learn how to add support for text search, bookmarks, annotations and more.

Let's take adding support for text search as an example for how to add the feature with Foxit Mobile PDF SDK.

a) Create a folder named “menu” located under the folder “app/src/main/res”, and then create a Layout XML file named “main.xml” in that folder. This file is used to add menu item to the menu. Here, we just add a **search** menu item as follows:

main.xml:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
        android:id="@+id/search"
        android:title="search"/>
</menu>
```

Note *In this project, we use the simplest menu button to quickly experience the features in the UI extensions library. Due to Android system’s upgrade, the menu button has been removed in higher versions. We recommend that change the targetSdkVersion to 10 when you try this guide project.*

b) In the “activity_main.xml” file, add an “id” to the layout like below:

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.foxit.test_android.MainActivity"
    android:id="@+id/parentLayout">

    <com.foxit.sdk.PDFViewCtrl
        android:id="@+id/pdfviewer"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical|horizontal"/>
</RelativeLayout>
```

c) In the “MainActivity.java” file we are now going to add the code necessary for including the search functionality. The required code additions are shown below and further down you will find a full example of what the “MainActivity.java” file should look like.

Import the following classes:

```
import android.view.Menu;
import android.view.MenuItem;

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

import android.widget.RelativeLayout;

import com.foxit.uiextensions.UIExtensionsManager;
import com.foxit.uiextensions.modules.SearchModule;
import com.foxit.uiextensions.modules.SearchView;
```

Set the *system theme to “No Title” mode and set the window to Fullscreen.*

```
this.requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

Instantiate a RelativeLayout object to get the parent layout.

```
private RelativeLayout parent = null;
...
parent = (RelativeLayout) findViewById(R.id.parentLayout);
```

Instantiate a UIExtensionsManager object and set it to PDFViewCtrl.

```
private UIExtensionsManager uiextensionsManager = null;
...

uiextensionsManager = new UIExtensionsManager (this, parent, pdfViewCtrl);
pdfViewCtrl.setUIExtensionsManager(uiextensionsManager);
```

Instantiate a SearchModule object to load a search module, and instantiate a SearchView object to start a search. Add two functions, onCreateOptionsMenu and onOptionsItemSelected like below:

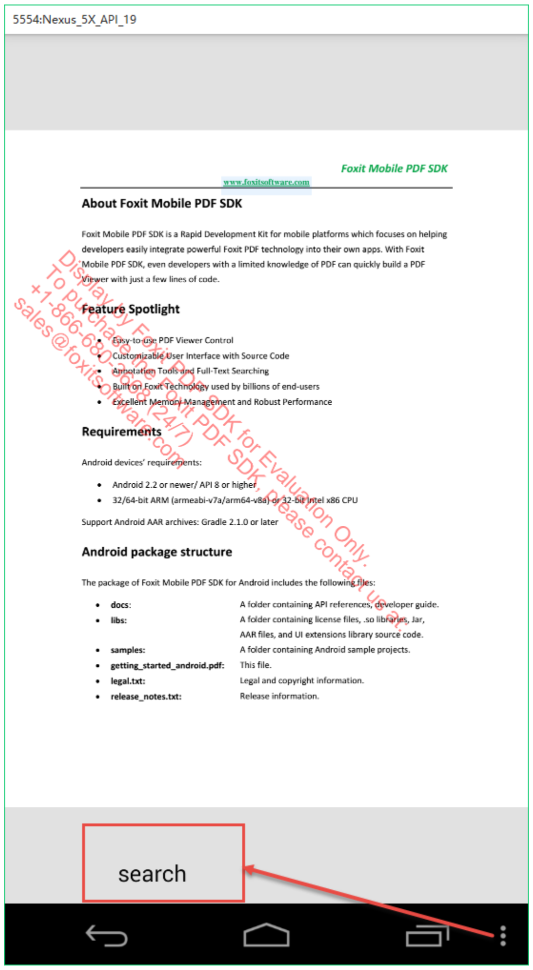
```
private SearchModule searchModule = null;
private SearchView searchView = null;
...

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the menu if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle menu item clicks here.
    if (item.getItemId() == R.id.search) {

        if (searchModule == null) {
            searchModule = new SearchModule(this, parent, pdfViewCtrl);
            searchModule.loadModule();
        }
        searchView = searchModule.getSearchView();
        searchView.show();
    }
    return true;
}
```

The whole update of “MainActivity.java” is as follows:

```
package com.foxit.test_android;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;
import android.view.WindowManager;
import android.widget.RelativeLayout;

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

import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.pdf.PDFDoc;

import com.foxit.uiextensions.UIExtensionsManager;
import com.foxit.uiextensions.modules.SearchModule;
import com.foxit.uiextensions.modules.SearchView;


public class MainActivity extends FragmentActivity {

    private PDFViewCtrl pdfViewCtrl = null;
    private RelativeLayout parent = null;
    private UIExtensionsManager uiextensionsManager = null;
    private SearchModule searchModule = null;
    private SearchView searchView = null;

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

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

        // Load "librdk.so" library.
        System.loadLibrary("rdk");
        try {
            // initialize the library.
            Library.init(sn, key);
        } catch (PDFException e) {
            e.printStackTrace();
            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);

        // Inflate the view and get a reference to PDFViewCtrl.
        setContentView(R.layout.activity_main);
        pdfViewCtrl = (PDFViewCtrl) findViewById(R.id.pdfviewer);

        // Load a document.
        String path = "/mnt/sdcard/getting_started_android.pdf";
        try {
            PDFDoc document = PDFDoc.createFromFilePath(path);
            document.load(null);
            pdfViewCtrl.setDoc(document);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        parent = (RelativeLayout) findViewById(R.id.parentLayout);

