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.

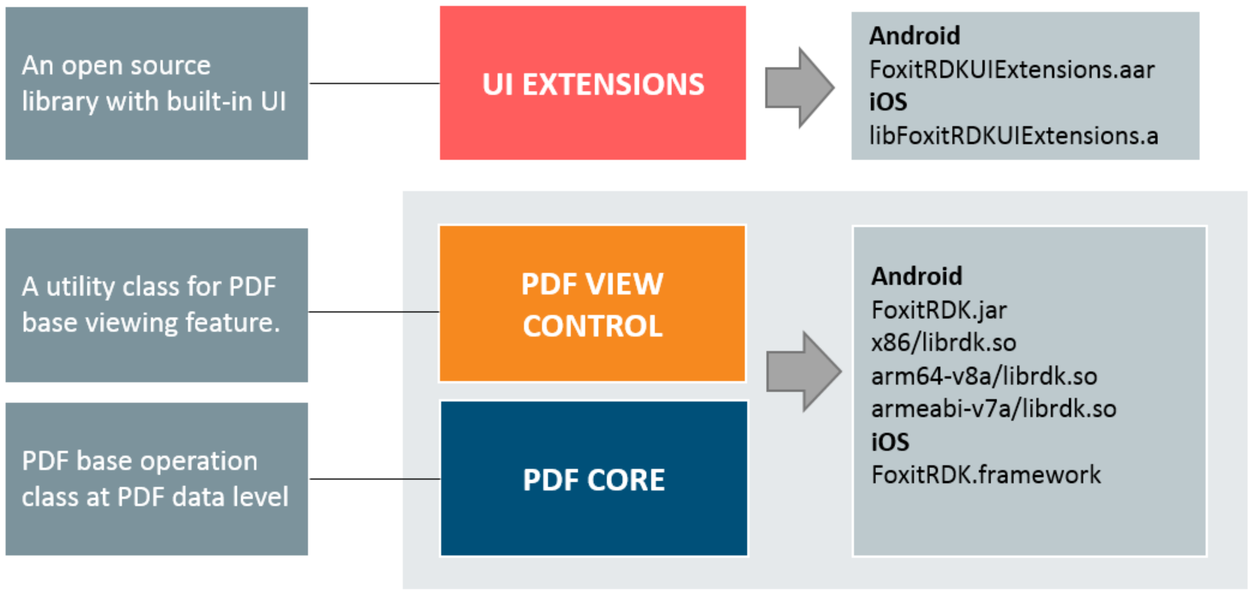
- **Premium World-side Support**

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 iOS 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 iOS platform. The following sections introduce the structure of the installation package, how to run a demo, and how to create your own project in Xcode

2.1 Requirements

- iOS 8.0 or higher
- Xcode 7.0 or newer

2.2 What is in the Package

Download the "foxit_mobile_pdf_sdk_ios_en.zip" package, and extract it to a new directory like "foxit_mobile_pdf_sdk_ios_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, SDK framework, ui extensions library and source code.
samples:	A folder containing iOS sample projects.
getting_started_ios.pdf:	A quick guide for Foxit Mobile PDF SDK for iOS.
legal.txt:	Legal and copyright information.
release_notes.txt:	Release information.

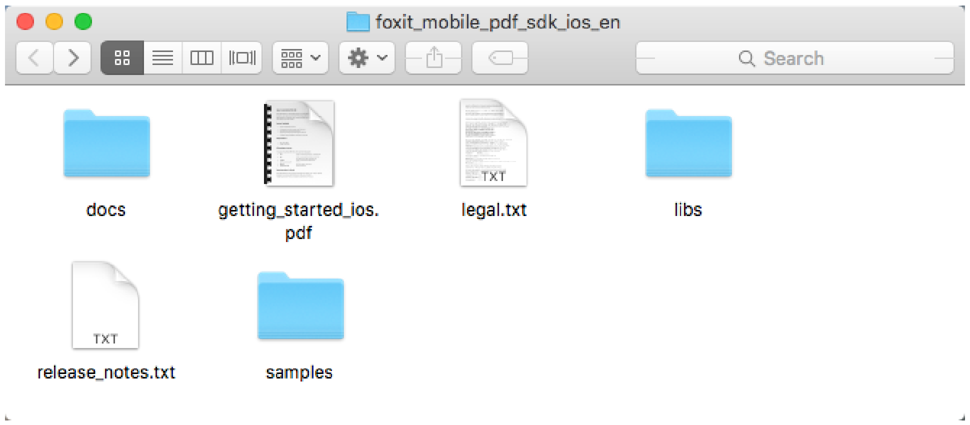


Figure 2-1

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 iOS.

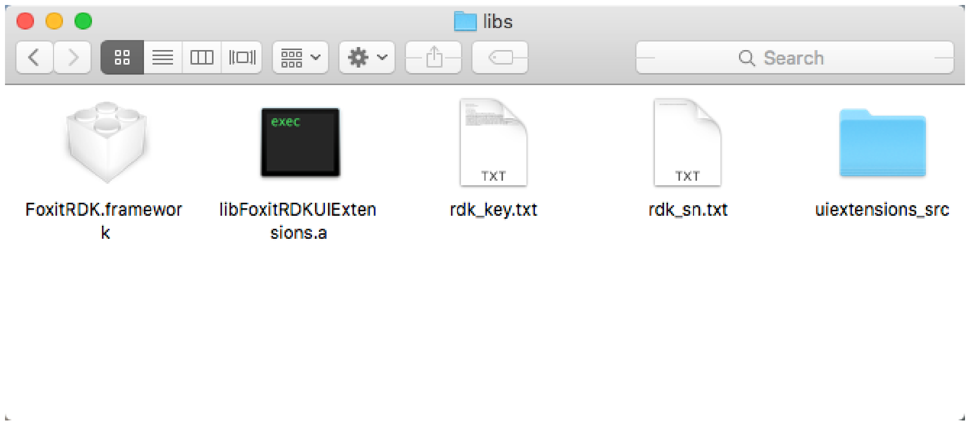


Figure 2-2

- ***FoxitRDK.framework*** – The framework that includes the Foxit Mobile PDF SDK dynamic library and associated header files.
- ***libFoxitRDKUIExtensions.a*** – It’s a universal static library (for both simulator and iOS device) generated by the “**uiextensions**” project found in the “libs/uiextensions_src” folder.
- ***uiextensions*** *project*- found in the “libs/uiextensions_src” 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 iOS 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” project.

2.3 How to run a demo

Download and install Xcode IDE (https://developer.apple.com/download/).

Note*: In this guide, we do not cover the installation of Xcode. You can refer to Apple’s developer site if you haven’t installed it already.*

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

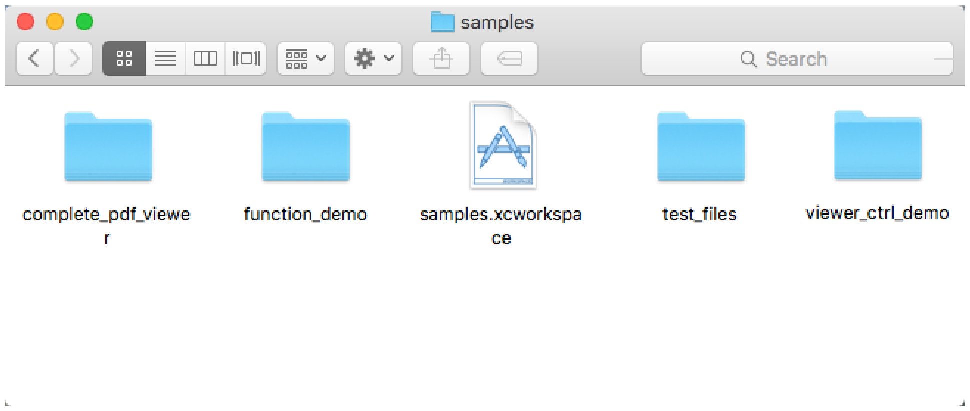


Figure 2-3

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 with PDF. This demo includes the following features:

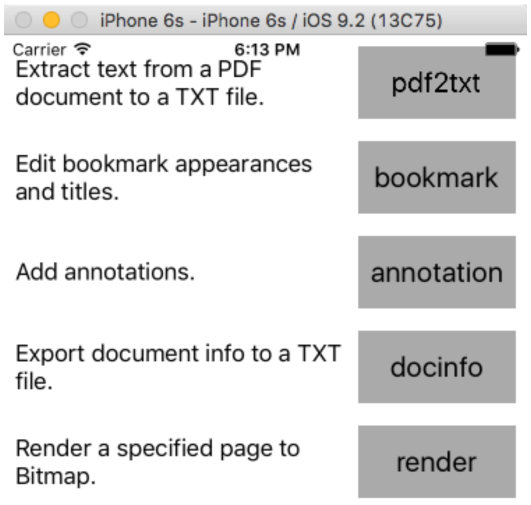
- **pdf2txt**: extract all the text from a PDF document, and then save them to a text file.
- **bookmark**: change the appearance of a PDF's bookmark, and rename its title.
- **annotation**: add some annotations to one page of a PDF.
- **docinfo**: export the basic information of a PDF to a text file.
- **render**: render a specific page of a PDF to a bitmap, and save the bitmap.

To run it in Xcode, follow the steps below:

a) Double-click **function_demo.xcodeproj** found in the "samples/function_demo" folder to open the demo in Xcode.

   ***Note**: There is another way to open the demo in Xocde: double-click **samples_xcworkspace** found in the "samples" folder. It is a workspace including the three demos.*

b) Click on "Product -> Run" to run the demo on an iOS device or simulator. In this guide, an iPhone 6s Simulator will be used as an example. After building the demo successfully, the features are listed like the Figure 2-4.



Extract text from a PDF document to a TXT file.

Edit bookmark appearances and titles.

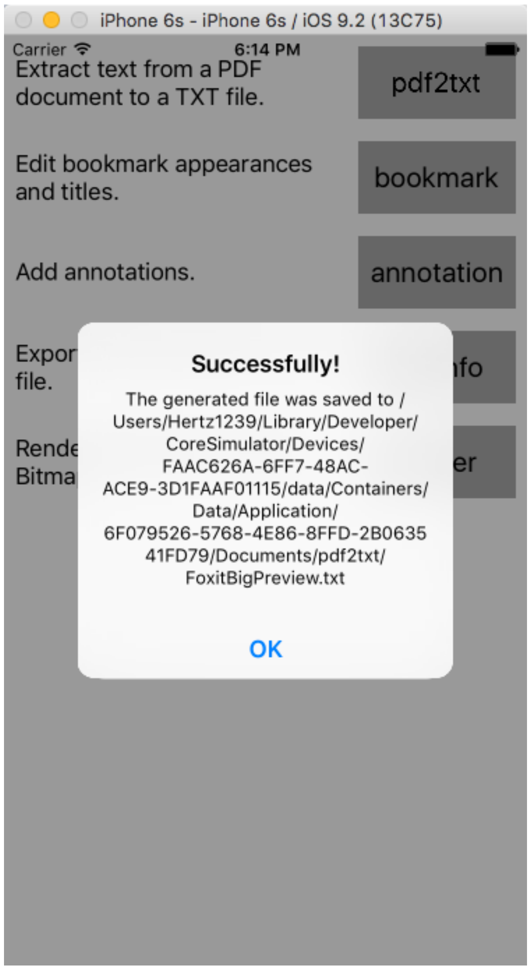
Add annotations.

Export document info to a TXT file.

Render a specified page to Bitmap.

Figure 2-4

c) 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 2-5. It shows where the text file was saved to. Just run the demo and try the features.


iPhone 6s - iPhone 6s / iOS 9.2 (13C75)
Carrier
6:14 PM
Extract text from a PDF document to a TXT file.
pdf2txt
Edit bookmark appearances and titles.
bookmark
Add annotations.
annotation
Expor
file.
nfo
Rende
Bitma
er
OK

Successfully!