        // Initialize a UIExtensionsManager object and set it to PDFViewCtrl.
        uiextensionsManager = new UIExtensionsManager(this, parent, pdfViewCtrl);
        pdfViewCtrl.setUIExtensionsManager(uiextensionsManager);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will automatically handle clicks
on the Home/Up.
        if (item.getItemId() == R.id.search) {
            if (searchModule == null) {
                searchModule = new SearchModule(this, parent, pdfViewCtrl);
                searchModule.loadModule();
            }
            searchView = searchModule.getSearchView();
            searchView.show();
        }
        return true;
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        try {
            Library.release();
        } catch (PDFException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (pdfViewCtrl != null)
            pdfViewCtrl.requestLayout();
    }
}
```

Now that we have finished adding support for text search into the project, let's run it on an AVD targeting Android 4.4.2. You will see that the "getting_started_android.pdf" document is automatically displayed, click the menu to see the "**search**" feature as shown in the Figure 2-21.

***Note:** when running the project, if you encounter an error like "Error running app: Instant Run requires 'Tools|Android|Enable ADB integration' to be enabled", just follow the prompt to enable the ADB integration in "**Tools** -> **Android** -> **Enable ADB integration**".*


5554:Nexus_5X_API_19

Foxit Mobile PDF SDK
www.foxitsoftware.com

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

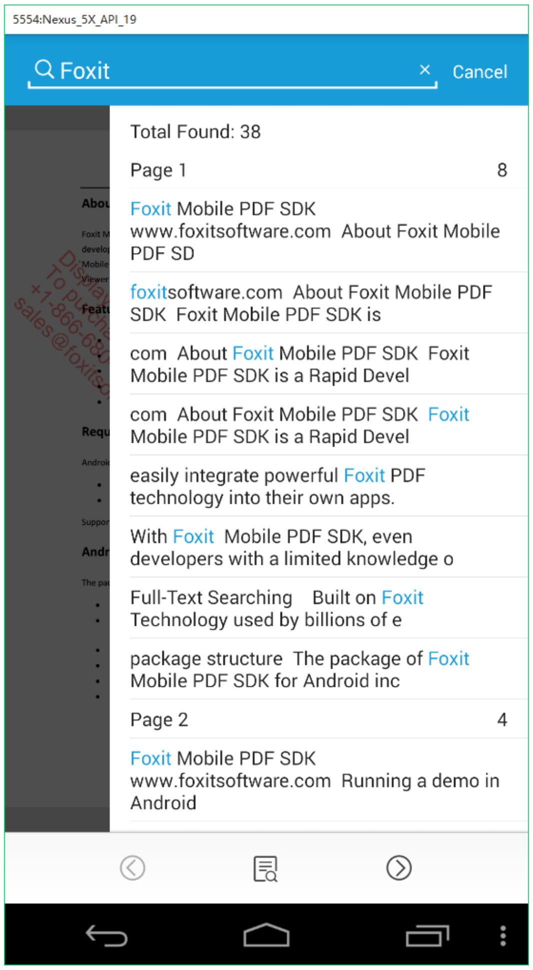
The package of Foxit Mobile PDF SDK for Android includes the following files:

• **docs**:	A folder containing API references, developer guide.
• **libs**:	A folder containing license files, .so libraries, Jar, AAR files, and UI extensions library source code.
• **samples:**	A folder containing Android sample projects.
• **getting_started_android.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Displayed by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 2-21

Click on the "**search**", you can search anything as you like. For example, input "Foxit", press "Enter", and then all of the search results will be listed as shown in the Figure 2-22.



5554:Nexus_5X_API_19

Foxit   Cancel

Total Found: 38

Page 1	8

Foxit Mobile PDF SDK www.foxitsoftware.com About Foxit Mobile PDF SD

foxitsoftware.com About Foxit Mobile PDF SDK Foxit Mobile PDF SDK is

com About Foxit Mobile PDF SDK Foxit Mobile PDF SDK is a Rapid Devel

com About Foxit Mobile PDF SDK Foxit Mobile PDF SDK is a Rapid Devel

easily integrate powerful Foxit PDF technology into their own apps.

With Foxit Mobile PDF SDK, even developers with a limited knowledge o

Full-Text Searching   Built on Foxit Technology used by billions of e

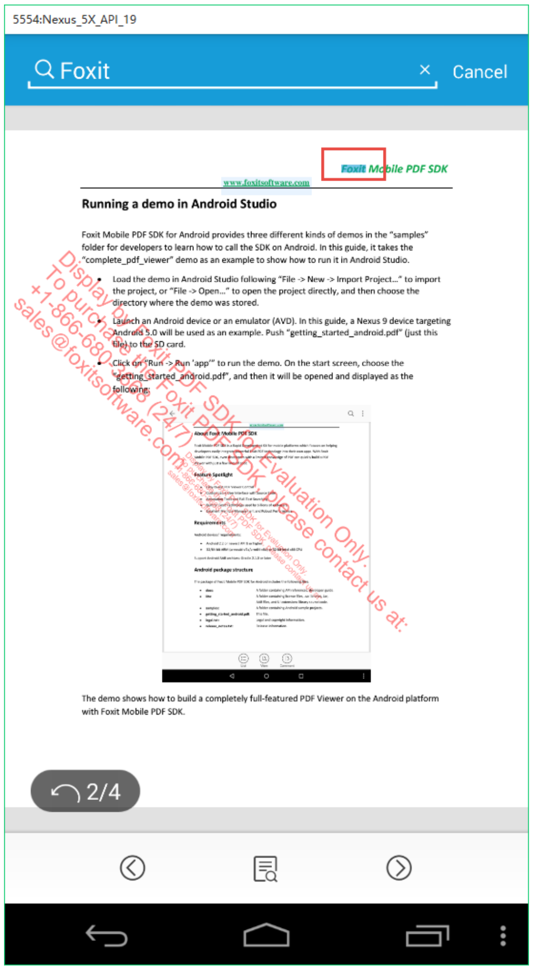
package structure The package of Foxit Mobile PDF SDK for Android inc

Page 2	4

Foxit Mobile PDF SDK www.foxitsoftware.com Running a demo in Android

Figure 2-22

Click any result or any page entry in the list to jump to the specific location. Here, we click the "Page 2" entry in the list, and then it will jump to the corresponding page and the first matched word will be highlighted as shown in the Figure 2-23. You can click the previous or next button to find the previous or next search result.



5554:Nexus_5X_API_19

Foxit Cancel

Foxit Mobile PDF SDK

www.foxitsoftware.com

Running a demo in Android Studio

Foxit Mobile PDF SDK for Android provides three different kinds of demos in the "samples" folder for developers to learn how to call the SDK on Android. In this guide, it takes the "complete_pdf_viewer" demo as an example to show how to run it in Android Studio.

- Load the demo in Android Studio following "File -> New -> Import Project..." to import the project, or "File -> Open..." to open the project directly, and then choose the directory where the demo was stored.
- Launch an Android device or an emulator (AVD). In this guide, a Nexus 9 device targeting Android 5.0 will be used as an example. Push "getting_started_android.pdf" (just this file) to the SD card.
- Click on "Run -> Run 'app'" to run the demo. On the start screen, choose the "getting_started_android.pdf", and then it will be opened and displayed as the following:

Display by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

The demo shows how to build a completely full-featured PDF Viewer on the Android platform with Foxit Mobile PDF SDK.

Figure 2-23

You can refer to the above code or the demos found in the download package to add support for bookmarks, annotations and any other available feature that you wish to add.

3 Customizing the UI implementation

Customizing the UI implementation is straightforward. Foxit Mobile PDF SDK provides the source code of the UI extensions library that contains ready-to-use UI module implementations, which lets the developers have full control of styling the appearance as desired.

To customize the UI implementation, you need to follow these steps:

First, add the following files into your app. They are all found in the "libs" folder.

- ***uiextensions_src*** project - It is an open source library that contains some ready-to-use UI module implementations, which can help developers rapidly embed a fully functional PDF reader into their Android app. Of course, developers are not forced to use the default UI, they can freely customize and design the UI for their specific apps through the "uiextensions_src" project.
- ***FoxitRDK.jar*** - used by the Java platform. It includes all of the Java APIs of Foxit Mobile PDF SDK.
- ***librdk.so*** (libs/armeabi-v7a, libs/arm64-v8a, libs/x86) - called by the Java APIs in the FoxitRDK.jar. It is the heart of the SDK and contains the core functionalities of Foxit Mobile PDF SDK, and it is built separately for each architecture. Currently it is available for armeabi-v7a, arm64-v8a, and x86.

Note *The **uiextensions_src** project has a dependency on **FoxitRDK.jar**. It is best to put them in the same directory. If they are not in the same directory, you will need to modify the reference path in the "**build.gradle**" file of the **uiextensions_src** project manually.*

Second, find the specific layout XML files that you want to customize in the ***uiextensions_src*** project, then modify them based on your requirements.

Now, for your convenience, we will show you how to customize the UI implementation in the "**viewer_ctrl_demo**" project found in the "samples" folder.

UI Customization Example

Step 1: Add the ***uiextensions_src*** project into the demo making sure that it is in the same folder as the FoxitRDK.jar file**.** This folder should already be in the right location if you have not changed the default folder hierarchy.

Note *The demo already includes references to the **FoxitRDK.jar** and **librdk.so** files, so we just need to add the **uiextensions_src** project through configuring the "settings.gradle" file. When to include the **uiextensions_src** project as a dependency, the reference to the **FoxitRDKUIExtensions.aar** needs to be removed.*

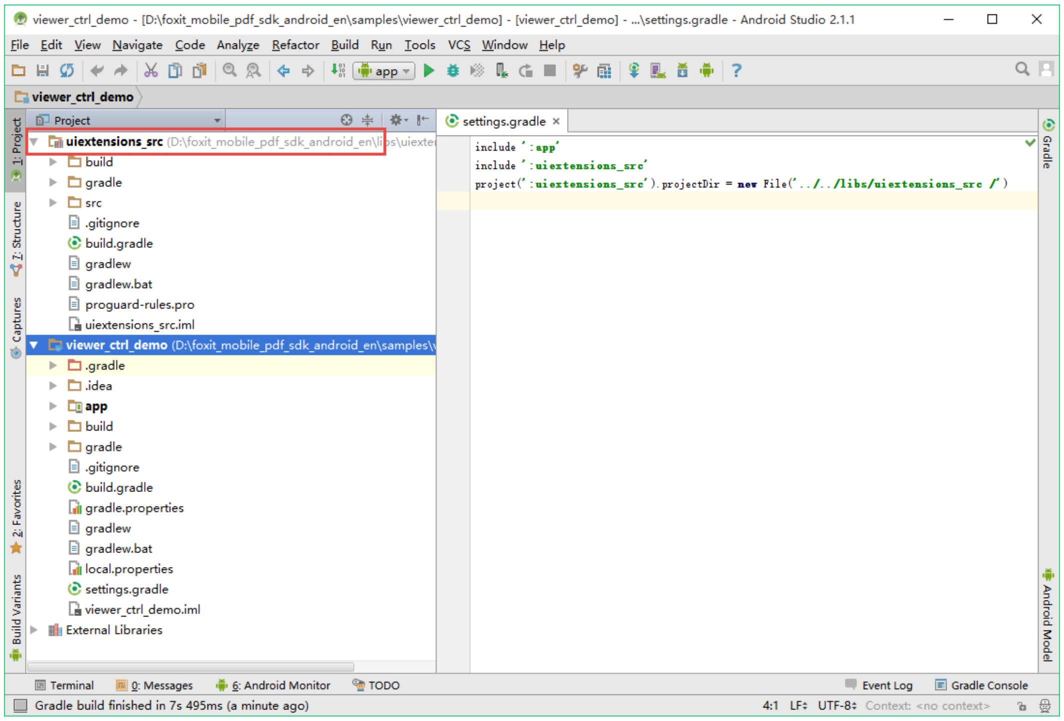
Load the "**viewer_ctrl_demo**" in Android Studio. Then, follow the steps below:

a) In the "settings.gradle" file, add the following code to include the ***uiextensions_src*** *project.*

settings.gradle:

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

Rebuild the gradle, then the ***uiextensions_src*** project will be added as shown in the Figure 3-1.


viewer_ctrl_demo - [D:\foxit_mobile_pdf_sdk_android_en\samples\viewer_ctrl_demo] - [viewer_ctrl_demo] - ...\settings.gradle - Android Studio 2.1.1

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

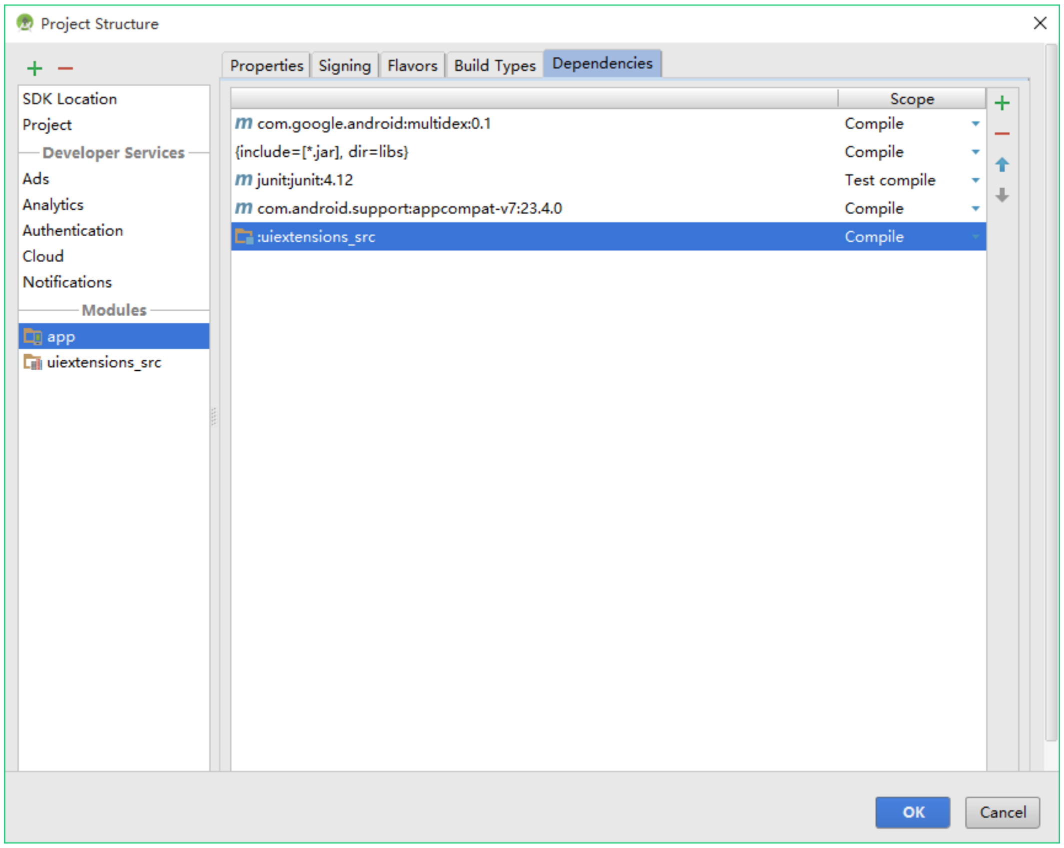
Gradle build finished in 7s 495ms (a minute ago)

Figure 3-1

b) Include the ***uiextensions_src*** project as a dependency into the demo. Inside the app's "build.gradle" file, add the *compile project(":uiextensions_src")* line and comment out the *compile(name:'FoxitRDKUIExtensions', ext:'aar')* line as follows:

```
dependencies {
    compile 'com.google.android:multidex:0.1'
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.4.0'
    compile project(":uiextensions_src")
    //compile(name:'FoxitRDKUIExtensions', ext:'aar')
}
```

After making the change, rebuild this gradle. Then, select "**File** -> **Project Structure…**" to open the **Project Structure** dialog. In the dialog click "**Modules** -> **app**", and select the **Dependencies** option, then you can see that the demo has a dependency on the ***uiextensions_src*** project as shown in the Figure 3-2.



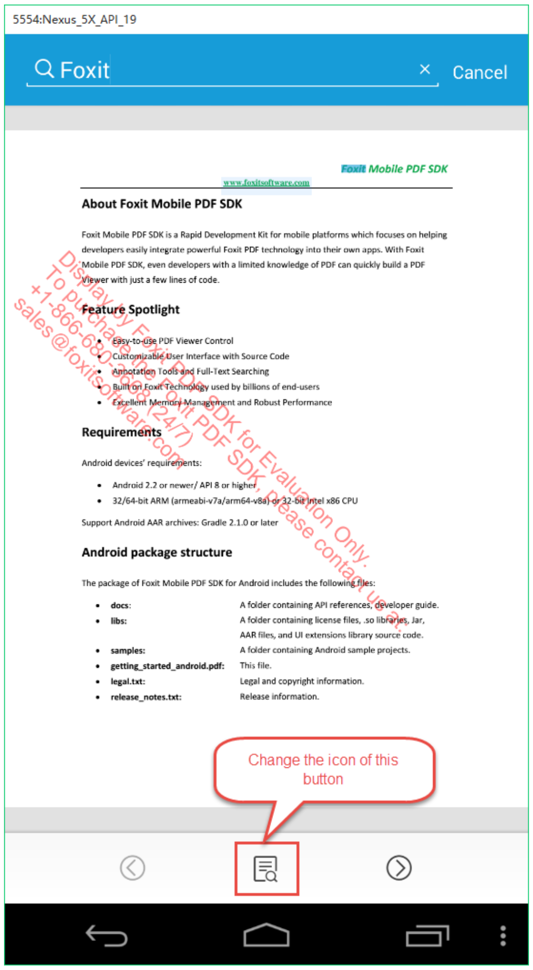
	Scope
com.google.android:multidex:0.1	Compile
{include=[*.jar], dir=libs}	Compile
junit:junit:4.12	Test compile
com.android.support:appcompat-v7:23.4.0	Compile
:uiextensions_src	Compile

Figure 3-2

Congratulations! You have completed the first step.

Step 2: Find and modify the layout files related to the UI that you want to customize.

Now we will show you a simple example that changes one button's icon in the search panel as shown in the Figure 3-3.



Foxit Mobile PDF SDK
www.foxitsoftware.com

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

The package of Foxit Mobile PDF SDK for Android includes the following files:

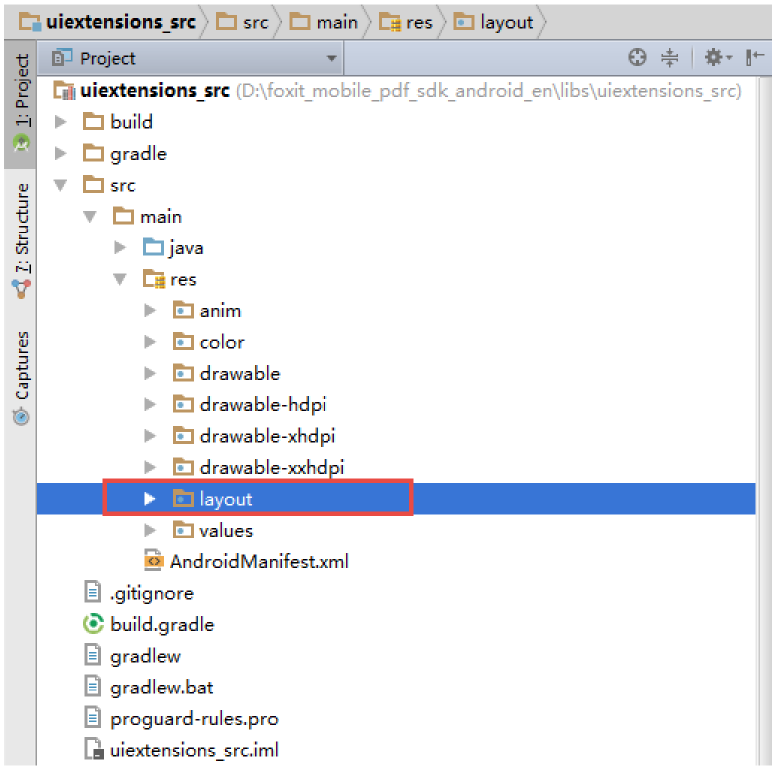
• **docs**:	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, .so libraries, Jar, AAR files, and UI extensions library source code.
• **samples:**	A folder containing Android sample projects.
• **getting_started_android.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Displayed by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 3-3

To replace the icon we only need to find the place which stores the icon for this button, then use another icon with the same name to replace it.

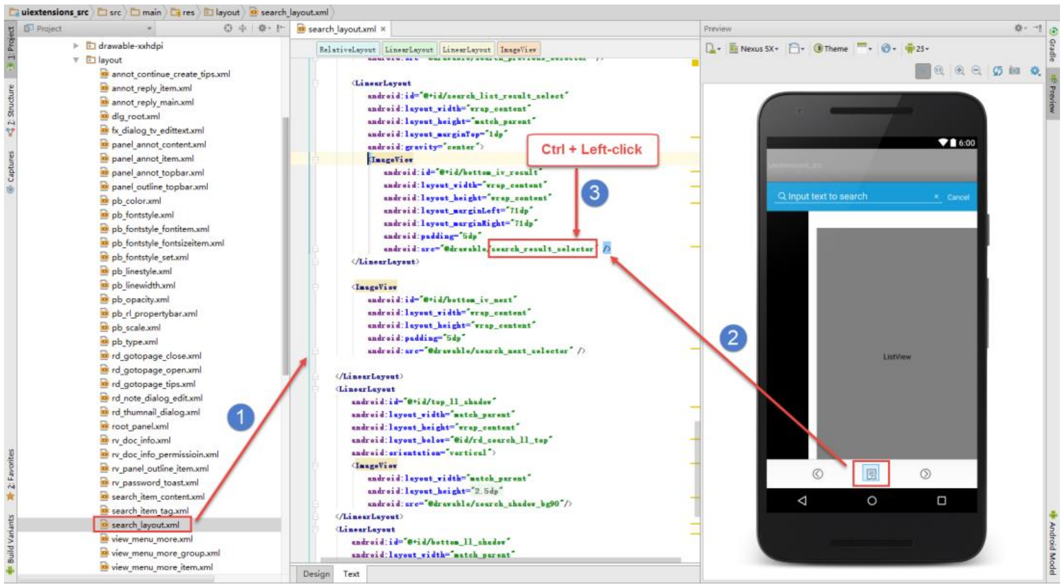
In the project, click "**uiextensions_src** -> **src** -> **res** -> **layout**" as shown in the Figure 3-4.