The generated file was saved to /Users/Hertz1239/Library/Developer/CoreSimulator/Devices/FAAC626A-6FF7-48AC-ACE9-3D1FAAF01115/data/Containers/Data/Application/6F079526-5768-4E86-8FFD-2B063541FD79/Documents/pdf2txt/FoxitBigPreview.txt

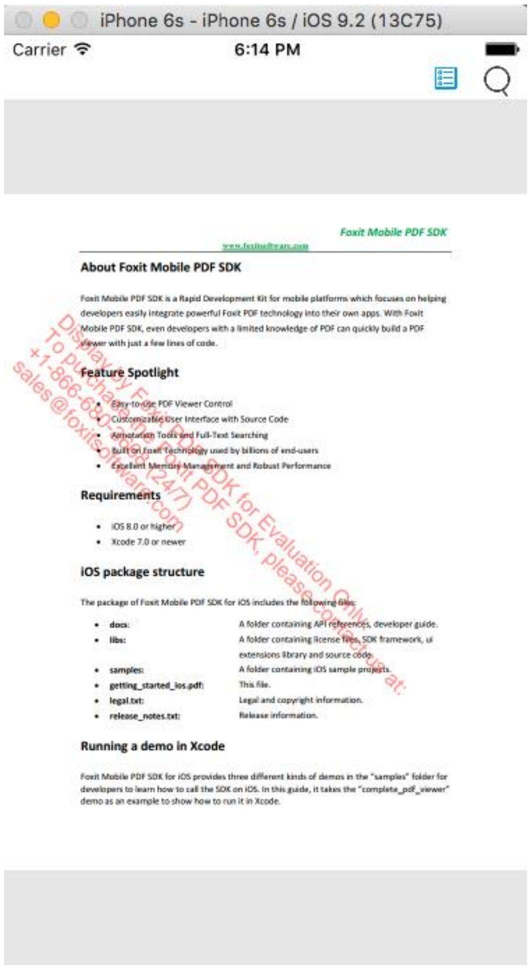
Figure 2-5

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.), outline and text search. 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 Xcode, please refer to the setup steps outlined in the Function demo.

Figure 2-6 shows what the demo looks like after it was built successfully. Here, an iPhone 6s Simulator will be used as an example to run the demo.



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

- iOS 8.0 or higher
- Xcode 7.0 or newer

iOS package structure

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

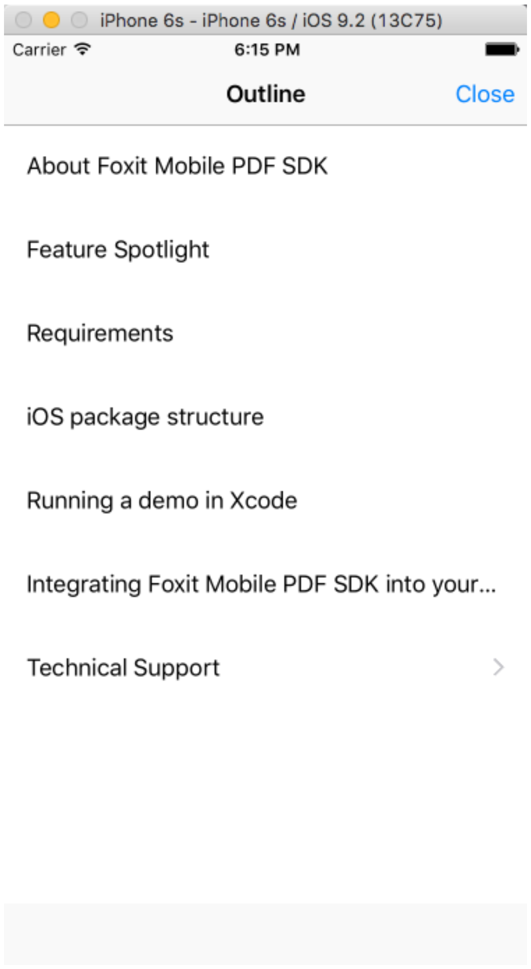
• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, SDK framework, ui extensions library and source code.
• **samples:**	A folder containing iOS sample projects.
• **getting_started_ios.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Running a demo in Xcode

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

Figure 2-6

This demo provides the features like text search and outline. For example, click ☰, then the outline (outline is the technical terms for bookmark in the PDF specification) of this document will be displayed as shown in the Figure 2-7.


iPhone 6s - iPhone 6s / iOS 9.2 (13C75)

Carrier 6:15 PM

Outline Close

About Foxit Mobile PDF SDK

Feature Spotlight

Requirements

iOS package structure

Running a demo in Xcode

Integrating Foxit Mobile PDF SDK into your...

Technical Support

Figure 2-7

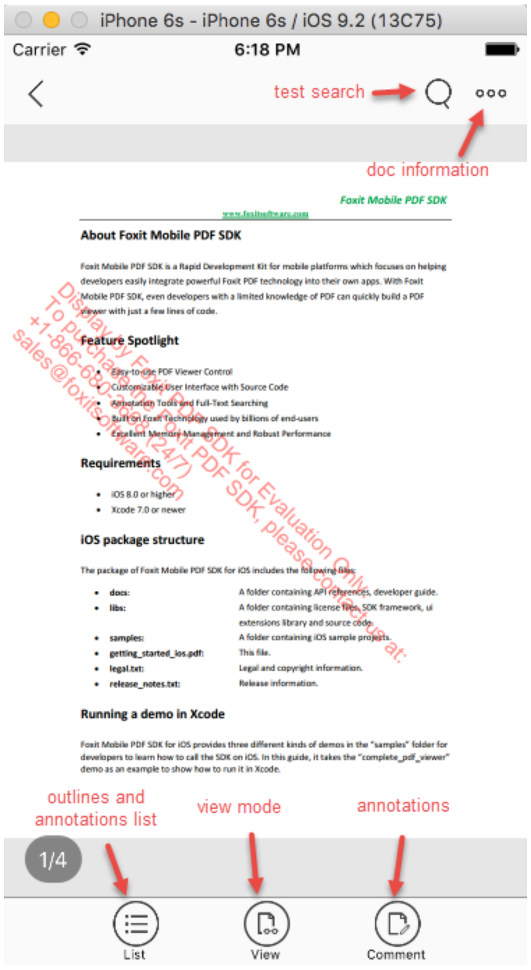
2.3.3 Complete PDF viewer demo

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 Xcode, please refer to the setup steps outlined in the Function demo.

Here, an iPhone 6s Simulator will also be used as an example to run the demo. After building the demo successfully, on the start screen, click the "getting_started_ios.pdf" document, and then it will be opened and displayed as shown in the Figure 2-8.

Note *If you want to use other PDF files to test this demo, you need to put them onto the "Document" folder of the device.*



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

- iOS 8.0 or higher
- Xcode 7.0 or newer

iOS package structure

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

• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, SDK framework, ui extensions library and source code.
• **samples:**	A folder containing iOS sample projects.
• **getting_started_ios.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

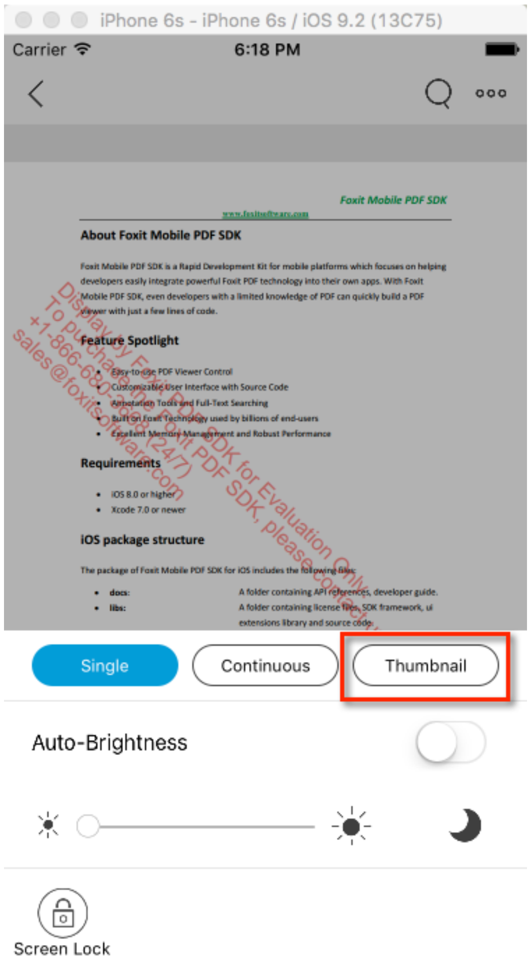
Running a demo in Xcode

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

Figure 2-8

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 2-9, and then the thumbnail of the document will be displayed as shown in the Figure 2-10.



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

- iOS 8.0 or higher
- Xcode 7.0 or newer

iOS package structure

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

• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, SDK framework, ui extensions library and source code.

Figure 2-9

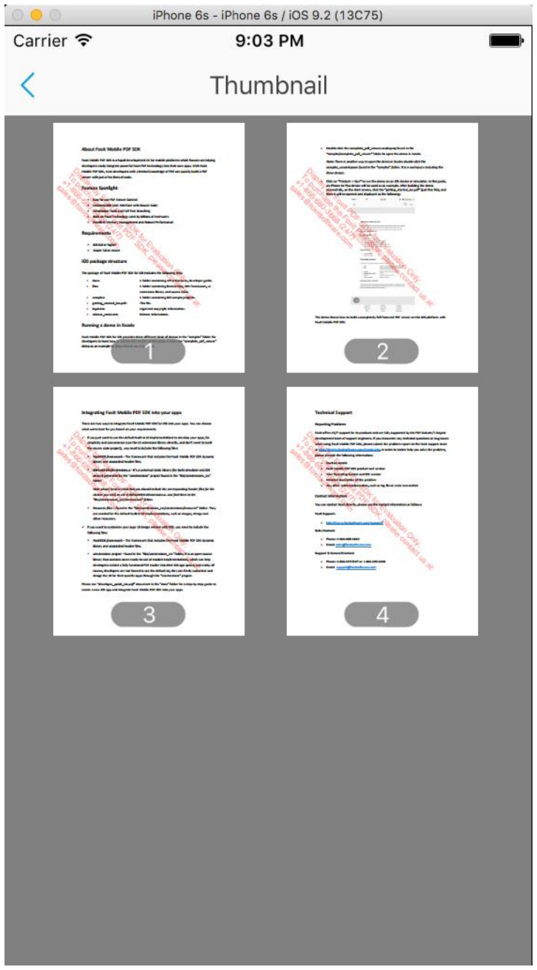


Figure 2-10

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

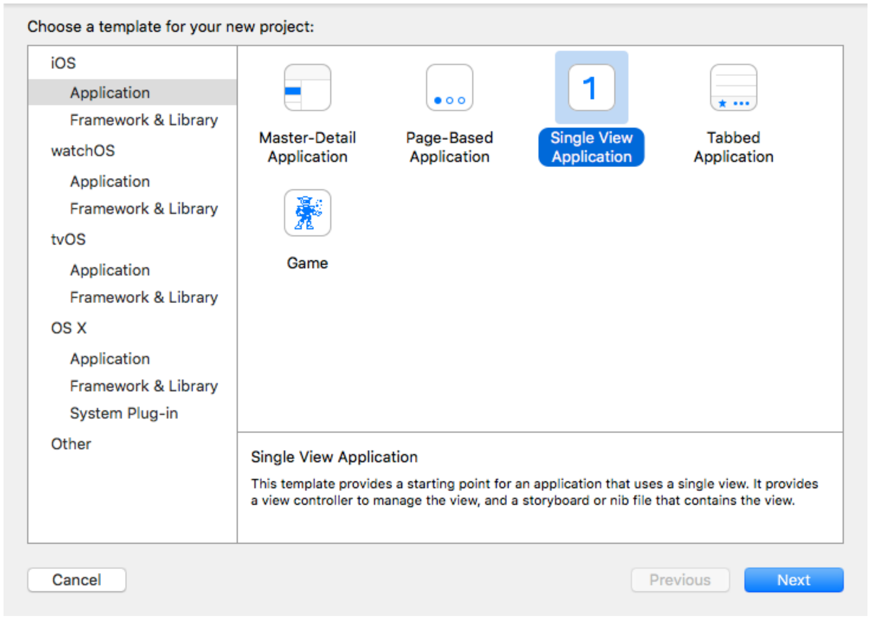
This section will help you to quickly get started with using Foxit Mobile PDF SDK to make an iOS app with step-by-step instructions provided. From now, you can get familiar with Foxit Mobile PDF SDK and create your first PDF iOS app in Xcode. This section includes the following steps:

1) Create a new iOS 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 iOS project

In this guide, we use Xcode 7.2.1 to create a new iOS project.