```
uiextensions_src (D:\foxit_mobile_pdf_sdk_android_en\libs\uiextensions_src)
  build
  gradle
  src
    main
      java
      res
        anim
        color
        drawable
        drawable-hdpi
        drawable-xhdpi
        drawable-xxhdpi
        layout
        values
      AndroidManifest.xml
  .gitignore
  build.gradle
  gradlew
  gradlew.bat
  proguard-rules.pro
  uiextensions_src.iml
```

Figure 3-4

In the layout list, find the "**search_layout.xml**" file, and double-click it. Find the button in the Preview window, and click it to navigate to the related code as shown in the Figure 3-5.



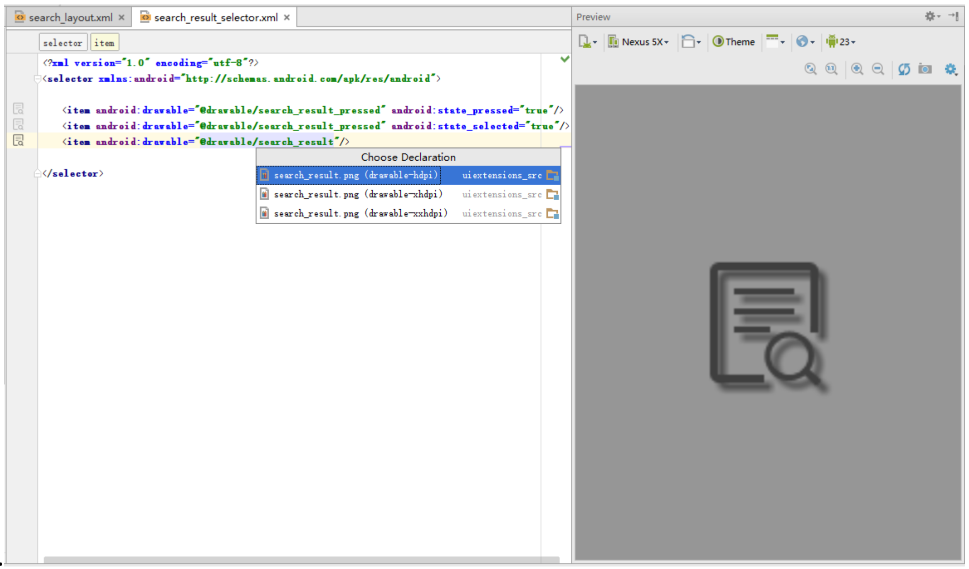
```
drawable-xxhdpi
layout
  annot_continue_create_tips.xml
  annot_reply_item.xml
  annot_reply_main.xml
  dlg_root.xml
  fx_dialog_tv_edittext.xml
  panel_annot_content.xml
  panel_annot_item.xml
  panel_annot_topbar.xml
  panel_outline_topbar.xml
  pb_color.xml
  pb_fontstyle.xml
  pb_fontstyle_fontitem.xml
  pb_fontstyle_fontsizeitem.xml
  pb_fontstyle_set.xml
  pb_linestyle.xml
  pb_linewidth.xml
  pb_opacity.xml
  pb_rl_propertybar.xml
  pb_scale.xml
  pb_type.xml
  rd_gotopage_close.xml
  rd_gotopage_open.xml
  rd_gotopage_tips.xml
  rd_note_dialog_edit.xml
  rd_thumnail_dialog.xml
  root_panel.xml
  rv_doc_info.xml
  rv_doc_info_permissioin.xml
  rv_panel_outline_item.xml
  rv_password_toast.xml
  search_item_content.xml
  search_item_tag.xml
  search_layout.xml
  view_menu_more.xml
  view_menu_more_group.xml
  view_menu_more_item.xml
```

```xml
    <LinearLayout
        android:id="@+id/search_list_result_select"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_marginTop="1dp"
        android:gravity="center">
        <ImageView
            android:id="@+id/bottom_iv_result"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="71dp"
            android:layout_marginRight="71dp"
            android:padding="5dp"
            android:src="@drawable/search_result_selector" />
    </LinearLayout>

    <ImageView
        android:id="@+id/bottom_iv_next"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dp"
        android:src="@drawable/search_next_selector" />

</LinearLayout>
<LinearLayout
    android:id="@+id/top_ll_shadow"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/rd_search_ll_top"
    android:orientation="vertical">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="2.5dp"
        android:src="@drawable/search_shadow_bg90"/>
</LinearLayout>
<LinearLayout
    android:id="@+id/bottom_ll_shadow"
    android:layout_width="match_parent"
```

Figure 3-5

After finishing the three steps described in the above picture, go to the "**search_result_selector.xml**" as shown in the Figure 3-6. We can see that the icon is stored in the "drawable-xxx" folder with the name of "search_result.png" by holding Ctrl and left-clicking on "search_result". Just replace it with your own icon.

Note *Foxit Mobile PDF SDK provides three sets of icons for devices with different DPI requirements to make sure that your apps can run smoothly on every device.*



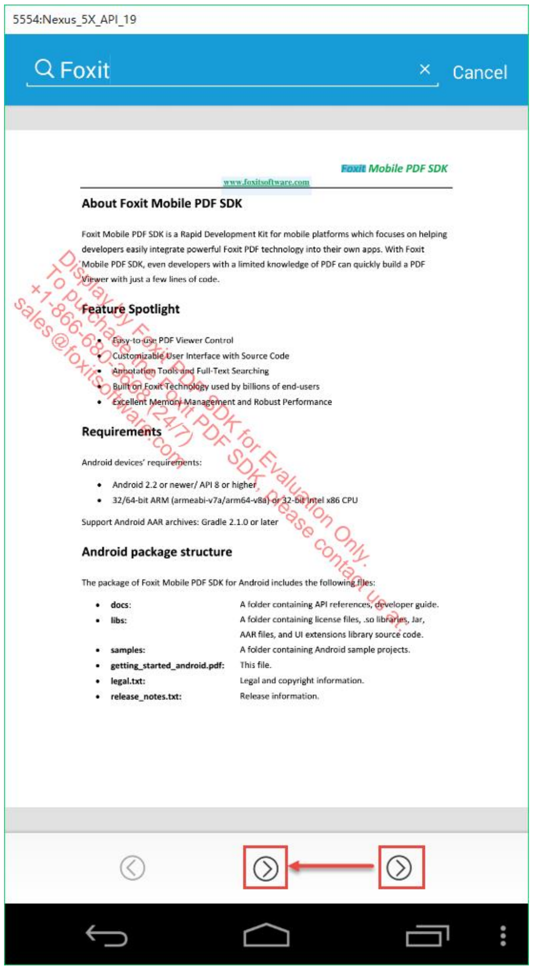
```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@drawable/search_result_pressed" android:state_pressed="true"/>
    <item android:drawable="@drawable/search_result_pressed" android:state_selected="true"/>
    <item android:drawable="@drawable/search_result"/>