Open Xcode, choose **File** -> **New** -> **Project**…, and then select **iOS** -> **Application** -> **Single View Application** as shown in the Figure 2-11. Click **Next**.

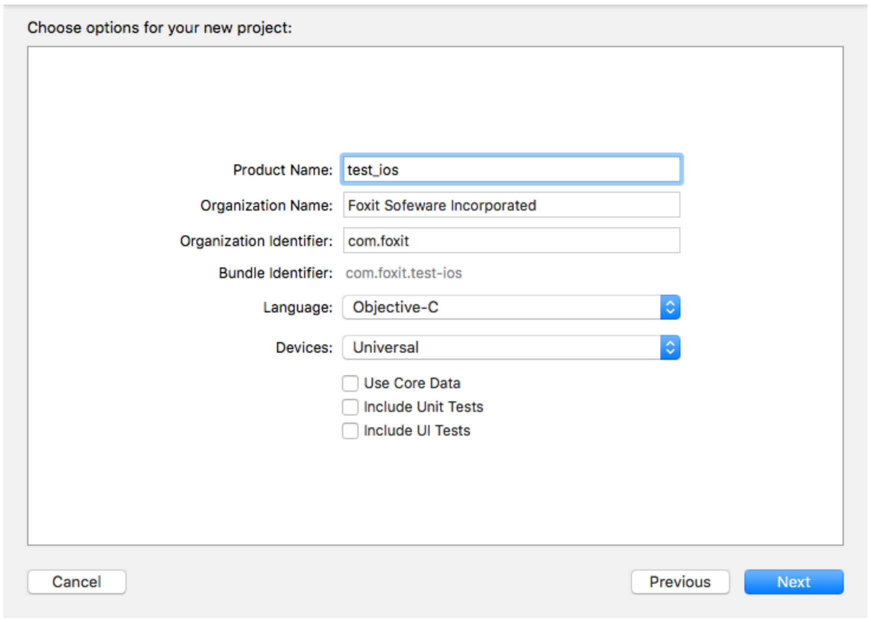


Single View Application

This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.

Figure 2-11

Choose the options for your new project as shown in the Figure 2-12. For simplicity, we don’t check the Unit Tests and UI Tests which are used for automated testing. Then, Click **Next**.

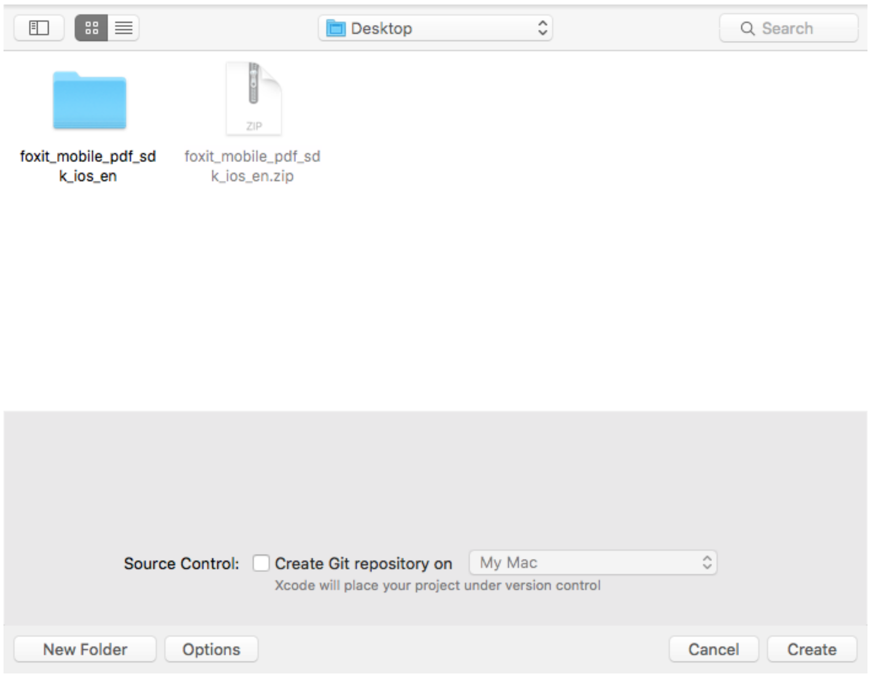


Choose options for your new project:

Product Name: test_ios
Organization Name: Foxit Sofeware Incorporated
Organization Identifier: com.foxit
Bundle Identifier: com.foxit.test-ios
Language: Objective-C
Devices: Universal
☐ Use Core Data
☐ Include Unit Tests
☐ Include UI Tests

Figure 2-12

Place the project to the location as desired. The option "version control" is not actually important for building your first PDF app, so let's use the default setting. Here, we place the project to the desktop as shown in the Figure 2-13. Then, click **Create**.



Source Control: ☐ Create Git repository on My Mac
Xcode will place your project under version control

Figure 2-13

2.4.2 Integrate Foxit Mobile PDF SDK into your apps

Note: *In this section, we will use the default built-in UI implementations to develop the app, for simplicity and convenience (use the UI extensions library directly, and don't need to build the source code project), we need to add the following files to the test_ios project.*

- ***FoxitRDK.framework*** - The framework that includes the Foxit Mobile PDF SDK dynamic library and associated header files.
- ***libFoxitRDKUIExtensions.a*** – It's a universal static library (for both simulator and iOS device) generated by the "**uiextensions**" project found in the "libs/uiextensions_src" folder.

  Note *please keep in mind that you should include the corresponding header files for the classes you need to use in libFoxitRDKUIExtensions.a. Just find them in the "libs/uiextensions_src/uiextensions" folder.*
- ***Resource files*** – found in the "libs/uiextensions_src/uiextensions/resource" folder. They are needed for the default built-in UI implementations, such as images, strings and other resources.

Note: *The UI extensions library (**libFoxitRDKUIExtensions.a**) and resource files are not required in the section "Display a PDF document", so we just add "**FoxitRDK.framework**" to the test_ios project first. In the section "Add support for Text Search, Bookmarks, and Annotations", we will introduce how to add the UI extensions library and resource files.*

To add the dynamic framework "***FoxitRDK.framework***" into the *test_ios* project, please follows the steps below:

a) Right-click the "*test_ios*" project, select **Add Files to "test_ios"**... as shown in the Figure 2-14.

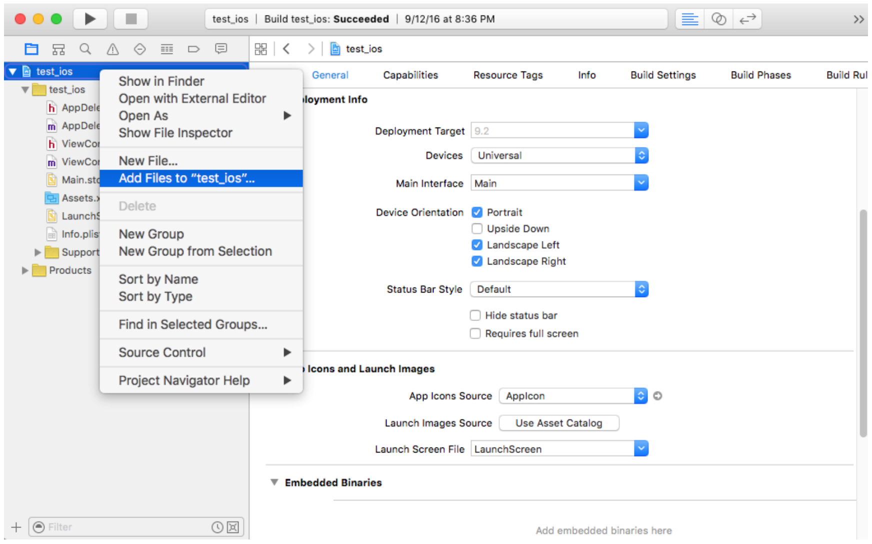
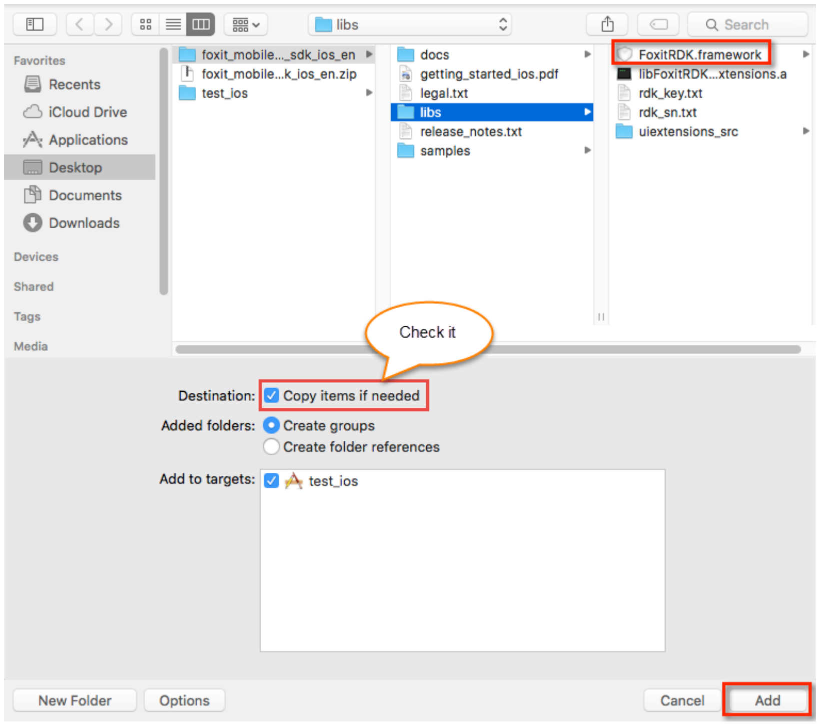


Figure 2-14

b) Find and choose "***FoxitRDK.framework***" in the "libs" folder of the download package, and then click **Add** as shown in the Figure 2-15.

Note *Make sure to check the "**Copy items if needed**" option.*



Destination: [✓ Copy items if needed]
Added folders: [✓ Create groups] Create folder references
Add to targets: [✓ test_ios]

Figure 2-15

Then, the *test_ios* project will look like the Figure 2-16.