</selector>
```

Choose Declaration

```
search_result.png (drawable-hdpi)     uiextensions_src
search_result.png (drawable-xhdpi)    uiextensions_src
search_result.png (drawable-xxhdpi)   uiextensions_src
```

Figure 3-6

For example, we use the icon of the search next button ("search_next.png" stored in the same folder with "search_result.png") to replace it. Then, rerun the demo, try the search feature and we can see that the icon of the bottom search button has changed as shown in the Figure 3-7.



About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

The package of Foxit Mobile PDF SDK for Android includes the following files:

docs:	A folder containing API references, developer guide.
libs:	A folder containing license files, .so libraries, Jar, AAR files, and UI extensions library source code.
samples:	A folder containing Android sample projects.
getting_started_android.pdf:	This file.
legal.txt:	Legal and copyright information.
release_notes.txt:	Release information.

Displayed by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 3-7

This is just a pretty simple example to show how to customize the UI implementation. You can refer to it and feel free to customize and design the UI for your specific apps through the ***uiextensions_src*** project.

4 Creating a custom tool

With Foxit Mobile PDF SDK, creating a custom tool is a simple process. There are several tools implemented in the UI extensions library already. These tools can be used as a base for developers to build upon or use as a reference to create a new tool. In order to create your own tool quickly, we suggest you take a look at the ***uiextensions_src*** project found in the "libs" folder.

To create a new tool, the most important step is to create a Java class that implements the "**ToolHandler.java**" interface.

In this section, we will make a Regional Screenshot Tool to show how to create a custom tool with Foxit Mobile PDF SDK. This tool can help the users who only want to select an area in a PDF page to capture, and then save it as an image. Now, let's do it.

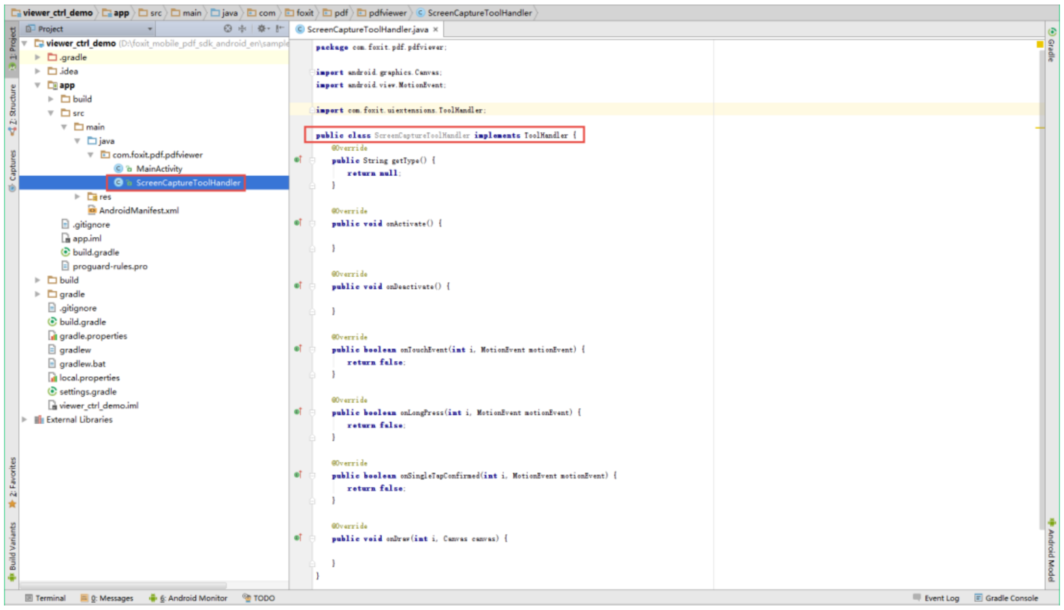
For convenience, we will build this tool based on the "**viewer_ctrl_demo**" project found in the "samples" folder. Steps required for implementing this tool are as follows:

- Create a Java class named **ScreenCaptureToolHandler** that implements the "**ToolHandler.java**" interface.
- Handle the **onTouchEvent** and **onDraw** events.
- Instantiate a **ScreenCaptureToolHandler** object, and then register it to the tool manager.
- Set the **ScreenCaptureToolHandler** object as the current tool handler.

Step 1: Create a Java class named **ScreenCaptureToolHandler** that implements the "**ToolHandler.java**" interface.

a) Load the "**viewer_ctrl_demo**" project in Android Studio. Create a Java class named "**ScreenCaptureToolHandler**" in the "com.foxit.pdf.viewctrl" package.

b) Let the **ScreenCaptureToolHandler.java** class implement the **ToolHandler** interface as shown in the Figure 4-1.



```
package com.foxit.pdf.pdfviewer;

import android.graphics.Canvas;
import android.view.MotionEvent;

import com.foxit.uiextensions.ToolHandler;

public class ScreenCaptureToolHandler implements ToolHandler {
    @Override
    public String getType() {
        return null;
    }

    @Override
    public void onActivate() {

    }

    @Override
    public void onDeactivate() {

    }

    @Override
    public boolean onTouchEvent(int i, MotionEvent motionEvent) {
        return false;
    }

    @Override
    public boolean onLongPress(int i, MotionEvent motionEvent) {
        return false;
    }

    @Override
    public boolean onSingleTapConfirmed(int i, MotionEvent motionEvent) {
        return false;
    }

    @Override
    public void onDraw(int i, Canvas canvas) {

    }
}
```

Figure 4-1

Step 2: Handle the **onTouchEvent** and **onDraw** events.

Update **ScreenCaptureToolHandler.java** as follows:

```
package com.foxit.pdf.pdfviewer;

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

import com.foxit.sdk.PDFViewCtrl;
import com.foxit.sdk.common.CommonDefines;
import com.foxit.sdk.common.PDFException;
import com.foxit.sdk.pdf.PDFPage;
import com.foxit.sdk.pdf.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;
    private ViewGroup mParent;

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

    @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.convertDisplayViewPtToPageViewPt(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 = Renderer.create(bmp);

            // Get the display matrix.
            Matrix matrix = page.getDisplayMatrix(0, 0, width, height, 0);
            int progress = renderer.startRender(page, matrix, null);
            while (progress == CommonDefines.e_progressToBeContinued) {
                progress = renderer.continueRender();
            }
            // 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 i, MotionEvent motionEvent) {
        return false;
    }

    @Override
    public boolean onSingleTapConfirmed(int i, MotionEvent motionEvent) {
        return false;
    }

    //Handle the drawing event.
    @Override
    public void onDraw(int pageIndex, Canvas canvas) {
        if (((UIExtensionsManager)
mPdfViewCtrl.getUIExtensionsManager()).getCurrentToolHandler() != this)
            return;
        if (mLastPageIndex != pageIndex) {
            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();
    }
}
```

Step 3: Instantiate a **ScreenCaptureToolHandler** object and then register it to the UIExtensionsManager.