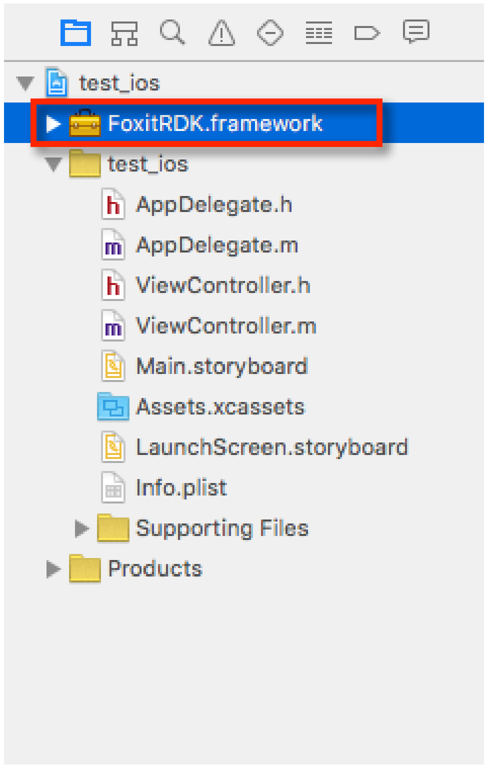


Figure 2-16

c) Add the dynamic framework "***FoxitRDK.framework***" to the Xcode's **Embedded Binaries**. Left-click the project, find **Embedded Binaries** in the **General** tab, and press on the **+** button as shown in the Figure 2-17.

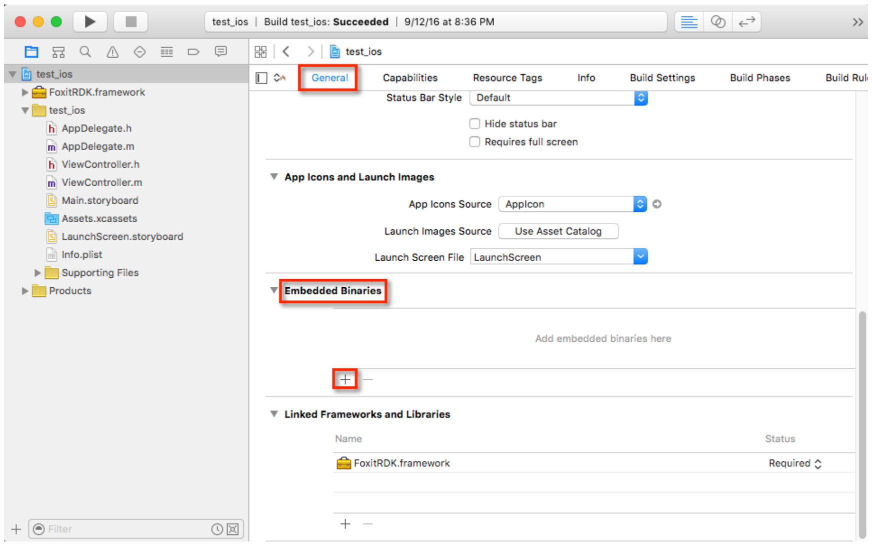


Figure 2-17

Then, choose the "***FoxitRDK.framework***" to add, and the **Embedded Binaries** will be like the Figure 2-18**.**

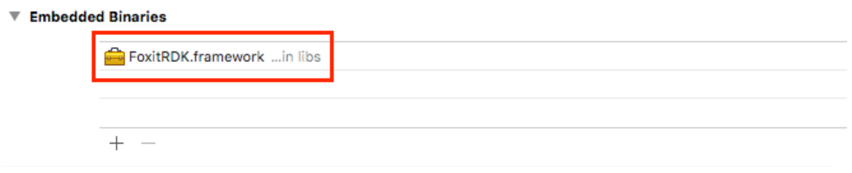


Figure 2-18

Now, we have added "***FoxitRDK.framework***" to the *test_ios* project successfully.

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 *[FSLibrary init:sn key: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. Finish the initialization in the didFinishLaunchingWithOptions method within the AppDelegate.m file.

```
#import "FoxitRDK/FSPDFObjC.h"

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {

    NSString* sn = @"";
    NSString* key =@"";
    enum FS_ERRORCODE eRet = [FSLibrary init:sn key:key];
    if (e_errSuccess != eRet) {
        return NO;
    }
    return YES;
}
```

***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=").*

Then we just have to call the *[FSLibrary release]* function to release the library in the applicationWillTerminate method.

```
- (void)applicationWillTerminate:(UIApplication *)application {
    [FSLibrary release];
}
```

In short, make sure that the AppDelegate.m file includes the following code:

```
#import "AppDelegate.h"
#import "FoxitRDK/FSPDFObjC.h"

@interface AppDelegate ()
```

```
@end

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {

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

    enum FS_ERRORCODE eRet = [FSLibrary init:sn key:key];
    if (e_errSuccess != eRet) {
        return NO;
    }
    return YES;
}

- (void)applicationWillTerminate:(UIApplication *)application {
    [FSLibrary release];
}

@end
```

2.4.4 Display a PDF document

So far, we have added “***FoxitRDK.framework***” to the *test_ios* project, and finished the initialization of the Foxit Mobile PDF SDK. 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 need to display a PDF document.*

First of all, add a PDF file to the project which will be used as the test file. For example, we use “getting_started_ios.pdf” found in the download package. Right-click the *test_ios* project, and select **Add Files to “test_ios”**… to add this file. After adding, you can see the PDF in the Xcode’s **Copy Bundle Resources** as shown in the Figure 2-19.

***Note** You can add the PDF to **Copy Bundle Resources** directly. Just right-click the test_ios project, find **Copy Bundle Resources** in the **Build Phases** tab, press on the **+** button, and choose the file to add. You can refer to any PDF file, just add it to the Xcode’s **Copy Bundle Resources**.*

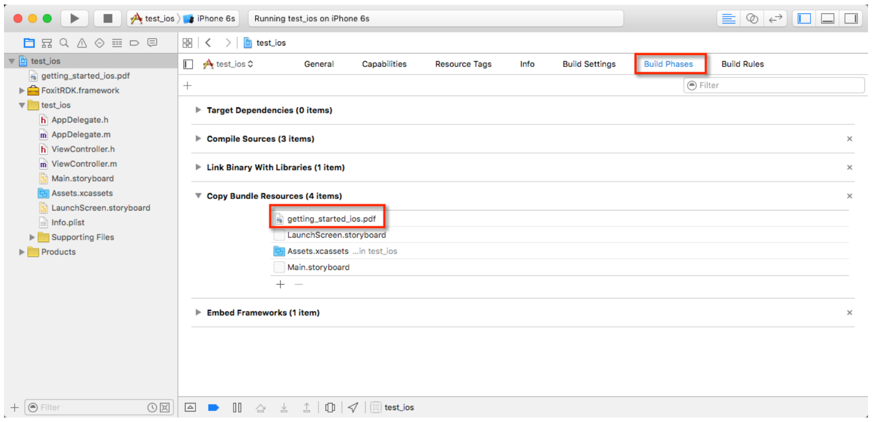


Figure 2-19

Then, add the following code to ViewController.m to display a PDF document. It's really easy to present a PDF on screen. All you need is to create a **FSPDFDoc** object and then show it with a **FSPDFViewCtrl** object.

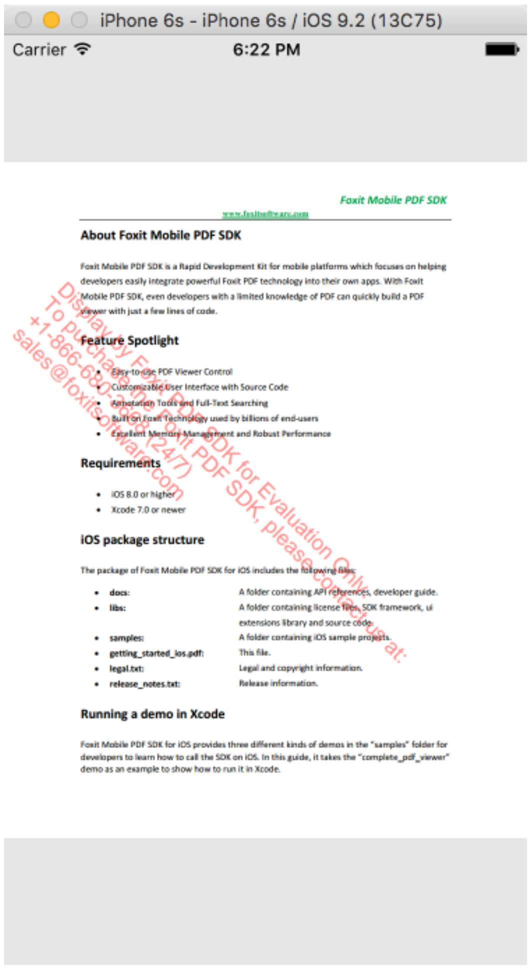
Update ViewController.m as follows:

```
#import "ViewController.h"
#import "FoxitRDK/FSPDFViewControl.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];

     // Get the path of a PDF
     NSString* pdfPath = [[NSBundle mainBundle] pathForResource:@"getting_started_ios"
ofType:@"pdf"];

     // Initialize a PDFDoc object with the path to the PDF file
     FSPDFDoc* pdfdoc = [FSPDFDoc createFromFilePath:pdfPath];
     if(e_errSuccess != [pdfdoc load:nil]) {
         return;
     }

    // Initialize a FSPDFViewCtrl object with the size of the entire screen
    FSPDFViewCtrl* pdfViewCtrl;
    pdfViewCtrl = [[FSPDFViewCtrl alloc] initWithFrame: [self.view bounds]];

    // Set the document to display
    [pdfViewCtrl setDoc:pdfdoc];

    // Add the pdfViewCtrl to the root view
    [self.view addSubview:pdfViewCtrl];

}
```

```
- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

Fantastic! We have now finished building a simple iOS 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 simulator.

In this guide, we build and run the project on an iPhone 6s Simulator, and you will see that the "getting_started_ios.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!



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

- iOS 8.0 or higher
- Xcode 7.0 or newer

iOS package structure

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

• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, SDK framework, ui extensions library and source code.
• **samples:**	A folder containing iOS sample projects.
• **getting_started_ios.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Running a demo in Xcode

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

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.

In the previous sections, we have introduced how to add ***FoxitRDK.framework*** to a project and how to build a simple iOS app for displaying a PDF document. Now, let's extend the simple app (*test_ios*) further to learn how to add support for text search, bookmarks, and 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.

Prepare work

Step 1: Add the UI extensions library (***libFoxitRDKUIExtensions.a***) to the project.

Note *In this app, we use the default built-in UI implementations to develop it, for simplicity and convenience, we will add* ***libFoxitRDKUIExtensions.a*** *to the test_ios project.*

Right-click the *test_ios* project, and select **Add Files to "test_ios"**… to add the extensions library. After adding, you can see the library in the Xcode's **Link Binary With Libraries** as shown in the Figure 2-21.

Note *You can add the library to* ***Link Binary With Libraries*** *directly. Just right-click the test_ios project, find* ***Link Binary With Libraries*** *in the* ***Build Phases*** *tab, press on the* ***+*** *button, and choose the file to add. In either case, please check the "****Copy items if needed****" option when choosing the file.*

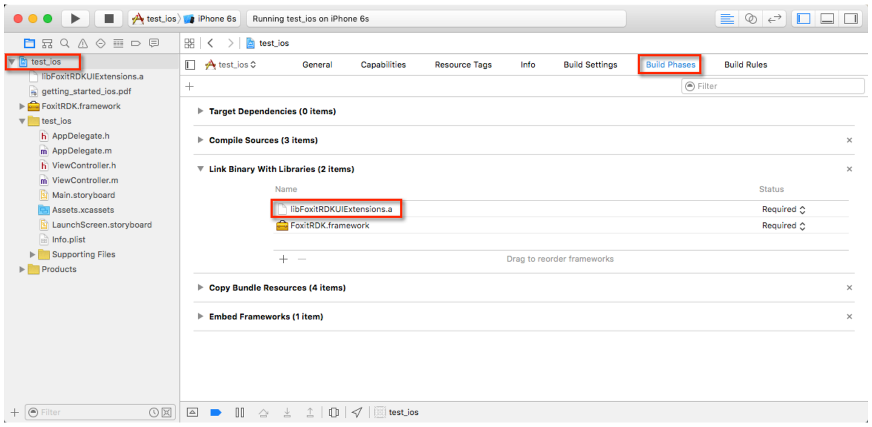
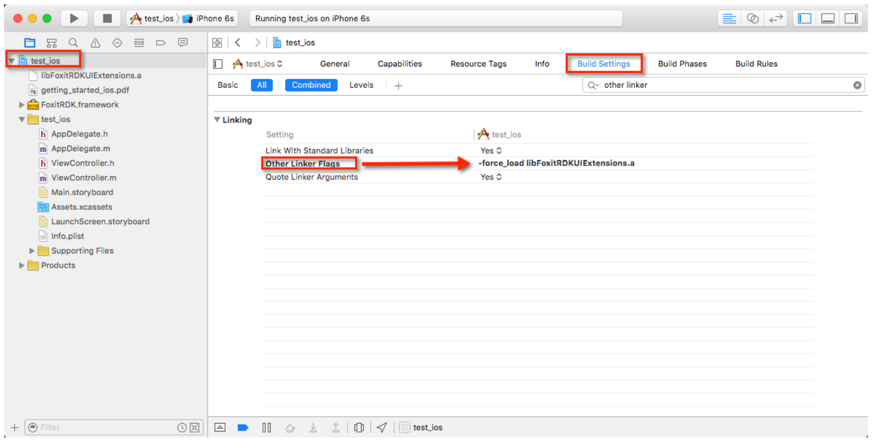


Figure 2-21

Step 2: Add "***-force_load libFoxitRDKUIExtensions.a***" to **Other Linker Flags** in the **Build Settings** tab as shown in the Figure 2-23. It is used to load all of the members that implement any Objective-C class or category in the static library



Linking

Setting	test_ios
Link With Standard Libraries	Yes
Other Linker Flags	-force_load libFoxitRDKUIExtensions.a
Quote Linker Arguments	Yes

Figure 2-22

Step 3: Copy the **uiextensions** folder from the "libs/uiextensions_src" of the download package to the "test_ios". This file contains the header files for ***libFoxitRDKUIExtensions.a***.

Note *This project only needs the "UIExtensionsManager.h" file. So you can just add this header file found in the "libs/uiextensions_src/uiextensions" of the download package to the project.*

Step 4: Add the **Resource** files that are needed for the built-in UI implementations to the *test_ios* project.

Right-click the *test_ios* project, and select **Add Files to "test_ios"**... to add the Resource files. Find and choose the folder as shown in the Figure 2-23.

Note *If you didn't copy the **uiextensions** file to the "test_ios", please find and choose the file in the "libs/uiextensions_src/uiextensions" of the download package.*

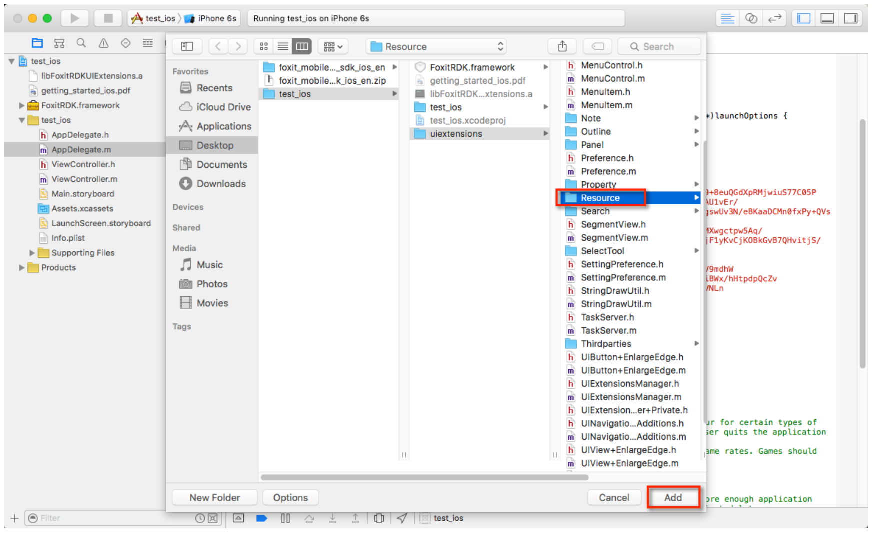


Figure 2-23

After adding, the *test_ios* project will look like the Figure 2-24.

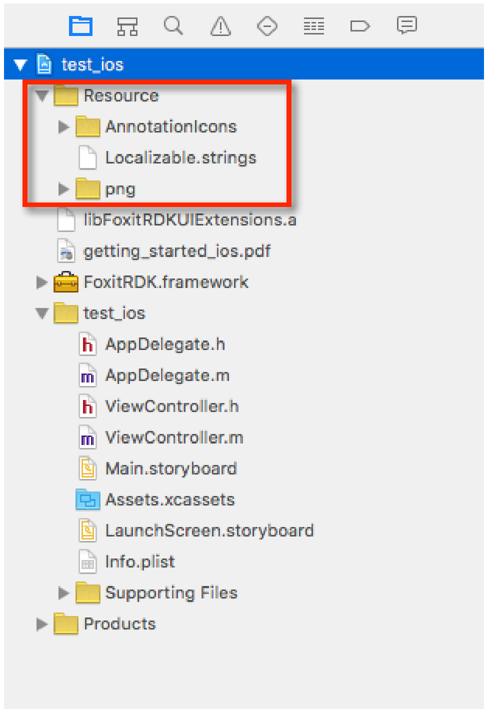


Figure 2-24

Step 5: Change the name of ViewController.m to ViewController.mm, which makes the file become an Objective-C++ file. It will ensure that the C++ standard library that is required by Foxit Mobile PDF SDK will be included at link time.

Note *The project should have at least one .mm or .cpp file, but which file is not important. Here, we change the filename of "ViewController.m"to "ViewController.mm".*

Add code to finish the support for text search

In the "ViewController.mm" 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 "ViewController.mm" file should look like.

Step 1: Initialize a UIExtensionsManager object and set it to PDFViewCtrl.

```
#import "../uiextensions/UIExtensionsManager.h"

UIExtensionsManager* extensionsManager;
...
extensionsManager = [[UIExtensionsManager alloc] initWithPDFViewControl:pdfViewCtrl];
pdfViewCtrl.extensionsManager = extensionsManager;
```

Step 2: Register the search event listener.

```
@interface ViewController () <ISearchEventListener>
...
[extensionsManager registerSearchEventListener:self];
```

Step 3: Add a search button, handle the search event, and set it to the root view.

```
UIButton* searchButton;
...
searchButton = [[UIButton alloc] initWithFrame:CGRectMake(280, 40, 80, 40)];
[searchButton setBackgroundColor:[UIColor grayColor]];
[searchButton setTitle: @"Search" forState: UIControlStateNormal];
[searchButton addTarget:self action:@selector(showSearchBar)
forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:searchButton];
```

The searchButton click event:

```
- (void)showSearchBar
{
    if (extensionsManager.currentAnnot) {

        [extensionsManager setCurrentAnnot:nil];
    }

    [extensionsManager showSearchBar:YES];
}
```

The search event listener

```
- (void)onSearchStarted {
    searchButton.hidden = YES;
}
```

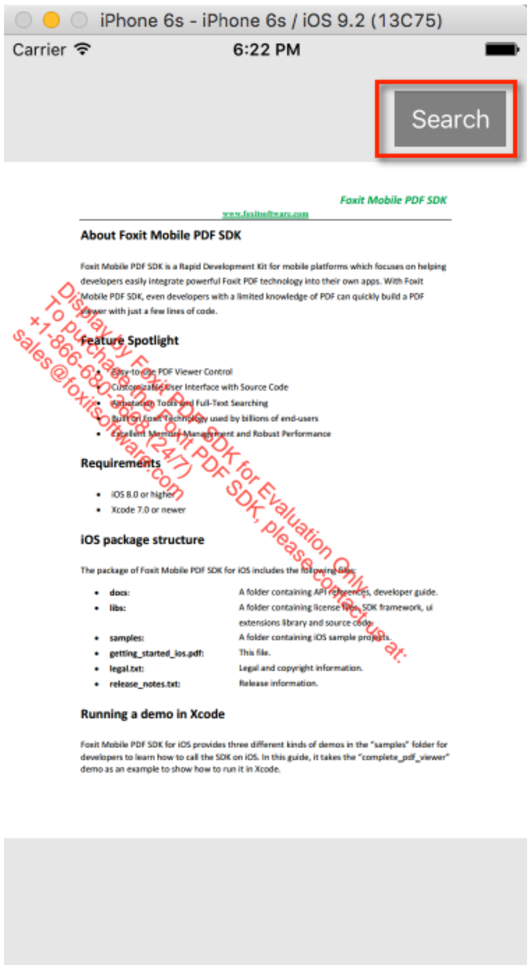
```
- (void)onSearchCanceled {
    searchButton.hidden = NO;
}
```

The whole update of ViewController.mm is as follows:

```
#import "ViewController.h"
#import "FoxitRDK/FSPDFViewControl.h"
#import "../uiextensions/UIExtensionsManager.h"

@interface ViewController () <ISearchEventListener>


@end

@implementation ViewController
{
    UIExtensionsManager* extensionsManager;
    UIButton* searchButton;
}

- (void)viewDidLoad {
    [super viewDidLoad];

    // Get the path of a PDF
    NSString* pdfPath = [[NSBundle mainBundle] pathForResource:@"getting_started_ios"
ofType:@"pdf"];

    // Initialize a PDFDoc object with the path to the PDF file
    FSPDFDoc* pdfdoc = [FSPDFDoc createFromFilePath:pdfPath];
    if(e_errSuccess != [pdfdoc load:nil]) {
        return;
    }

    // Initialize a FSPDFViewCtrl object with the size of the entire screen
    FSPDFViewCtrl* pdfViewCtrl;
    pdfViewCtrl = [[FSPDFViewCtrl alloc] initWithFrame: [self.view bounds]];

    // Set the document to display
    [pdfViewCtrl setDoc:pdfdoc];

    // Add the pdfViewCtrl to the root view
    [self.view addSubview:pdfViewCtrl];


    // Initialize a UIExtensionsManager object and set it to pdfViewCtrl
    extensionsManager = [[UIExtensionsManager alloc] initWithPDFViewControl:pdfViewCtrl];
    pdfViewCtrl.extensionsManager = extensionsManager;

    // Register the search event listener
    [extensionsManager registerSearchEventListener:self];


    // add a search button and handle the button click event
    searchButton = [[UIButton alloc] initWithFrame:CGRectMake(280, 40, 80, 40)];
    [searchButton setBackgroundColor:[UIColor grayColor]];
    [searchButton setTitle: @"Search" forState: UIControlStateNormal];
    [searchButton addTarget:self action:@selector(showSearchBar)
forControlEvents:UIControlEventTouchUpInside];
```

```
    // Add the searchButton to the root view
    [self.view addSubview:searchButton];
}

#pragma searchButton click event

- (void)showSearchBar
{
    if (extensionsManager.currentAnnot) {

        [extensionsManager setCurrentAnnot:nil];
    }

    [extensionsManager showSearchBar:YES];
}


#pragma ISearchEventListener

- (void)onSearchStarted {
    searchButton.hidden = YES;
}

- (void)onSearchCanceled {
    searchButton.hidden = NO;
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.