```
private ScreenCaptureToolHandler screenCapture = null;
...
screenCapture = new SelectRectToolHandler(this, parent, pdfViewCtrl);
uiExtensionsManager.registerToolHandler(screenCapture);
```

Step 4: Set the **ScreenCaptureToolHandler** object as the current tool handler.

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

Now, we have really finished creating a custom tool. In order to see what the tool looks like, we need to make it run. Just add a menu item and add the code referred in Step 3 and Step 4 to MainActivity.java.

First, add a menu item in **Main.xml** found in the “app/src/main/res/menu” as follows.

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

Then, add the following code to the **onOptionsItemSelected()** function in **MainActivity.java**.

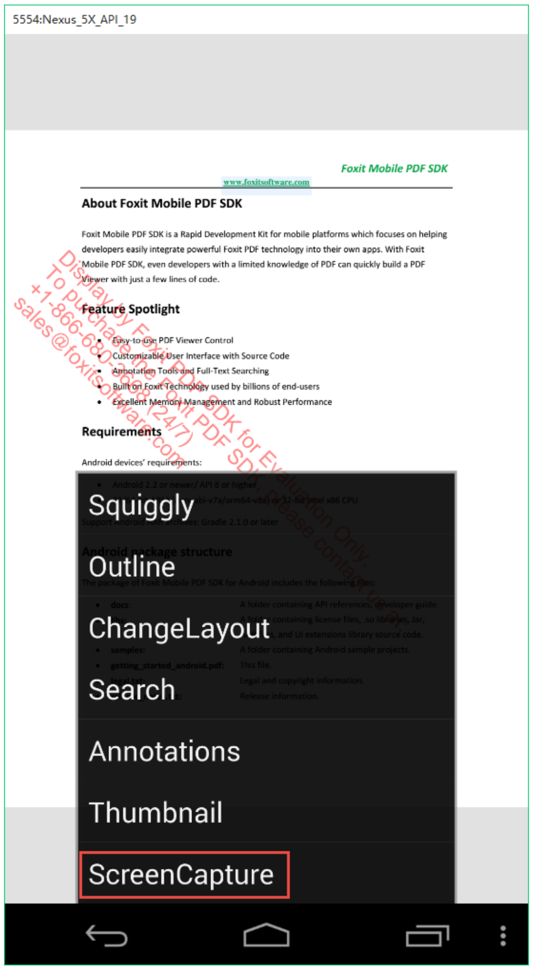
```
if (itemId == R.id.screencapture) {
    if (screenCapture == null) {
        screenCapture = new ScreenCaptureToolHandler(this, parent, pdfViewCtrl);
        uiExtensionsManager.registerToolHandler(screenCapture);
    }
    uiExtensionsManager.setCurrentToolHandler(screenCapture);
}
```

Please remember to instantiate a **ScreenCaptureToolHandler** object at first, like (`private ScreenCaptureToolHandler screenCapture = null;`).

After finishing all of the above work, build and run the demo.

***Note** Here, we run the demo on an AVD targeting 4.4.2. Please make sure you have pushed the "getting_started_android.pdf" document into the emulator's SD card.*

After building the demo successfully, click the menu to find the **ScreenCapture** option as shown in the Figure 4-2.



Foxit Mobile PDF SDK
www.foxitsoftware.com

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

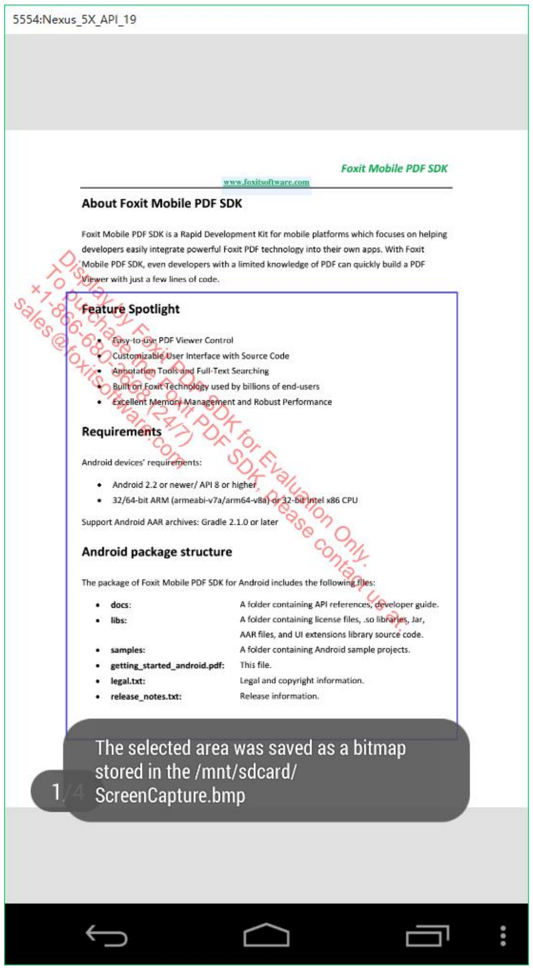
Requirements

Android devices' requirements:

Display by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 4-2

Click the **ScreenCapture**, select a rectangular area, and then a message box will be popped up as shown in the Figure 4-3. It shows where the bitmap (selected area) was saved to.


5554:Nexus_5X_API_19

Foxit Mobile PDF SDK
www.foxitsoftware.com

About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for mobile platforms which focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a PDF Viewer with just a few lines of code.

Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

The package of Foxit Mobile PDF SDK for Android includes the following files:

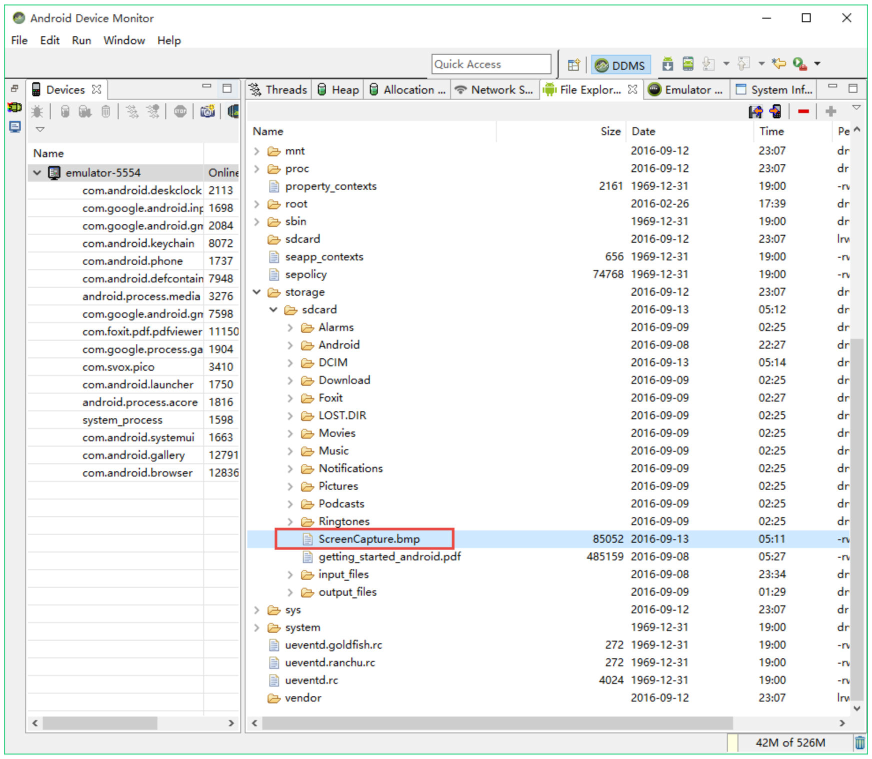
• **docs**:	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, .so libraries, Jar, AAR files, and UI extensions library source code.
• **samples:**	A folder containing Android sample projects.
• **getting_started_android.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Displayed by Foxit PDF SDK for Evaluation Only.
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

The selected area was saved as a bitmap stored in the /mnt/sdcard/ScreenCapture.bmp

Figure 4-3

In order to verify whether the tool captures the selected area successfully, we need to find the screenshot. Open the **Android Device Monitor**, we can see the screenshot named "ScreenCapture.bmp" in the SD card as shown in the Figure 4-4.

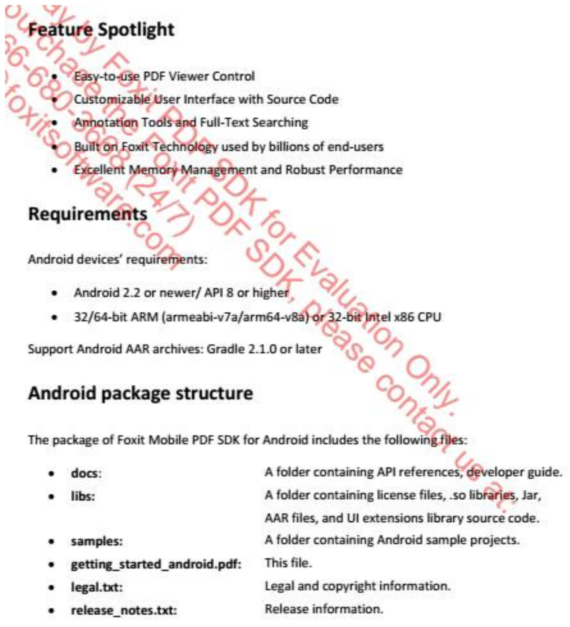


Name	
emulator-5554	Online
com.android.deskclock	2113
com.google.android.inp	1698
com.google.android.gn	2084
com.android.keychain	8072
com.android.phone	1737
com.android.defcontain	7948
android.process.media	3276
com.google.android.gn	7598
com.foxit.pdf.pdfviewer	11150
com.google.process.ga	1904
com.svox.pico	3410
com.android.launcher	1750
android.process.acore	1816
system_process	1598
com.android.systemui	1663
com.android.gallery	12791
com.android.browser	12836

Name	Size	Date	Time	Pe
mnt		2016-09-12	23:07	dr
proc		2016-09-12	23:07	dr
property_contexts	2161	1969-12-31	19:00	-rv
root		2016-02-26	17:39	dr
sbin		1969-12-31	19:00	dr
sdcard		2016-09-12	23:07	lrv
seapp_contexts	656	1969-12-31	19:00	-rv
sepolicy	74768	1969-12-31	19:00	-rv
storage		2016-09-12	23:07	dr
sdcard		2016-09-13	05:12	dr
Alarms		2016-09-09	02:25	dr
Android		2016-09-08	22:27	dr
DCIM		2016-09-13	05:14	dr
Download		2016-09-09	02:25	dr
Foxit		2016-09-09	02:27	dr
LOST.DIR		2016-09-09	02:25	dr
Movies		2016-09-09	02:25	dr
Music		2016-09-09	02:25	dr
Notifications		2016-09-09	02:25	dr
Pictures		2016-09-09	02:25	dr
Podcasts		2016-09-09	02:25	dr
Ringtones		2016-09-09	02:25	dr
ScreenCapture.bmp	85052	2016-09-13	05:11	-rv
getting_started_android.pdf	485159	2016-09-08	05:27	-rv
input_files		2016-09-08	23:34	dr
output_files		2016-09-09	01:29	dr
sys		2016-09-12	23:07	dr
system		1969-12-31	19:00	dr
ueventd.goldfish.rc	272	1969-12-31	19:00	-rv
ueventd.ranchu.rc	272	1969-12-31	19:00	-rv
ueventd.rc	4024	1969-12-31	19:00	-rv
vendor		2016-09-12	23:07	lrv

Figure 4-4

Pull the ScreenCapture.bmp from the emulator, we can see the image as shown in the Figure 4-5.



Feature Spotlight

- Easy-to-use PDF Viewer Control
- Customizable User Interface with Source Code
- Annotation Tools and Full-Text Searching
- Built on Foxit Technology used by billions of end-users
- Excellent Memory Management and Robust Performance

Requirements

Android devices' requirements:

- Android 2.2 or newer/ API 8 or higher
- 32/64-bit ARM (armeabi-v7a/arm64-v8a) or 32-bit Intel x86 CPU

Support Android AAR archives: Gradle 2.1.0 or later

Android package structure

The package of Foxit Mobile PDF SDK for Android includes the following files:

• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, .so libraries, Jar, AAR files, and UI extensions library source code.
• **samples:**	A folder containing Android sample projects.
• **getting_started_android.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Foxit PDF SDK for Evaluation Only.
Foxit PDF SDK, please contact us at:
(24/7)
foxitsoftware.com

Figure 4-5

As you can see we have successfully created a Regional Screenshot Tool. This is just an example to show how to create a custom tool with Foxit Mobile PDF SDK. You can refer to it or our demos to develop the tools you want.

5 Technical Support

Reporting Problems

Foxit offers 24/7 support for its products and are fully supported by the PDF industry's largest development team of support engineers. If you encounter any technical questions or bug issues when using Foxit Mobile PDF SDK, please submit the problem report to the Foxit support team at http://tickets.foxitsoftware.com/create.php. In order to better help you solve the problem, please provide the following information:

- Contact details
- Foxit Mobile PDF SDK product and version
- Your Operating System and IDE version
- Detailed description of the problem
- Any other related information, such as log file or error screenshot

Contact Information

You can contact Foxit directly, please use the contact information as follows:

Foxit Support:

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

Sales Contact:

- Phone: 1-866-680-3668
- Email: sales@foxitsoftware.com

Support & General Contact:

- Phone: 1-866-MYFOXIT or 1-866-693-6948
- Email: support@foxitsoftware.com