}

@end
```

Now that we have finished adding support for text search in the project, let’s run it on an iPhone 6s Simulator. You will see that the “getting_started_ios.pdf” document is automatically displayed as shown in the Figure 2-25.



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

- iOS 8.0 or higher
- Xcode 7.0 or newer

iOS package structure

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

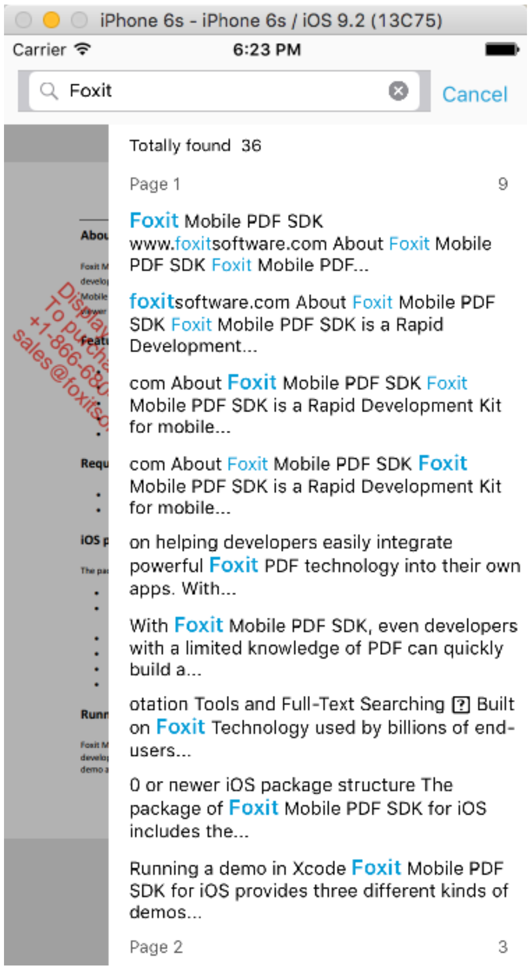
• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, SDK framework, ui extensions library and source code.
• **samples:**	A folder containing iOS sample projects.
• **getting_started_ios.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Running a demo in Xcode

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

Figure 2-25

Click on the "**Search**" button, 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 2-26.


iPhone 6s - iPhone 6s / iOS 9.2 (13C75)
Carrier 6:23 PM
Foxit Cancel

Totally found 36

Page 1 9

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

foxitsoftware.com About Foxit Mobile PDF SDK Foxit Mobile PDF SDK is a Rapid Development...

com About Foxit Mobile PDF SDK Foxit Mobile PDF SDK is a Rapid Development Kit for mobile...

com About Foxit Mobile PDF SDK Foxit Mobile PDF SDK is a Rapid Development Kit for mobile...

on helping developers easily integrate powerful Foxit PDF technology into their own apps. With...

With Foxit Mobile PDF SDK, even developers with a limited knowledge of PDF can quickly build a...

otation Tools and Full-Text Searching ? Built on Foxit Technology used by billions of end-users...

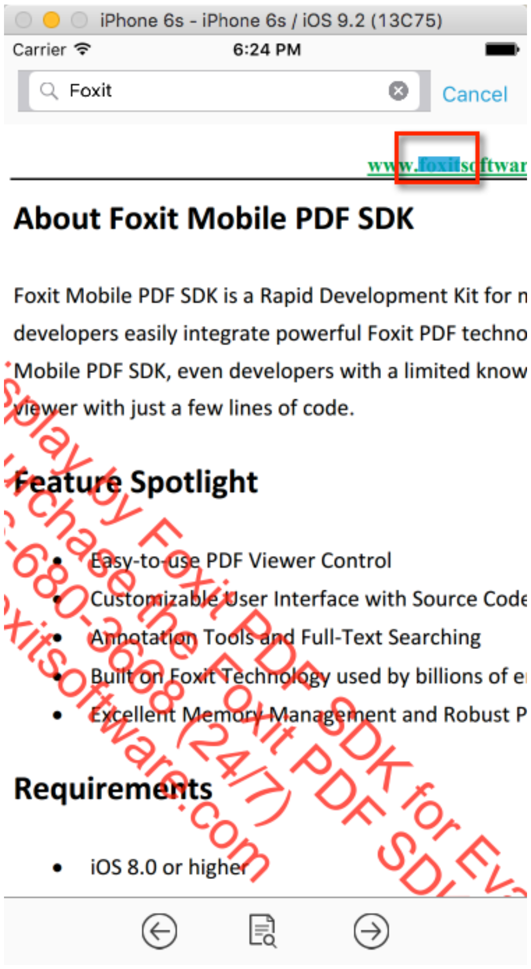
0 or newer iOS package structure The package of Foxit Mobile PDF SDK for iOS includes the...

Running a demo in Xcode Foxit Mobile PDF SDK for iOS provides three different kinds of demos...

Page 2 3

Figure 2-26

Click any result in the list to jump to the specific location. Here, we click the result in the second paragraph in the list, and then it will jump to the place where the result is, and the word will be highlighted as shown in the Figure 2-27 (zoom in on the page to see it clearly). You can click the previous or next button to find the previous or next search result.



About Foxit Mobile PDF SDK

Foxit Mobile PDF SDK is a Rapid Development Kit for n
developers easily integrate powerful Foxit PDF techno
Mobile PDF SDK, even developers with a limited know
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 e
- Excellent Memory Management and Robust P

Requirements

- iOS 8.0 or higher

Figure 2-27

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 features 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 required files into your app.

- ***FoxitRDK.framework*** – The framework that includes the Foxit Mobile PDF SDK dynamic library and associated header files. It can be found in the "libs" folder.
- ***uiextensions*** *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 iOS 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" project. It can be found in the "libs/uiextensions_src" folder.

  Note *The built-in UI customization can be done in the* ***uiextensions*** *project, and then you can add the new* ***libFoxitRDKUIExtensions.a*** *library generated by the modified* ***uiextensions*** *project to your app instead of the whole* ***uiextensions*** *project.*

Second, find the specific code or images related to the UI that you want to customize in the ***uiextensions*** project, then modify them based on your requirements.

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

UI Customization Example

Step 1: Add the ***uiextensions*** project into the demo.

Note *We will add the* ***uiextensions*** *project to the demo which is convenient for us to see the custom results. The demo already includes* ***FoxitRDK.framework****, so we just need to add the* ***uiextensions*** *project.*

Load the "**viewer_ctrl_demo**" project in Xcode. Drag "**uiextensions.xcodeproj**" found in the "libs/uiextensions_src" of the download package to the "**viewer_ctrl_demo**" project as shown in the Figure 3-1.

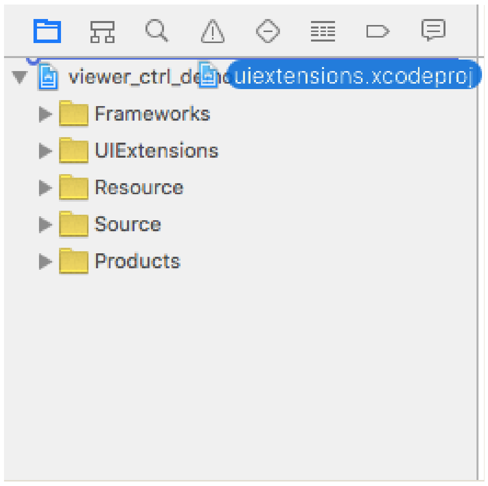
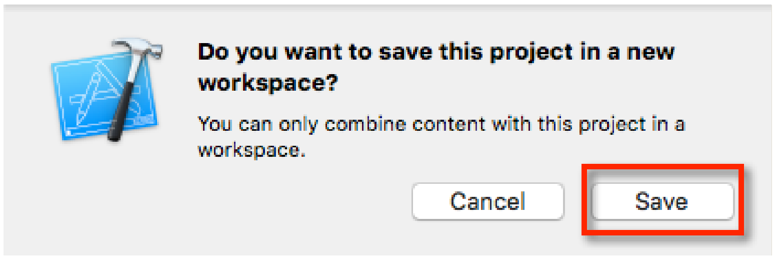


Figure 3-1

Then, it will pop up a dialog box which prompts you whether to save the project in a new workspace as shown in the Figure 3-2. Click **Save**.



Do you want to save this project in a new workspace?

You can only combine content with this project in a workspace.

Figure 3-2

Save the workspace to the "samples" folder, and name "custom_viewer" as shown in the Figure 3-3. Click **Save**.

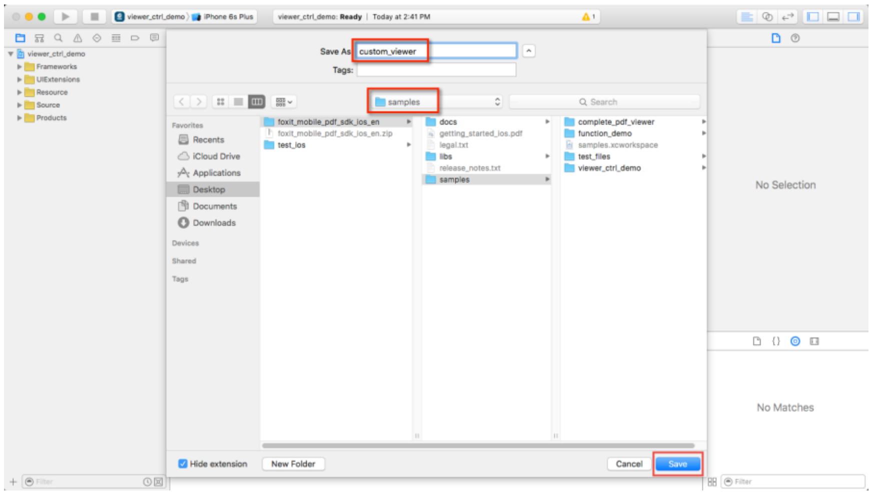


Figure 3-3

Now, the workspace looks like the Figure 3-4.

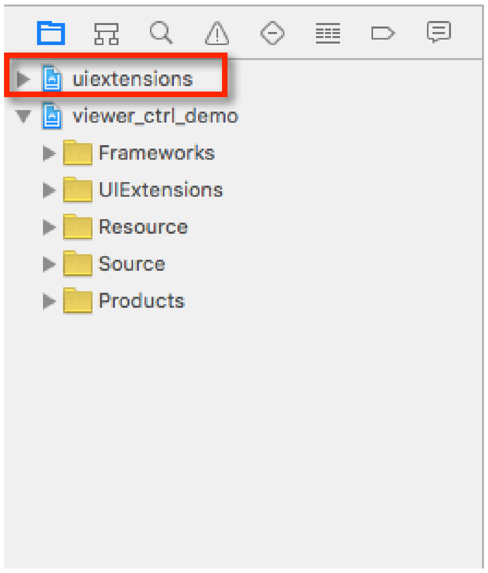
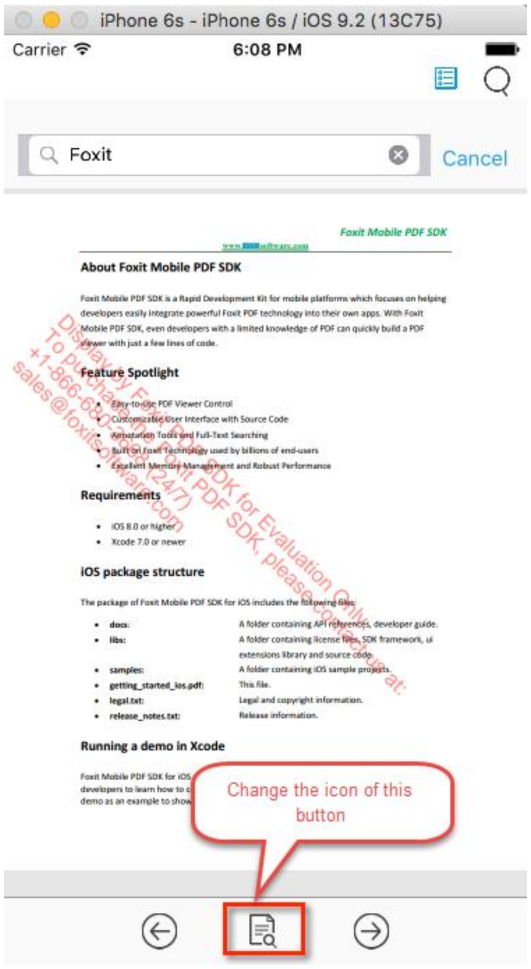


Figure 3-4

Congratulations! You have completed the first step.

Step 2: Find and modify the code or images 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 3-5.


iPhone 6s - iPhone 6s / iOS 9.2 (13C75)
Carrier
6:08 PM
Foxit
Cancel
Change the icon of this button

Foxit Mobile PDF SDK

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

- iOS 8.0 or higher
- Xcode 7.0 or newer

iOS package structure

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

• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, SDK framework, ui extensions library and source code.
• **samples:**	A folder containing iOS sample projects.
• **getting_started_ios.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Running a demo in Xcode

Foxit Mobile PDF SDK for iOS
developers to learn how to c
demo as an example to show

Figure 3-5

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 "**Resource** -> **png** -> **image** -> **Search**" as shown in the Figure 3-6. It's easy to find the image that we want to replace. The resource files are stored according to the features, so you can locate the related code through the icon's name.

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

There are three folders used to store the image resources as follows:

- ✓ *Image: used for the non-retina devices.*
- ✓ *Image2x: used for the retina devices.*
- ✓ *Image3x: used for larger screen iPhones, such as iPhone 6 Plus and 6s Plus.*

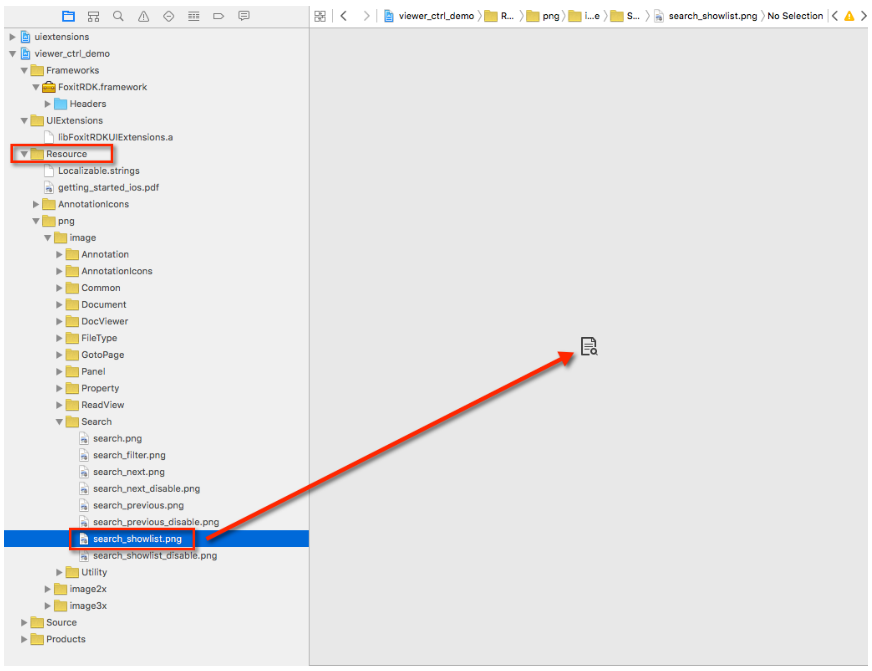
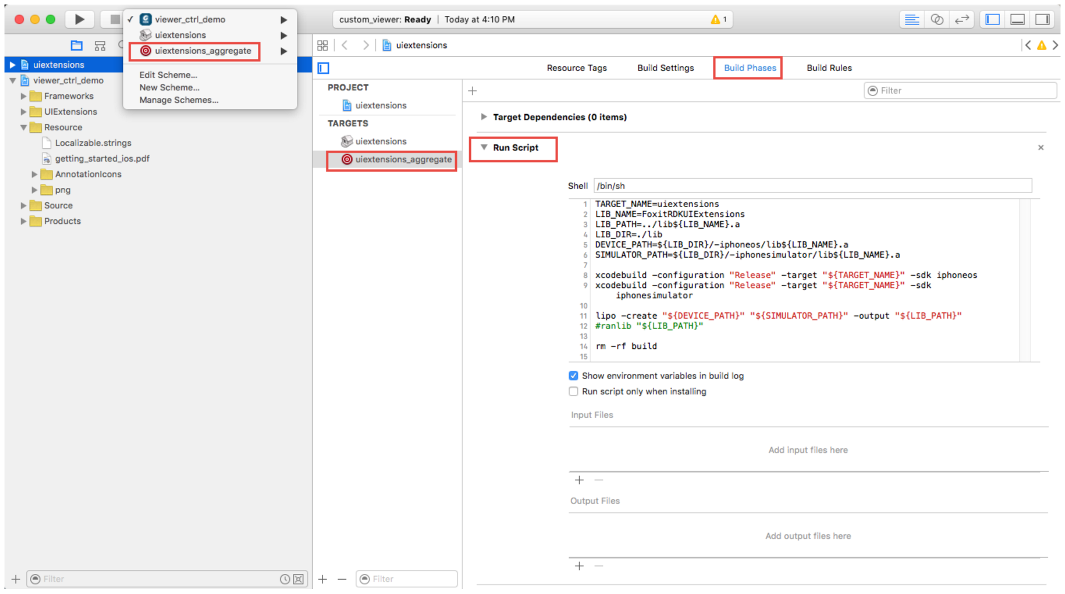


Figure 3-6

Right now, just replace "search_showlist.png" with your own icon. For example, we use the icon of the top search button (search.png) to replace it.

An iPhone 6s Simulator will be used as an example to run the demo, so replace the icon in the "libs/uiextensions_src/uiextensions/Resource/png/image2x" folder.

First, build and run the ***uiextensions_aggregate*** *project* as shown in the Figure 3-7.



Shell /bin/sh

```
TARGET_NAME=uiextensions
LIB_NAME=FoxitRDKUIExtensions
LIB_PATH=../lib${LIB_NAME}.a
LIB_DIR=./lib
DEVICE_PATH=${LIB_DIR}/-iphoneos/lib${LIB_NAME}.a
SIMULATOR_PATH=${LIB_DIR}/-iphonesimulator/lib${LIB_NAME}.a

xcodebuild -configuration "Release" -target "${TARGET_NAME}" -sdk iphoneos
xcodebuild -configuration "Release" -target "${TARGET_NAME}" -sdk
    iphonesimulator

lipo -create "${DEVICE_PATH}" "${SIMULATOR_PATH}" -output "${LIB_PATH}"
#ranlib "${LIB_PATH}"

rm -rf build
```

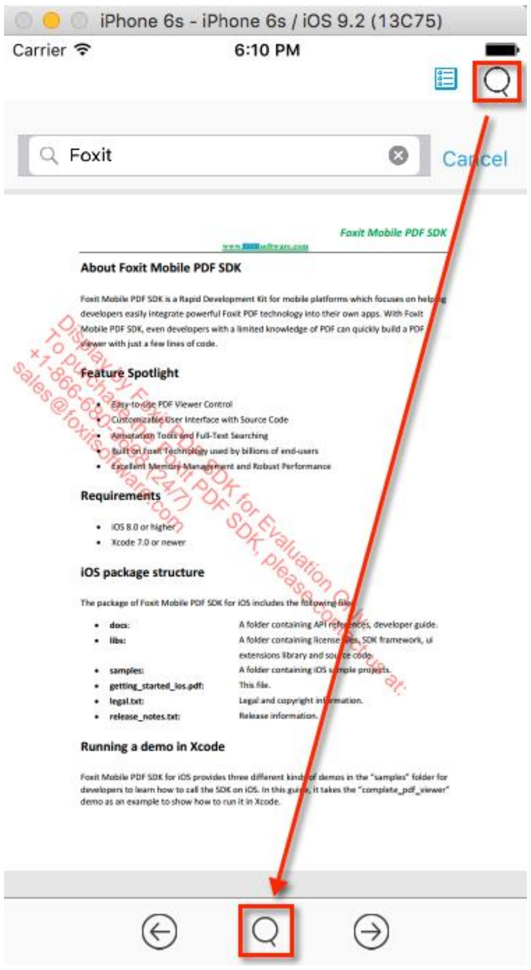
[✓] Show environment variables in build log
[ ] Run script only when installing

Figure 3-7

Here, we build and run ***uiextensions_aggregate****, then* the generated library will include the libraries for both simulator and iOS device. If you choose ***uiextensions*** *and run it on a simulator, then the generated library is a simulator library which can only be used for simulator.*

Note *The "**libFoxitRDKUIExtensions.a**" library in the "libs" folder of the download package is a universal static library which includes the libraries for both simulator and iOS device. It will be overwritten after building the "uiextensions" project successfully. In the "uiextension" project, it has already added the scripts to generate the universal static library as shown in the Figure 3-7.*

Then build and run the "**viewer_ctrl_demo**" project, after building successfully, try the search feature and we can see that the icon of the bottom search button has changed as shown in the Figure 3-8.



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

- iOS 8.0 or higher
- Xcode 7.0 or newer

iOS package structure

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

• **docs:**	A folder containing API references, developer guide.
• **libs:**	A folder containing license files, SDK framework, ui extensions library and source code.
• **samples:**	A folder containing iOS sample projects.
• **getting_started_ios.pdf:**	This file.
• **legal.txt:**	Legal and copyright information.
• **release_notes.txt:**	Release information.

Running a demo in Xcode

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

Figure 3-8

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*** 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 ***uiextension*** project found in the "libs" folder.

To create a new tool, the most important step is to declare a class that implements the "**IToolHandler**" 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 class named **ScreenCaptureToolHandler** that implements the "**IToolHandler**" interface.
- Handle the **onPageViewLongPress** and **onDraw** events.
- Instantiate a **ScreenCaptureToolHandler** object, and then register it to the extensions manager.
- Set the **ScreenCaptureToolHandler** object as the current tool handler.

Step 1: Create a class named **ScreenCaptureToolHandler** that implements the "**IToolHandler**" interface.

a) Load the "**viewer_ctrl_demo**" project in Xcode. Create a class named "**ScreenCaptureToolHandler**" in the "Source" folder, and create the corresponding header file.

b) Let the **ScreenCaptureToolHandler** class implement the I**ToolHandler** interface as follows:

```
@interface ScreenCaptureToolHandler : NSObject<IToolHandler>
```

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

Update **ScreenCaptureToolHandler.h** as follows:

```
#import <Foundation/Foundation.h>
#import <FoxitRDK/FSPDFViewControl.h>
#import "../../../libs/uiextensions_src/uiextensions/UIExtensionsManager.h"
@protocol IToolHandler;
```

```
@class TaskServer;

@interface ScreenCaptureToolHandler : NSObject<IToolHandler>

- (instancetype)initWithUIExtensionsManager:(UIExtensionsManager*)extensionsManager
taskServer:(TaskServer*)taskServer;
@end
```

Update **ScreenCaptureToolHandler.m** as follows:

```
#import "ScreenCaptureToolHandler.h"
#import "../../../libs/uiextensions_src/uiextensions/UIExtensionsManager.h"
#import <ImageIO/ImageIO.h>
#import <ImageIO/CGImageDestination.h>
#import <MobileCoreServices/UTCoreTypes.h>

@interface ScreenCaptureToolHandler ()

@end

@implementation ScreenCaptureToolHandler {
    UIExtensionsManager* _extensionsManager;
    FSPDFViewCtrl* _pdfViewCtrl;
    TaskServer* _taskServer;

    CGPoint startPoint;
    CGPoint endPoint;

}
@synthesize type;

- (instancetype)initWithUIExtensionsManager:(UIExtensionsManager*)extensionsManager
taskServer:(TaskServer*)taskServer
{
    self = [super init];
    if (self) {
        _extensionsManager = extensionsManager;
        _pdfViewCtrl = extensionsManager.pdfViewCtrl;
        _taskServer = taskServer;
    }
    return self;
}

-(NSString*)getName
{
    return @" ";
}

-(BOOL)isEnabled
{
    return YES;
}

-(void)onActivate
{

}

-(void)onDeactivate
{
}
// Save the image to a specified path.
- (void)saveJPGImage:(CGImageRef)imageRef path:(NSString *)path
{
    NSURL *fileURL = [NSURL fileURLWithPath:path];
    CGImageDestinationRef dr = CGImageDestinationCreateWithURL((__bridge CFURLRef)fileURL, 
kUTTypeJPEG , 1, NULL);
    
    CGImageDestinationAddImage(dr, imageRef, NULL);
    CGImageDestinationFinalize(dr);
    
    CFRelease(dr);
}

// Handle the PageView Gesture and Touch event
- (BOOL)onPageViewLongPress:(int)pageIndex recognizer:(UILongPressGestureRecognizer 
*)recognizer
{
    if (recognizer.state == UIGestureRecognizerStateBegan)
    {
        startPoint = [recognizer locationInView:[_pdfViewCtrl getPageView:pageIndex]];
        endPoint = startPoint;
    }
    else if (recognizer.state == UIGestureRecognizerStateChanged)
    {
        
        endPoint = [recognizer locationInView:[_pdfViewCtrl getPageView:pageIndex]];
        
        // Refresh the page view, then the onDraw event will be triggered.
        [_pdfViewCtrl refresh:pageIndex];
    }
    else if (recognizer.state == UIGestureRecognizerStateEnded || recognizer.state == 
UIGestureRecognizerStateCancelled)
    {
        // Get the size of the Rect.
        CGSize size = {fabs(endPoint.x-startPoint.x), fabs(endPoint.y-startPoint.y)};
        CGPoint origin = {startPoint.x<endPoint.x?startPoint.x:endPoint.x, 
startPoint.y<endPoint.y?startPoint.y:endPoint.y};
        // Get the Rect.
        CGRect rect = {origin, size};
        
        int newDibWidth = rect.size.width;
        int newDibHeight = rect.size.height;
        UIView* pageView = [_pdfViewCtrl getPageView:pageIndex];
        CGRect bound = pageView.bounds;
        
        // Create a bitmap with the size of the selected area.
        int imgSize = newDibWidth*newDibHeight*4;
        void* pBuff = malloc(newDibWidth*newDibHeight*4);
        FSBitmap* fsbitmap = [FSBitmap create:newDibWidth height:newDibHeight format:e_dibArgb 
buffer:pBuff pitch:newDibWidth*4];
        [fsbitmap fillRect:0xFFFFFFFF rect:nil];
        FSRenderer* fsrenderer = [FSRenderer create:fsbitmap rgbOrder:YES];
        FSPDFPage* page = [_pdfViewCtrl.currentDoc getPage:pageIndex];
        
        // Calculate the display matrix.
        FSMatrix* fsmatrix = [page getDisplayMatrix:-rect.origin.x yPos:-rect.origin.y 
xSize:bound.size.width ySize:bound.size.height rotate:0];
```

```
        // Set the render content, then start to render the selected area to the bitmap.
        [fsrenderer setRenderContent:e_renderPage|e_renderAnnot];
        [fsrenderer startRender:page matrix:fsmatrix pause:nil];
        [fsrenderer continueRender];

        // Convert FSBitmap to CGImage.
        CGDataProviderRef provider = CGDataProviderCreateWithData(NULL, pBuff, imgSize, nil);
        CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();
        CGBitmapInfo bitmapInfo = kCGBitmapByteOrderDefault|kCGImageAlphaLast;

        CGImageRef image = CGImageCreate(newDibWidth,newDibHeight, 8, 32, newDibWidth * 4,
                                         colorSpace, bitmapInfo,
                                         provider, NULL, YES, kCGRenderingIntentDefault);

        // Save the image to a specified path.
        NSString* jpgPath =@"/Users/Foxit/Desktop/ScreenCapture.jpg";
        [self saveJPGImage:image path:jpgPath];

        UIAlertView *alert = [[UIAlertView alloc]initWithTitle:@""
                                                       message:@" The selected area was saved
as a JPG stored in the /Users/Foxit/Desktop/ScreenCapture.jpg" delegate:nil
cancelButtonTitle:NSLocalizedString(@"OK", @"OK") otherButtonTitles:nil];
        [alert show];

        if (fsrenderer != nil) {
            [fsrenderer dealloc];
        }

        return YES;
    }
    return YES;
}

- (BOOL)onPageViewTap:(int)pageIndex recognizer:(UITapGestureRecognizer *)recognizer
{
    return NO;
}

- (BOOL)onPageViewPan:(int)pageIndex recognizer:(UIPanGestureRecognizer *)recognizer
{
    return NO;

}

- (BOOL)onPageViewShouldBegin:(int)pageIndex recognizer:(UIGestureRecognizer
*)gestureRecognizer
{
    if (self != [_extensionsManager getCurrentToolHandler]) {
        return NO;
    }
    return YES;
}

- (BOOL)onPageViewTouchesBegan:(int)pageIndex touches:(NSSet*)touches
withEvent:(UIEvent*)event
{
    return NO;
}
```

```
- (BOOL)onPageViewTouchesMoved:(int)pageIndex touches:(NSSet *)touches withEvent:(UIEvent
*)event
{
    return NO;
}

- (BOOL)onPageViewTouchesEnded:(int)pageIndex touches:(NSSet *)touches withEvent:(UIEvent
*)event
{
    return NO;
}

- (BOOL)onPageViewTouchesCancelled:(int)pageIndex touches:(NSSet *)touches withEvent:(UIEvent
*)event
{
    return NO;
}

// Handle the drawing event.
-(void)onDraw:(int)pageIndex inContext:(CGContextRef)context
{
    if ([_extensionsManager getCurrentToolHandler] != self) {
        return;
    }

    CGContextSetLineWidth(context, 2);
    CGContextSetLineCap(context, kCGLineCapSquare);
    UIColor *color = [UIColor redColor];
    CGContextSetStrokeColorWithColor(context, [color CGColor]);
    CGPoint points[] = {startPoint,CGPointMake(endPoint.x,
startPoint.y),endPoint,CGPointMake(startPoint.x, endPoint.y)};
    CGContextAddLines(context,points,4);
    CGContextClosePath(context);
    CGContextStrokePath(context);
}

@end
```

***Note** In the above code, you should specify an existing path to save the image. Here, the path is "@"/Users/Foxit/Desktop/ScreenCapture.jpg"", please replace it with a valid path.*

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

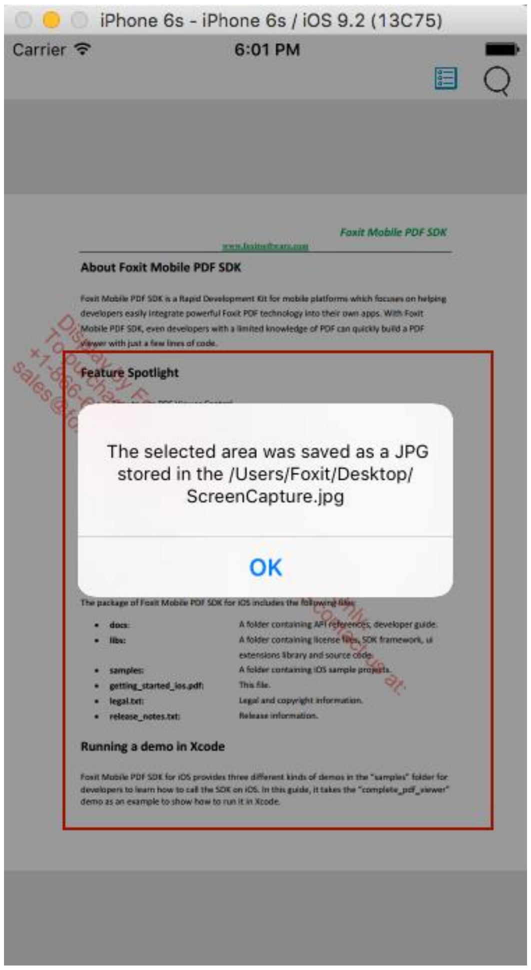
```
#import "ScreenCaptureToolHandler.h"
...

@property (nonatomic, strong) ScreenCaptureToolHandler* screencaptureToolHandler;
...

self.screencaptureToolHandler = [[[ScreenCaptureToolHandler alloc] initWithUIExtensionsManager:
self.extensionsManager taskServer:nil] autorelease];
[self.extensionsManager registerToolHandler:self.screencaptureToolHandler];
```

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

```
[self.extensionsManager setCurrentToolHandler:self.screencaptureToolHandler];
```

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 the code referred in Step 3 and Step 4 to ViewController.mm.

After finishing all of the above work, build and run the demo. An iPhone 6s Simulator will be used as an example to run the project. After building the demo successfully, long press and select a rectangular area, and then a message box will be popped up as shown in the Figure 4-1. It shows where the image (selected area) was saved to.



The selected area was saved as a JPG stored in the /Users/Foxit/Desktop/ ScreenCapture.jpg

Foxit Mobile PDF SDK

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

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

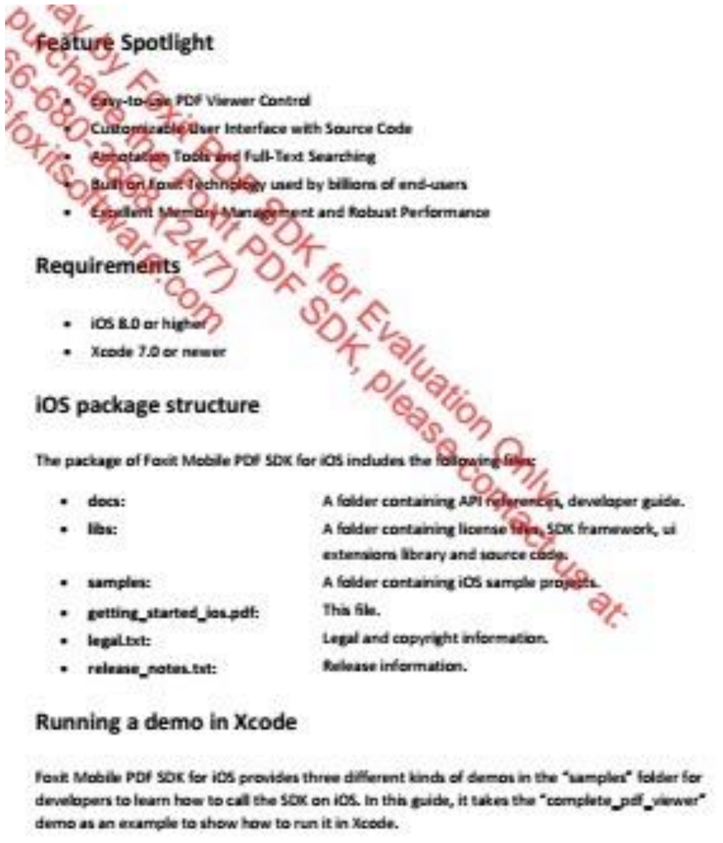
- docs: A folder containing API references, developer guide.
- libs: A folder containing license files, SDK framework, ui extensions library and source code.
- samples: A folder containing iOS sample projects.
- getting_started_ios.pdf: This file.
- legal.txt: Legal and copyright information.
- release_notes.txt: Release information.

Running a demo in Xcode

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

Figure 4-1

In order to verify whether the tool captures the selected area successfully, we need to find the screenshot. Go to "desktop", we can see the image as shown in the Figure 4-2.



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

- iOS 8.0 or higher
- Xcode 7.0 or newer

iOS package structure

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

• docs:	A folder containing API references, developer guide.
• libs:	A folder containing license files, SDK framework, ui extensions library and source code.
• samples:	A folder containing iOS sample projects.
• getting_started_ios.pdf:	This file.
• legal.txt:	Legal and copyright information.
• release_notes.txt:	Release information.

Running a demo in Xcode

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

Figure 4-2

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 the 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